The HP-GL/2 Reference Guide

A Handbook for Program Developers

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

The HP-GL/2 Reference Guide
A Handbook for Program Developers

ADDISON-WESLEY PUBLISHING COMPANY, INC.
Reading, Massachusetts Menlo Park, California New York
Don Mills, Ontario Wokingham, England Amsterdam Bonn
Sydney Singapore Tokyo Madrid San Juan

Notice

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental consequential damages in connection with the furnishing, performance, or use of this material.

Library of Congress Cataloging-in-Publication Data

The HP-GL/2 reference guide : a handbook for program developers /
Hewlett Packard. -- 1st ed.
p. cm.
ISBN 0-201-56308-8
1. HP-GL/2 (Computer program language) 2. Computer graphics.
I. Hewlett-Packard Company.
QA76.73.H6H6 1990 90-344
006.6'869--dc20 CIP

San Diego Division
16399 W. Bernardo Drive
San Diego, CA 92127-1899, U.S.A.

ISBN: 0-201-56308-8
ABCDEFHIJ-HA-9543210
First printing, April 1990

How to Use This Guide

This is a generic guide to HP-GL/2, Hewlett-Packard's Graphics Language standard supported by many Hewlett-Packard graphics peripherals. This manual describes each of the HP-GL/2 instructions without relying on a specific device or technology.

You must use a programming language in addition to HP-GL/2. However, this manual will not teach you how to program your computer. Your method of programming will depend on your computer system, the programming language you use, and your level of expertise. The manual, though, does give recommendations for getting the most from your HP-GL/2 device.

The manual is divided into two parts. The first gives an overview of the language and discusses plotting and programming concepts. The second describes the HP-GL/2 kernel instructions (supported by all HP-GL/2 devices) and extension instructions. The extensions help you make the best use of specific device technologies and, therefore, may not be supported on all HP-GL/2 devices. For example, because you cannot digitize on electrostatic plotters, those devices do not support the Digitizing Extension.

Organization

Introduction: Chapters 1–3 give you the basic information you need to begin HP-GL/2 programming, including tips for flexible and effective programs.

HP-GL/2 Kernel: Chapters 4–8 explain all of the instructions in each functional group of the language core.

HP-GL/2 Extensions: Chapters 9–12 describe each of the extensions that give flexibility in supporting other technologies.

In addition, the glossary will help you understand words with which you may be unfamiliar, and the index will help you quickly find the information you need.

Manual Terms and Conventions

The symbols and type styles used to represent the syntax requirements of HP-GL/2 are discussed in chapter 3, *Writing Efficient Programs*.

Numbers are presented using SI (International System of Units) standards. Numbers with more than four digits are placed in groups of three—separated by a space instead of a comma—counting to both the left and right of the decimal point (e.g., 54 321.123 45).

BOLDFACE TYPE denotes an ASCII control character, such as **ESC** (Escape).

All references to RS-232-C interface in this manual apply equally to RS-232-C and CCITT V.24 interfaces. The term RS-232-C is used for simplicity.

Additional Documentation

The *HP-GL/2 Comparison Guide* shows you how HP-GL/2 is implemented across a wide variety of HP-GL/2 devices. The *HP-GL/2 Comparison Guide* also indicates what devices support which extensions.

Each product's *User's Guide* includes an appendix that lists the kernel and extension instructions supported by that product.

Table of Contents

Chapter 4: The Configuration and Status Group

Chapter 5: The Vector Group

Chapter 8: The Character Group

Chapter 9: The Technical Graphics Extension

Chapter 10: The Palette Extension

The HP-GL/2 Reference Guide

A Handbook for Program Developers

1

Introduction to HP-GL/2

HP-GL/2 is the standardized version of the Hewlett-Packard Graphics Language. It is designed to provide consistent functionality that can be supported across a wide range of peripherals, reducing program development efforts. HP-GL/2 allows you great flexibility in creating images and maximizes your device's throughput and the future compatibility of your program with other HP-GL/2 devices.

HP-GL/2 Overview

HP-GL/2 is made up of a core set of instructions (called the HP-GL/2 kernel) and several extensions. All HP-GL/2 devices support the kernel and the Technical Graphics Extension (the Technical Graphics Extension may not be supported on devices for which HP-GL/2 is an added option). Most HP-GL/2 devices will also support the Palette Extension. The remaining extensions make use of specific technologies and are, therefore, device-specific. The following shows the HP-GL/2 model.

All HP-GL/2 Devices	Device-Specific HP-GL/2 Extensions			
HP-GL/2 kernel	Technical Graphics Extension	Palette Extension	Dual-Context Extension	Digitizing Extension

HP-GL/2 Kernel

The kernel is the foundation of the HP-GL/2 and contains most of the instructions. All HP-GL/2 devices support the kernel instructions. The kernel instructions are divided into five functional groups: configuration and status, vector, polygon, line and fill attributes, and character.

The Configuration and Status Group

The instructions in this group help you set up your plot by reestablishing default conditions, scaling, and manipulating the plotting area (through soft-clip limits and rotation). This group contains no graphics instructions but does set up your plotting area for the graphics information it is about to receive. The Configuration and Status Group is described in detail in chapter 4.

The Configuration and Status Group Instructions	
DF, Default Values	PG, Advance Full Page
IN, Initialize	RO, Rotate Coordinate System
IP, Input P1 and P2	RP, Replot
IR, Input Relative P1 and P2	SC, Scale
IW, Input Window	

The Vector Group

The instructions in this group enable you to draw vector graphics (lines and arcs) using either absolute or relative coordinates. The Vector Group is described in detail in chapter 5.

The Vector Group Instructions	
AA, Arc Absolute	PD, Pen Down
AR, Arc Relative	PE, Polyline Encoded
AT, Absolute Arc Three Point	PR, Plot Relative
CI, Circle	PU, Pen Up
PA, Plot Absolute	RT, Relative Arc Three Point

The Polygon Group

The instructions in this group use the polygon buffer in your HP-GL/2 device. Some of the instructions draw shapes while others act on the contents of the polygon buffer (filling or edging). The Polygon Group is described in detail in chapter 6.

The Polygon Group Instructions	
EA, Edge Rectangle Absolute	PM, Polygon Mode
ER, Edge Rectangle Relative	RA, Fill Rectangle Absolute
EW, Edge Wedge	RR, Fill Rectangle Relative
EP, Edge Polygon	WG, Fill Wedge
FP, Fill Polygon	

The Line and Fill Attributes Group

The instructions in this group enable you to enhance your plots through the use of different line and fill types. You will also learn how to manipulate area fill patterns and specify varying widths for different pen widths. The Line and Fill Attributes Group is described in detail in chapter 7.

The Line and Fill Attributes Group Instructions	
AC, Anchor Corner	RF, Raster Fill Definition
FT, Fill Type	SM, Symbol Mode
LA, Line Attributes	SP, Select Pen
LT, Line Type	UL, User-Defined Line Type
PW, Pen Width	WU, Pen Width Unit Selection

The Character Group

The instructions in this group enable you to select different fonts or character sets and manipulate their direction, size, and appearance. The Character Group is described in detail in chapter 8.

The Character Group Instructions	
AD, Alternate Font Definition	LO, Label Origin
CF, Character Fill Mode	SA, Select Alternate Font
CP, Character Plot	SD, Standard Font Definition
DI, Absolute Direction	SI, Absolute Character Size
DR, Relative Direction	SL, Character Slant
DT, Define Label Terminator	SR, Relative Character Size
DV, Define Variable Text Path	SS, Select Standard Font
ES, Extra Space	TD, Transparent Data
LB, Label	

Technical Graphics Extension

The instructions in this extension add flexibility often required in technical fields such as CAD, architectural rendering, IC layout, etc. All HP-GL/2 devices support the Technical Graphics Extension (though it may not be supported on devices for which HP-GL/2 is an option). This extension is described in detail in chapter 9.

Technical Graphics Extension Instructions	
BP, Begin Plot	OE, Output Error
CT, Chord Tolerance Mode	OH, Output Hard-Clip Limits
DL, Download Character	OI, Output Identification
EC, Enable Cutter	OP, Output P1 and P2
FR, Frame Advance	OS, Output Status
MC, Merge Control	PS, Plot Size
MG, Message	QL, Quality Level
MT, Media Type	ST, Sort
NR, Not Ready	VS, Velocity Select

Palette Extension

The instructions in this extension help maximize raster technology. However, this extension is not limited to HP-GL/2 raster devices. Pen plotters may support this extension and default some instructions in accordance with pen plotter technology. The Palette Extension is described in detail in chapter 10.

Palette Extension Instructions	
CR, Set Color Range for Relative Color Data	PC, Pen Color Assignment
NP, Number of Pens	SV, Screened Vectors
	TR, Transparency Mode

Dual-Context Extension

The instructions in this extension help satisfy desktop presentation requirements, specifically the merging of word-processed text and raster graphics images with vector graphics images. Check to see whether or not your device supports this extension. The Dual-Context Extension is described in detail in chapter 11.

Dual-Context Extension Instructions	
ESC%#A, Enter PCL Mode	FN, Secondary Font Selection by ID
ESCE, Reset	SB, Scalable or Bitmap Fonts
FI, Primary Font Selection by ID	

Digitizing Extension

The instructions in this group can be used on pen plotters only. Raster devices are incapable of digitizing an image. The Digitizing Extension is described in detail in chapter 12.

Digitizing Extension Instructions	
DC, Digitize Clear	OD, Output Digitized Point and Pen Status
DP, Digitize Point	

2

Plotting Concepts

This chapter gives an overview of some important plotting concepts. These concepts are basic to using a plotter. You will find this chapter useful if you have never done any graphics programming or if you have written programs for a graphics terminal but not a plotter. The plotting concepts this chapter discusses include the following.

- graphics limits
- the coordinate system
- units of measure
- scaling

This chapter also discusses plot rotation. If you already know these concepts, go on to chapter 3.

Graphics Limits

Your plotting area usually does not extend to the limits of your media. There is a physical limit beyond which your device cannot draw. This limit provides a neat margin for holding the plot and prevents the smearing of ink by pen plotter pinch rollers and loss of vacuum on electrostatic plotters.

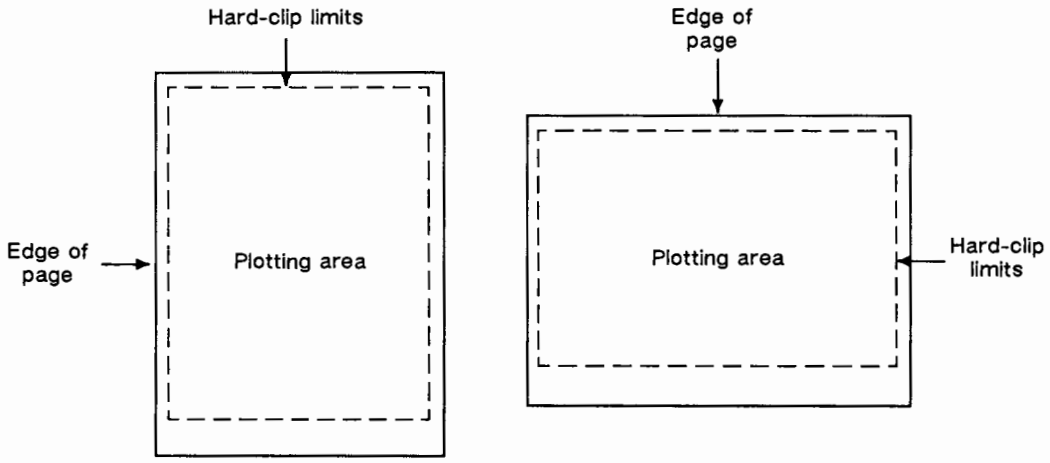
The plotter recognizes two types of graphics limits:

- hard-clip limits
- soft-clip limits

Hard-clip refers to a physical boundary, beyond which a pen cannot move. *Soft-clip* refers to a programmatically set boundary that allows pen movement within its borders only. You can set soft-clip limits at any time in your program. When you turn on or initialize the plotter, the hard-clip and soft-clip limits are identical.

Hard-Clip Limits

The hard-clip limits represent the physical boundary for pen movement. These are device-dependent boundaries. When the pen plotter senses the media size, it automatically sets the hard-clip limits inside the media edge. Pen plotters usually have a margin at the front of the plotter that is larger than on the other three sides of the media. The following illustration shows how the hard-clip limits locate the plotting area on pen plotter media in portrait and landscape orientations. Actual margin sizes are device-dependent.



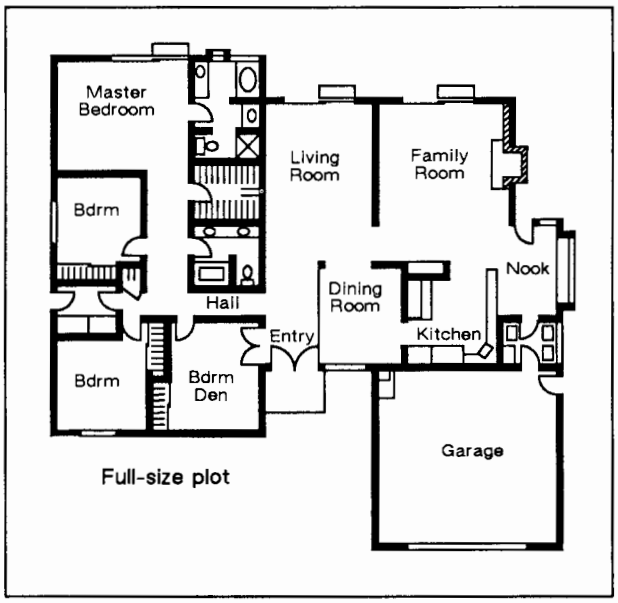
If you are using a device with HP-GL/2 capability, your printing area may be located on the page differently. Refer to your device's documentation for page size information.

Soft-Clip Limits

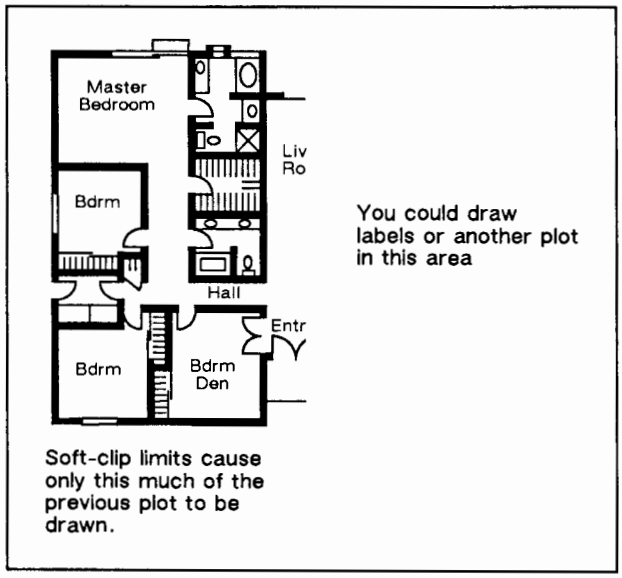
Soft-clip limits temporarily restrict pen movement to a specified area of the page. Because the soft-clip limits let you draw attention to a particular set of data, they are often called *windows*. Usually, you will use soft-clip limits to ensure that nothing will be drawn beyond that portion of the page.

Using soft-clip limits gives flexibility with respect to where you plot data and how much of your data is plotted. For more information on windowing, refer to chapter 4, *The Configuration and Status Group*.

For example, look at the following floor plan.



After plotting the full plan, suppose you decide to draw a new plot showing only the sleeping quarters, using the space on the right for text. To create the new plot, just add soft-clip limits to temporarily restrict plotting to the left half. Then, add program lines to extend the soft-clip limits and plot the text. Refer to the next illustration.

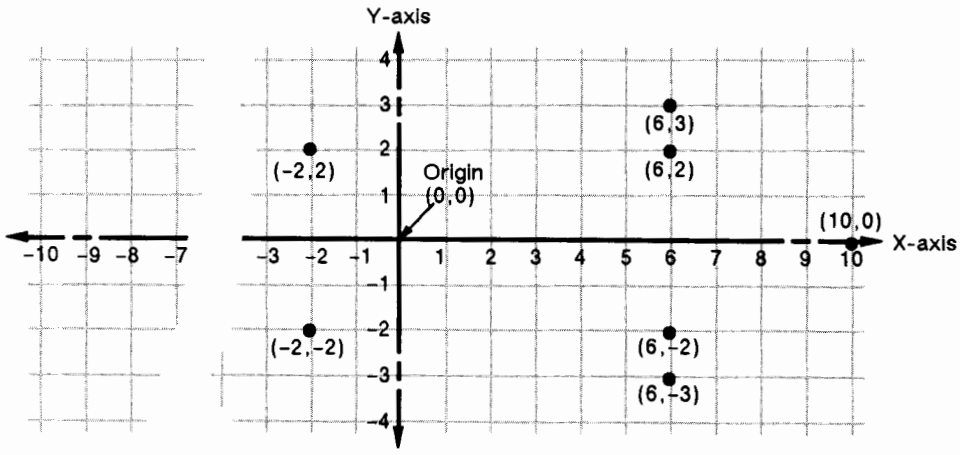


Plotting with the Coordinate System

The plotter uses the Cartesian coordinate system: a set of two perpendicular, scaled axes. These axes are usually called the X- and Y-axes. You can locate any point on the XY plane by specifying its X- and Y-coordinates. You can visualize the system as a grid. By convention the X-axis (abscissa) is parallel to the long side of the paper and the Y-axis (ordinate) is parallel to the short side.

Refer to the illustration below. The point where the axes intersect is the *origin* of the system. All points on each axis are measured in positive and negative values from the origin. Positive X values are plotted to the right of the origin; positive Y values are plotted above the origin.

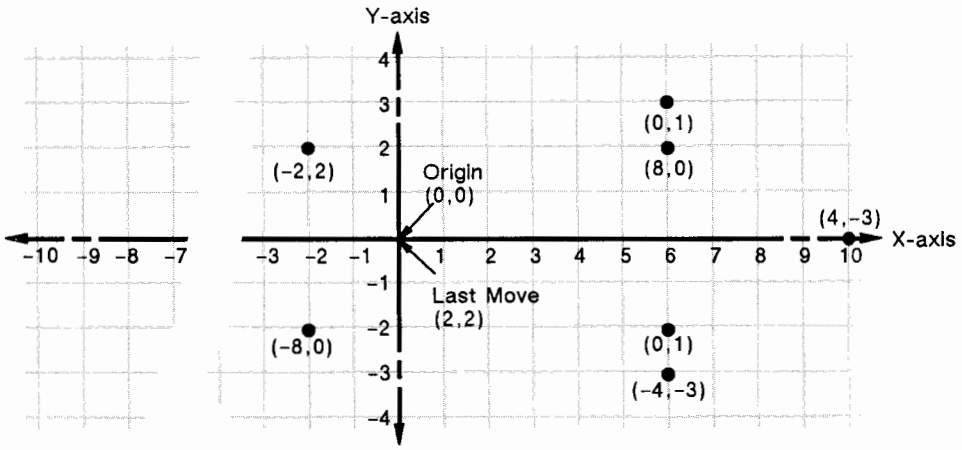
To locate any point in the plotting area, move from the origin so many units along the X-axis, then so many units along the Y-axis to your point. Each point, then, is referenced by the combination of its X-coordinate and Y-coordinate. Together, they are known as an X,Y coordinate pair. The coordinate pair of the origin is (0,0). To specify a location, you must always specify a complete X,Y coordinate pair; first the X-coordinate, then the Y-coordinate. This manual shows the coordinate pair in parentheses (X,Y) for clarity. Do not use the parentheses when programming.



Look at the illustration again to locate these points: (0,0); (-2,2); (6,2); (6,3); (10,0); (6,-3); (6,-2); (-2,-2); (0,0). Now draw a straight line between each point in the order listed. (You should have drawn an arrow.) This is how you define a picture on a two-dimensional coordinate system.

Moving within the Coordinate System

You can specify locations for your pen in one of two ways: absolute or relative coordinates. The previous section describes coordinates as absolute locations with respect to the origin. Relative coordinates are located *relative* to the last pen location. To draw the same arrow as described in the previous section, you would use the following relative coordinate values (in this order): (0,0); (–2,2); (8,0); (0,1); (4,–3); (–4,–3); (0,1); (–8,0); (2,2).



Note that the first two coordinate pairs are the same for absolute and relative locations. The third coordinate pair above (8,0) differs from its absolute counterpart (6,2) in the previous section. All absolute coordinate pairs specify a fixed location on the page with respect to the origin (0,0). Therefore, the location (6,2) will always be in the same location on the page. Relative coordinates, however, only specify a location relative to the last specified location. In the example above, (8,0) specifies a location that is 8 positive X-units and 0 Y-units from the previous location, not from the origin. Note too that the coordinate pair (0,1) appears twice in the example above, each one moving the pen to a different location on the page.

Units of Measure and Scaling

You can express coordinates as one of two types of units:

- plotter units
- user units

A plotter unit is a fixed size. However, when defining and drawing your plot, another unit of measure may make more sense. You can define a unit of measure called a user unit. The plotter always moves in plotter units. The plotter internally converts user units to plotter units before locating the coordinates on paper.

Plotter Units

A plotter unit is the smallest move the plotter can make (this is also known as the plotter's addressable resolution). Note the following measurements involving plotter units.

1 plotter unit	=	0.025 mm or 0.000 98 in.
40 plotter units	=	1 mm
1016 plotter units	=	1 in.

Because plotter units are the smallest move the plotter can make, it interprets all X,Y coordinates for plotter units as integers (whole numbers). Refer to your plotter's user manual to determine the numeric range for plotter units. For practical purposes, though, the effective range of plotter units is limited to the size of the paper you are using.

User Units

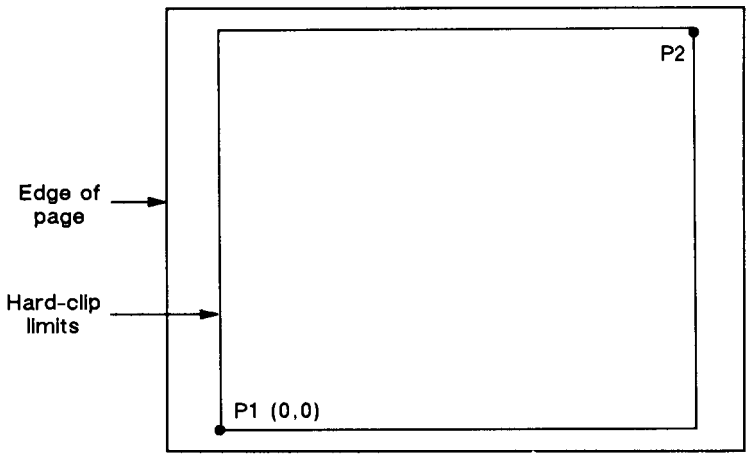
User units let you customize your coordinate system to your particular needs. For example, you might want to make a plot showing maximum temperatures along the Y-axis and the days of the month along the X-axis. You can let user units represent distances, elevations, units of time, temperatures, or whatever measurement meets your needs.

User units are flexible. You can control the number of units along each axis and assign the origin (0,0) to any location on the plotting area. You can also specify coordinates with fractions, making it easier to accurately represent your data. You define user units through a process known as *scaling*.

Scaling

Scaling establishes user units and relies on the relationship between two points: P1 and P2. These two points are called scaling points because they take on the user-unit values that you specify with the Scale instruction (SC).

P1 and P2 define opposite corners of a rectangular plotting area. The default locations for P1 and P2 are at the hard-clip limits. P1 is also the default location of the origin of the coordinate system. When you turn on or initialize your plotter, P1 has the coordinate value (0,0).



Default Orientation of P1 and P2

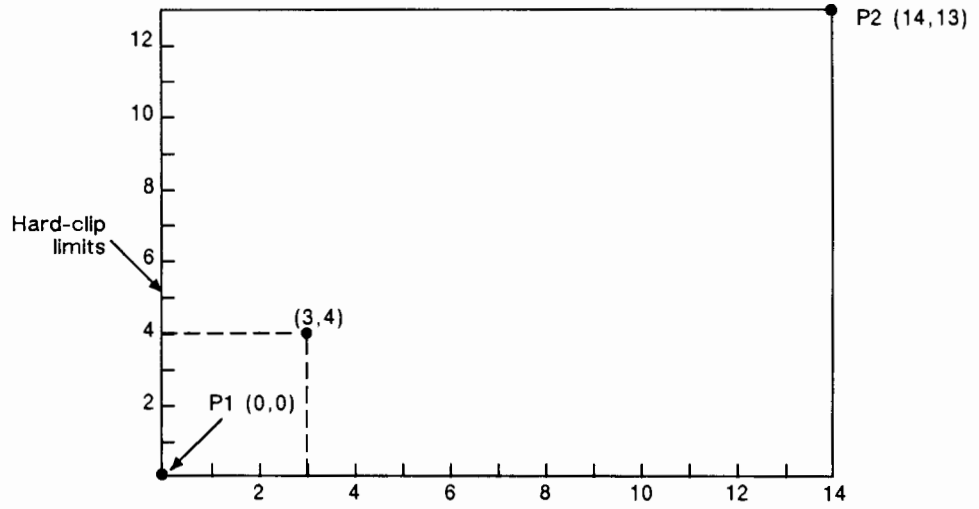
You can make this rectangle any size and place it anywhere depending on the plotter-unit coordinates you specify for P1 and P2. Note that you can place the P1/P2 rectangle either partially or entirely off the media; plotting, however, is restricted to the hard-clip limits. You set P1 and P2 using the Input P1 and P2 (IP) or Input Relative P1 and P2 (IR) instruction.

Using Scaling

When you scale a plot, you define your own units of measurement which the plotter internally converts to plotter units. In effect, you are assigning a scale (or ratio) of plotter units to user units. This is similar to scaling on maps where 1 inch (a user unit) represents a specific number of miles. Scaling lets you plot in units that make sense to you and are easy for you to work with. For example, since 400 plotter units equal 1 centimeter, you can set up your user-unit coordinate grid so that you can design your plot to a centimeter scale.

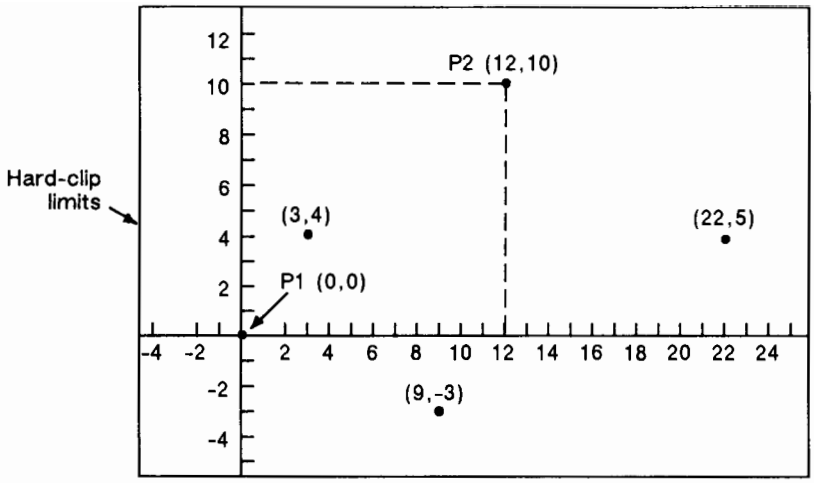
With the Scale (SC) instruction, you specify the number of user units you want along each axis. The number of sections defines the size of your user units.

The following example will help you to understand the Scale instruction. To divide the X-axis into 12 units representing months, and the Y-axis into 10 units representing thousands of dollars, specify the X-axis to scale from 0 to 12, and the Y-axis to scale from 0 to 10. P1 retains the location value of (0,0); this is its user-unit value. P2 becomes (12,10). The entire plotting area is now divided into the desired units. Subsequent plotting instructions will use these units. If you tell the plotter to move to the point (3,4), the plotter will move to the location equivalent to (3,4) user units, *not* (3,4) plotter units.



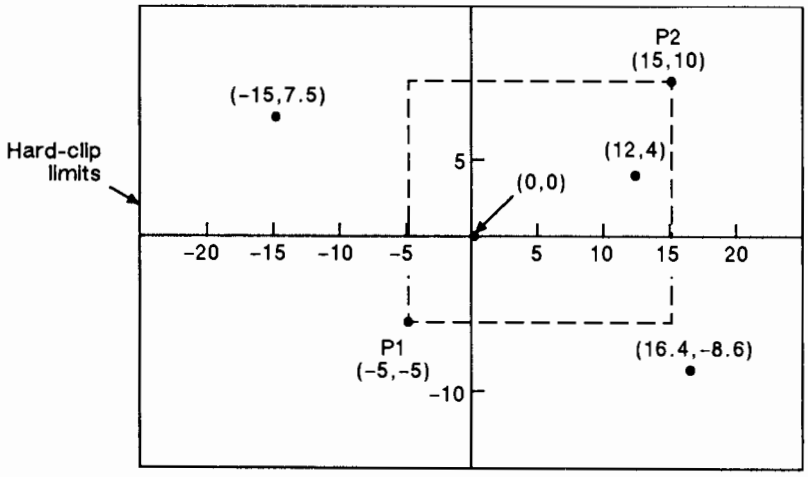
User-Unit Scaling with Default P1 and P2

If you move the locations of P1 and P2, the size of the user units will change. The previous illustration showed P1 and P2 in their default locations (the lower-left and upper-right corners, respectively, of the hard-clip limits). In the following example, P1 and P2 have the same user-unit *values* (set with SC), but their physical *locations* have been changed (using IP). Note that the size of the user units decreased.



Same User-Unit Scaling with New P1 and P2

The framework set by the scaling points P1 and P2 is *not* a graphics limit. The user-unit coordinate system extends across the entire plotting area. You can plot to a point beyond P1 or P2 as long as you are within the hard-clip limits, as shown in the following illustration.

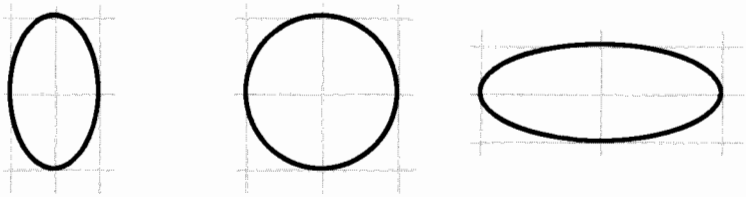


New P1 and P2 User-Unit Scaling with Negative Values

Isotropic and Anisotropic Scaling

The Scale instruction also lets you specify whether your units will be of equal size on the X- and Y-axes (*isotropic* scaling) or unequal (*anisotropic* scaling).

To preserve the proportions of shapes such as circles, wedges, and arcs, you must first set up isotropic scaling. In isotropic scaling, the spacing of units along the X-axis is the same as unit spacing along the Y-axis. In this way, shapes such as circles and wedges are drawn without distortion. When the scaling is anisotropic, a circle becomes an ellipse. Compare the shapes in the following illustration. Note that the specified radius for each of the shapes is the same (1 user unit); only the scaling changes.



Anisotropic
1 Y-unit > 1 X-unit

Isotropic
1 X-unit = 1 Y-unit

Anisotropic
1 X-unit > 1 Y-unit

For more information on scaling, refer to chapter 4, *The Configuration and Status Group*.

3

Writing Efficient Programs

Before you can write graphics programs for your HP-GL/2 peripheral, you must be familiar with your computer and a programming language the computer understands. Use any ASCII-based computer and language that outputs literal strings and uses input and output statements to transfer information to and from a peripheral device. This manual assumes you already have experience in the programming language you select.

This chapter discusses the following fundamental programming concepts.

- HP-GL/2 and your programming language.
- HP-GL/2 syntax.
- Writing an efficient program.

If you are an inexperienced HP-GL/2 programmer, use the following guidelines to ensure success.

- Know your equipment; this includes the computer and HP-GL/2 plotter. Know how to write, edit, save, and run a program. Determine how your computer sends (outputs) information to and reads (inputs) information from peripheral devices.
- Know your programming language. All of the programming examples in this manual are written in BASIC. But even if you aren't programming in BASIC, get a BASIC book and look up unfamiliar statements. Their definitions will help you find and use the equivalent statement in your language. Always use the syntax your language requires.
- Enter HP-GL/2 instructions exactly as they appear in the text. Every letter, symbol, and number is significant. Common mistakes are substituting the letter "*O*" for the number zero (Ø), or the lowercase letter "*l*" for the number one (1). Also, substituting a semicolon for a comma or omitting a quotation mark can make the difference between failure and success.

Using HP-GL/2 with Programming Languages

How you send an instruction to your plotter depends on the programming language you are using. Some languages include some graphics statements, but these are primarily for the computer's monitor. Three common languages used for plotting are BASIC, FORTRAN, and Pascal.

BASIC (Beginner's All-purpose Symbolic Instruction Code) is one of the most common languages used for plotting. It uses statements resembling English to perform many complex operations. Note that graphics statements included in your version of BASIC are probably designed for your computer monitor, not for your plotter.

FORTRAN (FORmula TRANslator) is a problem-oriented language. Programmers can think in terms of the problem and write programs as algebraic expressions and arithmetic statements.

Pascal is a block-structured language requiring structured programming, but still allowing many operator and control statements.

Follow these steps to send an instruction in your language.

1. Use whatever opening statements are required to define the computer's output port and establish the plotter as the recipient of an output string. These are called *configuration statements* and usually must be the first statements in your program. Configuration statements are explained in detail in the next section.

2. Send the string to the plotter using an output statement. Output statements are explained after configuration statements.

If you are programming in a language other than BASIC, FORTRAN, or Pascal, you will have to identify input, output, and program statements that perform the same function as the BASIC statements used in this manual. Refer to your computer documentation for information about sending literal strings to a peripheral (in this case, literal strings of HP-GL/2 to your device).

NOTE: Before sending any HP-GL/2 instructions to your plotter, you must be certain that your computer and plotter are communicating properly. Refer to your peripheral's documentation for interconnection instructions.■

The Configuration Statement

The configuration statement directs computer input and/or output to the plotter. It is specific to your computer language and the type of interface (Centronics, HP-IB, or RS-232-C) you are using. You must use a configuration statement at the beginning of every program. The following is an example of a configuration statement.

```
10  OPEN "COM1:9600,N,8,1,RS,CS65535,DS,CD" AS #1
```

The BASIC configuration statement shown above is for the HP Vectra personal computer (also other IBM compatibles) RS-232-C interface. It **OPEN**s **COM1** (port 1) for output at **9600** baud, No parity, **8** data bits, **1** stop bit, suppresses RTS (**RS**), waits 65 535 milliseconds before returning a device timeout error (**CS65535**), ignores the DSR and Carrier Detect lines (**DS,CD**), and calls the file **#1**.

Your configuration statement will vary according to the programming language and the hardware configuration of your computer. A table of sample configuration statements follows. Your peripheral's computer setup information may have additional examples.

Computer	Language	Example

HP 3000
DEC VAX
	FORTRAN
	(RS-232-C)
		`WRITE (9,14)`
		`10  FORMAT ("instruction")`
		
		(The WRITE statement also functions as a configuration statement. Unless you specify otherwise, the instruction is automatically directed to the standard output device, which is typically your terminal.)

HP 9000, Series 300	HP BASIC (HP-IB)	`OUTPUT 705 ; ("instruction")` (In this case, the output statement also functions as a configuration statement, directing output to the plotter at select code 7, HP-IB address 5.)
	Pascal (HP-IB)	`WRITELN ('instruction')` (The WRITELN statement also functions as a configuration statement. Unless you specify otherwise, the instruction is automatically directed to a file called "output.")

Output Statements

After the configuration statement, use output statements to send instructions to the plotter as a string. Output statements vary with the computer and language. To give you an idea of the variety of statements available, the following table lists a few computers and languages with their associated output statements. The examples use the HP-GL/2 instruction (*SP1*) as the instruction being sent.

Computer	Language	Example
HP Vectra PC IBM PC/PC-XT/AT	GW BASIC	`PRINT #1, "SP1"`
HP 3000 DEC VAX	FORTRAN	`WRITE (6,1Ø)` `1Ø FORMAT (X,3HSP1)`
HP 9000, Series 300	HP BASIC	`OUTPUT 7Ø5; "SP1"`
	Pascal	`WRITELN ('SP1')`

Input Statements

To get information *from* your HP-GL/2 device, you must first send an HP-GL/2 output statement to the plotter directing it to send information back to the computer. Your device must support the Technical Graphics Extension to use output instructions. After sending the output instruction, use a computer language input statement to read the output from your peripheral. The following table lists a few computers and languages with their associated input statements. The input statements are reading a four-digit response from an HP-GL/2 output instruction, such as the Output P1 and P2 (OP) instruction.

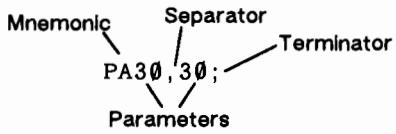
Computer	Language	Example
HP Vectra PC IBM PC/PC-XT/AT	GW BASIC	`INPUT #1, A,B,C,D`
HP 3000 DEC VAX	FORTRAN	`READ (5,4Ø)` `4Ø FORMAT (417)`
HP 9000, Series 300	HP BASIC	`OUTPUT 7Ø5;A,B,C,D`
	Pascal	`READ (A,B,C,D);`

Understanding HP-GL/2 Syntax

HP-GL/2 instructions have four components: a mnemonic, parameter(s), separator(s), and a terminator.

- **Mnemonic** — The two-letter mnemonic is designed to remind you of the instruction's function. The mnemonic can be upper- or lowercase.
- **Parameter(s)** — Some parameters are required, some optional. Optional parameters, when specified, may require additional, qualifying parameters. Some instructions have no parameters.
- **Separator(s)** — When you use parameters, you must separate them with a comma or space, and/or with a + or - sign. (Commas are recommended because some computers eliminate spaces, especially when sending variables.)
- **Terminator** — All instructions require a terminator. The recommended method is to terminate an instruction with a subsequent HP-GL/2 mnemonic. You may use semicolons to terminate all HP-GL/2 instructions. (Dropping the semicolon is recommended because it enhances throughput by approximately 10%.) Some instructions (PG, PE, and all output instructions) *must* be terminated with a semicolon.

The following illustration describes a typical HP-GL/2 instruction. Note that it uses a semicolon to show that the instruction is being terminated. The recommended practice is to terminate this instruction with the next HP-GL/2 mnemonic.



The following illustration shows the flexibility of the syntax. Each variation of the two-instruction sequence is permissible; however, the first method is recommended. The recommended method uses the first letter of the next mnemonic to terminate instructions, uses no space between the mnemonic and its parameters, and separates parameters with a comma. This method sends fewer bytes of information to the HP-GL/2 device, significantly reducing transmission time.

```
PDPU10,20          PD;PU10,20;          PD PU 10 20;
```

Recommended

All the program examples in this manual use the recommended method. The next section explains how the syntax of individual instructions is presented in this manual.

Notations Used to Express Syntax

The following describes the notations used in the syntax section of the instruction descriptions at the end of each chapter.

MNemonic	Always two uppercase characters.
parameters	All parameters are shown in italics.
( )	Parameters in parentheses are optional.
label	Any number of labeling characters.
(,...)	Any number of the previous parameter (you must have an even number of X,Y coordinates).
;	Instruction terminator. A semicolon is often optional and is shown in parentheses in most instruction syntax. Exceptions are the Polyline Encoded (PE) and output instructions, both of which *must* be terminated by a semicolon. We recommend you place a semicolon after the last HP-GL/2 instruction in your program to terminate the instruction and ensure the proper completion of your program.
[TERM]	The terminator sent back to your computer by the plotter at the end of the response to an output instruction. The default output terminators are: a carriage return (**CR**) (RS-232-C interfaces) and carriage return and line feed (**CR LF**) (HP-IB interfaces).

NOTE: Remember that while X,Y coordinates are shown in parentheses in text—e.g., (3,4) or (0,0)—the parentheses are not part of the syntax. Do not enter these parentheses in your instructions.■

Omitting Optional Parameters

Some instructions have optional parameters that take on default values if they are omitted. When you omit a parameter, you must omit *all subsequent parameters in the same instruction*; however, the Define Label Terminator (DT) instruction is an exception.

For example, the Line Type (LT) instruction has three optional parameters: type, pattern length, and mode. The following shows all three being used.

```
LT6,25,1
```

If you change line types, and do not specify the second and third parameters, the plotter uses the default values for the second and third parameters. When you omit the second parameter, you must also omit the third parameter, as shown next.

```
LT5
```

Note that the following interprets *1* as the second parameter, not the third.

```
LT6,1
```

Parameter Formats

You must give parameters in the format (type of units) required by each HP-GL/2 instruction. The required format is stated in the parameter table of each instruction's description, and described as follows. Note that numeric ranges are *minimums*; your peripheral may support a larger range.

- **Integer —** An integer within the range -8 388 608 (-2^{23}) to 8 388 607 ($2^{23}-1$).* The plotter automatically rounds fractional parameters to the nearest integer. Using a number outside this range causes an error.

  Plotter units are always integer.

- **Clamped Integer —** An integer within the range -32 767 (-2^{15}) to 32 767 ($2^{15}-1$). The plotter automatically rounds fractional parameters to the nearest integer. Sending a number outside this range does not cause an error, but the number is "clamped" to the limits of the range.

  Certain instructions have parameters which are restricted to a smaller range. These ranges are listed in the parameter tables for each instruction. Sending a number outside the reduced parameter range causes an error.

* The range extends beyond the hard-clip limits. Coordinate data located beyond the hard-clip limits is not plotted.

- **Real** — A number in which the integer portion is within the range -8 388 608 (-2^{23}) to 8 388 607 ($2^{23}-1$),* and the optional decimal fraction has a maximum of 10 significant digits (additional digits are ignored). You may omit the decimal point when no decimal fraction is specified. If you don't specify a sign, the parameter is assumed to be positive. Using a number outside this range causes an error.

  User units are real.

- **Clamped Real** — A number in which the integer portion is within the range -32 767.9999 (-2^{15}) to 32 767.9999 ($2^{15}-1$), and the optional decimal fraction has a maximum of 10 significant digits (additional digits are ignored). You may omit the decimal point when no decimal fraction is specified. If you don't specify a sign, the parameter is assumed to be positive. Sending a number outside this range does not cause an error, but the number is "clamped" to the limits of the range.

  Certain instructions have parameters which are restricted to a smaller range. These ranges are listed in the parameter tables for each instruction. Sending a number outside the reduced parameter range causes an error.

- **Character** — Any character or sequence of characters.

- **Newstring** — Any character or sequence of characters with quotes; used primarily with the MG (Message) and BP (Begin Picture) instructions.

When you see the term *current units* in a parameter table, the format of that parameter depends on whether scaling is on or off. When scaling is *on*, the format is interpreted as real (user units); when scaling is *off*, the format is interpreted as integer (plotter units).

NOTE: The plotter does not understand exponential format (e.g., 6.03E8). If you are using a computer or language that uses exponential format, you must use integer variables or a formatting technique to output fixed-point real numbers.■

* The range extends beyond the hard-clip limits. Coordinate data located beyond the hard-clip limits is not plotted.

Using the Program Examples

Examples are presented as complete programs written in a version of GW-BASIC for the MS-DOS®* 2.11 (or higher) operating system. This version of BASIC is used by many popular personal computers.

If you are using GW-BASIC,®** replace line 10 with the proper configuration statement for your computer; then enter and run the program examples as shown.

If you are not using GW-BASIC, you may need to insert the proper configuration statement at the beginning of the program and change some of the program statements.

Below is an example of the way programs appear in this manual.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS7000,5000SP1"
30  PRINT #1, "PA2000,3000PD2000,4000,PU2500,4000"
40  PRINT #1, "PD2000,3500,2500,3000PU2250,3500"
50  PRINT #1, "PD2250,2500,2750,2500PU2375,3250"
60  PRINT #1, "CI1000,30"
70  PRINT #1, "PUSP0PG;"
80  END
```

Outline of an Efficient HP-GL/2 Program

The following is a generic outline for your HP-GL/2 program.

- Initialize the plotter.
- Define a palette.
- Use compacted data.
- Close the program.

The following sections describe each of these in more detail.

* MS-DOS is a U.S. registered trademark of Microsoft Corporation.
** GW-BASIC is a U.S. registered trademark of Microsoft Corportation.

Initialize the Plotter

The following HP-GL/2 instructions are necessary to initialize the plotter and prepare for the plot data. Parameters are not included. *Use the instructions in the order in which they are listed.*

Instruction	Function
ESC%-1B	Some HP-GL/2 peripherals understand both HP-GL/2 and PCL. This instruction ensures that the plotter is in HP-GL/2 mode.
BP	**Begin Plot:** This instruction tells the plotter that a new picture is coming. BP is necessary to ensure that HP-GL/2 plots are written to a clean page. It also ensures that plotters with HP-GL/2 and HP-GL emulation are switched to the HP-GL/2 mode. Refer to chapter 9, *The Technical Graphics Extension*.
IN	**Initialize Plotter:** This instruction initializes the plotter to specific defaults. IN1 overrides the front panel settings with the factory defaults and could affect how a user has set front panel features. We do not recommend using IN1 unless your application requires you to do so. Refer to chapter 4, *The Configuration and Status Group*.
PS	**Plot Size:** This instruction allows you to specify the plot size, including long-axis sizes. It is very important for electrostatic plotters as the media will then advance only the size of the page specified when a page advance instruction is received. This saves considerably on expensive media. This instruction is also used to specify large plot sizes so you only need to send data once to the plotter to do long-axis plotting. Refer to chapter 9, *The Technical Graphics Extension*.

Many of the program examples in this manual begin as follows (some examples use different values for the PS instruction).

```
CHR$(27)+"%-1BBPINPS7000,5000"
```

Additional instructions are used to define plot quality. Although some devices allow the user to set the quality level using a plotter's control panel, it might be better to provide the capability through software so that the plotter can be used in a networked environment by users with differing needs. These instructions include MT (Media Type), QL (Quality Level), and VS (Velocity Control). Refer to chapter 9, *The Technical Graphics Extension*, for more information.

Define a Palette

To extend your program to raster devices as well as pen plotters, predefine a palette of pen widths and pen colors before beginning the data stream. Once defined, pen colors or widths should not be updated. This is necessary because pen plotters need to receive a different pen number for each different pen color or width desired.

As usual, it's up to the user to put a physical pen of the desired color and width in the correct pen plotter carousel position.

Also, in order to support the full range of HP-GL/2 monochrome and color plotters, programs need to allow users to choose the desired color and width for each pen number used in a drawing.

The instructions used to create the palette are listed below.

Instruction	Function
NP	**Number of Pens:** Defines the size of the HP-GL/2 pen palette. Pen attributes (color, width) are defined using the instructions described below. The number of pens defined must be a power of two, and the pens are numbered beginning with zero, not one. Refer to chapter 10, *The Palette Extension*. *Pen plotters*: The number of logical pens is fixed at 32. However, the plotter performs a modulo function on any pen number over 8 until it is able to select a physical pen from the carousel. *Monochrome electrostatic plotters*: The default number of logical pens is 2 (black and white).
PC	**Pen Color:** Defines a color for a specific pen. Black-and-white devices default to 2 pens, 0 and 1 (white and black); color devices default to 8, 0 through 7 (white, black, red, green, yellow, blue, magenta, and cyan, in that order). This instruction works with the NP (Number of Pens) instruction to let you create a palette of pen colors. Refer to chapter 10, *The Palette Extension*. *Pen plotters*: This instruction is not recognized. Pens will be selected based on the Select Pen (SP) instruction only. *Monochrome electrostatic plotters*: Monochrome plotters will print lines of varying grayscale depending on the color used to draw lines. Drawing screened vectors or raster fills using a nonblack pen will produce unpredictable grayscaled results due to this interaction.
PW	**Pen Width:** Defines the width of a specific pen. To provide optimum throughput, pen widths should be assigned to individual pens and accessed by selecting a pen (instead of just selecting a width for subsequent vectors). Refer to chapter 7, *The Line and Fill Attributes Group*. *Pen plotters:* Pen widths can be assigned to any of the 32 logical pens, but the widths will be stroked out using the 8 physical pens in the carousel. Because stroking is slow, PW will most likely be disabled via the plotter's front panel. The plotter will then draw vectors using the physical pen selected, regardless of the specified width.

For example, a typical palette definition for a three-pen plot might resemble the following.

```
"NP8PC1,255,0,0PC2,0,255,0PC3,0,0,255PW.3,1PW.25,2PW.7,3"
```

This indicates a palette of 8 pens (NP8), establishes color definitions for pens 1, 2, and 3 using the PC instruction (once for each pen with specific red, green, blue values), and sets the pen width for each pen using the PW instruction (e.g., pen 1 has a width of 0.3 mm).

Use Compacted Graphics

The key factor in this part of your program is maximizing throughput. Using a high-speed interface such as Centronics or HP-IB is just one part of this. Efficient use of HP-GL/2 and compacting vector data is also a significant, and just as important, part.

Instruction	Function
PE	**Polyline Encoded:** We cannot stress enough the importance of using the PE instruction to send vectors to the plotter as well as the use of a high-speed interface. Use of PE will significantly reduce the amount of data necessary to create a plot—in many cases by 60–70%.

Refer to chapters 5 through 8 for additional instructions you can use in the data stream.

Close the Program

The following instructions should be sent *in this order* at the end of the plot. These instructions put the pen away (pen plotters) and begin rasterization on raster plotters or long-axis plotting on pen plotters (if the plot fits within the current page on a pen plotter, it will be plotted as the instructions are received).

Instruction	Function
PUSPØ	**Pen Up, Select Pen Ø:** These instructions lift the pen on pen plotters and put it away. On electrostatic plotters, these instructions lift the pen from the media and change the color to that of pen 0.
PG;	**Page Advance:** If the plotter needs to buffer the entire image before plotting, the PG instruction tells the plotter all the vectors have been received and plotting can begin. When the plot is complete, the page advances to the next page. Unless this instruction is followed by a Replot (RP) instruction, the semicolon is required. After plotting, the page is ejected from the plotter.
RPn;	**Replot:** This instruction creates additional copies from the information in the device's buffer/disk. This always follows the PG instruction—there cannot be any other instructions between the PG and RP instructions.

All of the programming examples in this manual end with the following sequence.

```
PUSPØPG;
```

The following makes one additional copy of the plot.

```
PUSPØPGRP;
```

Increasing Your Plotter's Throughput

The *throughput* of your plotter refers to the time it takes to complete a plot. To take full advantage of your peripheral's speed, use a parallel interface, such as Centronics or HP-IB. For the best throughput with an RS-232-C interface, use the highest baud rate the device supports.

Here are some ways you can further enhance your throughput by reducing the amount of data transmitted over the interface.

- **Use the Polyline Encoded (PE) instruction.** HP-GL/2 uses base 10 (decimal) for vector coordinates, but the PE instruction uses base 64 or 32 to encode vector coordinates. This vector encoding sends approximately one-quarter the amount of data bytes, which takes approximately one-quarter the amount of transmission time. This reduction in transmission time can significantly improve your total throughput.

- **Eliminate terminators (semicolons) between instructions in your program.** Eliminating terminators in your programs can increase your plotter's throughput by approximately 10%. (Note that PE, PG, and output instructions *must* be followed by a semicolon.)

- **Take advantage of the plotter's graphics instructions in your plots.** For example, use the plotter's rectangle instruction, which allows you to draw a rectangle by sending only one coordinate pair instead of four.

- **Use the plotter's internal character set,** particularly for quick-check plots. Internal sets require only one byte per character.

Program Errors and Lost Mode

If your plot is not drawn as you expected, you probably have an error in your program. When you suspect you have an error *and* your device supports the Technical Graphics Extension, send the Output Error (OE) instruction to read the error. The plotter will send back a number representing the plotter's error status.

Note that you cannot use output instructions on a Centronics interface.

Program Errors

The following table lists each error number, its meaning, and the probable cause of the error. Note that the plotter may or may not complete your plot correctly, depending on the severity of the error. Error number 0 indicates that no error has occured.

Error Number	Meaning	Probable Cause
1	Unrecognized instruction (or not recognized in current state).	A mnemonic is incorrect or missing; an alphabetic character was specified in a parameter when a numeric character was expected.
2	Wrong number of parameters.	Too few or too many parameters; an incomplete X,Y coordinate pair.
3	Out-of-range or invalid parameter.	A parameter is beyond the allowed range.
6	Position overflow.	A single label is so long that it exceeds the numeric range, or the number of relative moves is so great that it caused an overflow.
7	Buffer overflow/out of memory.	There are too many vectors specified for a polygon or downloadable character.

For the following error conditions, the plotter will ignore the HP-GL/2 instruction.

- Unrecognized instruction.
- Instruction missing required parameters.
- Instruction with an out-of-range parameter.

The plotter partially ignores an instruction in the following circumstances.

- The instruction is sent with more parameters than necessary. (The plotter ignores the extra parameters and executes the instruction as normal.)
- A PA, PD, PR, or PU instruction has a parameter exceeding its numeric range. (The plotter ignores the out-of-range parameter and all subsequent parameters for that instruction.)
- An AD, LA, or SD instruction has a parameter exceeding its format range (e.g., integer or real). (The plotter ignores the out-of-range parameter and all subsequent parameters for that instruction. However, if the parameter is within the range of its format type, but exceeds the allowable range for the specific parameter, the plotter ignores the parameter and executes all subsequent parameters.)

Lost Mode

Though you may specify a parameter, or series of parameters, within the plotter's allowed range, note that scaled values may be subsequently unscaled by the plotter into resolution units (e.g., 9600 units per inch) that exceed your plotter's internally representable number range. When this occurs, the plotter generates error 6 (position overflow) and the plotter enters "lost" mode. In lost mode, all relative drawing instructions are ignored until the plotter receives an instruction specifying an absolute move to a point within the plotter's number range.

The instructions IN, PG, RP, and PA (with in-range parameters) clear lost mode. When in absolute plotting mode, the PD and PU with in-range parameters will also clear lost mode. When PD clears lost mode, a line is drawn from the last valid current location to the first point in the PD parameter sequence. When PA clears lost mode, the plotter will not draw until it receives a Pen Down instruction.

If lost in polygon mode, only *PM2* clears lost mode, closing the current polygon. *PM1* will not clear lost mode, but will close the current subpolygon. The polygon buffer will contain the points up to, but not including, the point that set lost mode. The buffer will also contain the points *after* lost mode was cleared; however, points stored after lost mode was cleared are defined as pen up until a Pen Down instruction is received.

4

The Configuration and Status Group

The Configuration and Status Group instructions help you achieve the following.

- Establish default conditions and values for programmatical features.
- Scale (especially useful when you want to reproduce a plot onto larger or smaller media).
- Establish a window (soft-clip limits).
- Draw equal-sized and mirror-imaged plots.
- Rotate the coordinate system.

The following instructions are described in this chapter.

Instruction	Summary
DF, Default	Sets most programmable features to their default conditions.
IN, Initialize	Sets all programmable features to their default conditions.
IP, Input P1 and P2	Establishes new or default locations for the scaling points P1 and P2.
IR, Input Relative P1 and P2	Establishes P1 and P2 locations with respect to the hard-clip limits.
IW, Input Window	Sets up a window (soft-clip limits).
PG, Advance Full Page	Terminates the plot*; advances the page.
RO, Rotate Coordinate System	Rotates the coordinate system.
RP, Replot	Plots multiple copies of a stored plot.*
SC, Scale	Establishes a user-unit coordinate system.

* Device-dependent: check the manual for your HP-GL/2 device or option to see if it supports this instruction.

The following is a summary of the parameter formats and their *minimum* ranges. Your device may support a greater range than listed here.

Parameter Format	Minimum Ranges
Integer	2^{23} to $2^{23}-1$ (-8 388 608 to 8 388 607)
Real	2^{23} to $2^{23}-1$ (-8 388 608.000 0 to 8 388 607.999 9)
Clamped Integer	2^{15} to $2^{15}-1$ (-32 768 to 32 767)
Clamped Real	2^{15} to $2^{15}-1$ (-32 768.000 0 to 32 767.999 9)

Establishing Default Conditions

Establish default conditions at the beginning of each program to prevent unexpected results "borrowed" from a previous plot. Many instruction parameters remain in effect until you change them or you reset the plotter to default conditions. Always establish default conditions at the beginning of each program, unless you want to use "leftover" parameters.

There are four ways to establish default conditions:

- Turn the plotter off and on.
- Use the Initialize (IN) instruction.
- Use the Default Values (DF) instruction.
- Use the control-panel reset function, if available on your device.

Using the IN instruction and turning the plotter off and on both restore the device to its factory defaults. This process is called *initialization*. The DF instruction restores user defaults programmed into the control panel (if one is available). DF is not as powerful as IN; the conditions set by each are described later in this chapter.

The Scaling Points P1 and P2

When you scale a plot, you define your own units of measurement. Scaling relies on the relationship between two points: P1 and P2. These two points are called the scaling points because they take on the user-unit values you specify with the Scale (SC) instruction.

P1 and P2 always represent an absolute plotter-unit location. They define opposite corners of a rectangular area. The P1/P2 rectangular area is not a graphics limit. Plotting is not restricted to the P1/P2 area. You can change the size of the rectangular area and move it anywhere depending on the plotter-unit coordinates you specify using the Input P1 and P2 (IP) or Input Relative P1 and P2 (IR) instructions. The biggest benefit of scaling is that your plot will retain the same relative proportions on any size media.*

* This will not always be true when using IP *with* parameters. Use IP *without* parameters, or use IR.

Using the Scaling Instruction

Scaling lets you establish units of measure with which you are familiar, or which are more logical for your plot. The Scale instruction (SC) determines the number of user units along the X- and Y-axes between P1 and P2. The actual size of the units depends on the locations of P1 and P2 and the range of user units set up by the SC instruction.

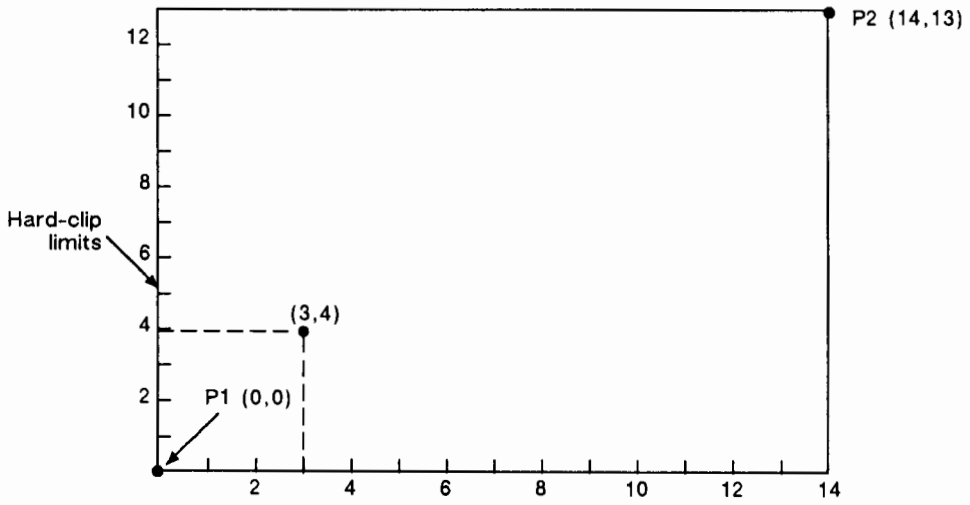
There are three types of scaling:

- Anisotropic
- Isotropic
- Point-factor

Anisotropic scaling indicates that the size of the units along the X-axis is not the same as the size of the units on the Y-axis. Isotropic scaling, then, indicates that the units are the same size on both axes. Point-factor scaling sets up a ratio of plotter units to user units.

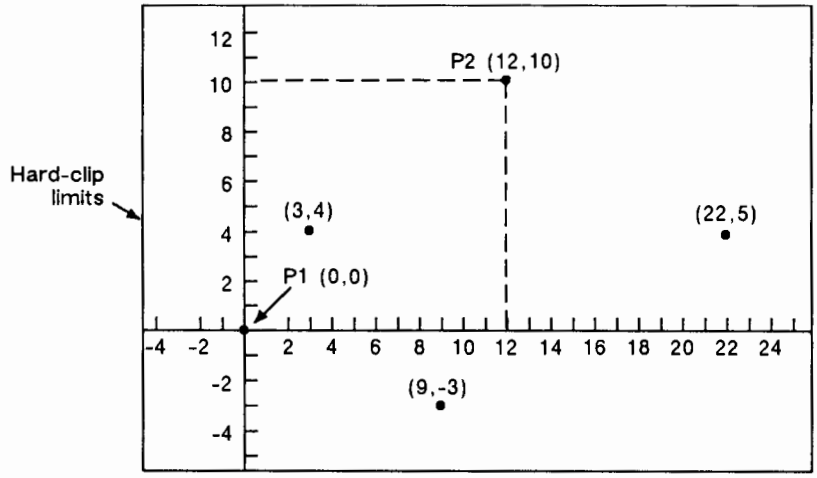
The SC instruction does not change the locations of P1 and P2, only their coordinate values. Also, scaling is not limited to the rectangular area defined by P1 and P2, but extends to the limits of the plotting range.

For example, to divide the X-axis into 12 units representing months, and the Y-axis into 10 units representing thousands of dollars, specify the X-axis to scale from 0 to 12, and the Y-axis to scale from 0 to 10. P1 becomes the origin, with user-unit coordinate (0,0), and P2 becomes (12,10). The entire plotting area is now divided into the desired units. Subsequent plotting instructions will use these units. If you tell the plotter to move to the point (3,4), the plotter will move to the location equivalent to (3,4) user units, *not* (3,4) plotter units.



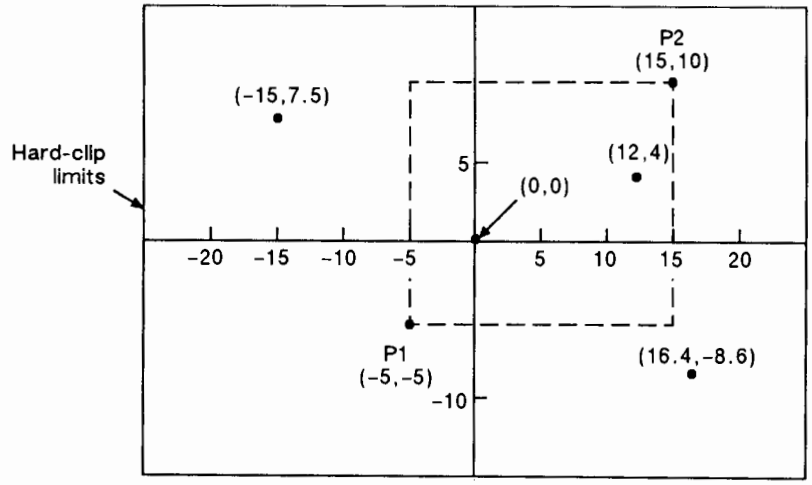
User-Unit Scaling with Default P1 and P2

If you subsequently move the locations of P1 and P2, the size of the user units changes. The previous example showed P1 and P2 in their default locations (the lower-left and upper-right corners, respectively, of the hard-clip limits). In the following example, P1 and P2 have the same user-units *values* (set with SC), but their physical *locations* have been changed (using IP). Note that the size of the user units decreased.



Same User-Unit Scaling with New P1 and P2

To further illustrate the flexibility of user-unit scaling, the next illustration shows the P1 and P2 locations with negative user-unit values.



New P1 and P2; User-Unit Scaling with Negative Values

Remember, the framework set by the scaling points P1 and P2 is *not* a graphics limit. The user-unit coordinate system extends across the entire plotting area. You can plot to a point beyond P1 or P2 as long as you are within the hard-clip limits.

Using Scaling Effectively

The following sections show you how to combine scaling and P1/P2 concepts to do the following.

- Enlarging or reducing the size of a plot.
- Drawing equal-sized pictures on the same page.
- Creating mirror-imaged plots.

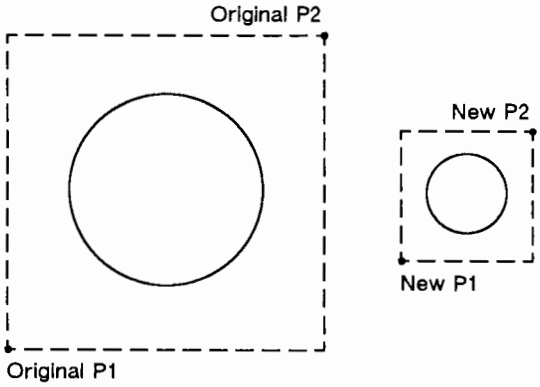
Enlarging or Reducing a Picture

The basic technique for changing a picture's size is to scale the plotting area defined by P1 and P2, then move the locations of P1 and P2 so they define a smaller or larger area. (Only scaled plots are affected by the changes in the P1/P2 locations.) This is especially useful when you want to plot the picture on a specific portion of the page.

To maintain the proportions of scaled plots, set P1 and P2 so they define an area with the same aspect ratio as the original scaling rectangle. For example, if the area defined by P1 and P2 is 3000 × 2000 plotter units in its X- and Y-axes, respectively, its aspect ratio is 3:2. To enlarge the plot, set P1 and P2 to define a larger area that maintains a 3:2 ratio.

The following illustrates this technique using a square (isotropic) P1/P2 scaling rectangle (the ratio is 1:1) with a scale of 0 to 10 in both axes. After a circle is drawn within the scaled area, the locations of P1 and P2 move to form a new rectangular area that maintains the 1:1 ratio. Note that the circle plotted in the new area fits proportionally within the P1/P2 area.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS7000,5000"
30  PRINT #1, "IP0,0,2000,2000SC0,10,0,10SP1"
40  PRINT #1, "PA5,5CI3"
50  PRINT #1, "IP2500,500,3500,1500"
60  PRINT #1, "PA5,5CI3"
70  PRINT #1, "PUSP0PG;"
80  END
```

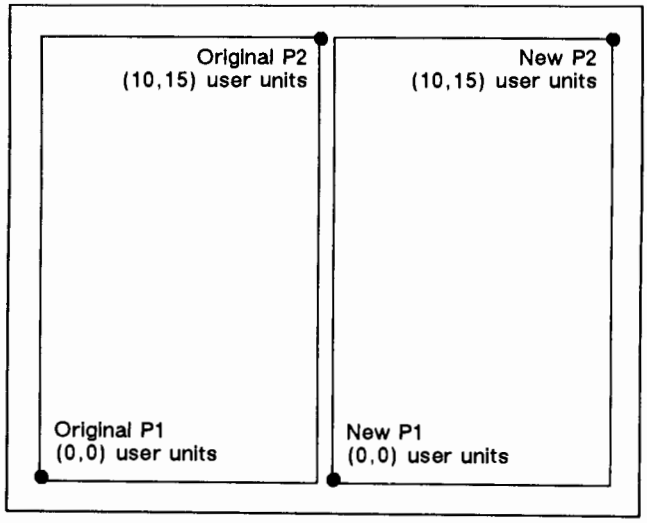


Drawing Equal-Sized Pictures on One Page

You may occasionally want to plot more than one drawing on the same page for a side-by-side comparison. This can be useful for comparing parts, assemblies, layouts, or other similar information. The easiest way to draw equal-sized pictures on one piece of paper is to take advantage of the fact that P2 follows P1 whenever you change the location of P1.

The following example locates P1 and P2 on the left side of the paper and scales the area. After drawing the P1/P2 rectangle, P1 is moved to the right side of the paper. P2, though not specified, tracks P1 so the scaling rectangle retains its same dimensions. The plotted rectangle around the second area shows P2 in its new location.

```
10  'Insert configuration statement here
20  PRINT #1, "CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS8800,7300IP0,0,4350,7300"
40  PRINT #1, "SC0,10,0,15SP1"
50  PRINT #1, "PA0,0PD10,0,10,15,0,15,0,0PU"
60  PRINT #1, "IP4450,0"
70  PRINT #1, "PA0,0PD10,0,10,15,0,15,0,0"
80  PRINT #1, "PUSP0PG;"
90  END
```



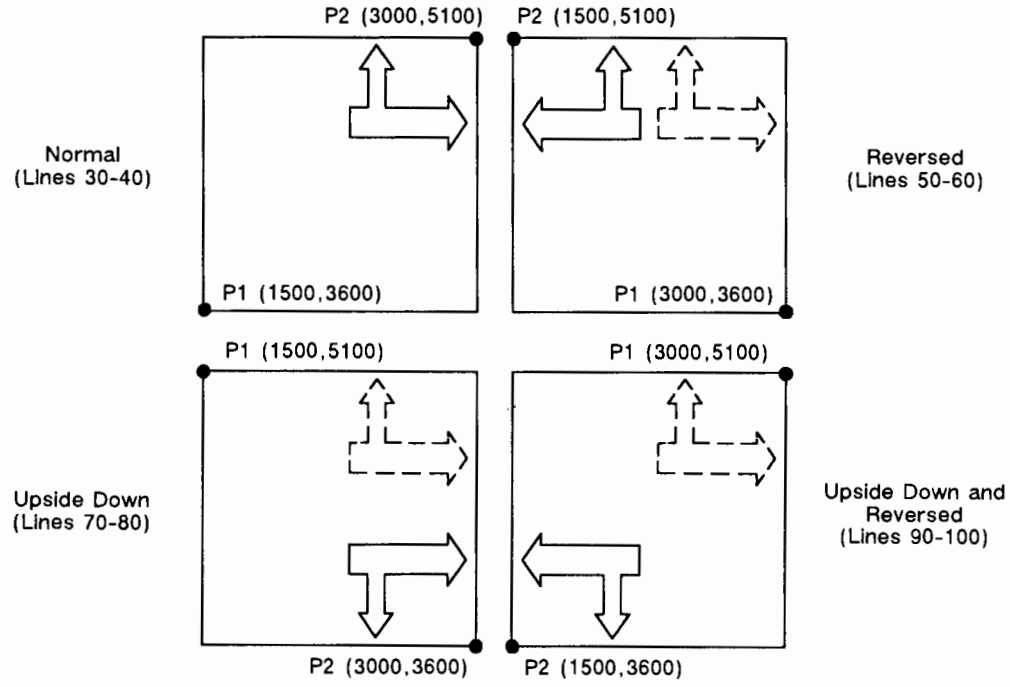
NOTE: These P1/P2 frames are not windows or graphics limits; the pen can plot anywhere within the hard-clip limits. Note that the new P1 and P2 retain their scaled values. This lets you use the same coordinates on both halves of the page. If you do not assign a scale to P1 and P2, you must calculate the new plotter-unit coordinates for the plot on the second half of the page.■

Creating Mirror Images

For most plots, you will probably set P1 in the lower-left corner and P2 in the upper-right corner of the scaling area. However, you can change the relationship of P1 and P2 to produce mirror imaging. You can mirror-image any *scaled* plot by changing the relative locations of P1 and P2.

The following example uses a subroutine to draw the same picture (an arrow) four times. The only change is to the locations of P1 and P2 before each arrow is drawn. This shows all of the possible mirrored orientations. (The original plot is shown in each picture so you can compare the orientation of the mirror image.)

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS7000,5000SP1"
30  PRINT #1, "IP1000,0,2500,1600SC-15,15,-10,10"
40  GOSUB 130
50  PRINT #1, "IP2500,0,1000,1600"
60  GOSUB 130
70  PRINT #1, "IP1000,1600,2500,0"
80  GOSUB 130
90  PRINT #1, "IP2500,1600,1000,0"
100 GOSUB 130
110 PRINT #1, "SP0PG;"
120 END
130 PRINT #1, "PA1,2PD1,4,3,4,3,7,2,7,4,9,6,7,5,7"
140 PRINT #1, "PD5,4,12,4,12,5,14,3,12,1,12,2,1,2PU"
150 RETURN
```

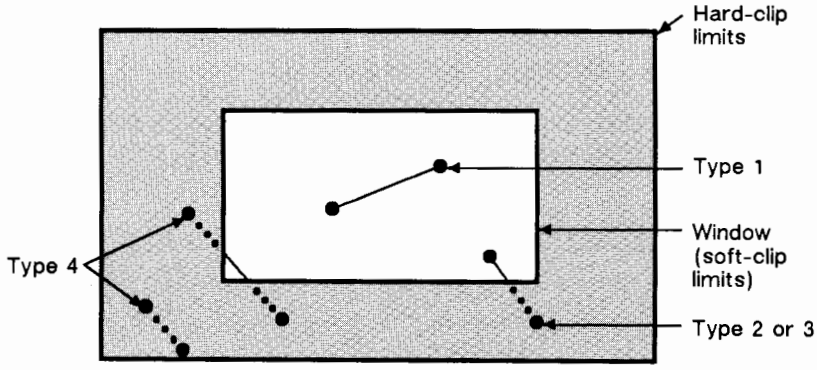


Windowing: Setting Up Soft-Clip Limits

Soft-clip limits temporarily restrict pen movement to a rectangular area, or *window*. When you initialize or set the plotter to default conditions, the soft-clip limits are the same as the hard-clip limits. You set the soft-clip window using the Input Window (IW) instruction. The window has the same effect as the hard-clip limits—the plotter does not draw outside the window.

The following illustration shows the four types of line segments you can specify from one point to another.

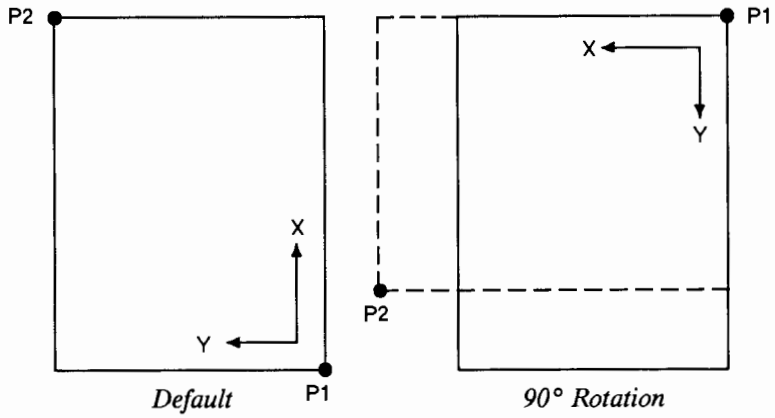
Type	From Last Point	To New Point
1	Inside window area	Inside window area
2	Inside window area	Outside window area
3	Outside window area	Inside window area
4	Outside window area	Outside window area



The IW instruction lets you control the size of the plotting area so that you can draw a particular portion of a plot. You can use the remaining area for labels, or another plot. Refer to *Graphics Limits* in chapter 2 and the IW instruction description later in this chapter.

Rotating a Picture

The plotter always sets the X-axis parallel to the longest edge of your plot. However, you can change this orientation using the Rotate Coordinate System (RO) instruction to rotate the coordinate system counterclockwise 90, 180, or 270 degrees. The following shows the default (for most HP-GL/2 devices*) and rotated orientation of the axes and locations of P1 and P2.



Note that P2 is now off the page. This occurs because the X,Y coordinates of P1 and P2 do not change. To set P1 and P2 at the hard-clip limits, use either the IP or IR instruction after the RO instruction (refer to the RO instruction for more information). If you reset your coordinate system to its default orientation, remember to reset P1 and P2 (using either the IP or IR instruction again).

Ending Your Program and Advancing the Page

When using a raster plotter, you must indicate the end of your plot before the plotter will rasterize and draw it. For printers with HP-GL/2 capability, this may require you to exit HP-GL/2 mode and send a form feed (FF) character in the PCL context. The Advance Full Page (PG) instruction automatically signals the end of incoming data for the plotter and starts the rasterization process. You may be able to use a control-panel function to achieve the same result, but the PG instruction is more efficient. It frees the user for more important duties. *In multiuser environments, PG is a necessary plot separator.*

PG advances the rollfeed paper (on pen and raster plotters) the length set by the Plot Size (PS) instruction, or the default page size if there is no PS instruction. (The PS instruction is described in chapter 9, *The Technical Graphics Extension.*)

If your device supports it, you can make additional copies of a plot by following the PG instruction with the Replot (RP) instruction. Generally, your plotter must have an area, such as disk space or a memory buffer, to support this feature. Since the plot is already stored in the plotter, using RP frees the computer while the copies are drawn.

* If you are using a printer with HP-GL/2 capability, your default orientation may be different. Check the documentation for your device or HP-GL/2 option.

DF, Default Values

USE: Returns the plotter to certain default settings (with the exception of settings already made from the plotter's control panel). Use DF to return the plotter to a known state while maintaining the current locations of P1 and P2. When you use DF at the beginning of a program, graphics parameters such as character size, slant, or scaling are not inherited from another program.

SYNTAX: DF*(;)*

REMARKS: The DF instruction resets the plotter to the following conditions.

Function	Equivalent Instruction	Default Condition
Anchor Corner	AC	Anchor set to lower-left corner of hard-clip limits.
Alternate Font Definition	AD	*Device-dependent.*
Character Fill Mode	CF	Solid fill, no edging.
Absolute Direction	DI	Character direction parallel to X-axis.
Define Label Terminator	DT	**ETX** and nonprinting mode.
Define Variable Text Path	DV	Text printed left to right with normal line feed.
Extra Space	ES	Turns off extra spacing.
Fill Type	FT	Solid bidirectional fill (all previously specified parameters for other fill types return to their defaults).
Input Window	IW	Hard-clip limits.
Line Attributes	LA	Butt caps, mitered joins, and miter limit = 5.
Label Origin	LO1	Standard labeling starting at current location.
Line Type	LT	Solid line, relative mode, pattern length = 4% of diagonal distance from P1 to P2.
Plotting Mode	PA	Absolute plotting.
Polygon Mode	PM0PM2	Polygon buffer cleared.
Raster Fill	RF	Solid fill in the current color.
Scale	SC	User-unit scaling off.
Standard Font Definition	SD	*Device-dependent.*

(Continued)

Function	Equivalent Instruction	Default Condition
Character Size Absolute	SI	Turns off size transformation.
Character Slant	SL	No slant.
Symbol Mode	SM	Off.
Select Font	SS	Standard font selected (*device-dependent*).
Transparent Data	TD	Normal printing mode.
User-Defined Line Type	UL	Defaults all 8 line types.

In addition, the plotter updates the carriage-return point for labeling to the current pen location. (Refer to chapter 8, *The Character Group*, for more information on the carriage-return point.)

The DF instruction does *not* affect the following plotter conditions.

- Locations of P1 and P2.
- Current pen, its location, width, width unit selection, colors, or up/down position.
- Plot size.
- Plot rotation.
- Generated errors (not cleared).

The DF instruction also affects the following HP-GL/2 extension instructions.

- **Technical Graphics Extension** (chapter 9):
  MT, Media Type
  QL, Quality Level
  ST, Sort
  VS, Velocity Select
- **Palette Extension** (chapter 10):
  CR, Set Color Range for Relative Color Data
  NP, Number of Pens
  PC, Pen Color Assignment
  SV, Screened Vectors
  TR, Transparency Mode

Related Instructions	Group/Extension
IN, Initialize	*The Configuration and Status Group*
BP, Begin Plot	*The Technical Graphics Extension*

IN, Initialize

USE: Resets all programmable plotter functions to their default settings. Use IN to return the plotter to a known state and to cancel settings that may have been changed by a previous program. *The IN instruction clears existing HP-GL/2 error conditions without affecting handshake protocol.*

SYNTAX: IN*n(;)*
or
IN(;)

Parameter	Format	Functional Range	Default
n	clamped integer	1*	no parameter

* This is the only valid number for this instruction. The full range is -2^{23} to $2^{23}-1$. If a fraction is present, it is rounded to the nearest integer; e.g., 0.6 is rounded to 1.

REMARKS:

- **No Parameter** — Defaults to the feature setting(s) specified using the device's control panel and sets all other programmable HP-GL/2 features to their factory defaults.
- **n** — Defaults all programmable HP-GL/2 features to the factory-set conditions. Any number other than 1 is treated like *IN* (no parameter).

  NOTE: An *IN1* is a temporary override of control-panel defaults; a subsequent *IN* will restore control-panel defaults. *IN1* is not recommended unless it is imperative that the program control the plotter.■

In this manual, all program examples begin with (*IN*) to clear unwanted conditions from the previous program. The IN instruction sets the plotter to the same conditions as the DF instruction, *plus* the following conditions.

- Returns pen location to lower-left corner of the hard-clip limits (*PA0,0*).
- Raises the pen (*PU*).
- Cancels plot rotation (*RO*).
- Sets P1 and P2 to the lower-left and upper-right corners, respectively, of the hard-clip limits (*IP*).
- Sets pen width mode to metric; units are in millimeters (*WU*).
- Sets the pen width to 0.35 mm (*PW*).
- Sets plot length for rollfeed plotters to approximately 1½ times the paper width (*PS*).
- Clears HP-GL/2 errors.

Related Instructions	Group/Extension
DF, Default Values	*The Configuration and Status Group*
BP, Begin Plot OS, Output Status	*The Technical Graphics Extension*

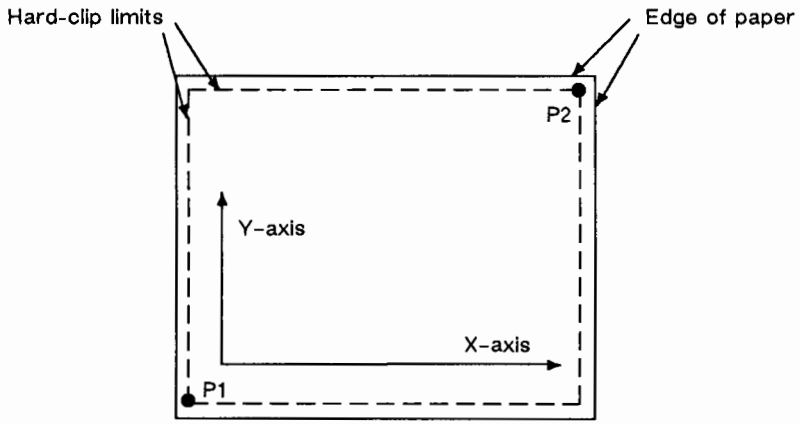
IP, Input P1 and P2

USE: Establishes new or default locations for the scaling points P1 and P2. P1 and P2 are used by the Scale (SC) instruction to establish user-unit scaling. You can also use IP to control character size (SR), label direction (DR), pen width (WU), spacing for hatch fill patterns (FT), and line type pattern length (LT).

SYNTAX: IP*P1X,P1Y(,P2X,P2Y;)*
or
IP(;)

Parameter	Format	Functional Range	Default
P1x,P1y(,P2x,P2y)	integer	*device-dependent*	hard-clip limits

REMARKS: The default location of P1 is in the lower-left corner of the hard-clip limits; the default location of P2 is in the upper-right corner of the hard-clip limits, as shown in the following illustration.



- **No Parameters** — Defaults P1 and P2 to the hard-clip limits, adjusted by any current axis rotation.

  NOTE: If an IP instruction without parameters is executed after the axes have been rotated, P1 and P2 locations change to reflect the rotation. Refer to the RO instruction for more information.■

- **X,Y Coordinates** — Specify the location of P1 (and, optionally, P2) in plotter units. Specifying P2 is not required. P2 tracks P1 and its coordinates change so that the distances of X and Y between it and P1 stay the same. This tracking process can cause P2 to end up outside the hard-clip limits.

  If either coordinate of P1 equals the corresponding coordinate of P2, the coordinate(s) of P2 is incremented by one.

The locations of P1 and P2 interact with the following instructions.

Instructions Affected by P1/P2	Group
IW, Input Window RO, Rotate Coordinate System SC, Scale	*The Configuration and Status Group*
FT, Fill Type LT, Line Type PW, Pen Width WU, Pen Width Unit Selection	*The Line and Fill Attributes Group*
DR, Relative Direction LB, Label SR, Relative Character Size	*The Character Group*

An IP instruction remains in effect until another IP instruction is executed, an IR instruction is executed, the plotter is initialized, or a Plot Size (PS) instruction is issued on a "clean" (unplotted) page.

Related Instructions	Group/Extension
IR, Input Relative P1 and P2 RO, Rotate Coordinate System SC, Scale	*The Configuration and Status Group*
OP, Output P1 and P2	*The Technical Graphics Extension*

ERRORS:

Condition	Error	Plotter Response
1 or 3 parameters	2	ignores instruction

IR, Input Relative P1 and P2

USE: Establishes new or default locations for the scaling points P1 and P2 relative to the hard-clip limits. P1 and P2 are used by the Scale (SC) instruction to establish user-unit scaling. You can also use IR to control character size (SR), label direction (DR), pen width (PW and WU), spacing for hatch fill patterns (FT), and line type pattern length (LT).

SYNTAX: IR*P1X,P1Y(,P2X,P2Y;)*
or
IR*(;)*

Parameter	Format	Functional Range	Default
P1x,P1y(,P2x,P2y)	clamped real	*device-dependent*	0,0,100,100%

REMARKS: When P1 and P2 are set with IR, the scaled area is page-size-independent. The P1/P2 rectangular area will occupy the same proportional space on any size media.

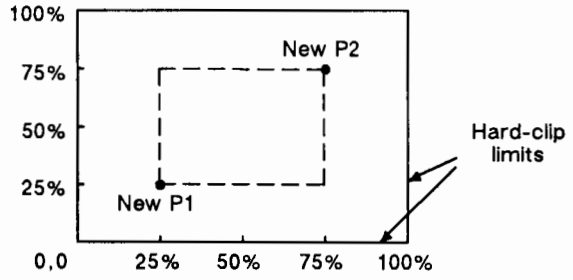
- **No Parameters** — Defaults P1 and P2 to the lower-left and upper-right corners of the hard-clip limits, respectively.

- **X,Y Coordinates** — Specify the location of P1 (and, optionally, P2) as *percentages* of the hard-clip limits. Specifying P2 is not required. If P2 parameters are omitted, P2 tracks P1 so that the X and Y distances between P1 and P2 remain the same. This tracking process can cause P2 to end up outside the hard-clip limits.

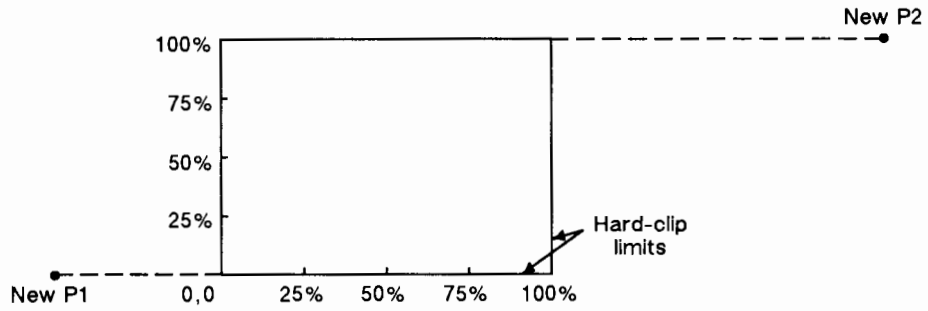
  If either coordinate of P1 equals the corresponding coordinate of P2, the coordinate(s) of P2 is incremented by one plotter unit.

EXAMPLE:

Sending the instruction *(IR25,25,75,75)* establishes new locations for P1 and P2 that create an area half as high and half as wide as the hard-clip limits in the center of the page. Refer to the following illustration.



You can set P1 or P2 outside the hard-clip limits by specifying parameters less than zero and greater than 100. For example, sending (*IR–50,0,200,100*) sets P1 and P2 as shown in the following illustration.



If you set P1 and P2 beyond the hard-clip limits, your plot will be scaled with respect to those locations; however, only the portion of the plot fitting within the hard-clip limits will be drawn.

NOTE: The specified percentages are converted to the equivalent plotter-unit coordinates; if the coordinate system orientation subsequently changes (e.g., by sending an RO instruction), the new P1/P2 locations maintain the same offsets from the new origin that they had from the previous origin.■

The locations of P1 and P2 interact with the following instructions.

Instructions Affected by P1/P2	Group
IW, Input Window RO, Rotate Coordinate System SC, Scale	*The Configuration and Status Group*
FT, Fill Type LT, Line Type PW, Pen Width WU, Pen Width Unit Selection	*The Line and Fill Attributes Group*
DR, Relative Direction LB, Label SR, Relative Character Size	*The Character Group*

An IR instruction remains in effect until another IR instruction is executed, an IP instruction is executed, the plotter is initialized, or a Plot Size (PS) instruction is issued on a "clean" (unplotted) page.

Related Instructions	Group/Extension
IP, Input P1 and P2 RO, Rotate Coordinate System SC, Scale	*The Configuration and Status Group*
OP, Output P1 and P2	*The Technical Graphics Extension*

ERRORS:

Condition	Error	Plotter Response
1 or 3 parameters	2	ignores instruction
more than 4 parameters	2	uses first 4 parameters

IW, Input Window

USE: Defines a rectangular area, or window, that establishes soft-clip limits. Subsequent programmed pen motion is restricted to this area. Use IW to restrict plotting to a specified area on the media.

SYNTAX: IW*XLL,YLL,XUR,YUR*(;)
or
IW(;)

Parameter	Format	Functional Range	Default
XLL,YLL,XUR,YUR	current units	*device-dependent*	hard-clip limits

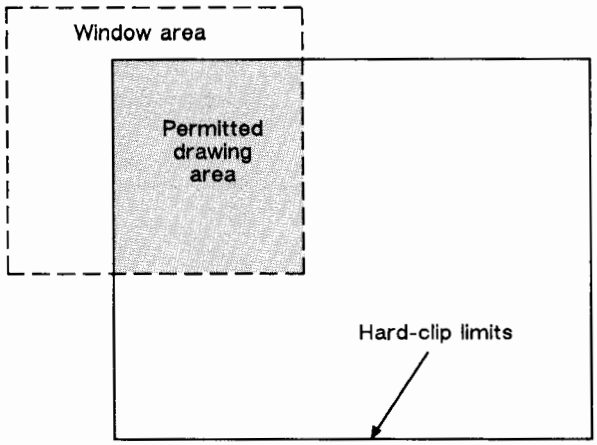
REMARKS: When you turn the plotter on, the window is automatically set to the hard-clip limits.

- **No Parameters** — Defaults the soft-clip limits to the hard-clip limits.

- **X,Y Coordinates** — Specify the opposite, diagonal corners of the window area, usually the lower-left (XLL,YLL) and upper-right (XUR,YUR) corners. Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off.

  When scaling is on, subsequent changes to P1 and P2 or changes in scaling cause the window to move on the physical page, but keep the same coordinate locations. Changes to scaling will also cause the window to be recalculated in the new coordinate system. However, sending a subsequent SC instruction binds the window to its equivalent plotter units. Then, the window does not change with any subsequent IP or IR instructions.

You can define a window that extends beyond the hard-clip limits; however, the plotter cannot plot beyond the hard-clip limits. All programmed pen motion is restricted to this area. For more information, refer to *Windowing: Setting Up Soft-Clip Limits* at the beginning of this chapter.

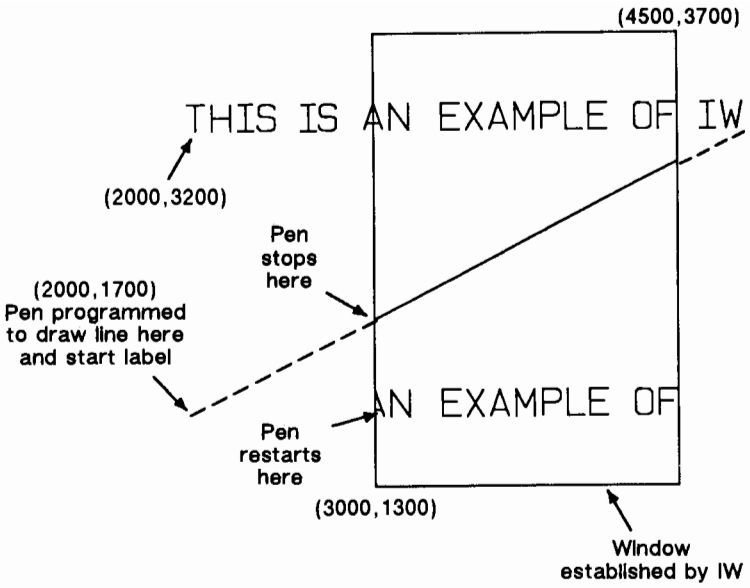


If the window falls entirely outside of the hard-clip limits, no plot will be drawn. This can happen when you define a window that is normally within the hard-clip limits and a subsequent Rotate Coordinate System (RO) instruction moves the window outside of the hard-clip limits.

The IW instruction remains in effect until another IW instruction is executed, the plotter is initialized or set to default conditions, or a Plot Size (PS) instruction is issued on a "clean" (unplotted) page.

EXAMPLE: The following draws a label, then establishes a window and draws the label again along with a line. Notice how the line and label are only clipped after the window has been established but not before. The *CHR$(3)* is the default label terminator **ETX**. Refer to chapter 8, *The Character Group*, for more information on sending and terminating labels.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS7000,5000SP1"
40  PRINT #1, "SI.2,.35PA2000,3200"
50  PRINT #1, "LBTHIS IS AN EXAMPLE OF IW"+CHR$(3)
60  PRINT #1, "IW3000,1300,4500,3700"
70  PRINT #1, "PD2000,1700"
80  PRINT #1, "LBTHIS IS AN EXAMPLE OF IW"+CHR$(3)
90  PRINT #1, "PU3000,1300PD4500,1300,4500,3700"
100 PRINT #1, "PD3000,3700,3000,1300"
110 PRINT #1, "PUSP0PG;"
120 END
```



Related Instructions	Group/Extension
IP, Input P1 and P2 IR, Input Relative P1 and P2 RO, Rotate Coordinate System SC, Scale	*The Configuration and Status Group*
FR, Frame Advance	*The Technical Graphics Extension*

PG, Advance Full Page

USE: *Devices with page advance capability:* Terminates the plot being sent, draws it, and advances the page.

Devices without page advance capability: If the media has been plotted on, this instruction is equivalent to the Not Ready (NR) instruction (chapter 9, *The Technical Graphics Extension*).

SYNTAX: PG*(n)*;
or
PG;

NOTE: The PG instruction, with or without parameters, *must* be terminated with a semicolon, unless it is followed by the RP instruction.■

Parameter	Format	Functional Range	Default
n	clamped integer	*device-dependent*	no parameter

REMARKS: Some devices require an end-of-file marker to designate the end of incoming data. The PG instruction is a common marker for HP-GL/2 devices (the RP instruction is another). When the PG instruction is received, the plot is drawn and the page is ejected. Also, the first Pen Down instruction after a page eject clears the stored plot. If the plotter expects but does not receive the PG instruction, it will wait for user interaction (through the control panel) before it will draw the plot.

PG moves the current pen location to the lower-left corner of the hard-clip limits on the next page and raises the pen. PG does not affect P1 and P2 values or plot rotation.

- **No Parameter** — Advances the page only if you have plotted on the current page. This is the recommended method of using the PG instruction.
- **n** — Advances the page whether or not you have plotted on the media.

NOTE: *On PCL devices with HP-GL/2 capability*, the PG instruction is ignored. You must exit to PCL context and send a form feed character (FF) to advance the page.■

Related Instructions	Group/Extension
PS, Plot Size RP, Replot	*The Configuration and Status Group*
BP, Begin Plot	*The Technical Graphics Extension*

RO, Rotate Coordinate System

USE: Rotates the plotter's coordinate system 90°, 180°, and 270° counterclockwise about the plotter-unit coordinate origin. Use RO to orient your plot vertically or horizontally, or to reverse the orientation.

SYNTAX: RO*angle(;)*
or
RO*(;)*

Parameter	Format	Functional Range	Default
angle	clamped integer	0°, 90°, 180°, or 270°	0°

REMARKS: The plotter interprets the parameters as follows:

- **No Parameter** — Defaults the orientation of the coordinate system to 0° (horizontal). Equivalent to *(RO0)*.
- **Angle** — Specifies the degree of rotation.

  0 Sets the orientation to horizontal (default). Note that using the PS instructions with parameters on a rollfeed device can change the orientation of the drawing on the media.

  90 Rotates and shifts the coordinate system counterclockwise 90 degrees to place the plotter-unit origin at the appropriate corner of the physical plotting surface.

  180 Rotates and shifts the coordinate system counterclockwise 180 degrees to place the plotter-unit origin at the appropriate corner of the physical plotting surface.

  270 Rotates and shifts the coordinate system counterclockwise 270 degrees to place the plotter-unit origin at the appropriate corner of the physical plotting surface.

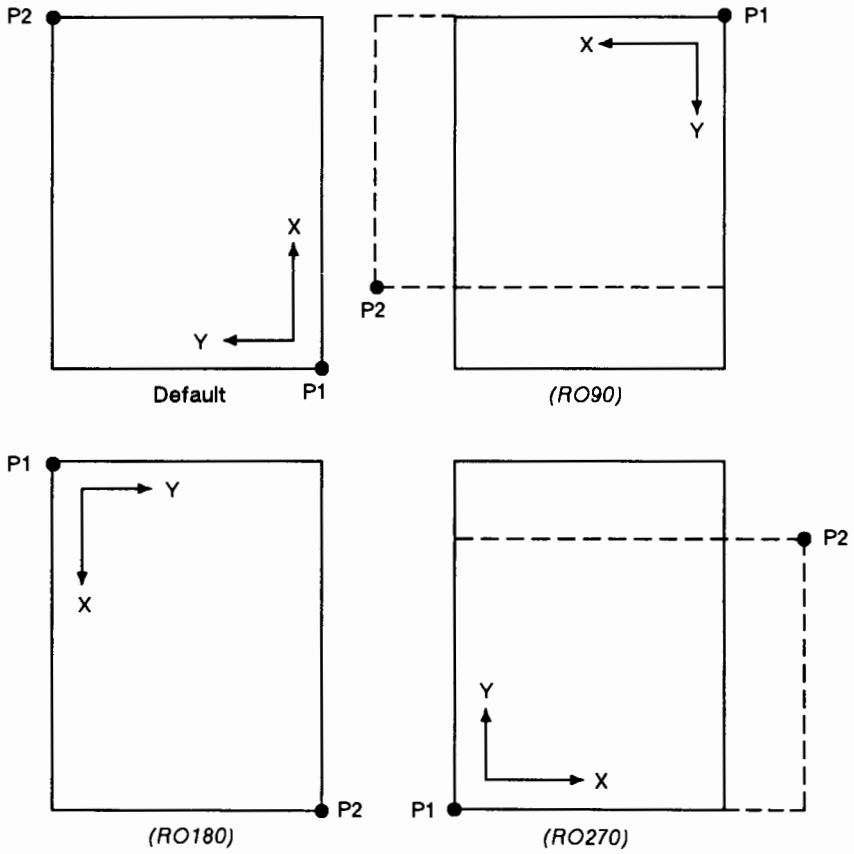
Rotations are not cumulative. The angle specified is the axis rotation. That is, *RO90RO90;* rotates the axis 90°, not 180°.

Note that the pen location does not change when you rotate the coordinate system. Instead, the plotter updates the pen's logical X,Y coordinate location to reflect the new orientation.

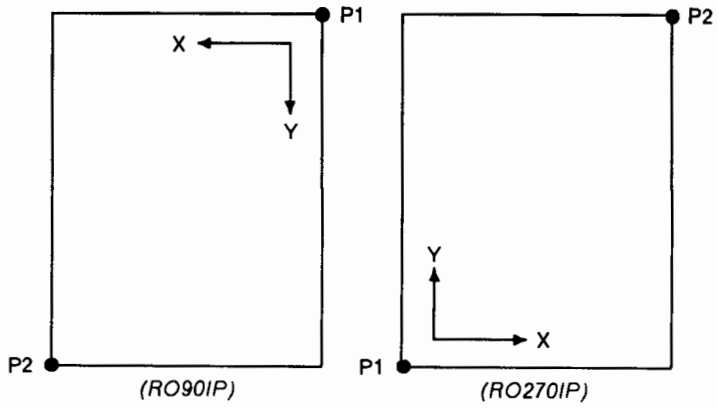
P1 and P2 rotate with the coordinate system. However, they maintain the same X,Y coordinate values as before the rotation. This means that P1 and P2 can be located outside of the hard-clip limits. Follow *(RO90)* or *(RO270)* with *(IP)* or *(IR)* to relocate points P1 and P2 to the hard-clip limits for the current orientation.

The current orientation remains in effect until it is changed by another RO instruction or the plotter is initialized.

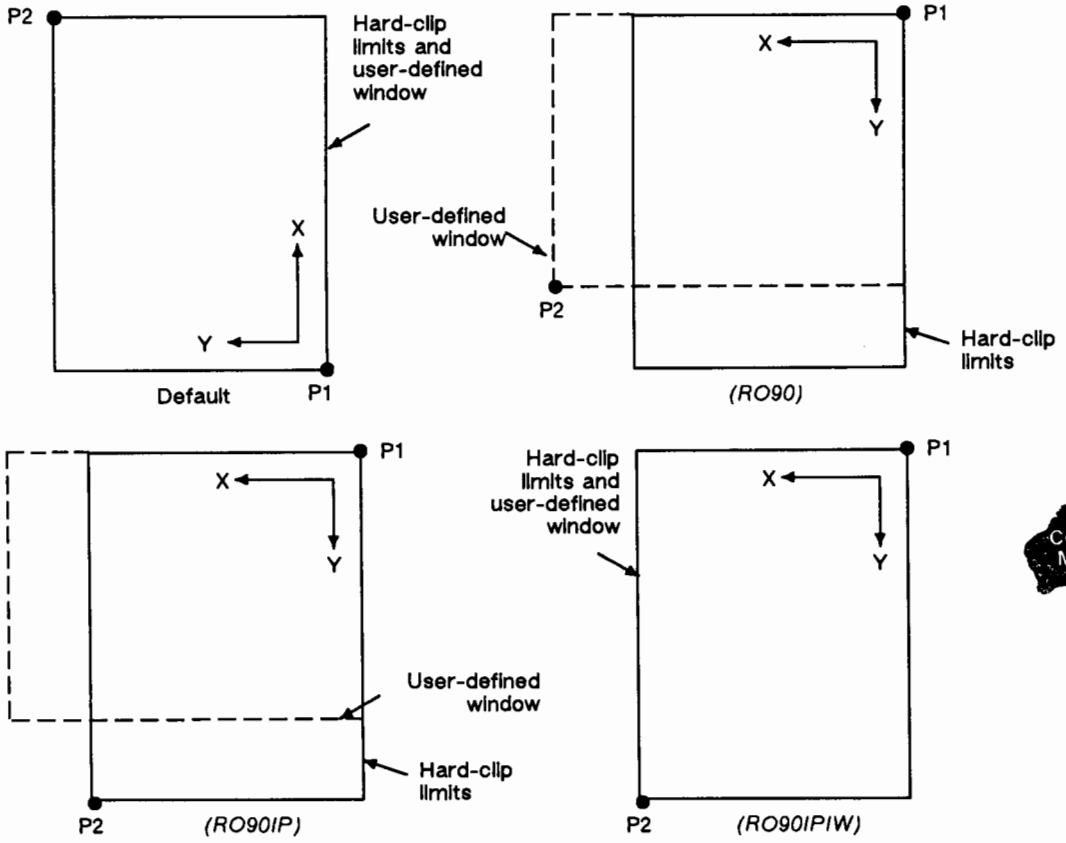
EXAMPLE: The following illustration shows the default orientation and the result of rotating the orientation without relocating P1 and P2.



The next illustration shows the locations of P1 and P2 when you follow the rotation with the IP instruction.



If you have previously defined a window (see the IW instruction), an RO may rotate the window so it is either entirely or partially outside the hard-clip limits. Data that is beyond the hard-clip limits cannot be drawn. Use *IW* to reset the window to the hard-clip limits. Note that *IP* does not affect the window limits.



Related Instructions	Group
IP, Input P1 and P2 IR, Input Relative P1 and P2 IW, Input Window	*The Configuration and Status Group*

RP, Replot

USE: Draws multiple copies of plots. Your plotter must have an internal hard disk or designated buffer area to store the plot. This instruction is ignored on devices that cannot store the plot data.

SYNTAX: RP(*n*);
or
RP;

NOTE: The RP instruction, with or without parameters, *must* be terminated with a semicolon.■

Parameter	Format	Functional Range	Default
n	clamped integer	*device-dependent*	1

REMARKS: Use the RP instruction at the end of your plot following the Advance Full Page (PG) instruction when you want more than one copy of a plot.

- **No Parameter** — Assumes you want one copy.
- **n** — Specifies the number of additional copies to be made.

The plotter ignores RP when printing the current page would produce no marks.

RP advances the page only if the plot has not already been terminated by a PG instruction. If *n* is less than or equal to zero and there is no previous PG instruction, the plotter advances the page and is otherwise ignored. If *n* is greater than the maximum value allowed for the plotter, the value is clamped to the maximum value.

NOTE: You may need to reload paper to print your plot more than once.■

Related Instructions	Group/Extension
PG, Advance Full Page	*The Configuration and Status Group*
BP, Begin Plot	*The Technical Graphics Extension*

SC, Scale

USE: Establishes a user-unit coordinate system by mapping user-defined coordinate values onto the scaling points P1 and P2. Use SC to plot in units convenient to your application (such as millimeters or inches). In addition, use SC to establish automatic isotropic scaling or to relocate the origin and set a specific ratio of plotter units to user units. For a discussion of the basic concept of scaling, refer to *Using the Scaling Instruction* earlier in this chapter.

SYNTAX: SC*X*MIN,*X*MAX,*Y*MIN,*Y*MAX*(,type(,left,bottom;))*
or
SC*X*MIN,*X*FACTOR,*Y*MIN,*Y*FACTOR,*type(;)*
or
SC*(;)*

Parameter	Format	Functional Range	Default
XMIN,XMAX, YMIN,YMAX	real	*device-dependent*	no default
type	clamped integer	0, 1, or 2	0
left	clamped real	0 to 100%	50%
bottom	clamped real	0 to 100%	50%
XFACTOR, YFACTOR	real	*device-dependent*	no default

REMARKS: There are three forms of scaling: anisotropic, isotropic, and point-factor. The *type* parameter tells the plotter which form you are using. Refer to the following table.

Scaling Form	Type	Description
Anisotropic	0	Establishes standard user-unit scaling.
Isotropic	1	Establishes user-unit scaling with units along the X- and Y-axes being the same size.
Point-factor	2	Establishes P1 user-unit location and a specific ratio of plotter units to user units.

- **No Parameters** — Turns off scaling; subsequent coordinates are in plotter units.

For Scaling Types 0 and 1

Scaling Form	Type	Syntax
Anisotropic	0	SC*X*MIN,*X*MAX,*Y*MIN,*Y*MAX*(,type;)*
Isotropic	1	SC*X*MIN,*X*MAX,*Y*MIN,*Y*MAX,*type(,left,bottom;)*

These forms of scaling establish a user-unit coordinate system by mapping user-defined coordinate values onto the scaling points P1 and P2. The type parameter selects between anisotropic (type 0) and isotropic (type 1) scaling.

- XMIN, XMAX, YMIN, YMAX — Represent the user-unit range along the X- and Y-axes, respectively. For example, *SC0,15,0,10* indicates 15 user units between P1x and P2x and 10 user units between P1Y and P2Y. As a result, the first and third parameters (XMIN and YMIN) are the coordinate pair that is mapped onto P1—in this case, the coordinate value of P1 is (0,0). The second and fourth parameters (XMAX and YMAX) are the coordinate pair that is mapped onto P2—in this case, the coordinate value of P2 is (15,10).

  This is different from the IP instruction, in which the parameters are expressed as X,Y coordinate pairs rather than as ranges.

NOTE: When XMIN equals XMAX, or YMIN equals YMAX, the plotter ignores the instruction and generates an error.■

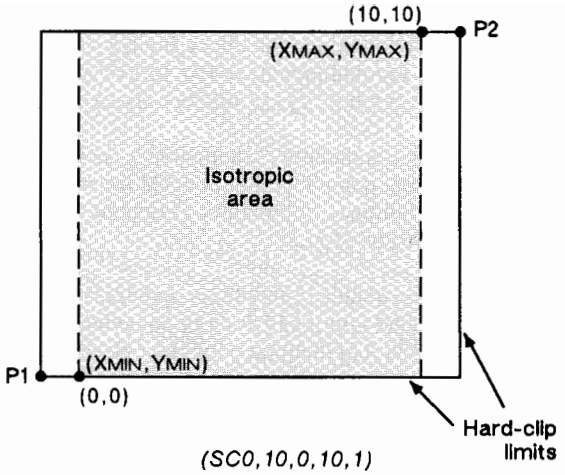
You can specify any values for the scaling ranges. As their names suggest, you will normally want to specify XMIN smaller than XMAX, and YMIN smaller than YMAX. For some applications, however, it may make more sense to let XMIN be a value greater than XMAX, and YMIN be greater than YMAX.

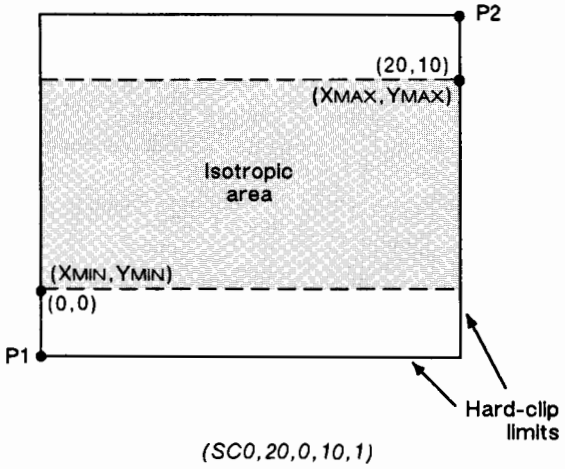
The parameters of the SC instruction are always mapped onto the current P1 and P2 locations. P1 and P2 retain these new values until scaling is turned off or another SC instruction redefines the user-unit values. Thus, the size of a user unit could change if any change is made in the location and distance between P1 and P2 *after* an SC instruction is executed.

- **Type** — Specifies anisotropic or isotropic scaling.

  0 Anisotropic scaling. Allows a user unit along the X-axis to be a different size than user units along the Y-axis. Geometric shapes are distorted when you use anisotropic scaling. For example, a circle might be drawn as an ellipse (oval-shaped). (*Left* and *bottom* parameters are ignored for anisotropic scaling.)

  1 Isotropic scaling. Produces user units that are the same size on both the X- and Y-axes. The following illustrations show how the plotter adjusts the location of (XMIN,YMIN) and (XMAX,YMAX) to create the largest possible scaling area within the P1/P2 limits. (Remember, the user units are always of equal size regardless of the shape of the isotropic area.)





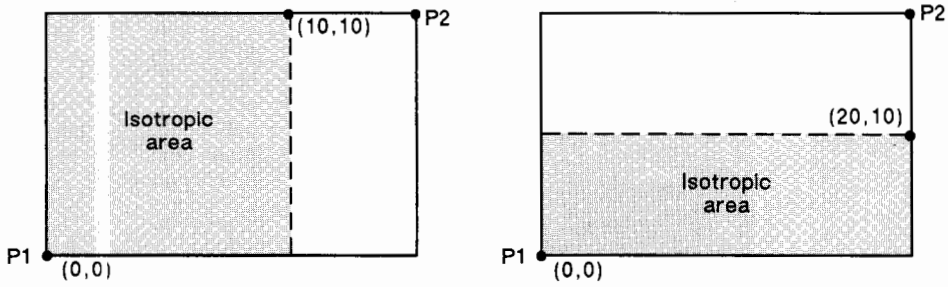
Isotropic Scaling with Default Left, Bottom Parameters

- **Left, Bottom** — Position the isotropic area in the P1/P2 limits. (These parameters are always specified together and are valid for isotropic scaling only.) The left parameter indicates the percentage of the unused space on the left of the isotropic area; the bottom parameter indicates the percentage of unused space below.

  The defaults for the *left* and *bottom* parameters are each 50%. This centers the isotropic area on the page with the unused space equally divided between left and right or top and bottom, as shown in the previous illustrations. (Directional implications of these parameters depend on P1/P2 orientation.)

  Although you must specify both parameters, the plotter applies only one: the left parameter applies when there is extra horizontal space; the bottom parameter applies when there is extra vertical space. The following examples illustrate left and bottom parameters of 0% and 100%.

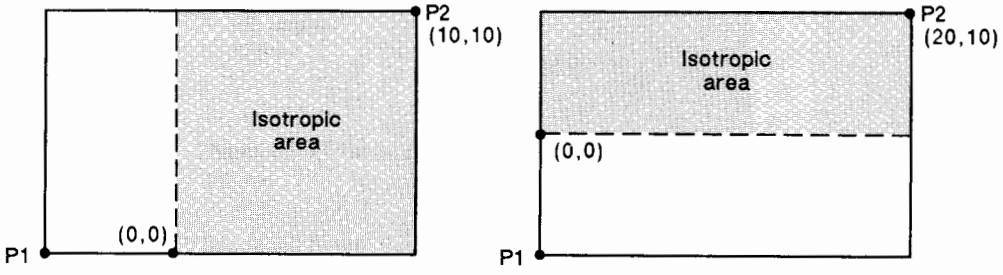
Left, Bottom = 0,0



(SC0,10,0,10,1,0,0)

(SC0,20,0,10,1,0,0)

Left, Bottom = 100,100



(SC0,10,0,10,1,100,100)

(SC0,20,0,10,1,100,100)

For Scaling Type 2

Scaling Form	Type	Syntax
Point-factor	2	SC*X*MIN,*X*FACTOR,*Y*MIN,*Y*FACTOR,*type(;)*

The third form, point-factor scaling, sets a specific ratio of plotter units to user units and establishes the user-unit coordinates of P1.

- XMIN,XFACTOR,YMIN,YFACTOR — Establish the user-unit coordinates of P1 and the ratio of plotter to user units. XMIN and YMIN are the user-unit coordinates of P1. XFACTOR sets the number of plotter units per user unit on the X-axis; YFACTOR sets the number of plotter units per user unit on the Y-axis.

- **Type** — Must be 2 for this type of scaling.

An SC instruction remains in effect until another SC instruction is executed, or the plotter is either initialized or set to default conditions.

NOTE: When XFACTOR or YFACTOR equals zero (0), the plotter ignores the instruction and generates an error.■

EXAMPLES: The following examples explain the effect of several parameter selections.

(*SC0,1,0,1,2*) moves the origin to P1 and establishes a one-to-one ratio of plotter to user units. This allows you to continue plotting in plotter units with the advantage of using real numbers.

(*SC0,40,0,40,2*) allows scaling in millimeters since 1 millimeter = 40 plotter units. Each user unit will be 1 millimeter.

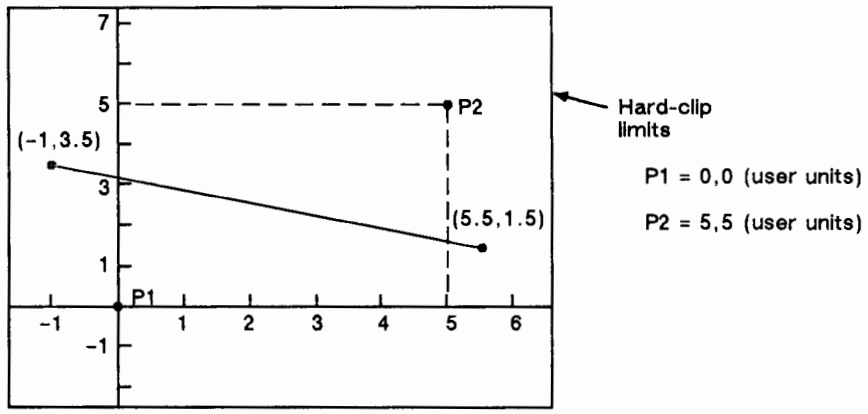
(*SC0,1.016,0,1.016,2*) allows scaling in thousandths of an inch since 1 inch = 1016 plotter units. Each user unit will be $1/1000$ of an inch.

NOTE: When combining the Scaling and Plot Size (PS) instructions and accurate scaling is a *must*, note the following. If you specify a plot size larger than your media's maximum plotting area, your plot size will be reduced to the hard-clip limits and your user units will be smaller than you intended. To correct, add an IP instruction so that the P1/P2 area is equal to the intended size of your PS instruction (IP0,0,*PSlength,PSwidth*). This moves P2 off the page, but guarantees accurate scaling. Place the IP instruction between the PS and SC instructions.■

While scaling is on (after sending an SC instruction with parameters), only those plotting instructions that can be issued as "current units" are interpreted as user units; the instructions that can be issued only in plotter units are still interpreted as plotter units. (The syntax section of each instruction tells you what kind of units each parameter requires.)

Remember, the SC parameters are mapped onto the current locations of P1 and P2. P1 and P2 do *not* represent a graphics limit; therefore, the new user-unit coordinate system extends across the entire range of the plotter-unit coordinate system. Thus, you can plot to a point beyond P1 or P2, as long as you are within the hard-clip limits or the current window.

For example, the following illustration shows a line that starts and ends outside the P1/P2 scaling rectangle. Notice that the endpoint coordinate locations are specified in user units, even though they are outside the P1/P2 area. If you established a window with its lower-left and upper-right corners at P1 and P2, respectively, you would not see the endpoints of the line, only that portion of the line that is within the current window limits.



Related Instructions	Group/Extension
IP, Input P1 and P2	*The Configuration and Status Group*
PS, Plot Size	*The Technical Graphics Extension*

ERRORS:

Condition	Error	Plotter Response
no parameters	none	turns scaling off
more than 7 parameters	2	executes first 7 parameters
6 parameters or less than 4 parameters	2	ignores instruction
$X_{MIN} = X_{MAX}$ or $Y_{MIN} = Y_{MAX}$ or number out of range	3	ignores instruction
$X_{FACTOR} = 0$ or $Y_{FACTOR} = 0$	3	ignores instruction

5

The Vector Group

The information in this chapter enables you to achieve the following.

- Use absolute and relative coordinates when plotting.
- Draw lines and arcs.
- Encode coordinates to greatly increase your plotter's throughput.

The following instructions are described in this chapter.

Instruction	Summary
AA, Arc Absolute	Draws an arc using absolute coordinates.
AR, Arc Relative	Draws an arc using relative coordinates.
AT, Absolute Arc Three Point	Draws an arc from the current pen location through two specified absolute points.
CI, Circle	Draws a circle with a specified radius.
PA, Plot Absolute	Enables movement to absolute coordinate locations—with respect to the origin (0,0).
PD, Pen Down	Lowers the pen to the media.
PE, Polyline Encoded	Increases throughput by incorporating common HP-GL/2 instructions into an encrypted format.
PR, Plot Relative	Enables movement relative to the current pen location.
PU, Pen Up	Lifts the pen from the media.
RT, Relative Arc Three Point	Draws an arc from the current pen location through two relative points.

The following is a summary of the parameter formats and their *minimum* ranges. Your device may support a greater range than listed here.

Parameter Format	Minimum Ranges
Integer	2^{23} to $2^{23}-1$ (-8 388 608 to 8 388 607)
Real	2^{23} to $2^{23}-1$ (-8 388 608.000 0 to 8 388 607.999 9)
Clamped Integer	2^{15} to $2^{15}-1$ (-32 768 to 32 767)
Clamped Real	2^{15} to $2^{15}-1$ (-32 768.000 0 to 32 767.999 9)

Pen Position and Location

Some devices (such as raster devices) use "logical" rather than physical pens. Like a physical pen, this logical pen must be selected if you want to draw. Instructions such as Pen Up or Pen Down and phrases such as "current pen position" or "moving the pen" apply to the logical pen just as they would to a physical pen.

Pen Position

Pen position refers to whether the pen is up or down. Use the Pen Up (PU) instruction to raise the pen; use PU with parameters to move the pen to the desired plotting location before drawing. Use the Pen Down (PD) instruction to lower the pen; use PD with parameters to draw from the current location to the specified location. You must be aware of the pen's position (up or down) to avoid drawing unwanted lines between figures.

Every time you use a PU or PD instruction, the plotter updates the pen-up/down position information. Most HP-GL/2 instructions plot according to the current pen-up/down position. The following lists the instructions that include an automatic PD instruction as part of their function. After performing their complete function, they return the pen to its previous up/down position.

Instructions with Automatic Pen Downs	Group
CI, Circle	*The Vector Group*
EA, Edge Rectangle Absolute EP, Edge Polygon ER, Edge Rectangle Relative EW, Edge Wedge FP, Fill Polygon RA, Fill Rectangle Absolute RR, Fill Rectangle Relative WG, Fill Wedge	*The Polygon Group*
LB, Label	*The Character Group*
SM, Symbol Mode	*The Line and Fill Attributes Group*

NOTE: Whenever the plotter receives a Pen Down instruction, it produces a dot at the next coordinate. If the pen is already down when the plotter receives an instruction with an automatic pen down, a second dot is produced at the same coordinate. The first unnecessary dot can mar your final output. For best results, include a Pen Up (PU) instruction before any instruction with an automatic pen down.■

The definition of each instruction will tell you whether it has an automatic pen down. If you find that part of your plot isn't drawn, make sure your program uses the PD instruction before the affected instructions. Turning on the the plotter or sending an IN instruction sets the pen in the up position.

Pen Location

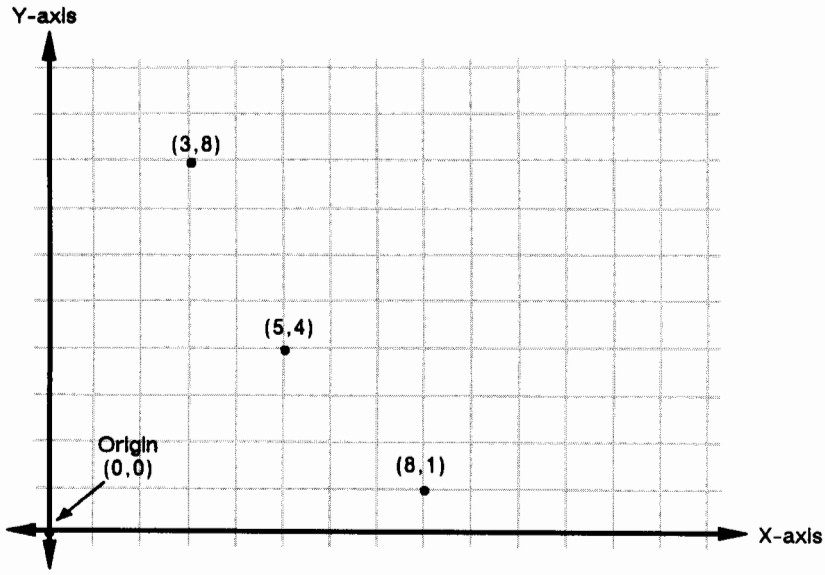
Pen location refers to the X,Y coordinates of the current plotting location (the point at which the next plotting instruction will begin). Most instructions, when completed, update the pen location to the current point. The next instruction then begins at that location. Some instructions do not update the current pen location. The definition of each instruction will tell you whether the current pen location is updated or restored. Use the Pen Up (PU) instruction with X,Y coordinates to lift the pen and move it to a new location.

The DF instruction does not reset the current pen location; the IN instruction moves it to the lower-left corner of the hard-clip limits. You must specify your beginning pen location for each plot. If you don't specify a starting location, your plot will begin at (0,0), or, if your plot starts with a DF instruction, your plot may begin at the same location where a previous plot finished.

Absolute and Relative Movement

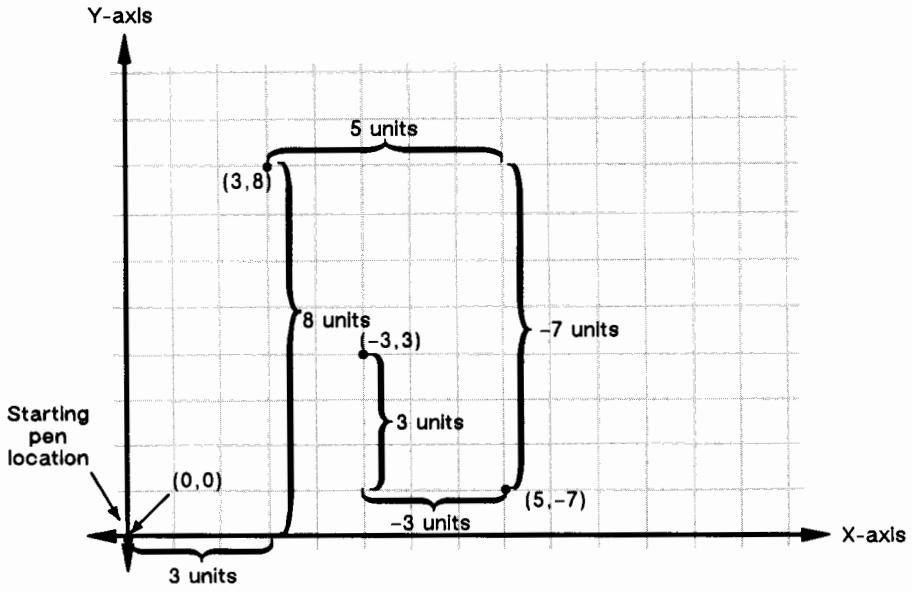
The Plot Absolute (PA) and Plot Relative (PR) instructions let you specify whether you want to draw using absolute or relative movement. Absolute movement uses X,Y coordinates to specify a fixed point relative to the origin (0,0). Relative movement uses X,Y *increments* to specify the number of units the pen moves from its current location. Both absolute and relative movement can be specified as user units (when scaling is on) or as plotter units (when scaling is off). All instructions that use relative increments include "relative" in their name (except the PE instruction).

The following illustrates absolute coordinates. The coordinate pairs (3,8), (5,4), and (8,1) are always in the same place with respect to the origin, no matter where the pen is when the coordinates are issued.



Absolute Coordinates

For an example of relative movement, assume that the pen is currently at the origin. To get to the same previously shown absolute points using relative coordinates, count 3 units to the right and 8 units up from the current pen location; these are both positive directions with respect to the origin. This is the relative location (3,8). Now move 5 positive X-units and 7 negative Y-units from this location to the lower point; this is the relative location (5,–7). From this location, move to the last point by moving 3 negative X-units and 3 positive Y-units (–3,3).



Relative Coordinates

Relative movement is very useful in many applications where you know the dimensions of the shape you want, but don't want to calculate the absolute coordinates. For example, if you knew you wanted a box 4 X-units by 8 Y-units, you could easily draw it without having to calculate the absolute coordinates. Using relative coordinates can also increase the throughput of your device.

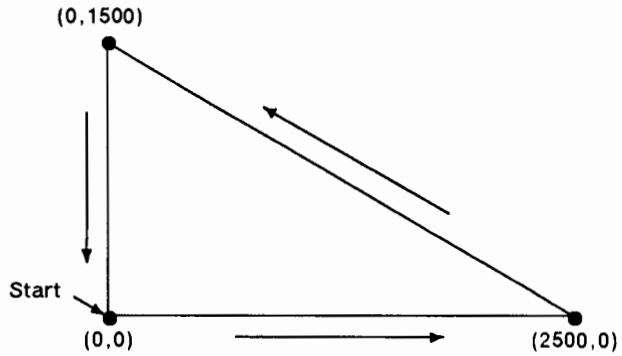
NOTE: Relative increments add to the current pen location. The plotter automatically converts the new relative location to absolute coordinates and updates the current pen location. When using relative movement, then, some round-off error may occur. Therefore, use absolute movement when you need to guarantee line endpoints.■

Drawing Lines

You can draw lines between two points (X,Y coordinate pairs) using the PD instruction and a series of absolute or relative coordinate pairs. (If one or both of the coordinate pair values fall outside the plotting area—window or hard-clip limits—the plotter draws only the portion of the line that falls within the plotting area.)

In the following example, note that the PA instruction sets absolute plotting, and the coordinate pair (0,0) specifies the beginning pen location.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS7000,5000"
30  PRINT #1, "SP1PA0,0PD2500,0,0,1500,0,0"
40  PRINT #1, "PUSP0PG;"
50  END
```



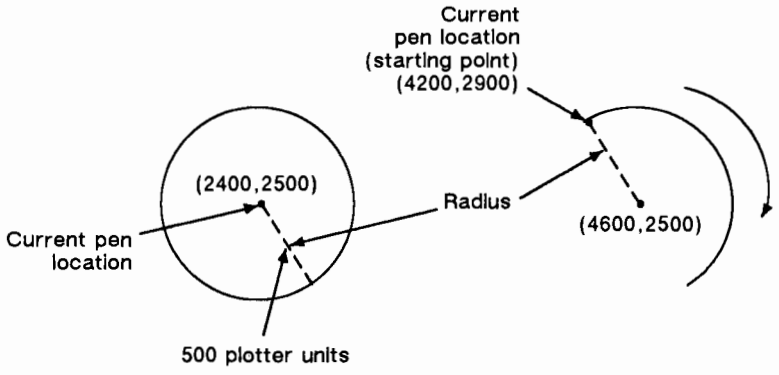
You can increase your plotter's throughput by using the Polyline Encoded (PE) instruction to send coordinates. The PE instruction requires that you convert coordinates from decimal to base 64 or 32. This conversion sends 60–70% less data, increasing your plotter's throughput, particularly over an RS-232-C interface. The PE instruction, with its parameters, is used in place of PA, PD, PR, PU, *and* SP.

Drawing Circles and Arcs

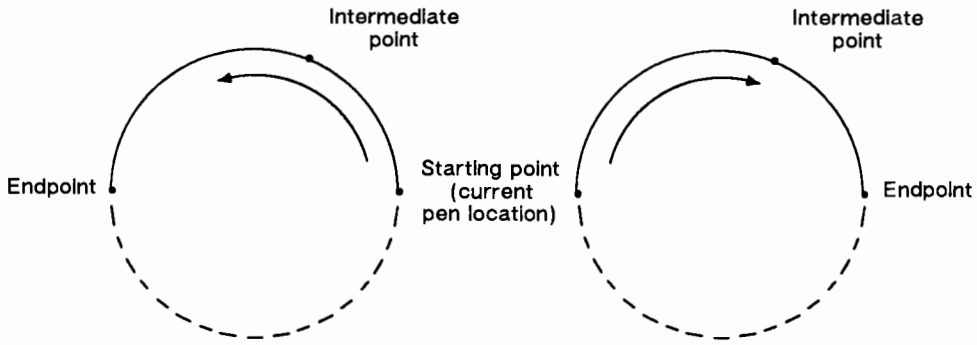
The Circle (CI) instruction uses your current pen location as the center of the circle; you specify the radius of the circle. The Arc Absolute (AA) and Arc Relative (AR) instructions use a similar method for drawing arcs; your current pen location becomes one end of the arc and you specify the center point (setting the radius) and the number of degrees through which you want the arc drawn.

The following is a simple program using CI and AA to draw a circle and an arc.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS7000,5000SP1PA2400,2500CI500"
40  PRINT #1, "PA4200,2900PDAA4600,2500,-180"
50  PRINT #1, "PUSP0PG;"
60  END
```



You can also draw arcs using the Absolute Arc Three Point (AT) and Relative Arc Three Point (RT) instructions. These instructions use three known points (your current pen location plus two points you specify) to calculate a circle and draw the appropriate segment of its circumference. The arc is drawn clockwise or counterclockwise as necessary, so that it passes through the intermediate point before the endpoint. Refer to the following illustration.



AA, Arc Absolute

USE: Draws an arc, using absolute coordinates, that starts at the current pen location and pivots around a specified center point.

SYNTAX: AA*XCENTER,YCENTER,sweep angle(,chord angle;)*

Parameter	Format	Functional Range	Default
XCENTER,YCENTER	current units	*device-dependent*	no default
sweep angle	clamped real	±360°	no default
chord angle	clamped real	0.5° to 180°	5°

REMARKS: The AA instruction draws the arc starting at the current pen location using the current pen-up/down position and line type and attributes. After the arc is drawn, the pen location remains at the end of the arc.

NOTE: Do *not* use an adaptive line type when drawing arcs with small chord angles. The plotter will attempt to draw the complete pattern in every chord; there are 72 chords in a circle using the default chord angle.■

- XCENTER, YCENTER — Specifies the location (in absolute coordinates) of the center of the arc. (The center of the arc is the center of the circle that would be drawn if the arc were 360 degrees.)

  Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off. If current scaling is not isotropic, the arc drawn is elliptical rather than circular.

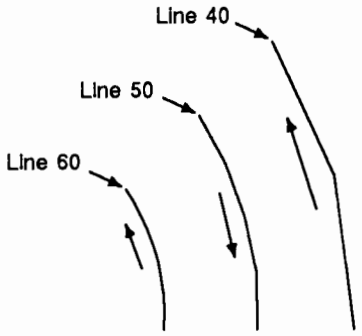
- **Sweep Angle** — Specifies the angle, in degrees, through which the arc is drawn. A positive angle draws counterclockwise from the current pen location; a negative angle draws clockwise.

- **Chord Angle** — Specifies, in degrees, the maximum angle to use in drawing the chords in the arc. The default chord angle is 5 degrees.

  NOTE: If you use the Chord Tolerance (CT) instruction and begin specifying chords in terms of deviation distance, this parameter is termed chord tolerance and is measured in current units. Refer to chapter 9, *The Technical Graphics Extension*, for complete information on chords and chord tolerance.■

EXAMPLE:

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS7000,5000"
30  PRINT #1, "PA2000,0SP1"
40  PRINT #1, "PDAA0,0,45,25"
50  PRINT #1, "PU1050,1060PDAA0,0,-45,10"
60  PRINT #1, "PU1000,0PDAA0,0,45"
70  PRINT #1, "PUSP0PG;"
80  END
```



Related Instructions	Group/Extension
AR, Arc Relative AT, Absolute Arc Three Point CI, Circle RT, Relative Arc Three Point	*The Vector Group*
LA, Line Attributes LT, Line Type PW, Pen Width	*The Line and Fill Attributes Group*
CT, Chord Tolerance Mode	*The Technical Graphics Extension*

AR, Arc Relative

USE: Draws an arc, using relative coordinates, that starts at the current pen location and pivots around a specified center point.

SYNTAX: AR*XINCREMENT,YINCREMENT,sweep angle(,chord angle;)*

Parameter	Format	Functional Range	Default
XINCREMENT,YINCREMENT	current units	*device-dependent*	no default
sweep angle	clamped real	±360°	no default
chord angle	current units	0.5° to 180°	5°

REMARKS: The AR instruction draws the arc starting at the current pen location using the current pen-up/down position and line type and attributes. After the arc is drawn, the pen location remains at the end of the arc.

NOTE: Do *not* use an adaptive line type when drawing arcs with small chord angles. The plotter will attempt to draw the complete pattern in every chord; there are 72 chords in a circle using the default chord angle.■

- XINCREMENT, YINCREMENT — Specifies the center of the arc relative to the current location. (The center of the arc is the center of the circle that would be drawn if the arc were 360 degrees.)

  Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off. If current scaling is not isotropic, the arc drawn is elliptical rather than circular.

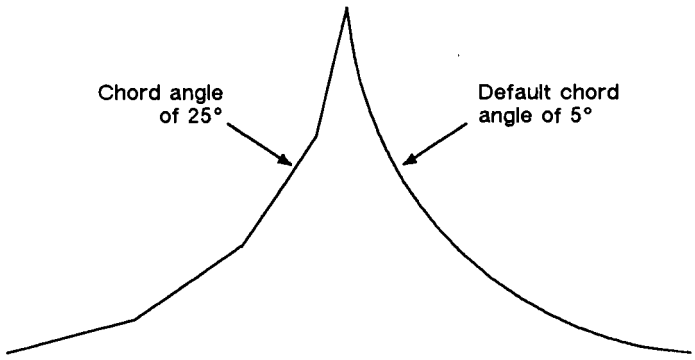
- **Sweep Angle** — Specifies the angle, in degrees, through which the arc is drawn. A positive angle draws counterclockwise from the current pen location; a negative angle draws clockwise.

- **Chord Angle** — Specifies, in degrees, the maximum angle to use in drawing the chords in the arc. The default chord angle is 5 degrees.

  NOTE: If you use the Chord Tolerance (CT) instruction and begin specifying chords in terms of deviation distance, this parameter is termed chord tolerance and is measured in current units. Refer to chapter 9, *The Technical Graphics Extension*, for complete information on chords and chord tolerance.■

EXAMPLE:

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
20  PRINT #1, "PS7000,5000SP1PA1500,1500PD"
30  PRINT #1, "AR0,2000,80,25AR2000,0,80"
40  PRINT #1, "PUSP0PG;"
50  END
```



Related Instructions	Group/Extension
AA, Arc Absolute AT, Absolute Arc Three Point CI, Circle RT, Relative Arc Three Point	*The Vector Group*
LA, Line Attributes LT, Line Type PW, Pen Width	*The Line and Fill Attributes Group*
CT, Chord Tolerance Mode	*The Technical Graphics Extension*

AT, Absolute Arc Three Point

USE: Draws an arc segment, using absolute coordinates, from a starting point through an intermediate point to an endpoint. Use AT when you know these three points of an arc.

SYNTAX: AT*XINTER,YINTER,XEND,YEND(,chord angle;)*

Parameter	Format	Functional Range	Default
XINTER, YINTER	current units	*device-dependent*	no default
XEND, YEND	current units	*device-dependent*	no default
chord angle	clamped real	0.5° to 180°	5°

REMARKS: The AT instruction uses the current pen location and two specified points to calculate a circle and draw the appropriate arc segment of its circumference. The arc starts at the current pen location, using the current pen, line type, line attributes, and pen-up/down position. You specify the intermediate and endpoints. After the arc is drawn, the pen location remains at the end of the arc.

NOTE: Do *not* use an adaptive line type when drawing arcs with small chord angles. The plotter will attempt to draw the complete pattern in every chord; there are 72 chords in a circle using the default chord angle.■

- XINTER,YINTER — Specifies the absolute location of an intermediate point of the arc. The arc is drawn clockwise or counterclockwise as necessary, so that it passes through the intermediate point before the endpoint.

- XEND,YEND — Specifies the absolute location of the endpoint of the arc.

- **Chord Angle** — Specifies, in degrees, the maximum angle to use in drawing the chords in the arc. The default chord angle is 5 degrees.

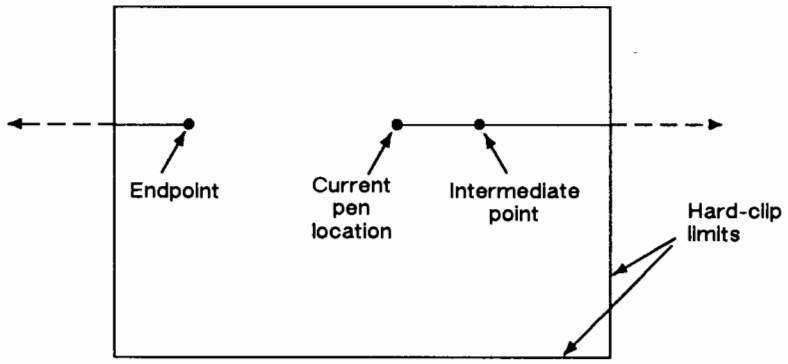
  NOTE: If you use the Chord Tolerance (CT) instruction and begin specifying chords in terms of deviation distance, this parameter is termed chord tolerance and is measured in current units. Refer to chapter 9, *The Technical Graphics Extension*, for complete information on chords and chord tolerance.■

Intermediate and endpoint coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off. If current scaling is not isotropic, the arc drawn is elliptical rather than circular. Note the following about locating the intermediate and endpoints.

- If the intermediate point and endpoint are the same as the current pen location, the plotter draws a dot.

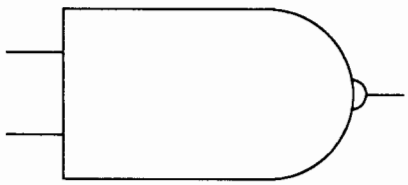
- If the intermediate point is the same as either the current pen location or the endpoint, a line is drawn between the current pen location and the endpoint.

- If the endpoint is the same as the current pen location, a circle is drawn, with its diameter being the line from the current pen position to the intermediate point.
- If the current pen position, intermediate point, and endpoint are collinear, a straight line is drawn.
- If the intermediate point does not lie between the current pen location and the endpoint, two lines are drawn to the edge of the hard-clip limits or window, one from the current pen location and the other from the endpoint, leaving a gap between them. Refer to the following illustration.



EXAMPLE:

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS7000,5000SP1PA1000,100PD2500,100"
40  PRINT #1, "PU650,1150PD1000,1150PU650,450PD1000,450"
50  PRINT #1, "PU1000,100PD1000,1500,2500,1500"
60  PRINT #1, "AT3200,800,2500,100PU3200,900PD"
60  PRINT #1, "AT3300,800,3200,700PU3300,800PD3500,800"
70  PRINT #1, "PUSP0PG;"
80  END
```



Related Instructions	Group/Extension
AA, Arc Absolute AR, Arc Relative CI, Circle RT, Relative Arc Three Point	*The Vector Group*
LA, Line Attributes LT, Line Type PW, Pen Width	*The Line and Fill Attributes Group*
CT, Chord Tolerance Mode	*The Technical Graphics Extension*

CI, Circle

USE: Draws the circumference of a circle using the specified radius and chord tolerance. If you want a filled circle, refer to the PM instruction.

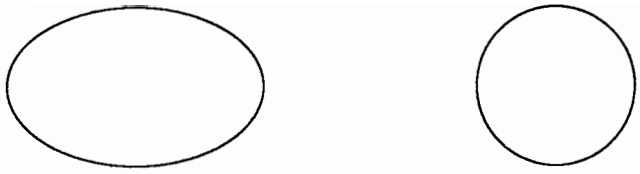
SYNTAX: CI*radius(,chord angle;)*

Parameter	Format	Functional Range	Default
radius	current units	*device-dependent*	no default
chord angle	clamped real	0.5° to 180°	5°

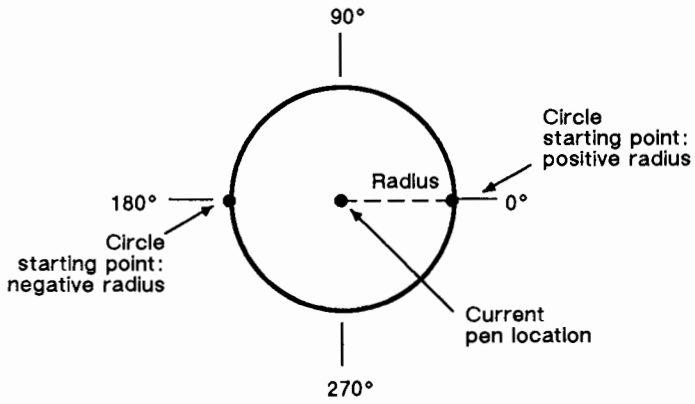
REMARKS: The CI instruction includes an automatic pen down. When a CI instruction is received, the pen lifts and moves from its current location (the center of the circle) to the starting point on the circumference, lowers the pen, draws the circle, then returns pen up to the center of the circle. After the circle is drawn, the previous pen-up/down position is restored. To avoid leaving a dot at the center of the circle, move to and from the circle's center with the pen up.

Each chord of the circle is drawn using the currently defined line type, width, and attributes. (Refer to chapter 7, *The Line and Fill Attributes Group*, for more information.) Do *not* use an adaptive (negative) line type to draw a circle as the plotter will attempt to draw a complete pattern for *every* chord (there are 72 chords in a circle using the default chord angle mode).

Always use isotropic scaling in plots that draw circles, unless you want your circles to "flex" with the plot. Anisotropic scaling may produce an ellipse. Refer to the discussion of scaling in chapter 2 and the Scale (SC) instruction description for more information.



- **Radius** — Specifies the distance from the current pen location (center of the circle) to the circumference of the circle. Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off. If you specify a negative radius, the circle begins and ends at the 180° reference point. The 0°/180° reference line is parallel to the X-axis.

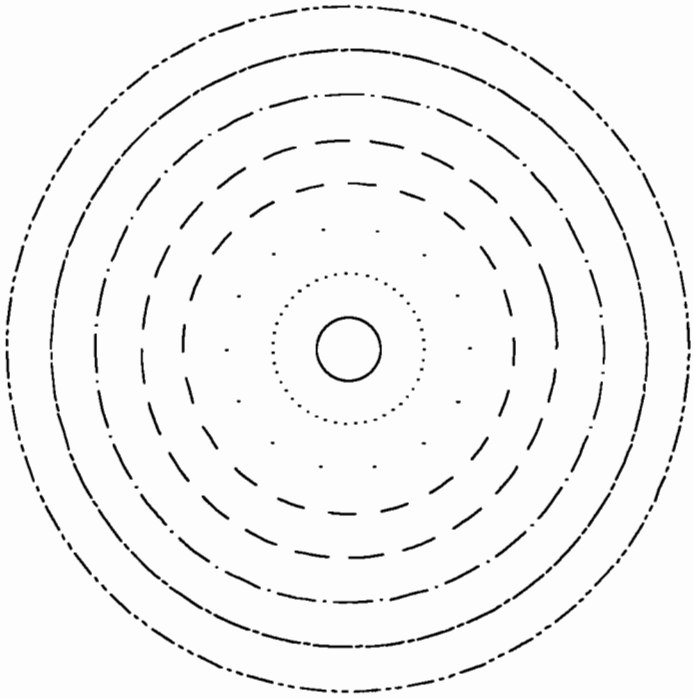


- **Chord Angle** — Specifies, in degrees, the maximum angle to use in drawing the chords in the circle. The default chord angle is 5 degrees.

  NOTE: If you use the Chord Tolerance (CT) instruction and begin specifying chords in terms of deviation distance, this parameter is termed chord tolerance and is measured in current units. Refer to chapter 9, *The Technical Graphics Extension*, for complete information on chords and chord tolerance.■

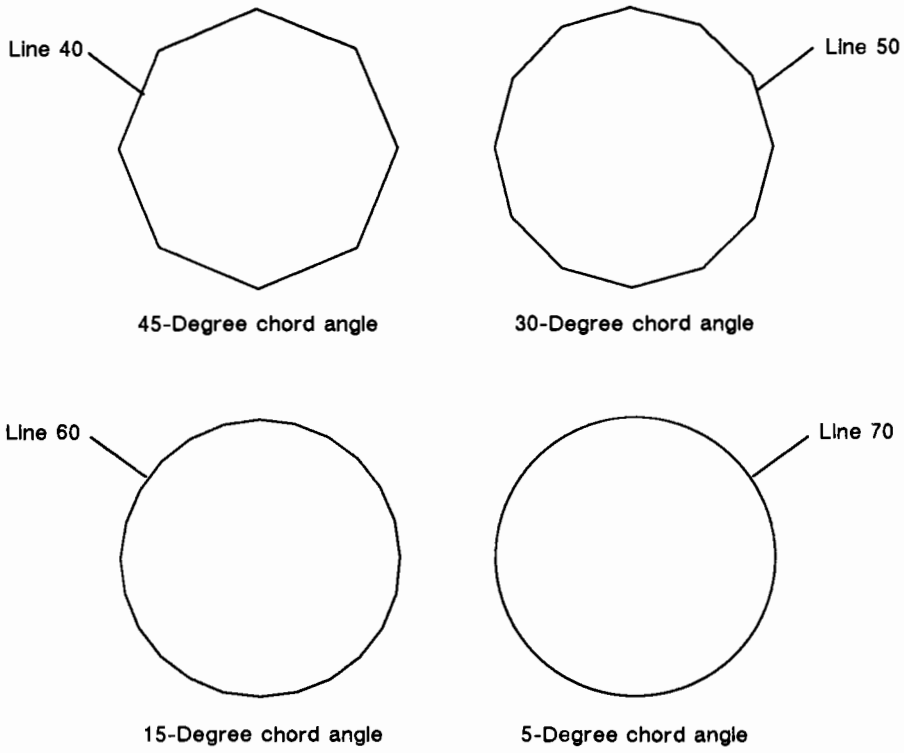
EXAMPLE: Drawing Circles with Different Radii and Line Types

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS7000,5000SP1"
30  PRINT #1, "SC-75,75,-75,75,1PA0,0"
40  PRINT #1, "LTCI5LT0CI-12LT1CI19"
50  PRINT #1, "LT2CI-26LT3CI33LT4CI-40"
60  PRINT #1, "LT5CI47LT6CI54"
70  PRINT #1, "PUSP0PG;"
80  END
```



EXAMPLE: Effects of Chord Angle on Circle Smoothness

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS7000,7000SP1SC-3000,3000,-2000,2000,1"
40  PRINT #1, "PA-1700,2000CI750,45"
50  PRINT #1, "PA300,2000CI750,30"
60  PRINT #1, "PA-1700,-200CI750,15"
70  PRINT #1, "PA300,-200CI750"
80  PRINT #1, "PUSP0PG;"
90  END
```



Related Instructions	Group/Extension
SC, Scale	*The Configuration/Status Group*
AA, Arc Absolute AR, Arc Relative RT, Relative Arc Three Point	*The Vector Group*
WG, Fill Wedge	*The Polygon Group*

(Continued)

Related Instructions	Group/Extension
LA, Line Attributes LT, Line Type PW, Pen Width	*The Line and Fill Attributes Group*
CT, Chord Tolerance Mode	*The Technical Graphics Extension*

PA, Plot Absolute

USE: Establishes absolute plotting and moves the pen to the specified absolute coordinates from the current pen location.

SYNTAX: PA *X,Y (,...;)*
or
PA*(;)*

Parameter	Format	Functional Range	Default
X,Y coordinates	current units	*device-dependent*	no default

REMARKS: Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off.

- **No Parameters** — Establishes absolute plotting for subsequent instructions.

- **X,Y Coordinates** — Specify the absolute location to which the pen moves. When you include more than one coordinate pair, the pen moves to each point in the order given, using the current pen-up/down position. If the pen is up, PA moves the pen to the point; if the pen is down, PA draws a line to the point. Lines are drawn using the current line width, type, and attributes.

  When you use the Symbol Mode (SM) instruction, PA draws the specified symbol at each X,Y coordinate. When you are in polygon mode (using the PM instruction), the X,Y coordinates enter the polygon buffer (and are used when the polygon is edged or filled). (Refer to chapter 6 for more information on polygon mode, to chapter 7 for more information on symbol mode.)

Related Instructions	Group
PE, Polyline Encoded PR, Plot Relative	*The Vector Group*

ERRORS:

Condition	Error	Plotter Response
odd number of coordinates	2	ignores last, unmatched coordinate

PD, Pen Down

USE: Lowers the plotter's physical or logical pen to the paper for drawing.

SYNTAX: PD *X,Y(,...;)*
or
PD*(;)*

Parameter	Format	Functional Range	Default
X,Y coordinates/ increments	current units	*device-dependent*	no default

REMARKS: Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off.

- **No Parameters** — Prepares plotter to draw to subsequent graphics instructions.

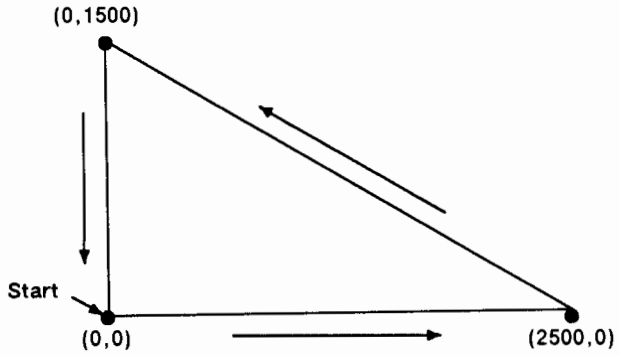
- **X,Y Coordinates/Increments** — Draws (in current units) to the point specified. You can specify as many X,Y coordinate pairs as you want. When you include more than one coordinate pair, the plotter draws to each point in the order given.

  Whether the PD instruction uses absolute movement (coordinates) or relative movement (increments) depends on the most recently executed PA or PR instruction. If you have not issued a PA or PR instruction, absolute plotting (PA) is used.

  When you use the Symbol Mode (SM) instruction, PD draws the specified symbol at each X,Y coordinate along with a line between each coordinate pair. When you are in polygon mode (using the PM instruction), the X,Y coordinates enter the polygon buffer (and are used when the polygon is edged or filled). (Refer to chapter 6 for more information on polygon mode, to chapter 7 for more information on symbol mode.)

EXAMPLE:

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS7000,5000SP1"
40  PRINT #1, "PA0,0PD2500,0,0,1500,0,0"
50  PRINT #1, "PUSP0PG;"
60  END
```



Related Instructions	Group
PA, Plot Absolute PE, Polyline Encoded PR, Plot Relative PU, Pen Up	*The Vector Group*

ERRORS:

Condition	Error	Plotter Response
odd number of coordinates	2	ignores last, unmatched coordinate

PE, Polyline Encoded

USE: Incorporates PA, PR, PU, PD, and SP instructions into an encrypted format that substantially decreases the size of your file and the time required for data transmission.

SYNTAX: PE*(flag)(value/X,Y)...(flag)(value/X,Y)*;
or
PE;

NOTE: Parameter values are self-terminating; *do not use commas* with this instruction. Also, you *must* use a semicolon to terminate PE.■

Parameter	Format	Functional Range	Default
flag	character	: , < , > , = , or 7	no default
value	character	*flag*-dependent*	no default
X,Y coordinates/ increments	character	*device-dependent*	no default

* Refer to the table following the parameter description.

REMARKS: Lines are drawn using the current line type and current units. The plotter draws to all points with the pen down unless a pen-up flag precedes the X,Y coordinates. If the final move is made with the pen up, the pen will remain in the up position; otherwise, the pen is left in the down position.

PE interprets coordinate pairs as relative coordinates unless they are preceded by an *absolute coordinates* value flag. Relative integer coordinates produce the most compact data stream. For best results, scale your plots so you use only integer coordinates and use relative plotting mode. After PE is executed, the previous plotting mode (absolute or relative) is restored and the carriage return point is updated.

The PE instruction represents vectors in base 64 (default) or base 32. In parameter value data, all spaces, delete characters, and control characters are ignored (regardless of their eighth bit).

- **No Parameters** — Updates the carriage return point (refer to chapter 8, *The Character Group*). PE without parameters does not affect the pen's current location or up/down position.

- **Flag** — Indicates how the plotter interprets subsequent values. Flags are ASCII characters. The plotter disregards the eighth bit of a flag, e.g., a character code of 61 and a character code of 189 both send an "=" (the absolute flag). *Flags are not encoded.*

Flag	Meaning	Description
:	Select Pen	Indicates that the subsequent value is encrypted to represent the desired pen number. A PE command without a pen-select flag uses the currently selected pen.
<	Pen Up	Raises the pen and moves to the subsequent coordinate pair value. (All coordinate pair values not preceded by a pen-up flag are considered pen-down moves.)*
>	Fractional Data	Indicates that the subsequent value specifies the number of fractional binary *bits* contained in the coordinate data. Default is zero.
=	Absolute	Indicates that the subsequent coordinate pair is defined by absolute coordinates.
7	7-bit Mode	Indicates that all subsequent coordinate pair values should be interpreted in 7-bit mode. Once you send a 7-bit flag, base 32 is used and the eighth bits are ignored for the remainder of the command.

* Because there is no implicit pen down within the PE instruction, we recommend you always follow a pen-up flag with a relative move of (0,0). This ensures that the next plotting coordinates will be drawn.

NOTE: Selecting a pen is not allowed when using PE in polygon mode. If you select a pen within PE while in polygon mode, the plotter ignores the rest of the command and sets error 1 (unrecognized instruction).■

- **Value** — Specifies data according to the preceding flag. For example, a value following a select-pen flag should be a pen number; values following an absolute flag should be coordinate pairs. *Flag values are encoded in the same manner as coordinate data.* Instructions for encoding flag values follow the parameter descriptions.

Value	Format	Range
pen number	integer	*device-dependent*
number of fractional binary bits	integer	*device-dependent*

  - **Pen Number** — Specifies the pen to be selected. The pen number must be encoded into a base 64 or base 32 equivalent. If your device supports the palette extension, refer to chapter 10 for more information on the Number of Pens (NP) instruction.

  - **Number of Fractional Binary Bits** — Specifies the number of fractional binary bits contained in the coordinate data. The number of fractional binary bits must be encoded into a base 64 or base 32 equivalent.

- **X,Y Coordinates** — Specifies a coordinate pair encoded into a base 64 (default) or a base 32 equivalent. Use base 64 if your system can send 8 bits of data without parity. Use 7-bit mode and base 32 coordinate values if your system requires a parity bit.

  If the current pen position goes out of this range, the plotter ignores plotting instructions until it receives an absolute PA or PE coordinate within the extended ranges. (Your device may allow extended ranges for relative movement beyond the minimum *integer* range described in this manual. Check your device's documentation for the PR and PE instructions.)

  When you are in symbol mode (refer to the SM instruction in chapter 7, *The Line and Fill Attributes Group*), PE draws the specified symbol at each X,Y coordinate. When you are in polygon mode (refer to the PM instruction in chapter 6, *The Polygon Group*), the X,Y coordinates enter the polygon buffer (are not drawn).

Encoding PE Values

Values are encoded into a base 64 (default) or base 32 equivalent (7-bit mode). Use base 64 if your system can send 8 bits of data without parity. Use 7-bit mode and base 32 coordinate values if your system requires a parity bit. *Do not encode PE flags.*

The following steps give a generic algorithm for encoding a number. Assume x is the number to be encoded. Use steps 1 and 2 only if you are encoding fractional data; otherwise, begin with step 3.

1. **Fraction adjustment.** If you are using fractional data, this step converts the number of decimal places in your data to the number of binary fractional bits. Assume n is the number of fractional binary bits specified by the fractional data flag.

   a. Multiply the number of decimal places contained in the data by 3.33.

   b. Round that number up to the next integer to get integer n.

   $n = \text{round (decimal places} \times 3.33)$

   $x = x \times 2^n$

2. **Round to an integer.** Round the results of step 1 to the nearest integer.

   $x = \text{round } (x)$

3. **Set the sign bit.** If x is positive, multiply it by two.

   if $(x \geq 0)$
   $x = 2 \times x$

   If x is negative, multiply the absolute value of x by 2 and add 1.

   if $(x < 0)$
   $x = 2 \times \text{abs}(x) + 1$

4. **Convert the number to base 64 or 32 and encode the data.** Convert x to a base 64 number if your system sends 8 bits without parity. Convert x to a base 32 number if your system sends 7 bits with parity (7-flag is sent).

   Encode each base 64 or 32 digit into the ASCII character range, as described below. Output each character as it is encoded, starting with the least significant digit. The most significant digit is used to terminate the number and is encoded into a different ASCII character range than the low-order digits.

   Each number in a coordinate pair is represented as zero or more nonterminator characters, followed by a terminator character. A character is a nonterminator or terminator depending on the range it is in; refer to the following table. For example, in base 64 there are 64 nonterminator and 64 terminator characters. Either kind represents a "digit."

Range Type	Nonterminator	Terminator
8-bit Range (base 64)	63–126	191–254
7-bit Range (base 32)	63–94	95–126

   Values following the fractional data or select-pen flag must also be encoded.

   Base 64. Encode all the low-order bits into the ASCII range 63 to 126. For a digit with value i, use ASCII character CHR$(63 + i). Encode the highest-order digit (or the single digit in a one-digit number) into the range 191 to 254.

   Base 32. Encode all the low-order bits into the ASCII range 63 to 94. For a digit with value i, use ASCII character CHR$(63 + i). Encode the highest-order digit (or the single digit in a one-digit number) into the range 95 to 126.

```
while n ≥ base
        output CHR$(63 + (n MOD base))
        n = n DIV base
    end
    if base = 64 then n = 191 + n
    if base = 32 then n = 95 + n
    output CHR$(n)
```

Programming Considerations

In order to use relative pen moves to draw to known absolute coordinates, you must know your initial location. The Label (LB) instruction, for example, updates the pen location to the end of the label, but you don't know how many units (plotter units or user units) the pen has moved. Since you cannot use output instructions in PE, your first coordinate pair following a label should be to absolute coordinates, not to a relative location. If this presents a problem in your program, take the following steps.

1. Create a flag called lost in your program.
2. After labeling (or any instruction which updates the current pen location), set lost to true.
3. If lost = true at the beginning of the PE instruction, use an absolute flag for the first coordinate pair only (subsequent coordinates are interpreted as relative).
4. Set lost to false.

NOTE: At the beginning of your application program, set lost to true. Then specify the next coordinate in absolute mode (PA or PE=).■

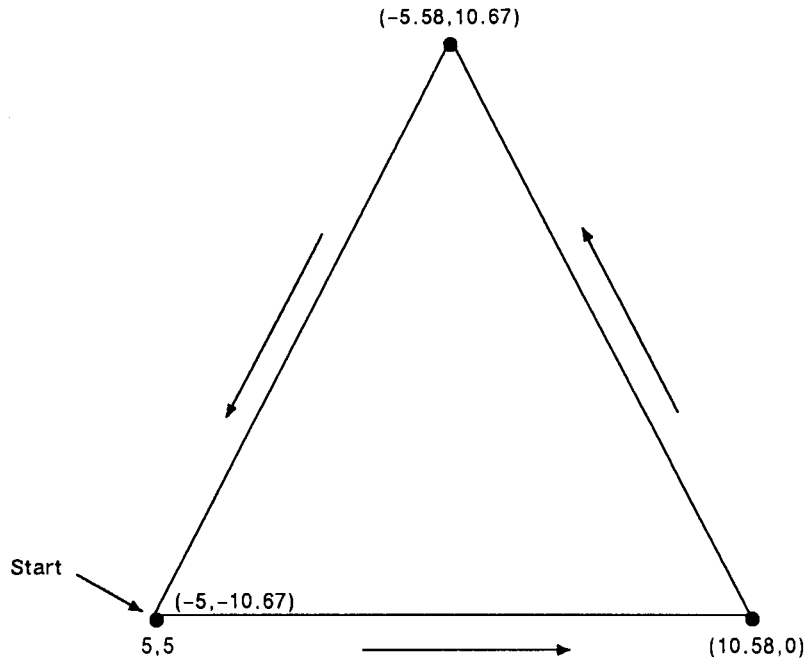
When converting and encoding data, note the following.

- n DIV 64 = n.shift right.6 bits. You can optimize your driver by shifting 6 bits to the right since shifting is faster than division.
- n MOD 64 = n.AND.63. The number is logically AND'd with 63.

EXAMPLE: The following program converts three relative real coordinate pairs to base 64.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS7000,5000SC1,20,1,20,1SP1PU5,5"
40  PRINT "Input number of fractional decimal places in data"
50  INPUT F
60  'calculate number of fractional binary bits
70  F = F * 3.33
80  F = INT(F)
90  A = F
100 IF F >= 0 THEN F = 2*F ELSE F = 2*ABS(F)+1
110 F = 191+F
120 PRINT #1, "PE>"+CHR$(F)
130 'convert coordinate data to base 64
140 FOR J = 1 to 6
150   READ C
160   C = C * (2^A)
170 C = INT(C)
180 IF C >= 0 THEN C = 2*C ELSE C = 2*ABS(C)+1
```

```
190 WHILE C >= 64
200 PRINT #1, CHR$(63+(C MOD 64))
210   C = C\64
220   WEND
230   C = 191+C
240   PRINT #1, CHR$(C)
250 NEXT J
260 PRINT #1, ";"
270 PRINT #1 "PUSP0PG;"
280 DATA 10.58,0,-5.58,10.67,-5,-10.67
290 END
```



Related Instructions	Group/Extension
PA, Plot Absolute PD, Pen Down PR, Plot Relative PU, Pen Up	*The Vector Group*
SP, Select Pen	*The Line and Fill Attributes Group*

ERRORS:

Condition	Error	Plotter Response
coordinate evaluates to a –0	3	ignores rest of instruction
entering ";" when a *y* value, pen number, or number of fractional bits was expected	2	terminates instruction and looks for next valid HP-GL/2 mnemonic

PR, Plot Relative

USE: Establishes relative plotting and moves the pen to specified points with each successive move relative to the current pen location.

SYNTAX: PR *X,Y(,...;)*
or
PR*(;)*

Parameter	Format	Functional Range	Default
X,Y increments	current units	*device-dependent**	no default

* Your device may allow extended ranges for PR and PE beyond the minimum *integer* range described in this manual. Check your device's documentation for the PR and PE instructions. If the current pen position goes out of this range, the plotter ignores plotting instructions until it receives an absolute PA or PE coordinate within the extended ranges.

REMARKS: The plotter interprets the parameters as follows.

- **No Parameters** — Defaults plotting mode to relative for subsequent instructions.

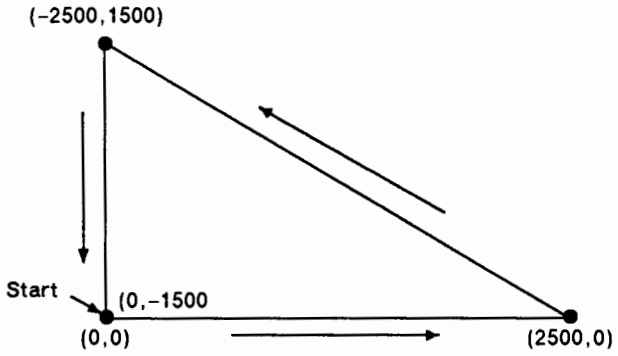
- **X, Y (Increments)** — Specify incremental moves relative to the current pen location. When you include more than one increment pair, the pen moves to each point in the order given (relative to the previous point), using the current pen-up/down position. If the pen is up, PR moves the pen to the point; if the pen is down, PR draws a line to the point. Lines are drawn using the current line width, type, and attributes.

  When you use the Symbol Mode (SM) instruction, PR draws the specified symbol at each X,Y coordinate along with a line between each coordinate pair. When you are in polygon mode (using the PM instruction), the X,Y coordinates enter the polygon buffer (and are used when the polygon is edged or filled). (Refer to chapter 6 for more information on polygon mode, to chapter 7 for more information on symbol mode.)

  Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off.

EXAMPLE:

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS7000,5000SP1"
40  PRINT #1, "PA0,0PDPR2500,0,-2500,1500,0,-1500"
50  PRINT #1, "PUSP0PG;"
60  END
```



Related Instructions	Group
PA, Plot Absolute PD, Pen Down PE, Polyline Encoded PU, Pen Up	*The Vector Group*

ERRORS:

Condition	Error	Plotter Response
odd number of coordinates	2	ignores last coordinate
number out of range	3	ignores wrong coordinate and any subsequent coordinates

PU, Pen Up

USE: Lifts the pen and moves to specified points without drawing.

SYNTAX: PU*X,Y(,...;)*
or
PU(;)

Parameter	Format	Functional Range	Default
X,Y coordinates/ increments	current units	*device-dependent*	no default

REMARKS: Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off.

- **No Parameters** — Prevents drawing subsequent graphics instructions (unless the instruction contains an automatic pen down).

- **X, Y Coordinates/Increments** — Move to the point(s) specified. You can specify as many X,Y coordinate pairs as you want. When you include more than one coordinate pair, the plotter moves to each point in the order given.

  When you use the Symbol Mode (SM) instruction, PU draws the specified symbol at each X,Y coordinate along with a line between each coordinate pair. When you are in polygon mode (using the PM instruction), the X,Y coordinates enter the polygon buffer (and are used when the polygon is edged or filled). (Refer to chapter 6 for more information on polygon mode, to chapter 7 for more information on symbol mode.)

  Whether the PU instruction uses coordinates or increments depends on the most recently executed PA or PR instruction. If you have not issued a PA or PR instruction, absolute plotting (PA) is used.

Related Instructions	Group
PA, Plot Absolute PD, Pen Down PE, Polyline Encoded PR, Plot Relative	*The Vector Group*

ERRORS:

Condition	Error	Plotter Response
odd number of coordinates	2	ignores last coordinate
number out of range	3	ignores wrong coordinate and any subsequent coordinates

RT, Relative Arc Three Point

USE: Draws an arc segment, using relative coordinates, from a starting point through an intermediate point to an endpoint. Use RT when you know these three points of an arc.

SYNTAX: RT*X*INCR INTER,*Y*INCR INTER,*X*INCR END,*Y*INCR END(,*chord angle*;)

Parameter	Format	Functional Range	Default
XINCR INTER, YINCR INTER	current units	*device-dependent*	no default
XINCR END, YINCR END	current units	*device-dependent*	no default
chord angle	clamped real	0.5° to 180°	5°

REMARKS: The RT instruction uses the current pen location and two specified points to calculate a circle and draw the appropriate arc segment of its circumference. The arc starts at the current pen location, using the current pen, line type, line attributes, and pen-up/down position. You specify the intermediate and endpoints. After drawing the arc, the pen location remains at the end of the arc.

NOTE: Do *not* use an adaptive line type when drawing arcs with small chord angles. The plotter will attempt to draw the complete pattern in every chord; there are 72 chords in a circle using the default chord angle.■

- XINCR INTER, YINCR INTER — Specify the location (in relative increments from the current pen location) of the intermediate point of the arc. The arc is drawn clockwise or counterclockwise, as necessary, so that it passes through the intermediate point before the endpoint.

- XINCR END,YINCR END — Specify the location (in relative increments from the current pen location) of the endpoint of the arc.

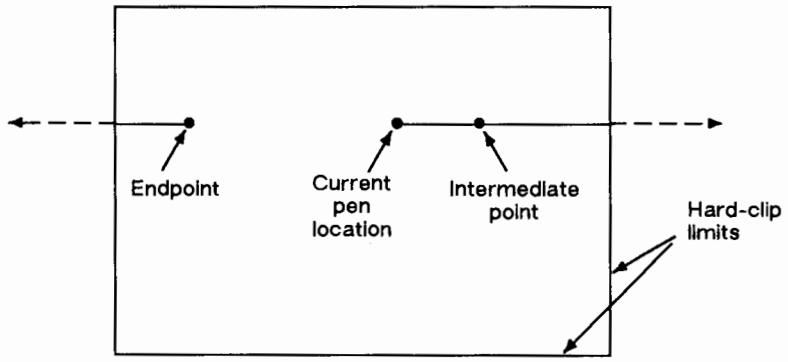
- **Chord Angle** — Specifies, in degrees, the maximum angle to use in drawing the chords in the arc. The default chord angle is 5 degrees.

  NOTE: If you use the Chord Tolerance (CT) instruction and begin specifying chords in terms of deviation distance, this parameter is termed chord tolerance and is measured in current units. Refer to chapter 9, *The Technical Graphics Extension*, for complete information on chords and chord tolerance.■

Intermediate and endpoint coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off. If current scaling is not isotropic, the arc drawn is elliptical rather than circular. Note the following about intermediate and endpoints.

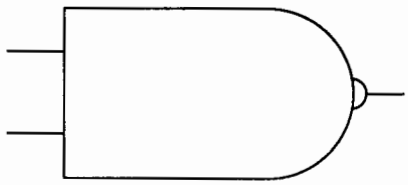
- If the intermediate point and endpoint are the same as the current pen location, the plotter draws a dot.

- If the intermediate point is the same as either the current pen location or the endpoint, a line is drawn between the current pen location and the endpoint.
- If the endpoint is the same as the current pen location, a circle is drawn, with its diameter being the line from the current pen position to the intermediate point.
- If the current pen position, intermediate point, and endpoint are collinear, a straight line is drawn.
- If the intermediate point does not lie between the current pen location and the endpoint, two lines are drawn to the hard-clip limits or the current window, one from the current pen location and the other from the endpoint, leaving a gap between them. Refer to the following illustration.



EXAMPLE:

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "INPS7000,5000SP1PA1000,100PRPD1500,0"
40  PRINT #1, "PU-1850,1050PD350,0PU-350,-700PD350,0"
50  PRINT #1, "PU0,-350PD0,1500PD1500,0RT700,-750,0,-1500"
60  PRINT #1, "PU700,850PDRT100,-100,0,-200PU100,100PD200,0"
70  PRINT #1, "PUSP0PG;"
80  END
```



Related Instructions	Group/Extension
AA, Arc Absolute AR, Arc Relative AT, Absolute Arc Three Point	*The Vector Group*
LA, Line Attributes LT, Line Type PW, Pen Width	*The Line and Fill Attributes Group*
CT, Chord Tolerance Mode	*The Technical Graphics Extension*

6

The Polygon Group

All of the instructions in this group use the polygon buffer in your plotter. Some of the instructions define and draw complete shapes while others act only on the contents of the polygon buffer. The information in this chapter enables you to achieve the following.

- Draw circles, wedges, and rectangles.
- Use polygon mode for drawing polygons, subpolygons, and circles.
- Approximate buffer use.

The following instructions are described in this chapter.

Instruction	Summary
EA, Edge Rectangle Absolute	Outlines a rectangle defined with absolute coordinates.
EP, Edge Polygon	Outlines the contents of the polygon buffer.
ER, Edge Rectangle Relative	Outlines a rectangle defined with relative coordinates.
EW, Edge Wedge	Defines and outlines a wedge-shaped polygon.
FP, Fill Polygon	Fills the contents of the polygon buffer.
PM, Polygon Mode	Allows you to create user-defined polygons in the polygon buffer.
RA, Fill Rectangle Absolute	Fills a rectangle specified with absolute coordinates.
RR, Fill Rectangle Relative	Fills a rectangle specified with relative coordinates.
WG, Fill Wedge	Defines and fills a wedge-shaped polygon.

The following is a summary of the parameter formats and their *minimum* ranges. Your device may support a greater range than listed here.

Parameter Format	Minimum Ranges
Integer	2^{23} to $2^{23}-1$ (-8 388 608 to 8 388 607)
Real	2^{23} to $2^{23}-1$ (-8 388 608.000 0 to 8 388 607.999 9)
Clamped Integer	2^{15} to $2^{15}-1$ (-32 768 to 32 767)
Clamped Real	2^{15} to $2^{15}-1$ (-32 768.000 0 to 32 767.999 9)

Using the Polygon Buffer

A buffer is a temporary storage area for information. The polygon buffer collects the instructions and coordinates that define your polygon. This polygon remains in the buffer until replaced by another polygon, or until the buffer is cleared by initializing the plotter. The following instructions use the polygon buffer (but do not require you to enter polygon mode first).

EA, Edge Rectangle Absolute
ER, Edge Rectangle Relative
EW, Edge Wedge
PM, Polygon Mode
RA, Fill Rectangle Absolute
RR, Fill Rectangle Relative
WG, Fill Wedge

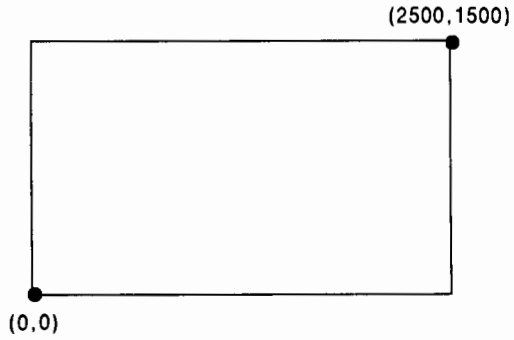
Check your device's documentation for the amount of polygon buffer space you have available. Your documentation will also tell you whether or not you can configure the buffers in your device and how this is done.

Drawing Rectangles

You can draw a rectangle by outlining (edging) the defined area, by filling the rectangular area, or by using a combination of both. To draw a rectangle, the plotter uses the current pen location for one corner; you give the coordinates for the diagonally opposite corner. The plotter draws the rectangle defined by these two points.

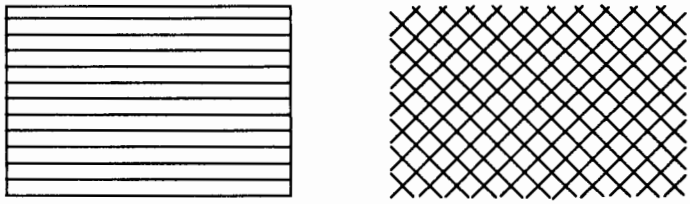
Outline a rectangle using the Edge Rectangle Absolute (EA) or Edge Rectangle Relative (ER) instructions. The following simple program uses EA to draw a rectangle.

```
10 'Insert configuration statement here
20 PRINT #1, CHR$(27)+"%-1BBPIN"
30 PRINT #1, "PS7000,5000SP1PA0,0"
40 PRINT #1, "EA2500,1500"
50 PRINT #1, "PUSP0PG;"
60 END
```



Fill a rectangular area using the Fill Rectangle Absolute (RA) or the Fill Rectangle Relative (RR) instructions. Both produce filled rectangles using the default or current fill pattern (refer to chapter 7, *The Line and Fill Attributes Group*, for more information on fill types). When you use an open fill type, you may also want to edge (or outline) the rectangle for better definition. The following program draws two filled rectangles; the first rectangle is filled and edged using relative increments, the second rectangle is filled using absolute coordinates and is not edged.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS7000,5000SP1PA0,0"
30  PRINT #1, "FT3RR1500,1000ER1500,1000"
40  PRINT #1, "PR2000,0FT4,100,45RA3500,1000"
50  PRINT #1, "PUSP0PG;"
60  END
```

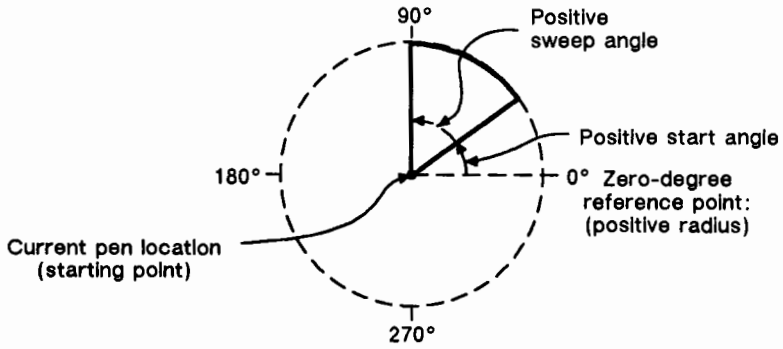


NOTE: Because rectangles are defined in the polygon buffer, you can easily edge filled rectangles using the Edge Polygon (EP) instruction.■

Drawing Wedges

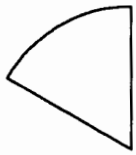
A wedge is a section of a circle. Wedges are commonly used to draw pie charts. You can draw a wedge by outlining (edging) the defined area using the Edge Wedge (EW) instruction, or you can create shaded (filled) wedges using the Fill Wedge (WG) instruction.

The wedge instructions use your current pen location as the center point; you specify the radius, the start angle, and the sweep angle. The radius determines the length of the two sides of the wedge. The sign (positive or negative) of the radius determines the location of a "zero-degree" reference point. The start angle is the number of degrees from the zero-degree reference point at which you want to draw the first radius. The sweep angle is the number of degrees through which you want to draw the arc. To draw or fill a circle, simply specify a 360-degree sweep angle. The following illustration shows the different parameters of a wedge with a positive radius.



The following simple program draws a wedge using the EW instruction. The radius of the wedge is 600 plotter units; the wedge begins 90° from the zero-degree reference point and "sweeps" for 60°.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS7000,5000"
30  PRINT #1, "SP1PA2500,3500"
40  PRINT #1, "EW600,90,60"
50  PRINT #1, "PUSP0PG;"
60  END
```



The following program uses different fill types with wedges and circles.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS7000,5000SP1"
30  PRINT #1, "PA1400,2500WG600,150,120"
40  PRINT #1, "PA2300,2500FT3,75,45WG600,90,180"
50  PRINT #1, "FT1,0,0WG600,270,60"
60  PRINT #1, "FT4,60,45WG600,330,120"
70  PRINT #1, "PA3500,2500WG400,0,360"
80  PRINT #1, "PUSP0PG;"
90  PRINT #1, ""
100 END
```

Drawing Polygons

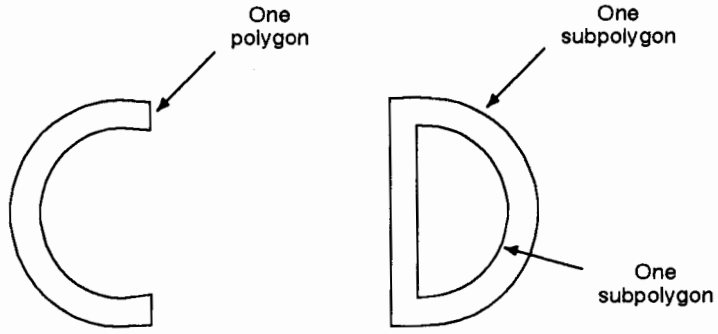
A polygon is one or more closed sequences of connected line segments (which may cross each other). Drawing polygons requires the use of *polygon mode*. The Polygon Mode (PM) instruction tells the plotter to store subsequent instructions and coordinates in the polygon buffer before plotting the shape. (Rectangles and wedges are polygons that have their own drawing instructions—the plotter automatically generates and stores their coordinates in the polygon buffer.)

You can use the following instructions in polygon mode to create polygons. These instructions are stored in the polygon buffer until they are replaced with another polygon or the plotter is initialized.

Polygon Definition Instructions	Group/Extension
AA, Arc Absolute AR, Arc Relative AT, Absolute Arc Three Point CI, Circle PA, Plot Absolute PD, Pen Down PE, Polyline Encoded PR, Plot Relative PU, Pen Up RT, Relative Arc Three Point	*The Vector Group*
PM1/PM2, Polygon Mode	*The Polygon Group*
BP, Begin Plot	*The Technical Graphics Extension*

Drawing Subpolygons

While in polygon mode, you can define either one polygon or a series of subpolygons. Like a polygon, a subpolygon is a closed sequence of connected line segments. For example, the block letter C is one complete polygon. However, the block letter D is actually two subpolygons: the outline and the "hole."



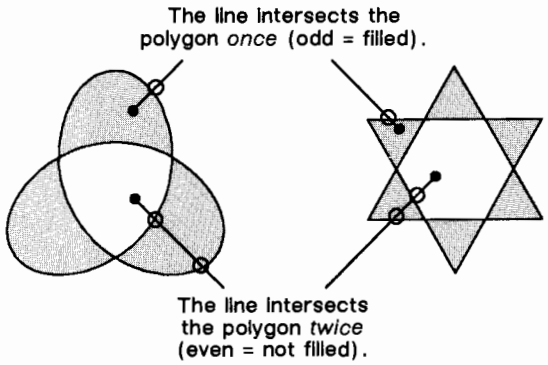
To create one polygon (e.g., the letter C), move the pen to the starting location for the polygon, then use the Polygon Mode (PM) instruction to enter polygon mode. Define the shape of the C using the appropriate instructions and coordinates, then exit polygon mode. Now draw the polygon using either the Edge Polygon (EP) or Fill Polygon (FP) instruction.

To create a series of subpolygons (e.g., the letter D), move the pen to the starting location of the first subpolygon, then enter polygon mode. Define the outer shape of the letter D using the appropriate instructions and coordinates, then close that subpolygon. Define the inner shape of the D, then exit polygon mode. Now draw the subpolygons using either the Edge Polygon (EP) or Fill Polygon (FP) instruction. Note that the line connecting the two subpolygons is not drawn or used as a fill boundary (see the next section).

For more information on entering and exiting polygon mode, refer to the Polygon Mode (PM) instruction near the end of this chapter. Note that you can define points with the pen up or down. However, the EP instruction draws only between points that were defined when the pen was down. The FP instruction fills between all points, regardless of whether they were defined when the pen was up or down.

Filling Polygons

There is a simple rule for determining what portion of a single polygon or series of subpolygons will be filled when you send a Fill Polygon (FP) instruction: FP fills an area if a line of infinite length, starting at any point in the area, intersects the polygon an odd number of times. An example of this "odd-even" rule follows.



Single polygon *Two subpolygons*

Drawing Circles in Polygon Mode

Polygon mode interprets the Circle (CI) instruction differently from the other HP-GL/2 instructions. The plotter treats a circle as a complete subpolygon. The plotter automatically closes the first polygon (if any) before starting the circle, and uses the first coordinates (if any) after the circle is drawn to start a new subpolygon.

If you have not completely closed your first polygon before sending the CI instruction, the plotter automatically closes the polygon by adding a point at the same location as the first coordinates in the subpolygon. This can change your current pen location and the placement of the circle in your polygon and result in an inaccurate polygon.

NOTE: In polygon mode, the smaller a circle's chord tolerance, the more chords will be stored in the polygon buffer to draw it.■

Approximating Polygon Buffer Use

You can use the following formula to estimate how much buffer space a polygon consumes. Each point in a polygon uses 8 bytes. For example, assume your plotter has 16 900 bytes available in its polygon buffer. If you divide 16 900 by 8, you get 2112.5. If you allow some extra for subpolygon closures and pen-up/down position, you can estimate that a polygon with up to 1500 points easily fits in the polygon buffer.

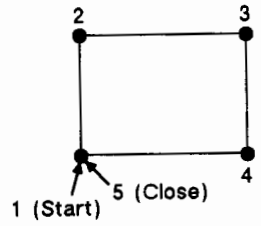
The following explains how to count the number of points in a polygon.

$$\#\text{ of points in polygon} \times 8 = \text{buffer space consumed by polygon}$$

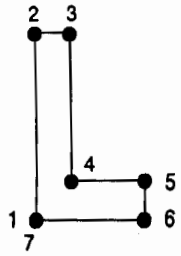
(If your program supports downloadable characters, refer to the Download Character [DL] instruction in chapter 9, *The Technical Graphics Extension*, and your device's documentation to estimate how many bytes are being used by the downloadable character buffer.)

Counting the Points in a Polygon

The starting pen location and each subsequent point define a polygon. As shown in the following illustration, a rectangle is defined by five points, not four. This is because the starting location is counted again as the ending location.



The following shape has seven points.



Counting the Points in a Circle or Arc

When a circle or arc defines a polygon, the number of points depends on the number of chords in the arc. When you are using chord angles to determine chord tolerance, use the following formula to determine the number of points used to draw a circle or arc.

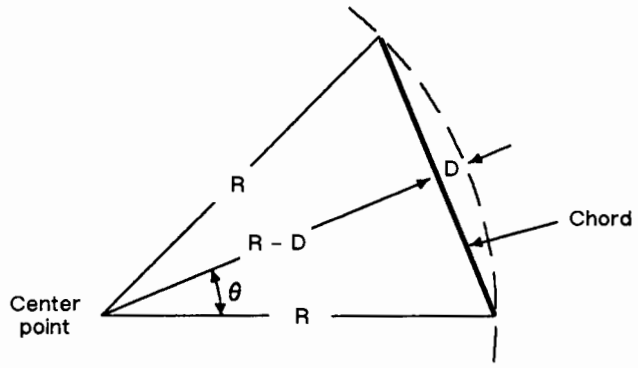
$$\#\text{ of points} = \frac{\text{arc angle (degrees)}}{\text{chord angle (degrees)}} + 1$$

Using this formula, a full circle with the default chord angle of 5° consists of 73 points (360/5 + 1 = 73), and a 45° arc with a chord angle of 3° consists of 16 points (45/3 + 1 = 16).

NOTE: If the chord angle does not divide evenly into the arc, round up to the next integer before adding 1: 45/2 + 1 = 23 + 1 = 24.■

When using deviation distance rather than degrees, the calculation changes. Use the following procedures to convert your deviation distance information to a chord angle, then use the formula on the previous page.

When you draw a circle or arc using a specific deviation distance, two things are known: the radius (R) and the deviation distance (D). When you bisect a chord, the distance to the chord is the radius minus the deviation distance (R - D). Refer to the following illustration.



The angle θ is determined by the following equation.

$$\theta = \text{ARCCOS}((R - D)/R)$$

Because you want the angle that created the whole chord, not the bisected chord, the chord angle, then, is twice θ. When you have the chord angle, use the chord angle formula to determine the number of points in the circle.

For example, consider a circle where the radius is 1000 and the deviation distance is 20: the angle θ equals ARCCOS (1000 - 20/1000) or 11.5 degrees. The chord angle that created the whole chord is 23 degrees (11.5 × 2). Using the chord angle formula above, the number of points is then (360/23) + 1 = 16 + 1 = 17.

EA, Edge Rectangle Absolute

USE: Defines and outlines a rectangle using absolute coordinates.

SYNTAX: EA*X,Y(;)*

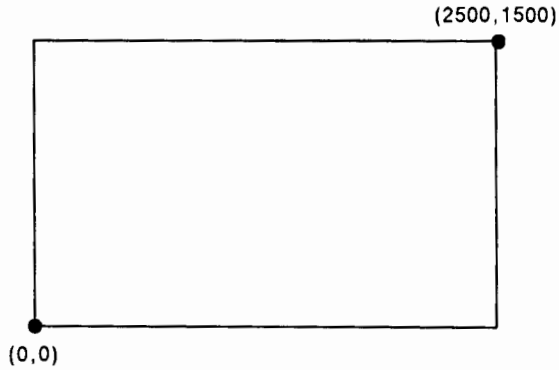
Parameter	Format	Functional Range	Default
X,Y coordinates	current units	*device-dependent*	no default

REMARKS: The EA instruction defines and edges a rectangle using the current pen, line type, line attributes, and absolute plotting. The edges of the rectangle will be parallel to the coordinate axes. The EA instruction includes an automatic pen down. When the instruction is completed, the original pen location and up/down position are restored.

- **X,Y Coordinates —** Specify the opposite corner of the rectangle from the current pen location. The current pen location is the starting point of the rectangle. Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off.

  If either the X- or Y-coordinate is the same as the respective coordinate of the current location, the plotter draws a line.

  NOTE: The following illustration shows the current pen location in the lower-left corner and the instruction's X,Y coordinates in the upper-right corner. However, these points can be in any diagonally opposite corners.■

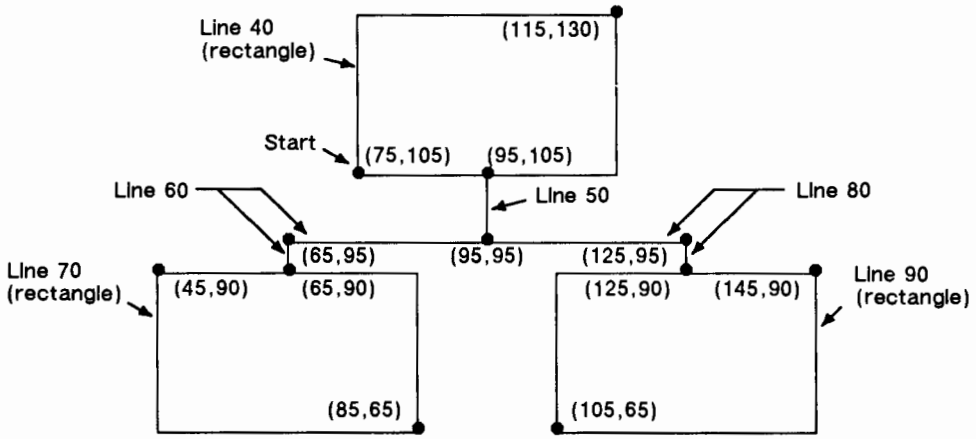


The only difference between the EA instruction and the RA instruction is that the EA instruction produces an outlined rectangle; the RA, a filled one.

The EA instruction clears the polygon buffer and then stores the defined rectangle in the polygon buffer before drawing. Refer to *Drawing Polygons* earlier in this chapter for more information.

EXAMPLE: This example uses absolute coordinates to draw the same plot shown with the ER instruction. Compare this program with the ER example program to understand the differences between the coordinates used.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS6000,8000SC0,150,0,150,1SP1"
40  PRINT #1, "PA75,105EA115,130"
50  PRINT #1, "PA95,105PD95,95"
60  PRINT #1, "PD65,95,65,90"
70  PRINT #1, "PU45,90EA85,65"
80  PRINT #1, "PU95,95PD125,95,125,90"
90  PRINT #1, "PU145,90EA105,65"
100 PRINT #1, "PUSP0PG;"
100 END
```



Related Instructions	Group
EP, Edge Polygon ER, Edge Rectangle Relative FP, Fill Polygon RA, Fill Rectangle Absolute RR, Fill Rectangle Relative	*The Polygon Group*

EP, Edge Polygon

USE: Outlines the polygon currently stored in the polygon buffer. Use EP to edge polygons that you defined in polygon mode and with the fill rectangle and wedge instructions (RA, RR, and WG).

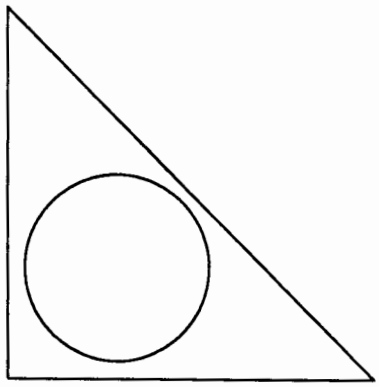
SYNTAX: EP(;)

REMARKS: The EP instruction outlines any polygon that is currently in the polygon buffer. This includes wedges and rectangles defined using the WG, RA, and RR instructions. EP accesses the data in the polygon buffer, but does *not* clear the buffer or change the data in any way.

The EP instruction edges only between points that were defined with the pen down, using the current pen, line type, and attributes. When the instruction is completed, the original pen location and up/down position are restored.

EXAMPLE: The following creates a shape in polygon mode, then uses EP to outline it.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS7000,5000SP1PA2000,0"
40  PRINT #1, "PM0PD0,2000,0,0,2000,0PM1"
50  PRINT #1, "PU600,600CI500PM2EP"
60  PRINT #1, "PUSP0PG;"
70  END
```



Related Instructions	Group/Extension
EA, Edge Rectangle Absolute ER, Edge Rectangle Relative EW, Edge Wedge PM, Polygon Mode	*The Polygon Group*
LA, Line Attributes LT, Line Type PW, Pen Width	*The Line and Fill Attributes Group*

ER, Edge Rectangle Relative

USE: Defines and outlines a rectangle using relative coordinates.

SYNTAX: ER*X,Y(;)*

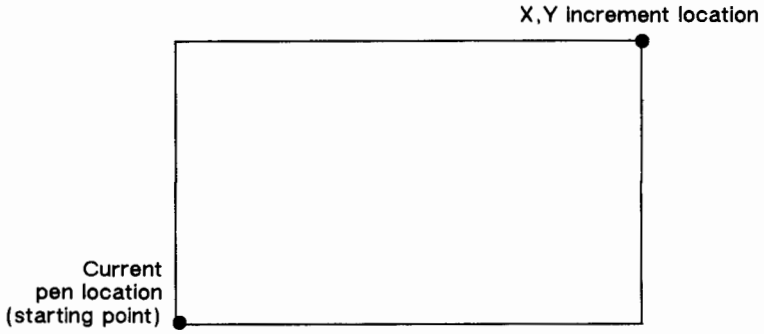
Parameter	Format	Functional Range	Default
X,Y increments	current units	*device-dependent*	no default

REMARKS: The ER instruction defines and edges a rectangle using the current pen, line type, line attributes, and relative coordinates. The edges of the rectangle will be parallel to the coordinate axes. The ER instruction includes an automatic pen down. When the instruction is completed, the original pen location and up/down position are restored.

- **X,Y Increments** — Specify the opposite corner of the rectangle from the current pen location. The current pen location is the starting point of the rectangle. Increments are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off.

  If either the X- or Y-increment is zero, the plotter draws a line.

  NOTE: The following illustration shows the current pen location in the lower-left corner and the instruction's X,Y increments in the upper-right corner. However, these points can be in any diagonally opposite corners.■

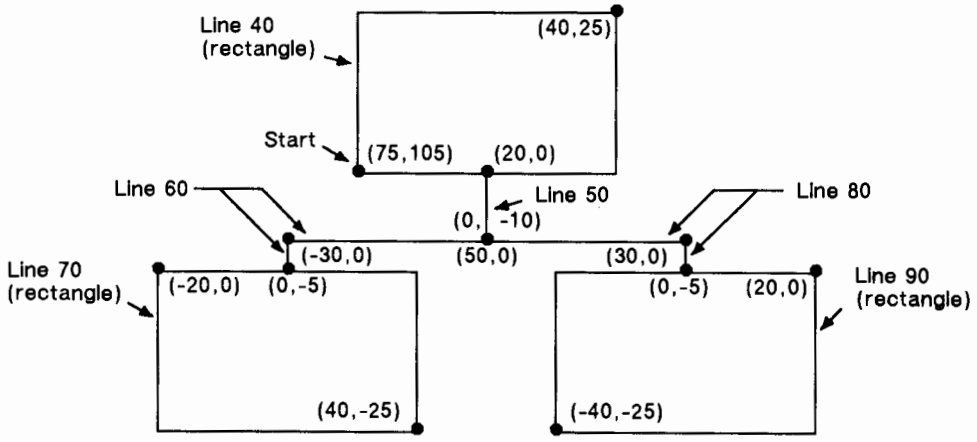


The only difference between the ER instruction and the RR instruction is that the ER instruction produces an outlined rectangle; the RR, a filled one.

The ER instruction clears the polygon buffer and then stores the defined rectangle in the polygon buffer before drawing. Refer to *Drawing Polygons* earlier in this chapter for more information.

EXAMPLE: This example uses relative coordinates to draw the same plot shown with the EA instruction. Compare this program with the EA example program to understand the differences between the coordinates used.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS6000,8000"
30  PRINT #1, "SC0,150,0,150,1SP1"
40  PRINT #1, "PA75,105ER40,25"
50  PRINT #1, "PR20,0PD0,-10"
60  PRINT #1, "PD-30,0,0,-5"
70  PRINT #1, "PU-20,0ER40,-25"
80  PRINT #1, "PU50,5PD30,0,0,-5"
90  PRINT #1, "PU20,0ER-40,-25"
100 PRINT #1, "PUSP0PG;"
110 END
```



Related Instructions	Group
EA, Edge Rectangle Absolute RA, Fill Rectangle Absolute RR, Fill Rectangle Relative	*The Polygon Group*

EW, Edge Wedge

USE: Outlines any wedge. Use EW to draw sectors of pie charts.

SYNTAX: EW*radius,start angle,sweep angle(,chord angle;)*

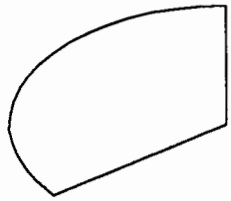
Parameter	Format	Functional Range	Default
radius	current units	*device-dependent*	no default
start angle	clamped real	±360°*	no default
sweep angle	clamped real	±360°	no default
chord angle	clamped real	0.5° to 180°	5°

* The plotter performs a modulo 360 on any start angle greater than 360.

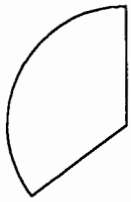
REMARKS: The EW instruction defines and edges a wedge using the current pen, line type, and attributes. The EW instruction includes an automatic pen down. When the instruction is completed, the original pen location and up/down position are restored.

The only difference between the EW instruction and the WG instruction is that the EW instruction produces an outlined wedge; the WG, a filled one.

Always use isotropic scaling in plots that draw wedges. Refer to the discussion of scaling in chapter 2 and the Scale (SC) instruction (chapter 4) for more information.



Anisotropic scaling



Isotropic scaling

- **Radius** — Specifies the distance from the current pen location to the start of the wedge's arc. Since the wedge is a portion of a circle, this parameter is the radius of the circle. It specifies the distance from the current pen location (which becomes the center of the circle) to any point on the circumference of the circle.

  The radius is interpreted in current units: as user units when scaling is on; as plotter units when scaling is off. The sign (positive or negative) of the radius determines the location of the zero-degree reference point. The illustrations following the parameter descriptions show the location of the zero-degree reference point for a positive and a negative radius.

- **Start Angle** — Specifies the beginning point for the arc as the number of degrees from the zero-degree reference point. A positive start angle positions the radius counter-clockwise from the zero-degree reference point; a negative start angle positions the radius clockwise from the zero-degree reference point.

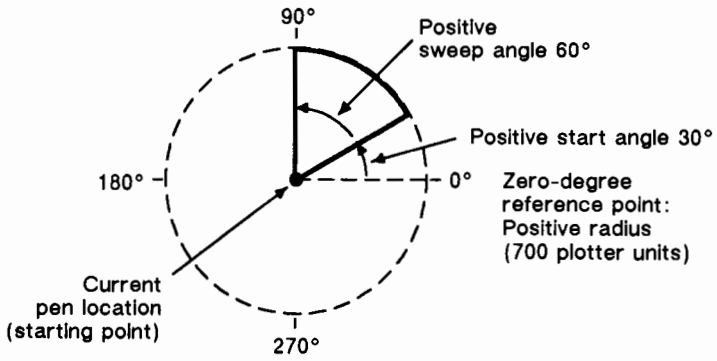
NOTE: Counterclockwise is considered the direction from the positive X-axis toward the positive Y-axis of the coordinate system. Changes to orientation, P1/P2 locations, and scaling can each have an effect on clockwise and counterclockwise direction.■

- **Sweep Angle** — Specifies the number of degrees through which the arc is drawn. A positive sweep angle draws the arc counterclockwise; a negative sweep angle draws the arc clockwise. If you specify a sweep angle greater than 360 degrees, a 360-degree angle is used.

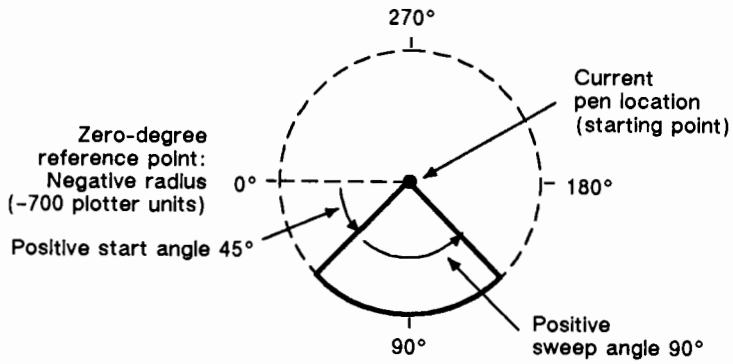
- **Chord Angle** — Specifies, in degrees, the maximum angle to use in drawing the arc. The default chord angle is 5 degrees.

  NOTE: If you use the Chord Tolerance (CT) instruction and begin specifying chords in terms of deviation distance, this parameter is termed chord tolerance and is measured in current units. Refer to chapter 9, *The Technical Graphics Extension*, for complete information on chords and chord tolerance.■

Positive Radius *(EW700,30,60)*

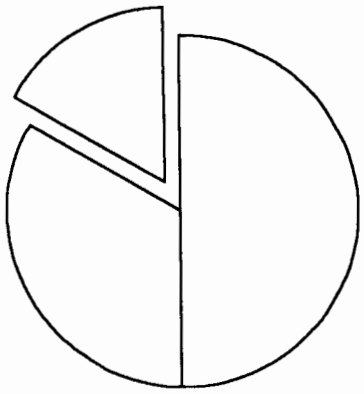


Negative Radius *(EW-700,45,90)*



EXAMPLE:

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS7000,5000SP1"
30  PRINT #1, "SC-3000,3000,-2000,2000,1"
40  PRINT #1, "PA0,0"
50  PRINT #1, "EW-1000,90,180"
60  PRINT #1, "EW-1000,330,120"
70  PRINT #1, "PR-60,110"
80  PRINT #1, "EW-1000,270,60"
90  PRINT #1, "PUSP0PG;"
100 END
```



Related Instructions	Group/Extension
WG, Fill Wedge	*The Polygon Group*
SC, Scale	*The Configuration/Status Group*
LA, Line Attributes LT, Line Type PW, Pen Width	*The Line and Fill Attributes Group*
CT, Chord Tolerance Mode	*The Technical Graphics Extension*

ERRORS:

Condition	Error	Plotter Response
polygon buffer overflow	7	edges contents of buffer

FP, Fill Polygon

USE: Fills the polygon currently in the polygon buffer. Use FP to fill polygons defined in polygon mode and by the edge rectangle and wedge instructions (EA, ER, and EW).

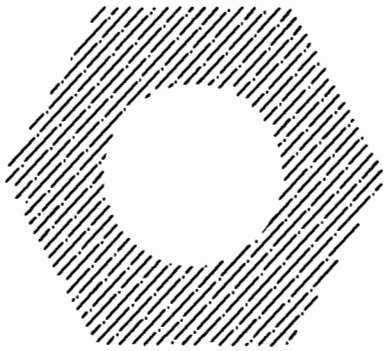
SYNTAX: FP(;)

REMARKS: The FP instruction fills any polygon that is currently in the polygon buffer. This includes wedges and rectangles defined using the EA, ER, and EW instructions. FP accesses the data in the polygon buffer, but does *not* clear the buffer or change the data in any way.

The FP instruction fills between points defined in polygon mode with both the pen up and down. The polygon is filled using the current pen, fill type, line type, and attributes (if the fill type is not solid). The FP instruction includes an automatic pen down. When the instruction is completed, the original pen location and up/down position are restored.

EXAMPLE: The following creates a polygon composed of two subpolygons. In this case, the FP instruction fills alternating areas, beginning with the outside area. Note, too, that *PM1* is not needed between the two circle instructions (line 40), since, in polygon mode, each defines a complete subpolygon.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS6000,8000SP1"
30  PRINT #1, "PA1500,1500"
40  PRINT #1, "PM0CI1000,60CI500PM2"
50  PRINT #1, "LT4FT3,50,45FP"
60  PRINT #1, "PUSP0PG;"
70  END
```



Related Instructions	Group
PM, Polygon Mode RA, Fill Rectangle Absolute RR, Fill Rectangle Relative WG, Fill Wedge	*The Polygon Group*
FT, Fill Type LA, Line Attributes LT, Line Type PW, Pen Width	*The Line and Fill Attributes Group*

PM, Polygon Mode

USE: Enters polygon mode for defining shapes, such as block letters or any unique area, and exits for subsequent filling and/or edging. Fill polygons using the Fill Polygon (FP) instruction and/or outline them using the Edge Polygon (EP) instruction.

SYNTAX: PM*polygon definition*(;)
or
PM(;)

Parameter	Format	Functional Range	Default
polygon definition	clamped integer	0, 1, or 2	0

REMARKS: In polygon mode, you define the area of the polygon(s) using graphics instructions. These instructions (and associated X,Y coordinates) are stored in the polygon buffer. The polygon is not plotted until you exit polygon mode and fill and/or outline the area.

- **No Parameter** — Clears the polygon buffer and enters polygon mode. Equivalent to (*PM0*).
- **Polygon Definition** — Defines polygon mode status as follows.

  0 — Clears the polygon buffer and enters polygon mode.

  1 — Closes the current polygon (or subpolygon).

  2 — Closes current polygon (or subpolygon) and exits polygon mode.

The following subsections explain how to use each parameter. The order in which you use these instructions is very important.

(PM0) or (PM)

Use *(PM0)* to clear the polygon buffer and enter polygon mode. While in polygon mode, you can use certain instructions to define your polygon. The following table lists these instructions.

Polygon Definition Instructions	Group/Extension
DF, Default Values IN, Initialize	*The Configuration/Status Group*
AA, Arc Absolute AR, Arc Relative AT, Absolute Arc Three Point PA, Plot Absolute PD, Pen Down PE, Polyline Encoded PR, Plot Relative PU, Pen Up RT, Relative Arc Three Point	*The Vector Group*
CI, Circle PM1/PM2, Polygon Mode	*The Polygon Group*
BP, Begin Plot*	*The Technical Graphics Extension*

* The BP instruction automatically closes polygon mode and initializes the plotter.

The polygon buffer stores the points (vertices) that define your polygon. These points are accessed later when you exit polygon mode and fill and/or edge the polygon.

NOTE: In polygon mode, the CI instruction is interpreted differently than other graphics instructions. Refer to *Drawing Circles in Polygon Mode* earlier in this chapter.■

When you define polygons, the pen location before *(PM0)* is the first point (vertex) of the polygon, and the first point stored in the polygon buffer. For example, if you execute the instructions *(PA0,1750PM0)*, the absolute coordinates (0,1750) are the first point of your polygon. Each subsequent pair of coordinates defines a point, or vertex, of the polygon.

You can define points with the pen up or down. However, the EP instruction draws only between points that are defined when the pen is down. The FP instruction fills the area(s) between all vertices, regardless of whether the pen is up or down when they are defined.

It is good programming practice to "close" the polygon before exiting polygon mode. Closing a polygon means adding the final vertex that defines a continuous shape; the last coordinates or increments represent the same location as the first. If you have not closed the polygon, executing *(PM1)* or *(PM2)* forces closure by adding a point to close the polygon. However, you might end up with an unexpected polygon by not closing it yourself.

You can also use the Initialize (IN) or Default Values (DF) instructions while in polygon mode. IN and DF exit polygon mode, clear the polygon buffer, and begin executing subsequent instructions immediately. Output instructions can also be used; they are not stored in the polygon buffer, but are executed immediately. You must exit polygon mode to execute other graphics instructions.

(PM1)

Use *PM1* to close the current polygon (or subpolygon) and remain in polygon mode. When you use *PM1*, the point specified after *PM1* becomes the first point of the next subpolygon. This move is *not* used as a boundary when filling a polygon with FP. When drawing the polygon, the pen will always move to this point in the up position, regardless of the current pen status. Each subsequent coordinate pair after (*PM1*) defines a point of the subpolygon.

(PM2)

Use (*PM2*) to close the current polygon (or subpolygon) and exit polygon mode. Remember, if you have not closed your polygon, executing (*PM2*) adds a point to close the polygon. Refer to *Pen Position* in chapter 5, *The Vector Group*.

EXAMPLE: The following draws the surface area of a three-prong receptacle as a series of subpolygons, then fills and edges it using the FP and EP instructions.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS7000,6000SP1"
30  PRINT #1, "PA2000,2000"
40  PRINT #1, "PM0"
50  PRINT #1, "PD3000,2000,3000,3000"
60  PRINT #1, "PD2000,3000,2000,2000"
70  PRINT #1, "PM1"
80  PRINT #1, "PD2080,2160,2480,2160,2480,2340,2080,2340,
      2080,2160"
90  PRINT #1, "PM1"
100 PRINT #1, "PD2080,2660,2480,2660,2480,2840,2080,2840,
      2080,2660"
110 PRINT #1, "PM1"
120 PRINT #1, "PD2920,2340,2920,2660,2720,2660"
130 PRINT #1, "AA2720,2500,180PD2920,2340"
140 PRINT #1, "PM2FPEP"
150 PRINT #1, "PUSP0PG;"
160 END
```

Related Instructions	Group
EP, Edge Polygon FP, Fill Polygon	*The Polygon Group*

ERRORS:

Condition	Error	Plotter Response
invalid instruction used in polygon mode	1	ignores invalid instruction
parameter out of range	3	ignores instruction
buffer overflow	7	ignores overflowing points

RA, Fill Rectangle Absolute

USE: Defines and fills a rectangle using absolute coordinates. Use RA to fill rectangular shapes in plots.

SYNTAX: RA*X,Y(;)*

Parameter	Format	Functional Range	Default
X,Y coordinates	current units	*device-dependent*	no default

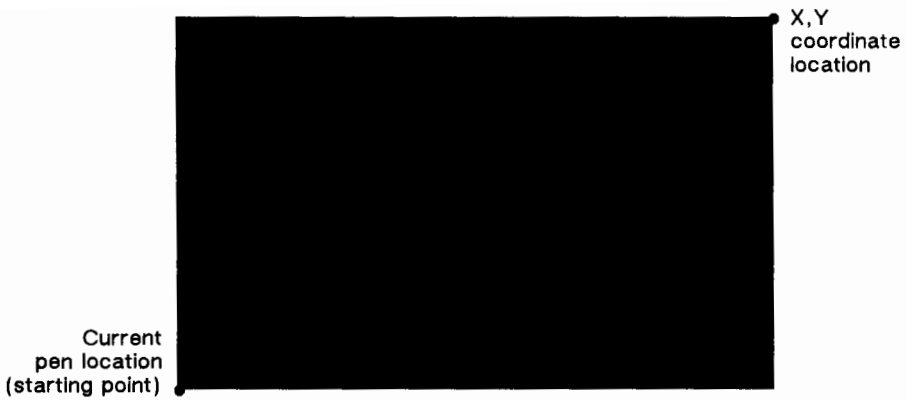
REMARKS: The RA instruction defines and fills a rectangle using the current pen, the current line attributes and fill types, and absolute plotting. The sides of the filled polygon will be parallel to the coordinate axes. The RA instruction includes an automatic pen down. When the instruction is completed, the original pen location and up/down position are restored.

- **X,Y Coordinates** — Specify the corner of the rectangle that is diagonally opposite from the current pen location, which is the starting point of the rectangle. Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off.

  If either the X- or Y-coordinate is the same as the respective coordinate of the current location, the plotter draws a line. Note that with certain hatch fill patterns, the line may fall in the white spaces of the fill and not be drawn.

  Because the coordinates are stored in the polygon buffer, you can overlay a second fill pattern by specifying another Fill Type (FT) and using the Fill Polygon (FP) instruction.

  NOTE: The following illustration shows the current pen location in the lower-left corner and the instruction's X,Y coordinates in the upper-right corner. However, these points can be in any diagonally opposite corners.■

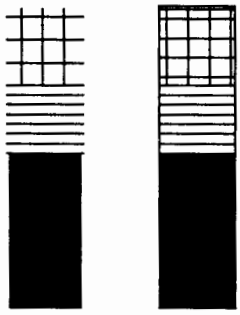


The only difference between the RA instruction and the EA instruction is that the RA instruction produces a filled rectangle; the EA, an outlined one.

The RA instruction clears the polygon buffer and then stores the defined rectangle in the polygon buffer before drawing. Refer to *Using the Polygon Buffer* earlier in this chapter.

EXAMPLE: This program uses RA with three different fill types (refer to the FT instruction described in chapter 7, *The Line and Fill Attributes Group*) to create rectangles such as those you might use in a PERT chart. The rectangles in the right bar are edged using the EP instruction.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS7000,5000SP1"
30  PRINT #1, "PA400,400RA800,1200"
40  PRINT #1, "PA400,1200FT3,50RA800,1600"
50  PRINT #1, "PA400,1600FT4RA800,2000"
60  PRINT #1, "PA1200,400FTRA1600,1200EP"
70  PRINT #1, "PA1200,1200FT3,50RA1600,1600EP"
80  PRINT #1, "PA1200,1600FT4RA1600,2000EP"
90  PRINT #1, "PUSP0PG;"
100 END
```



Related Instructions	Group
EA, Edge Rectangle Absolute EP, Edge Polygon ER, Edge Rectangle Relative RR, Fill Rectangle Relative	*The Polygon Group*
FT, Fill Type LT, Line Type	*The Line and Fill Attributes Group*

RR, Fill Rectangle Relative

USE: Defines and fills a rectangle using relative coordinates. Use RR to fill rectangular shapes in plots.

SYNTAX: RR*X,Y(;)*

Parameter	Format	Functional Range	Default
X,Y increments	current units	*device-dependent*	no default

REMARKS: The RR instruction defines and fills a rectangle using the current pen, the current line attributes and fill types, and relative plotting. The sides of the filled rectangle will be parallel to the coordinate axes. The RR instruction includes an automatic pen down. When the instruction is completed, the original pen location and up/down position are restored.

- **X,Y Increments** — Specify the corner of the rectangle that is diagonally opposite from the current pen location, which is the starting point of the rectangle. Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off.

  If either the X- or Y-coordinate is zero, the plotter draws a line. Note that with certain hatch fill patterns, the line may fall in the white spaces of the fill and not be drawn.

  Because the coordinates are stored in the polygon buffer, you can overlay a second fill pattern by specifying another Fill Type (FT) and using the Fill Polygon (FP) instruction.

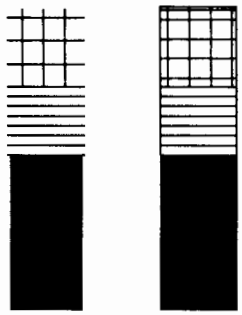
  NOTE: This illustration shows the current pen location in the lower-left corner and the instruction's X,Y increments in the upper-right corner. However, these points can be in any diagonally opposite corners.■

The only difference between the RR and the ER instructions is that the RR instruction produces a filled rectangle; the ER, an outlined one.

The RR instruction clears the polygon buffer and then stores the defined rectangle in the polygon buffer before drawing. A rectangle requires enough buffer space to hold five points. The default buffer size is sufficient. Refer to *Using the Polygon Buffer* earlier in this chapter.

EXAMPLE: This program uses RR with three different fill types (refer to the FT instruction described in chapter 7, *The Line and Fill Attributes Group*) to create rectangles such as those you might use in a bar chart. The rectangles in the right bar are edged using the EP instruction.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS7000,5000SP1PA400,400RR400,800"
40  PRINT #1, "PR0,800FT3,50RR400,400"
50  PRINT #1, "PR0,400FT4RR400,400"
60  PRINT #1, "PA1200,400FTRR400,800EP"
70  PRINT #1, "PR0,800FT3,50RR400,400EP"
80  PRINT #1, "PR0,400FT4RR400,400EP"
90  PRINT #1, "PUSP0PG;"
100 END
```



Related Instructions	Group
EA, Edge Rectangle Absolute ER, Edge Rectangle Relative RA, Fill Rectangle Absolute	*The Polygon Group*
FT, Fill Type LT, Line Type	*The Line and Fill Attributes Group*

WG, Fill Wedge

USE: Defines and fills any wedge. Use WG to draw filled sectors of a pie chart.

SYNTAX: WG*radius,start angle,sweep angle(,chord angle;)*

Parameter	Format	Functional Range	Default
radius	current unit	*device-dependent*	no default
start angle	clamped real	±360°*	no default
sweep angle	clamped real	±360°	no default
chord angle	clamped real	0.5° to 180°	5°

* The plotter performs a modulo 360 on any start angle greater than 360.

REMARKS: The WG instruction defines and fills a wedge using the current pen, fill type, and line types. The WG instruction includes an automatic pen down. When the instruction is completed, the original pen location and up/down position are restored.

The only difference between the WG instruction and the EW instruction is that the WG instruction produces a filled wedge; the EW, an outlined one.

We recommend that you always use isotropic scaling in plots that draw wedges, unless you want your wedges to "flex" with changes in scaling. Refer to the discussion of scaling in chapter 2 and the Scale (SC) instruction (chapter 4) for more information.

- **Radius** — Specifies the distance from the current pen location to the start of the wedge's arc. Since the wedge is a portion of a circle, this parameter is the radius of the circle. It specifies the distance from the current pen location (which becomes the center of the circle) to any point on the circumference of the circle.

The radius is interpreted in current units: as user units when scaling is on; as plotter units when scaling is off. The sign of the radius determines the location of the zero-degree reference point. The illustrations following the parameter descriptions show the location of the zero-degree reference point for a positive and a negative radius.

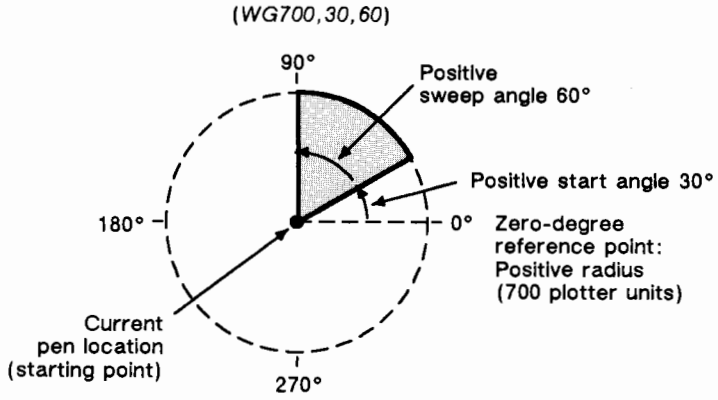
- **Start Angle** — Specifies the beginning point for the arc as the number of degrees from the zero-degree reference point. A positive start angle positions the radius counterclockwise from the zero-degree reference point; a negative start angle positions the radius clockwise from the zero-degree reference point.

  NOTE: Counterclockwise is considered the direction from the positive X-axis toward the positive Y-axis of the coordinate system. Changes to orientation, P1/P2 locations, and scaling can each have an effect on clockwise and counterclockwise direction.■

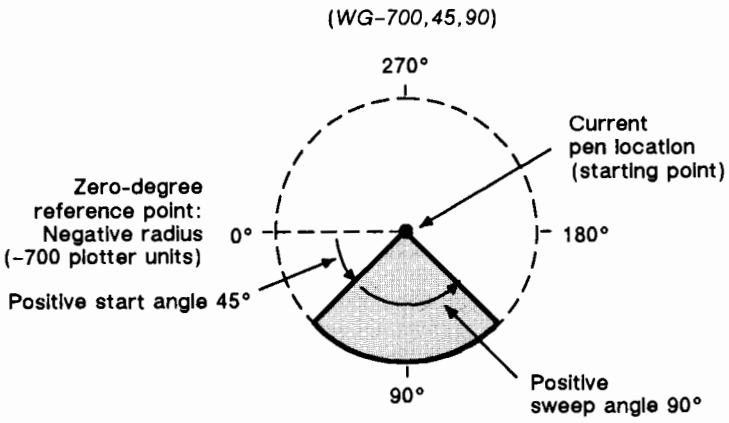
- **Sweep Angle** — Specifies the number of degrees through which the arc extends. A positive sweep angle defines the arc counterclockwise; a negative sweep angle defines the arc clockwise. If you specify a sweep angle greater than 360 degrees, a 360-degree angle is used.

- **Chord Angle** — Specifies, in degrees, the maximum angle to use in drawing the arc. The default chord angle is 5 degrees.

  NOTE: If you use the Chord Tolerance (CT) instruction and begin specifying chords in terms of deviation distance, this parameter is termed chord tolerance and is measured in current units. Refer to chapter 9, *The Technical Graphics Extension*, for complete information on chords and chord tolerance.■



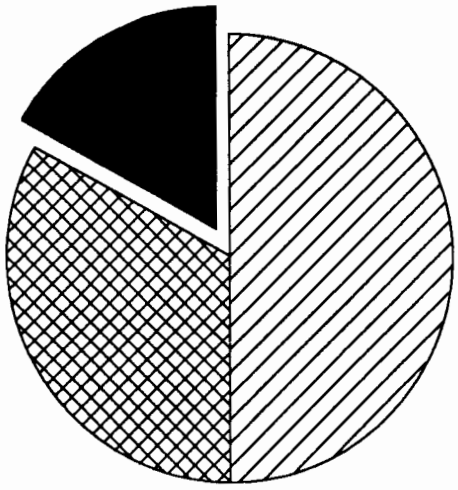
Positive Radius



Negative Radius

EXAMPLE: This program uses WG with three different fill types (refer to the Fill Type [FT] instruction described in chapter 7, *The Line and Fill Attributes Group*) to create wedges such as those you might use in a pie chart. Note that the wedges are edged using the EP instruction.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS5000,7000SP1"
30  PRINT #1, "SC-3000,3000,-2000,2000,1"
40  PRINT #1, "PA0,0FT3,75,45WG-1000,90,180EP"
50  PRINT #1, "FT4,60,45WG-1000,330,120EP"
60  PRINT #1, "PR-60,110FT1,0,0"
70  PRINT #1, "WG-1000,270,60EP"
80  PRINT #1, "PUSP0PG;"
90  END
```



Related Instructions	Group/Extension
EP, Edge Polygon EW, Edge Wedge	*The Polygon Group*
SC, Scale	*The Configuration/Status Group*
FT, Fill Type LA, Line Attributes LT, Line Type PW, Pen Width	*The Line and Fill Attributes Group*
CT, Chord Tolerance Mode	*The Technical Graphics Extension*

ERRORS:

Condition	Error	Plotter Response
polygon buffer overflow	7	fills contents of buffer

7

The Line and Fill Attributes Group

The information in this chapter enables you to achieve the following results.

- Enhance your plots with line types.
- Enhance your plots with fill types.
- Position fill type patterns.

The following instructions are described in this chapter.

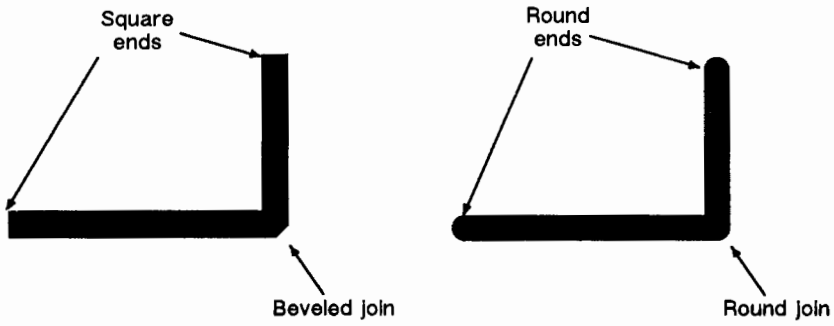
Instruction	Summary
AC, Anchor Corner	Specifies the starting point for fill patterns.
FT, Fill Type	Selects the pattern to use when filling polygons.
LA, Line Attributes	Specifies how line ends and joins are shaped.
LT, Line Type	Selects the line pattern to use for drawing lines.
PW, Pen Width	Assigns a specific width to a pen.
RF, Raster Fill Definition	Defines a pattern for use as area fill.
SM, Symbol Mode	Draws a symbol at each coordinate location.
SP, Select Pen	Selects a pen for plotting.
UL, User-Defined Line Type	Creates a specified line pattern.
WU, Pen Width Unit Selection	Specifies whether the pen width is expressed in millimeters or as a percentage of the P1/P2 distance.

The following is a summary of the parameter formats and their *minimum* ranges. Your device may support a greater range than listed here.

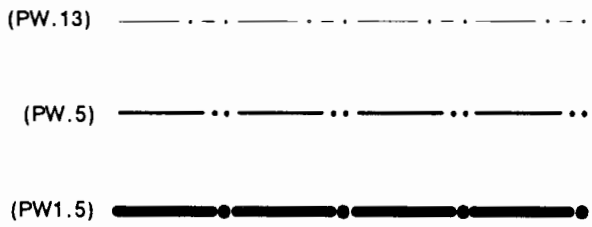
Parameter Format	Minimum Ranges
Integer	2^{23} to $2^{23}-1$ (-8 388 608 to 8 388 607)
Real	2^{23} to $2^{23}-1$ (-8 388 608.000 0 to 8 388 607.999 9)
Clamped Integer	2^{15} to $2^{15}-1$ (-32 768 to 32 767)
Clamped Real	2^{15} to $2^{15}-1$ (-32 768.000 0 to 32 767.999 9)

Using Line Attributes and Types

You can change the appearance of the lines you draw by using the Line Attributes (LA) and Line Type (LT) instructions. The Line Attributes instruction lets you specify how the ends of lines and corners of joined lines should appear, for example, square, round, or beveled.



Line types are repeated patterns of dots and/or dashes (including solid lines). The following shows some examples of line types. Note that you can vary the width of lines and line types by using the Pen Width (PW) instruction.



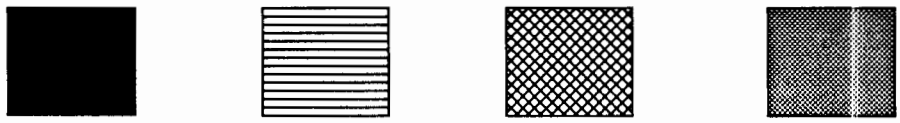
NOTE: Pen width does not necessarily set the width of lines in labels. The width of lines in labels is determined by the stroke weight attribute of the Alternate Font Definition (AD) or Standard Font Definition (SD) instruction. Refer to chapter 8, *The Character Group*, for more information.■

Once you specify a line type and line attributes, all lines created by the following instructions are drawn using the new line type and attributes. Line types and their interactions with fill patterns are discussed later in this chapter.

Instructions Affected by Line Types	Group
AA, Arc Absolute AR, Arc Relative AT, Absolute Arc Three Point PA, Plot Absolute PD, Pen Down PE, Polyline Encoded PR, Plot Relative RT, Relative Arc Three Point	*The Vector Group*
CI, Circle EA, Edge Rectangle Absolute EP, Edge Polygon ER, Edge Rectangle Relative EW, Edge Wedge FP, Fill Polygon RA, Fill Rectangle Absolute RR, Fill Rectangle Relative WG, Fill Wedge	*The Polygon Group*

Using Fill Types

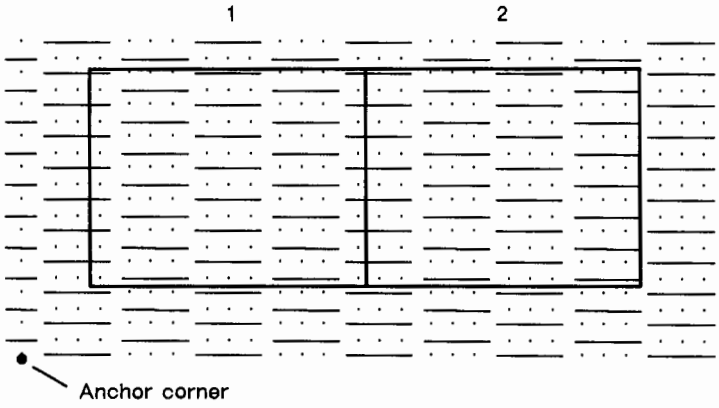
The Fill Type (FT) instruction adds detail to your plots and increases their visual effectiveness. The fill type affects the RA (Fill Rectangle Absolute), RR (Fill Rectangle Relative), WG (Fill Wedge), and FP (Fill Polygon) instructions. All HP-GL/2 plotters support solid, parallel line (hatch), and crosshatch fill types. Your plotter may also support raster fill types; these are primarily supported by raster devices.* Raster devices can also use the RF (Raster Fill) instruction to create a user-defined fill type. The following illustration shows the four fill types just mentioned.



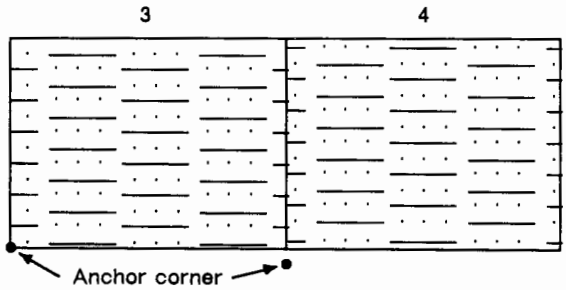
When you use hatched or crosshatched fill types, the lines are drawn using the currently selected line width, type, and attributes. For example, if you have selected a dashed line type and a hatched fill type, your figure will be filled with dashed, parallel lines.

* Pen plotters plot raster patterns as hatch patterns, approximating the density.

All fill types have an *anchor corner*, the starting point of the fill's pattern. Its default location is the plotter-unit origin (0,0). Conceptually, the fill type replicates outward from the anchor corner in the X- and Y-directions, as shown in the following illustration. Figures are filled by the portion of the fill type that occurs within their borders; refer to rectangles 1 and 2.



Use the AC (Anchor Corner) instruction to position the fill type in the figure. Rectangle 3 has the anchor corner set in its lower-left corner. Rectangle 4 has the anchor corner set below its lower-left corner to alter the pattern's position and thereby create a contrast between the two adjacent figures.



Selecting a Pen and Width

Whether your plotter uses physical pens or "logical" pens, you must select a pen before your plotter will draw anything. Select a pen using either the Select Pen (SP) instruction or the Polyline Encoded (PE) instruction. The SP instruction is described later in this chapter; the PE instruction is described in chapter 5, *The Vector Group*.

You can change the logical width of the pen using the Pen Width (PW) instruction. Subsequent lines are drawn in the new width. For pen plotters, the PW instruction causes the plotter to think that subsequent moves are being made with a wider or narrower pen than before, *whether or not you have switched to a pen with a different width*. The plotter will compensate by restroking lines to the approximate width. Use PW to distinguish lines and enhance your plots.

Pen (line) widths can be specified either in millimeters or as a percentage of the diagonal distance from P1 to P2. Use the WU (Pen Width Unit Selection) instruction to specify how the width (a parameter of PW) is specified. Since using WU (with or without parameters) affects all pen widths, send WU *before* PW.

AC, Anchor Corner

USE: Positions the starting point of any fill pattern. Use AC to ensure that the selected fill pattern will be positioned within the figure as expected.

SYNTAX: AC*X,Y(;)*
or
AC*(;)*

Parameter	Format	Functional Range	Default
X,Y coordinates	current units	*device-dependent*	no default

REMARKS: The "anchor corner" is the point at which any fill pattern starts, regardless of where it is actually drawn. Setting the anchor corner guarantees that a corner point of the selected fill pattern will be at the specified coordinate, aligned vertically and horizontally.

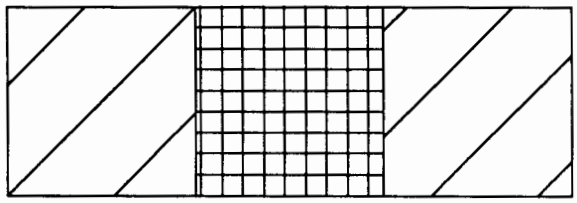
- **No Parameters** — Defaults the anchor corner to the plotter-unit origin (0,0). Equivalent to (*AC0,0*).

- **X,Y Coordinates** — Defines the position of the starting point for any fill pattern.

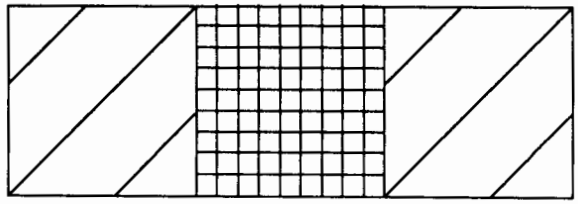
If the X,Y location is defined in plotter units, turning scaling on or changing the locations of P1 and P2 has no effect on the anchor corner location. If, however, the X,Y location is defined in user units, the physical location of the anchor corner moves with changes in scaling or to P1 and P2. For example, assume the scaling is from 0 to 10 in both axes and the anchor corner is specified as point (2,3). If the scaling is changed to 0 to 5 in both axes, the anchor corner is still (2,3), but it is in a different physical location now that the user units are a different size (the user units have become twice as large). Turning off scaling causes the anchor corner location to be frozen in the plotter-unit equivalent of its current user-unit value.

EXAMPLE: The following example shows, first, three adjacent squares whose fill patterns are anchored at the lower-left corner of the hard-clip limits. The fill pattern is continuous across the squares. Each square in the second set has an anchor corner set in its own lower-left corner. Notice how this helps distinguish between the adjacent figures.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS7000,5000SP1PA3000,3000"
40  PRINT #1, "FT3,400,45RR1000,1000EP"
50  PRINT #1, "PR1000,0FT4RR1000,1000EP"
60  PRINT #1, "PR1000,0FT3RR1000,1000EP"
70  PRINT #1, "PA3000,1500AC3000,1500RR1000,1000EP"
80  PRINT #1, "PA4000,1500AC4000,1500FT4RR1000,1000EP"
90  PRINT #1, "PA5000,1500AC5000,1500FT3RR1000,1000EP"
100 PRINT #1, "PUSP0PG;"
110 END
```



Anchor Corner at Lower-left Corner of Hard-Clip Limits



Anchor Corner at Lower-left Corner of Each Rectangle

Related Instructions	Group/Extension
FT, Fill Type RF, Raster Fill Type	*The Line and Fill Attributes Group*
RA, Fill Rectangle Absolute RR, Fill Rectangle Relative WG, Fill Wedge	*The Polygon Group*
SV, Screened Vectors	*The Palette Extension*

FT, Fill Type

USE: Selects the shading pattern used to fill polygons (FP), rectangles (RA or RR), characters (CF), or wedges (WG). Use FT to enhance plots with solid fill, parallel lines (hatching), crosshatching, or patterned (raster) fill.

SYNTAX: FT *fill type(,option1(,option2;))*
or
FT*(;)*

Parameter	Format	Functional Range	Default
fill type	clamped integer	1 to 4, 10, 11	1
option1, option2	clamped real	*type*-dependent	*type*-dependent

REMARKS: There are six forms of fill type (additional fill types are available if you are using your device in a dual-context environment; refer to chapter 11, *The Dual-Context Extension*). The type parameter tells the plotter which form you are using.

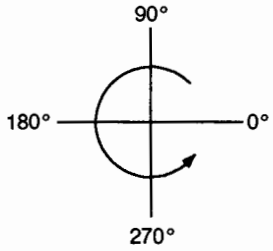
- **No Parameters** — Defaults the fill type to solid fill. Equivalent to *(FT1)*.
- **Fill Type** — Selects the fill pattern. The table below lists the parameter values and corresponding fill types.
- **Option1, Option2** — The definition of these optional parameters depends on the type of fill selected. The following table lists the options available for each fill type.

Fill Type	Description	Option1	Option2
1	solid bidirectional*	ignored	ignored
2	solid unidirectional*	ignored	ignored
3	hatched (parallel lines)	spacing of lines	angle of lines
4	crosshatched	spacing of lines	angle of lines
10	shading	shading level	ignored
11	user-defined	raster-fill index	pen

* Directionality is pertinent to pen plotters only. On raster plotters, only primary colors are guaranteed to be truly solid. Other colors may be produced by a dither pattern, and will be as close to solid as the device can produce.

For fill types 3 and 4, the *option1* parameter specifies the distance between the lines in the fill. This distance is specified in current units measured along the X-axis. The default spacing is 1% of the diagonal distance from P1 to P2. Subsequent changes in the P1/P2 locations affect this distance and therefore spacing. Option1 must be a positive number (if zero is specified, then 1% is used). (Spacing is interpreted as a percentage only in its default state; otherwise, it is interpreted as current units.)

For fill types 3 and 4, the *option2* parameter specifies an angle, in degrees, of the lines in the fill. This angle is referenced counterclockwise from the positive plotter-unit X-axis, as shown in the following illustration (0 and 180 are horizontal; 90 and 270 are vertical). The first set of lines for crosshatched fill types is drawn at the specified angle and the next set is drawn at that angle plus 90 degrees.



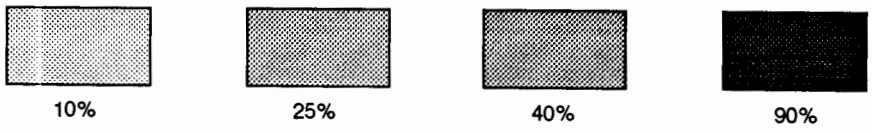
Types 3 and 4 use the current pen, line type, and attributes unless the spacing is such that the fill pattern is solid.

When the *option* parameters are omitted, the plotter uses the last specified parameters for that fill type. For example, if you specified fill type 3 with line spacing and angle parameters, then switched to fill type 4 without specifying any line parameters, the plotter would draw fill type 4 with default line spacing and angles because they have not yet been specified for that fill type. If you then switched back to fill type 3, you would not need to respecify the spacing and angle parameters because the plotter would use the values specified previously for the fill type 3 instruction.

If spacing is defined in plotter units, turning scaling on or changing the locations of P1 and P2 has no effect. If, however, the spacing is defined in user units, the spacing fluctuates with changes to P1 and P2 or scaling. Turning off scaling causes the spacing to be frozen in the plotter-unit equivalent of its current user-unit value.

NOTE: The endpoints of hatch fills are treated with the current line cap. Lines are not clipped to the polygon.■

For fill type 10, the *option1* parameter specifies the level of shading. The level is specified as a percentage.* The following illustration shows four levels of shading and the specified percentage.

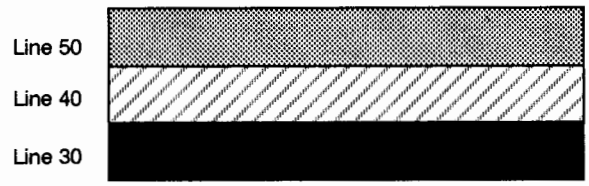


* The plotter uses at least nine predefined levels of shading equally divided between 0 and 100%.

For fill type 11, the *option1* parameter selects the corresponding user-defined raster fill. If the pattern was defined using only 1s and 0s (refer to the Raster Fill Type [RF] instruction), the pattern can be printed in the current pen color. Option2 is a Boolean that specifies whether or not the current color should be applied. If option2 is "1" (true), the color of the currently selected pen is applied to the 1s pixels in the pattern. If option2 is "0" or is not present, the pattern (of the 1s pixels) is printed in the color of pen number 1. If the pattern definition includes indexes other than 0 and 1, option2 is ignored. If you have not issued an RF instruction, the plotter will use solid fill.

EXAMPLE:

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS7000,5000SP1PA2000,2000"
40  PRINT #1, "FTRR2500,300EPPR0,300"
50  PRINT #1, "FT3,80,30RR2500,300EPPR0,300"
60  PRINT #1, "FT10,36RR2500,300EP"
70  PRINT #1, "PUSP0PG;"
80  END
```



Related Instructions	Group
LA, Line Attributes LT, Line Type PW, Pen Width RF, Raster Fill Type	*The Line and Fill Attributes Group*

LA, Line Attributes

USE: Specifies how line ends and joins (corners) are physically shaped. Use LA when drawing lines thicker than 0.35 mm to define their appearance. The LA instruction applies to lines drawn by the AA, AR, AT, CI, EA, EP, ER, EW, FP, PA, PD, PE, PR, RA, RR, RT, and WG instructions.

SYNTAX: LA*kind,value(,kind,value(,kind,value;))*
or
LA*(;)*

Parameter	Format	Functional Range	Default
kind	clamped integer	1 through 3	1
value	clamped integer (kinds 1 and 2)	*kind*-dependent*	*kind*-dependent*
	clamped real (kind 3)	*device-dependent**	5

* Refer to the table following the parameter descriptions.

REMARKS: There are three line attributes: line ends, line joins, and the miter limit. The LA parameters are used in pairs: the first parameter selects a line attribute and the second parameter defines the appearance of that attribute. The plotter uses the current line attributes when optional parameter pairs are omitted.

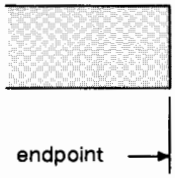
- **No Parameters** — Defaults the line attributes to butt ends, mitered joins, and a miter limit of 5. Equivalent to (*LA1,1,2,1,3,5*).
- **Kind** — Specifies the line attribute for which you are setting a value. Attributes and kind parameter values are listed in the following table.
- **Value** — Defines the characteristics of the attribute specified by the *kind* parameter. The available values are listed in the following table and described under each attribute.

Attribute	Kind	Value	Description
Line Ends*	1	1	Butt (default)
		2	Square
		3	Triangular
		4	Round
Line Joins*	2	1	Mitered (default)
		2	Mitered/beveled
		3	Triangular
		4	Round
		5	Beveled
		6	No join applied
Miter Limit	3	*device-dependent***	5 (default, refer to description under *Miter Limit*)

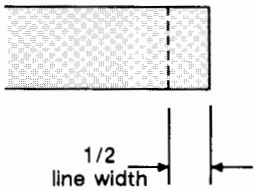
* Lines with a width of 0.35 mm or less always have rounded end caps, regardless of the current attribute setting.
** Full range is 0 to 32 767.9999, but values less than 1.1 are set to 1.1 and do not cause an error.

Line Ends

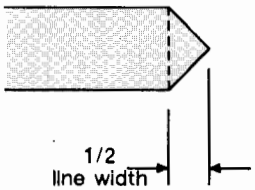
The value you specify for line ends determines how the ends of line segments are shaped. The following illustration describes the four types of line ends.



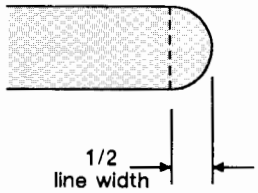
Butt ends (1) Terminate at the endpoints.



Square ends (2) Terminate one-half line width beyond the endpoints.



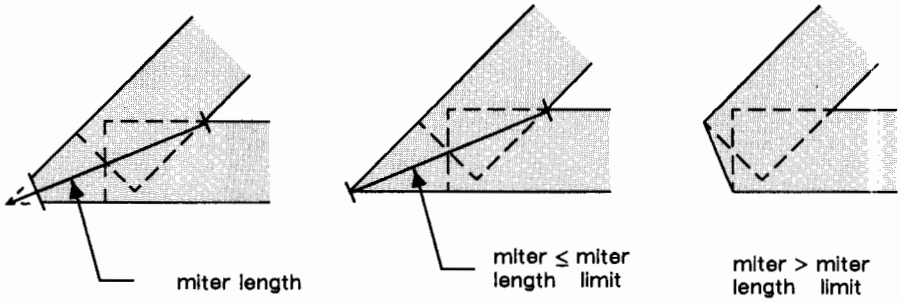
Triangular ends (3) Terminate one-half line width beyond the endpoints.



Round ends (4) Terminate in semicircle with a diameter equal to the current line width. Round ends require more data storage space than other ends.

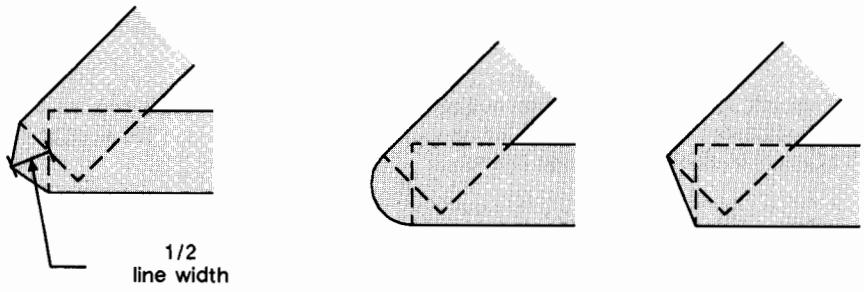
Line Joins

The value you specify for the line joins attribute determines how connecting line ends (corners) are shaped. The following illustration describes the five types of line joins. If the first and last points of a series of lines are the same, they join according to the current line join and miter limit.



Mitered join (1) Formed by two lines extending from the outer edge of each vector until they meet. The miter limit applies to this join.

Mitered/beveled join (2) Formed by two lines extending from the outer edge of each vector until they meet. If the miter length exceeds the miter limit, a beveled join is used.

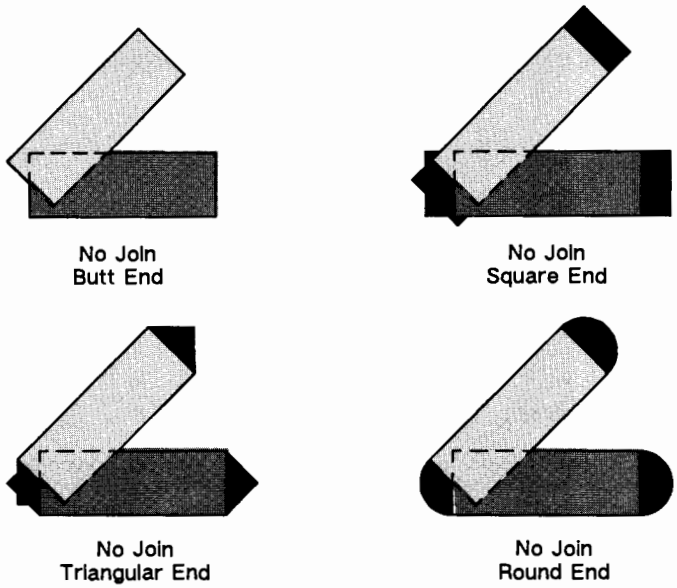


Triangular join (3) Formed by two lines extending from the outer edge of each vector to a point 1/2 line width beyond the end intersection of the vectors.

Rounded join (4) Formed by an arc with a diameter equal to the current line width. Round joins require more data storage space than other joins.

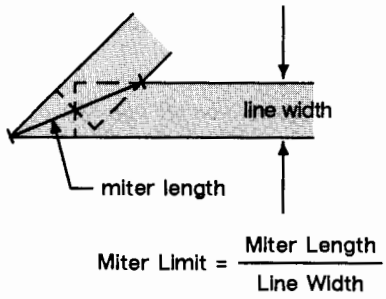
Beveled join (5) Formed by a line connecting the outer edge of one vector to the outer edge of the other vector.

When you select "no join," the currently selected line ends for the two lines merely overlap. Refer to the following illustration.



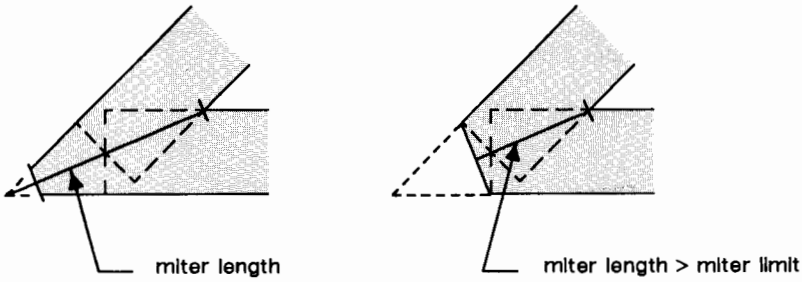
Miter Limit

The value you specify for miter limit determines the "length" of a mitered join, as shown in the following illustration. The miter limit is the ratio of the diagonal line through the join of two connecting lines to the width of the lines.



$$\text{Miter Limit} = \frac{\text{Miter Length}}{\text{Line Width}}$$

When the miter length exceeds the miter limit, the point of the miter is clipped to the miter limit; the maximum clipped miter is equivalent to a beveled join. The default miter limit is 5.



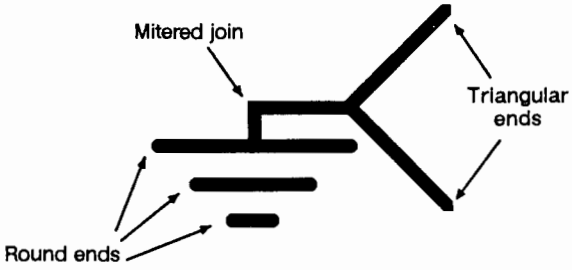
Clipped mitered join

Maximum clipped mitered join (beveled join)

An LA instruction remains in effect until another LA instruction is executed, or the plotter is initialized or set to default conditions.

EXAMPLE:


```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
20  PRINT #1, "PS7000,5000SP1PA4000,3000"
30  PRINT #1, "PW2LA1,3PD3500,2500,4000,2000"
40  PRINT #1, "PU3500,2500LA2,2,3,20PD3000,2500,3000,2300"
50  PRINT #1, "PU2500,2300LA1,4PD3500,2300"
60  PRINT #1, "PU2700,2100PD3300,2100"
70  PRINT #1, "PU2900,1900PD3100,1900"
80  PRINT #1, "PUSP0PG;"
90  END
```



Related Instructions	Group
LT, Line Type PW, Pen Width UL, User-Defined Line Type	*The Line and Fill Attributes Group*

LT, Line Type

USE: Specifies the line pattern to be used when drawing lines. Use LT to distinguish lines and enhance your plot. Note that the ends of dashed line segments in a line pattern are affected by current line attributes (refer to the LA instruction earlier in this chapter). The LT instruction does not affect labels.

SYNTAX: LT*line type(,pattern length(,mode;))*
or
LT*(;)*
or
LT*99(;)*

Parameter	Format	Functional Range	Default
line type	clamped integer	-8 to 8	solid line
		99	restores previous line type
pattern length	clamped real	>0	4% of the distance between P1 and P2
mode	clamped integer	0 or 1	0 (relative)

REMARKS: The LT instruction applies to lines drawn by the AA, AR, AT, CI, EA, EP, ER, EW, FP, PA, PD, PE, PR, RA, RR, RT, and WG instructions. Line types are drawn using the current line attributes as set by the Line Attributes (LA) instruction. For example, if you have used LA to specify rounded ends, the plotter draws each dash in a dashed line pattern with rounded ends.

An LT instruction remains in effect until another LT instruction is executed or the plotter is initialized or set to default conditions.

- **No Parameters** — Defaults the line type to solid line and saves the previous line type, pattern length, and residue (if any).

- **Line Type** — Specifies the line pattern for subsequent lines. Line patterns can be of fixed or adaptive type.

  Positive line types (1 — 8) are *fixed-length* line patterns. Any unused part of the pattern (residue) is carried over into the next line. The plotter saves the residue when any of the following instructions are received: CI, EA, EP, ER, EW, FP, PM, RA, RR, or WG. The residue is restored after these instructions are completed.

The following instructions clear current residue and vector endpoints.

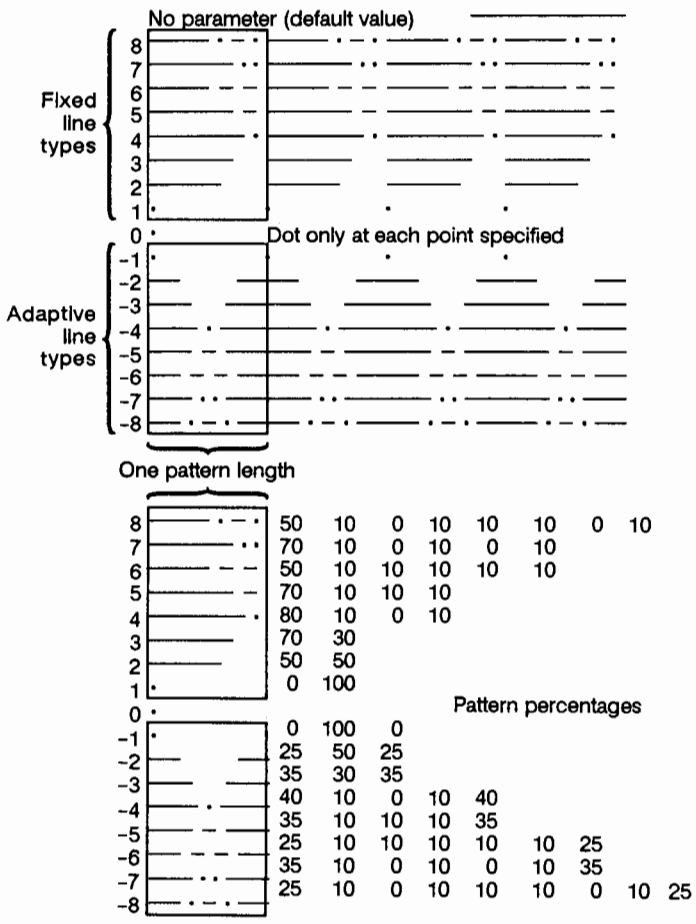
Instructions That Affect LT1 – LT8	Group
AC, Anchor Corner LA, Line Attributes LT Line Type—except (*LT*) and (*LT99*) PG, Advance Full Page PW, Pen Width RF, Raster Fill Definition SP, Select Pen UL, User-Defined Line Type WU Pen Width Unit Selection	*The Line and Fill Attributes Group*
DF, Default Values IN, Initialize IP, Input P1 and P2 IR, Input Relative P1 and P2 IW, Input Window RO, Rotate Coordinate System SC, Scale	*The Configuration and Status Group*
BP, Begin Plot	*The Technical Graphics Extension*

A zero line type (0) draws only a dot at the X,Y coordinates for the AA, AR, AT, CI, PA, PD, PR, and RT instructions. Zero pen down values and zero-length lines also produce dots. A dot is a one-plotter-unit-long vector, drawn using the current line end and pen width.

Negative line types (–1 – –8) are *adaptive-length* line patterns. The plotter adjusts the pattern length so that each line contains one or more complete patterns.

NOTE: Do *not* use an adaptive line type when drawing circles or arcs, or the edges of wedges or polygons. The plotter will attempt to draw the complete pattern in every chord; there are 72 chords in a circle using the default chord angle.■

Line patterns are composed of alternate pen-down and pen-up moves that are percentages of the pattern length (the first percentage is always pen-down). The following illustration first shows the line type patterns, then gives the pattern percentages. Note that you can create additional line patterns with the UL instruction (described later in this chapter).



99 (*LT99*) restores the previous line type (and residue if fixed-line type) only if the current line type is solid and pen location is the same as when you issued *LT*. The plotter ignores (*LT99*) if you are currently plotting with a nonsolid line type.

Sending any of the following instructions while plotting with a solid line type clears the previous line type and a subsequent (*LT99*) has no effect.

Instructions That Affect *LT99*	Group/Extension
AC, Anchor Corner LA, Line Attributes LT, Line Type—except (*LT*) and (*LT99*) PG, Advance Full Page PW, Pen Width RF, Raster Fill Definition SP, Select Pen UL, User-Defined Line Type WU Pen Width Unit Selection	*The Line and Fill Attributes Group*
DF, Default Values IN, Initialize IP, Input P1 and P2 IR, Input Relative P1 and P2 IW, Input Window RO, Rotate Coordinate System SC, Scale	*The Configuration and Status Group*
BP, Begin Plot	*The Technical Graphics Extension*

- **Pattern Length** — Specifies the length of one complete line pattern, either as a percentage of the diagonal distance between the scaling points P1 and P2 or in millimeters. You must specify a length greater than zero or the plotter ignores the instruction and generates an error. If you don't specify a length, the plotter uses the last value specified (4% of P1/P2 distance is the default).

- **Mode** — Specifies how the values of the pattern length parameter are interpreted. If you don't specify a mode, the plotter uses the last value specified. If the specified mode is not 0 or 1, the plotter generates an error and ignores the instruction.

  0 **Relative mode** (default). Interprets the pattern length parameter as a percentage of the diagonal distance between P1 and P2.

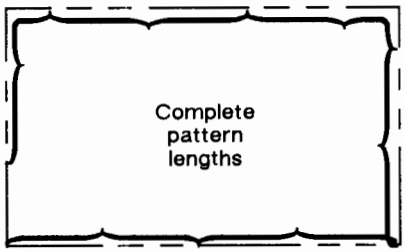
  When specified as a percentage, the pattern length changes along with changes in P1 and P2.

  1 **Absolute mode**. Interprets the pattern length parameter in millimeters.

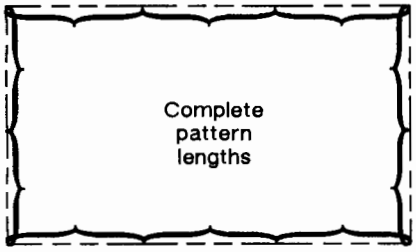
  When specified in millimeters, fixed-line type pattern lengths are a constant length, but adaptive-line type pattern lengths are adjusted downward to fit an integral number of patterns per vector.

If you do not specify the pattern length and mode parameters, then the plotter uses their current values. When using relative mode and isotropic scaling, the pattern length changes with changes to P1 and P2.

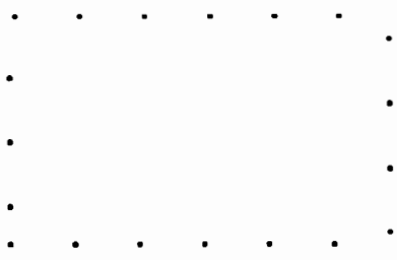
EXAMPLES:



Fixed *LT6*



Adaptive *LT-6*



Dots *LT1*

Dots *LT0*

Related Instructions	Group
FT, Fill Type PW, Pen Width UL, User-Defined Line Type	*The Line and Fill Attributes Group*
AA, Arc Absolute AR, Arc Relative AT, Absolute Arc Three Point PA, Plot Absolute PD, Pen Down PE, Polyline Encoded PR, Plot Relative RT, Relative Arc Three Point	*The Vector Group*
CI, Circle EA, Edge Rectangle Absolute EP, Edge Polygon ER, Edge Rectangle Relative EW, Edge Wedge FP, Fill Polygon RA, Fill Rectangle Absolute RR, Fill Rectangle Relative WG, Fill Wedge	*The Polygon Group*

PW, Pen Width

USE: Specifies the width of a pen. Subsequent lines are drawn in this width. Pen plotters will restroke lines to the approximate width. Use PW to distinguish lines and enhance your plots. Pen width can be specified as a fixed metric unit or relative to P1 and P2. Set the mode using the WU instruction (default is metric).

SYNTAX: PW*width(,pen;)*
or
PW*(;)*

Parameter	Format	Functional Range	Default
width	clamped real	*device-dependent*	dependent*
pen	integer	*device-dependent*	all pens

* Dependent on the mode set by the Pen Width Unit Selection (WU) instruction: if mode is metric, default width is 0.35 mm; if mode is relative, default width is 0.1% of the diagonal distance from P1 to P2.

REMARKS: If the pen is down when you change the width, the new width takes effect at the next pen down. For this reason, we recommend that you assign pen widths once at the beginning of your program. In this way, when you select a pen, it is already the proper width and all lines will be drawn appropriately.

A PW instruction remains in effect until another PW instruction or a WU instruction is executed. If you use WU to change the units of the width parameter (metric or relative), send the WU *before* PW. The Default Values (DF) instruction does *not* affect PW.

- **No Parameters** — Defaults the pen line width according to the current units set by WU: 0.35 mm if metric (default); 0.1% of the diagonal distance from P1 to P2 if relative.

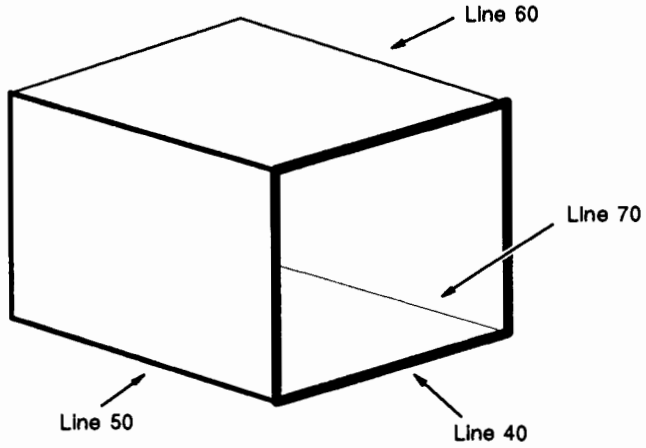
- **Width** — Specifies the line width. When the parameter is zero or less than the thinnest line available in your device, the plotter assumes the thinnest line width it can produce. Raster devices support pen widths up to 16 384 plotter units (409.6 mm). If the specified width is greater than the plotter's maximum, then the maximum available is used.

- **Pen** — Specifies the pen number to which the new width applies. If the pen parameter is not specified, the plotter applies the width to all pens. If the pen number is greater than the number of pens available in the plotter, or the pen number is negative, the plotter ignores the instruction.

NOTE: The width of lines in labels is determined by the stroke weight attribute of the Alternate Font Definition (AD) or Standard Font Definition (SD) instruction, not by the PW instruction. If the stroke weight for stick fonts is set to 9999, then labels will be drawn using the current pen width.■

EXAMPLE:

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS7000,5000SP1PA3500,2500"
40  PRINT #1, "PW1.5PD4500,2800,4500,1800,3500,1500,3500,2500"
50  PRINT #1, "PW.8PD2300,2900,2300,1900,3500,1500"
60  PRINT #1, "PW.5PU2300,2900PD3300,3200,4500,2800"
70  PRINT #1, "PW.25PU4500,1800PD3500,2100"
80  PRINT #1, "PUSP0PG;"
90  END
```



Related Instructions	Group
SP, Select Pen WU, Pen Width Unit Selection	*The Line and Fill Attributes Group*

ERRORS:

Condition	Error	Plotter Response
width < 0	3	ignores instruction
pen number out of range	3	ignores instruction

RF, Raster Fill Definition

USE: Defines a rectangular pattern that may be used in area fill and screened vectors. Use RF to create your own fill pattern before selecting it for use with the FT instruction.

SYNTAX: RF*index(,width,height,pen number(,...pen number;))*
or
RF*index(;)*
or
RF*(;)*

Parameter	Format	Functional Range	Default
index	clamped integer	1 to 8	1 (solid)
width	clamped integer	8, 16, 32, or 64	no default
height	clamped integer	8, 16, 32, or 64	no default
pen number	integer	*device-dependent*	no default

REMARKS: The RF instruction does not itself select a fill type; use the Fill Type (FT) instruction with a type parameter of 11 and the corresponding raster fill index number for the second parameter.

NOTE: If you redefine an RF index in the middle of a plot, the plotter may use the pattern in effect when rasterization begins, not the pattern designed for a previous polygon or vector. This is determined by your plotter's data storage capabilities.■

- **No Parameters** — Defaults all raster fill patterns to solid fill.

  NOTE: On raster plotters, only primary colors are guaranteed to be truly solid. Other colors may be produced by a dither pattern, and will be as close to solid as the device can produce, or the plotter may choose the nearest primary color.■

- **Index** — Specifies the index number of the pattern being defined. At least eight patterns can exist concurrently; your plotter may support more.

  When you send RF with an index parameter only (*RFn*), the corresponding pattern is defaulted to solid fill.

- **Width, Height** — Specify the width and height *in pixels* of the pattern being defined. These parameters must be powers of 2, from 8 to 64 pixels.

- **Pen Number** — Represents a pixel in the pattern being defined and indicates its color.

  0 White (default). Note that you can change the color of pen zero using the Pen Color Assignment (PC) instruction. Refer to chapter 10, *The Palette Extension*.

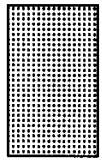
  >0 Color of the specified pen.

The pen number parameter defines pixels left to right, top to bottom. The total number of pen number parameters should be equal to the *width* × *height* parameters. For example, to define a pattern that is 8 × 16 pixels, you need 128 pen number parameters. If you do not include enough pen number parameters, the rest of the pixels are assumed to be white (or the color of pen zero). Patterns are printed in rows parallel to the plotter-unit X-axis.

If the pattern associated with a particular index is defined more than once during a single plot and all are used to fill objects, the resulting pattern(s) are device-dependent. A plotter that uses a direct bitmap will print the pattern that was defined when the object was filled, while a device that uses an intermediate data format may print the pattern last defined.

EXAMPLE:

```
10  'Insert configuration statement here
20  A$=",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0"
30  B$=",0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0"
40  PRINT #1, CHR$(27)+"%-1BBPIN"
50  PRINT #1, "PS7000,5000SP1PA3500,2500"
60  PRINT #1, "RF2,16,16"
70  FOR I = 1 TO 5
80    PRINT #1, A$
90  NEXT I
100 FOR M = 1 TO 6
110   PRINT #1, B$
120 NEXT M
130 FOR I = 1 TO 5
140   PRINT #1, A$
150 NEXT I
160 PRINT #1, ";"
170 PRINT #1, "FT11,2RR500,800EP"
180 PRINT #1, "PUSP0PG;"
190 END
```



Related Instructions	Group/Extension
FT, Fill Type	*The Line and Fill Attributes Group*
SV, Screened Vectors	*The Palette Extension*

SM, Symbol Mode

USE: Draws the specified symbol at each X,Y coordinate point with PA, PD, PE, PR, and PU instructions. Use SM to create scattergrams, indicate points on geometric drawings, and differentiate data points on multiline graphs.

SYNTAX: SM*character(;)*
or
SM*(;)*

Parameter	Format	Functional Range	Default
character	label	most printing characters (decimal codes 33–58, 60–126, 161, and 254)*	no default

* Decimal code 59 (the semicolon) is an HP-GL/2 terminator and cannot be used as a symbol *in any character set.* Use it only to cancel symbol mode – i.e., (*SM;*).

REMARKS: The SM instruction draws the specified symbol at each X,Y coordinate point for subsequent PA, PD, PE, PR, and PU instructions. The SM instruction includes an automatic pen down; after the symbol is drawn, the pen position is restored.

- **No Parameter** — Terminates symbol mode.

- **Character** — Draws the specified character at each subsequent X,Y coordinate. The symbol is drawn in addition to the usual function of each plotting instruction. The plotter centers the character cell at the coordinate location (for stick fonts, the plotter centers the character itself at the coordinate location).

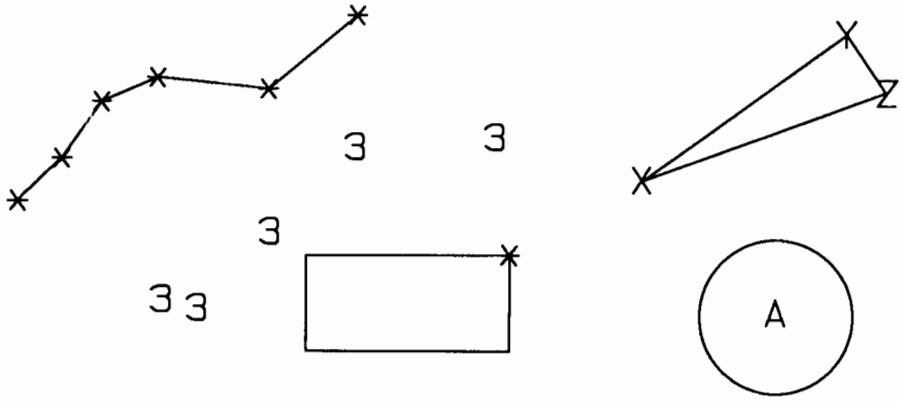
  The character is drawn in the currently selected font (either standard or alternate). If you change to a new character set, the symbol changes to the corresponding character in the new character set. If you have defined downloadable characters, you can use these in symbol mode plotting; refer to the Downloadable Character (DL) instruction. Also, the character fill (CF), size (SI and SR), slant (SL), and direction (DI and DR) instructions affect how the character is drawn (refer to chapter 8, *The Character Group*). Specifying a nonprinting character cancels symbol mode.

An SM instruction remains in effect until another SM instruction is executed or the plotter is initialized or set to default conditions.

EXAMPLE: The following program shows several uses of symbol mode: with the pen down for a line graph, with the pen up for a scattergram, and with the pen down for geometric drawings.

NOTE: Symbol mode works only with the PA, PD, PE, PR, and PU instructions. Notice that the circle and rectangle have symbols only for the PA instruction coordinate point.■

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "PS7000,5000SP1SC0,1,0,1,2"
40  PRINT #1, "SM*PA0,1000PD200,1230,400,1560"
50  PRINT #1, "PD700,1670,1300,1600,1800,2000PU"
60  PRINT #1, "SM3PA700,500,900,450,1300,850"
70  PRINT #1, "PA1750,1300,2500,1350PUSM"
80  PRINT #1, "PA3300,1100PDSMYPA4400,1890SMZ"
90  PRINT #1, "PA4600,1590SMXPA3300,1100PU"
100 PRINT #1, "SMAPA4000,400CI400"
110 PRINT #1, "SM*PA2600,700EA1500,200"
120 PRINT #1, "PUSP0PG;"
130 END
```



Related Instructions	Group
PA, Plot Absolute PD, Pen Down PE, Polyline Encoded PR, Plot Relative PU, Pen Up	*The Vector Group*

SP, Select Pen

USE: Selects the plotter's physical or logical pen for subsequent plotting. You must select a pen using this instruction or the PE instruction before the plotter can draw anything.

SYNTAX: SP*pen number(;)*
or
SP*(;)*

Parameter	Format	Functional Range	Default
pen number	integer	*device-dependent*	0

REMARKS:

- **No Parameters** — Equivalent to *SP0*. For pen plotters, SP without parameters stores the pen in the carousel; for raster devices, this selects pen zero.

- **Pen Number** — Selects the plotter's pen. Pen plotters will not draw unless an SP with a parameter greater than 0 is received.

  0 For pen plotters, SP without parameters stores the pen in the carousel, and subsequent plotting instructions are not drawn; for raster devices, this selects pen zero.

  >0 Selects the pen number. In raster plotters, pen numbers may also represent pens of the same color but with different widths. Refer to the manual for your product or HP-GL/2 option for more information.

  If the pen number is greater than the maximum number of pens in the plotter, the plotter performs the modulo function indicated below.

  The following modulo function is also used when the requested pen number is larger than the palette size currently in effect.

  pen number = ((*pen number* −1) mod number of pens −1) + 1.

The default palette for black-and-white raster plotters consists of two pens: one black and one white. The default palette for a color raster device is eight pens. The colors are defined for each pen as follows.

Number of Pens In Plotter	Pen Number	Color
2	0 1	White Black
8	0 1 2 3 4 5 6 7	White Black Red Green Yellow Blue Magenta Cyan

Use the Pen Width (PW) instruction to change the line width. Use the Pen Color Assignment (PC) instruction to change pen color assignments (grayscale in monochrome devices).

NOTE: If you are not using the Transparency Mode (TR) instruction or it is not implemented in your plotter, white is always transparent; that is, it is equivalent to no pen. For more information refer to chapter 10, *The Palette Extension*.■

Related Instructions	Group
PW, Pen Width WU, Pen Width Unit Selection	*The Line and Fill Attributes Group*
PE, Polyline Encoded	*The Vector Group*

UL, User-Defined Line Type

USE: Creates line types by specifying gap patterns. Use the LT instruction to select the pattern you've defined using UL.

SYNTAX: UL*index(,gap1,...gapn;)*
or
UL*(;)*

Parameter	Format	Functional Range	Default
index	clamped integer	1 through 8	no default
gaps	clamped real	1 to 20	*index*-dependent

REMARKS: The UL instruction allows you to define and store your own line types. The instruction does not itself select a line type.

- **No Parameters** — Defaults all line types; refer to the LT instruction.

- **Index** — Identifies the line type to be redefined. This parameter uses absolute values, so *(UL-n)* is the same as *(ULn)*. Redefining a standard fixed-line type automatically redefines the corresponding adaptive-line type. Specifying an index without gap parameters sets the line type to its default pattern.

- **Gaps** — Specify alternate pen-down and pen-up stretches in the line pattern. The first gap is a pen-down move. (All odd-numbered gaps are pen-down moves; even-numbered gaps are pen-up moves.) Gap values are converted to percentages of the pattern length parameter of the LT instruction.

  Up to 20 gaps are allowed for each user-defined line type. Gap values must be zero or positive. Zero gaps cause the pen-up/down position to alternate and the plotter produces a dot if a nonzero pen-up gap precedes and/or follows the zero value. The sum of the gap parameters must be greater than zero.

EXAMPLE:

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS7000,5000SP1"
30  PRINT #1, "PR1000,1000"
40  PRINT #1, "UL8,0,15,0,15,0,15,40,15"
50  PRINT #1, "LT8,10PD3000,0"
60  PRINT #1, "PUSP0PG;"
70  END
```

Related Instructions	Group
LA, Line Attributes LT, Line Type	*The Line and Fill Attributes Group*

ERRORS:

Condition	Error	Plotter Response
sum of gap parameters equals zero	3	ignores instruction
gap value is negative	3	ignores instruction
index is zero or |index| > 8	3	ignores instruction

WU, Pen Width Unit Selection

USE: Specifies how the width parameter of the Pen Width (PW) instruction is interpreted, in metric or relative units.

SYNTAX: WU*type(;)*
or
WU*(;)*

Parameter	Format	Functional Range	Default
type	clamped integer	0 or 1	0 (metric)

REMARKS: Since using WU with or without parameters defaults all pen widths, send WU *before* PW (which sets a new pen width).

- **No Parameters** — Defaults all pen widths to metric (0.35 mm).
- **Type** — Specifies how the width parameter of the Pen Width (PW) instruction is interpreted.

  0 **Metric.** Interprets the pen width parameter in millimeters.

  1 **Relative.** Interprets the pen width parameter as a percentage of the diagonal distance between P1 and P2.

If the specified type parameter is not 0 or 1, the plotter ignores the instruction.

A WU instruction remains in effect until another WU instruction is executed, or the plotter is initialized. WU is *not* affected by the Default Values (DF) instruction.

Related Instructions	Group
PW, Pen Width SP, Select Pen	*The Line and Fill Attributes Group*

8

The Character Group

The information in this chapter enables you to achieve the following results.

- Draw and position labels.
- Use variables in labels.
- Change label size, slant, and direction.
- Designate and select standard and alternate character sets.
- Plot with variable- and fixed-space fonts.
- Work with the character plot cell.

The following instructions are described in this chapter.

Instruction	Summary
AD, Alternate Font Definition	Specifies an alternate font for labeling.
CF, Character Fill Mode	Specifies how outline fonts will be rendered.
CP, Character Plot	Moves the pen the specified number of character plot cells from the current pen location.
DI, Absolute Direction	Specifies the slope of labels independent of P1 and P2 locations.
DR, Relative Direction	Specifies the slope of labels relative to P1 and P2 locations.
DT, Define Label Terminator	Defines the character or code that "turns off" labeling.
DV, Define Variable Text Path	Specifies the label path as right, left, up, or down.
ES, Extra Space	Increases or reduces space between label characters and lines.
LB, Label	Plots text using the currently selected font.
LO, Label Origin	Positions labels relative to the current pen location.

(Continued)

Instruction	Summary
SA, Select Alternate Font	Selects the font designated by AD for labeling.
SD, Standard Font Definition	Specifies the primary font for labeling.
SI, Absolute Character Size	Specifies an absolute character size (in centimeters).
SL, Character Slant	Specifies the slant at which labels are drawn.
SR, Relative Character Size	Specifies the size of characters as a percentage of the P1/P2 distance.
SS, Select Standard Font	Selects the font designated by SD for labeling.
TD, Transparent Data	Specifies whether control characters perform their function or print their character when labeling.

The following is a summary of the parameter formats and their *minimum* ranges. Your device may support a greater range than listed here.

Parameter Format	Minimum Ranges
Integer	2^{23} to $2^{23}-1$ (-8 388 608 to 8 388 607)
Real	2^{23} to $2^{23}-1$ (-8 388 608.000 0 to 8 388 607.999 9)
Clamped Integer	2^{15} to $2^{15}-1$ (-32 768 to 32 767)
Clamped Real	2^{15} to $2^{15}-1$ (-32 768.000 0 to 32 767.999 9)

Using Labels

Use the Label instruction (LB) to add text to plots, create text charts, or emphasize areas of a diagram or graph that need special attention or explanation. You can control almost all aspects of the label's appearance: its position, size, slant, spacing, and direction. This chapter covers the instructions that control these features. This chapter also tells you how to select character sets other than the default set. All of the instructions in this chapter are valid with the plotter's other fonts.

To use a font other than the default, use the SD (Standard Font Definition) or AD (Alternate Font Definition) instructions to *designate* the new font. Then, use the SS (Select Standard Font) or SA (Select Alternate Font) instructions to *select* the designated font for use. You can follow the LB (Label) instruction with virtually any characters, including non-printing control codes, such as a line feed or carriage return. When you are through with your label, you must use a special label terminator. Without the label terminator, your device will continue to label your picture with the remaining HP-GL/2 instructions and parameters. The default label terminator is the ASCII character **ETX** (decimal code 3). You can change the label terminator with the DT (Define Label Terminator) instruction.

Default Label Conditions

The following label default conditions are established when the plotter is initialized, or set to default conditions. To change these settings, refer to the appropriate chapter or instruction.

- **Character set** — HP Roman8.
- **Label terminator** — ASCII end-of-text character **ETX** (decimal code 3). Refer also to the DT instruction.
- **Label starting point** — Current pen location. Refer to the LO instruction.
- **Character size** — Device-dependent. Usual defaults are a character pitch of 9 and point size of 11.5 points. Refer to your device's HP-GL/2 option documentation for any of the following instructions: AD, SD, SI, and SR.
- **Label direction** — Horizontal. Refer to the DI, DR, and DV instructions.
- **Space between characters and lines** — Normal (no extra space). Refer to the ES instruction.
- **Character slant** — None (vertical). Refer to the SL instruction.
- **Character fill mode** — Solid, no edging. Refer to the CF instruction.

Designating and Selecting Fonts

If you always intend to label with the default fixed-space font, you may not need to use the SD or AD instructions for designating standard and alternate fonts (though you will want to become familiar with the font attributes you can set with these instructions). However, if you intend to use a different font, you must use these instructions to designate fonts before you can select those fonts (using either SA or SS) for labeling.

Standard and Alternate Fonts

The following outlines some of the principles to use when labeling with different fonts.

- Designate the standard and alternate fonts, using the SD and/or AD instructions, *before* labeling. If you are using the default font as your standard font, you need specify only your alternate font.

- Select either the standard or alternate font, using either the SS or SA instruction, before labeling.

  Note that labeling always begins with the standard font. If you want to start with the alternate font first, use the SA instruction before you label.

- Switch from the standard font to the alternate font using either SS and SA or the shift-in/shift-out method. If you are changing fonts within a label string, the shift-in/shift-out method is usually more efficient. Switch from the standard set to the alternate font using the ASCII shift-out control character (**SO**, decimal code 14). Switch from the alternate font to the standard font using the ASCII shift-in control character (**SI**, decimal code 15).

Special Characters

There are four ways to access special characters in any set.

1. Use the equivalent ANSI ASCII English character on your keyboard in the label string. For example, to draw the character "½" in set 5, you can use the "x" from an English keyboard.

   $\frac{1}{2}$

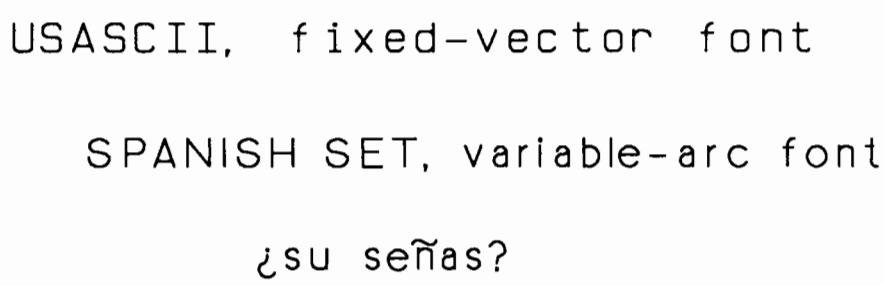
2. Use a computer-language-dependent function such as CHR$ to enter the decimal code. For example, to draw the character "½" in set 5, use CHR$(120).

   $\frac{1}{2}$

3. Use SS (line 40) and SA (line 50) to shift fonts to use a character from another set.

NOTE: Although some program lines are shown on two lines to fit on this page, you should write them on just one line.■

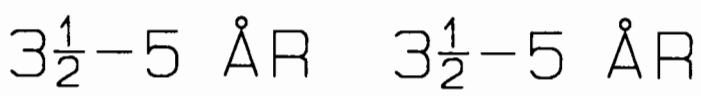
```
10  'Insert configuration statement here
20  PRINT #1, "INPS7000,5000SP1"
30  PRINT #1, "PA1000,4000SD1,21,2,0,7,48AD1,83,2,1,7,50"
40  PRINT #1, "SSLB USASCII, fixed-vector font"+CHR$(3)
50  PRINT #1, "CP-22,-2SALB SPANISH SET, variable-arc font"
      +CHR$(3)
60  PRINT #1, "CP-17,-2LB]su se"+CHR$(124)+"as?"+CHR$(3)
70  PRINT #1, "PG;"
80  END
```



4. If you need to use a special character from another set in the middle of a label string, using SS and SA to toggle between sets can be inefficient. Instead, use the control characters shift-in (**SI**, decimal code 15) and shift-out (**SO**, decimal code 14) to toggle between the sets. You can use either the keyboard method or the CHR$ method to shift-in/shift-out. The following example shows both methods.

NOTE: Although some program lines are shown on two lines to fit on this page, you should write them on just one line.■

```
10  'Insert configuration statement here
20  PRINT #1, "INPS8000,5000SP1"
30  PRINT #1, "PA300,1000SD1,115,3,3,4,30AD1,5,3,3,4,30"
40  PRINT #1, "LB3"+CHR$(14)+"x"+CHR$(15)+"-5]R"+CHR$(3)
50  PRINT #1, "CP2,0"
60  PRINT #1, "LB3"+CHR$(14)+CHR$(120)+CHR$(15)+"-5"
      +CHR$(93)+"R"+CHR$(3)
70  PRINT #1, "PG;"
80  END
```



Using Character Sets

Your plotter will implement at least one of three different types of fonts.

- **Scalable outline font** — Font characters can be displayed at any size. The characters are defined as a set of geometrical relationships. A scalable outline character can be resized (using SI and SR), rotated (using RO, DI, and DR), and distorted (using SL).

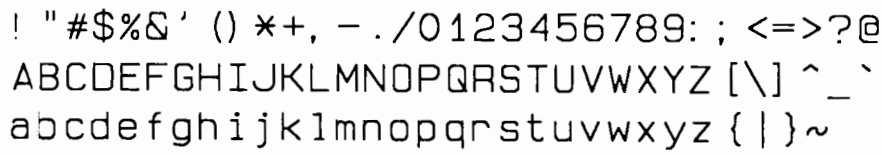
- **Bitmap font** — Raster pattern characters defined as a set of dots. A bitmap character cannot be transformed using DI, DR, SI, SR, or SL. Some devices do not support bitmap fonts. (For more information, refer to the Scalable or Bitmap Fonts [SB] instruction in chapter 11, *The Dual-Context Extension*.)

- **Stick font** — Characters are drawn as a series of vectors. The characters are defined as a set of endpoints. You can resize (using SI or SR), rotate (using DI, DR, and RO), and distort (using SL) stick fonts. Stick fonts are defined on a dimensionless grid. The main body of each character fits within a 32-by-32-unit box, with descenders extending beneath. All HP-GL/2 devices support stick fonts.

The examples in this chapter use stick fonts.

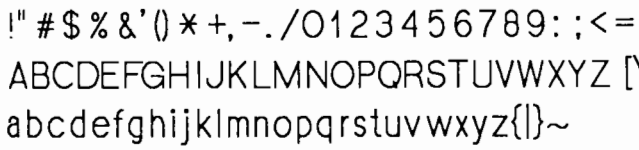
HP Roman8 is the default character set for all HP-GL/2 devices. Refer to your plotter's user or HP-GL/2 documentation to determine what character sets your plotter supports. Pen plotters generally support several character sets in one of the following types of fonts: vector, arc, and drafting.

- **Fixed-vector font** — The horizontal space for all characters is equal. All characters are always drawn using a fixed number of vectors.



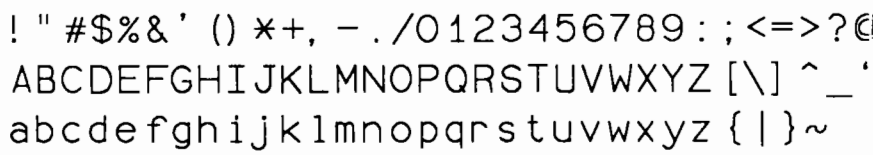
Fixed-Vector Font

- **Variable-arc font** — Characters are proportionately spaced (the amount of horizontal space occupied by each character varies with the character) and are drawn using arcs for smoother contours.



Variable-Arc Font

- **Fixed-arc font** — The horizontal space for all characters is equal. Characters are drawn using arcs for smoother contours.



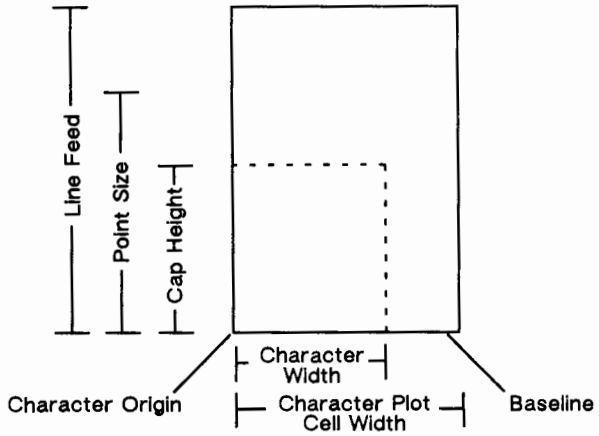
Fixed-Arc Font

- **Drafting font** — Characters are designed to provide reliable character recognition in situations where photo reduction may cause image degradation and loss of resolution. The characters are drawn in such a way as to avoid confusion between lines and figures, as described in the following table. The set also contains symbols used in drafting, such as ∞ or Δ. The HP Drafting font is a fixed-space vector font.

-B-	The bottom of the character is wider than the top.
-6-	Large body, stem curved but open.
-8-	Lower part larger than upper, full and round to avoid blur.
-9-	Large body, stem curved but open.

Working with the Character Plot (CP) Cell

In each font, the basis for each character or space is the character plot cell. This is also known as the CP cell. Think of the CP cell as a rectangular area around a character that includes blank areas above and to the right of the character. Refer to the following illustration.

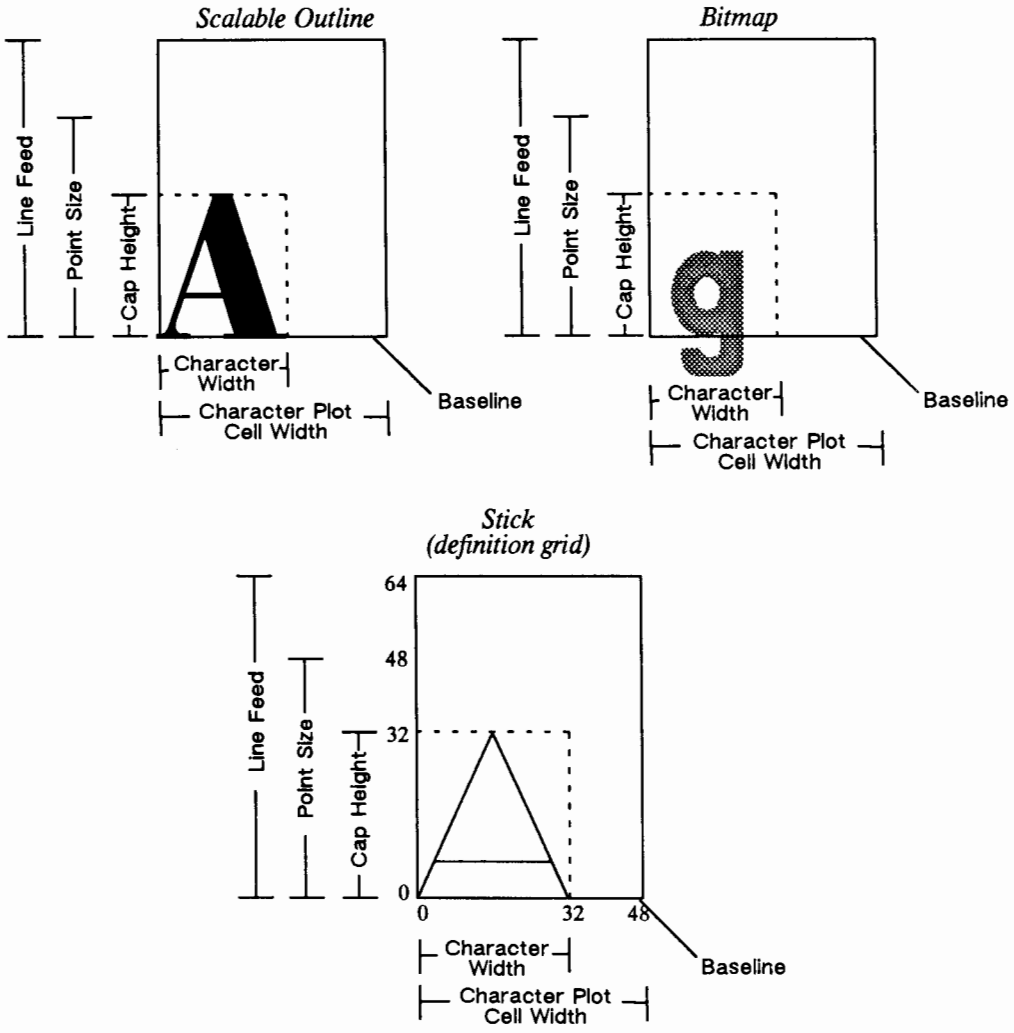


Term	Description
Baseline	The imaginary line on which a line of text rests. A character's descender (such as the bottom of a lowercase "g") extends below the baseline.
Line Feed	The distance from the baseline of a line of text to the baseline of the next character line above or below. For most fonts, the line feed is about 1.2 of the point size (1.33 of the point size for stick fonts).
Point Size	Traditional character measure roughly equivalent to the height of a capital letter plus the depth of a descender.
Cap Height	The distance from the baseline to the top of a capital letter.
Character Width	The horizontal area allocated for character rendering.
Character Plot Cell Width	The distance from the beginning of one character to the beginning of the next. The character plot cell width includes the character itself and the white space between it and the next character.
Character Plot (CP) Cell	A rectangular area defined by the character plot cell width and a line feed.
Character Origin	The lower-left corner of the character plot cell.

Your device will implement at least one of three different types of fonts.

- Scalable outline font
- Bitmap font
- Stick font

The following shows each type of font in relation to its character plot cell.

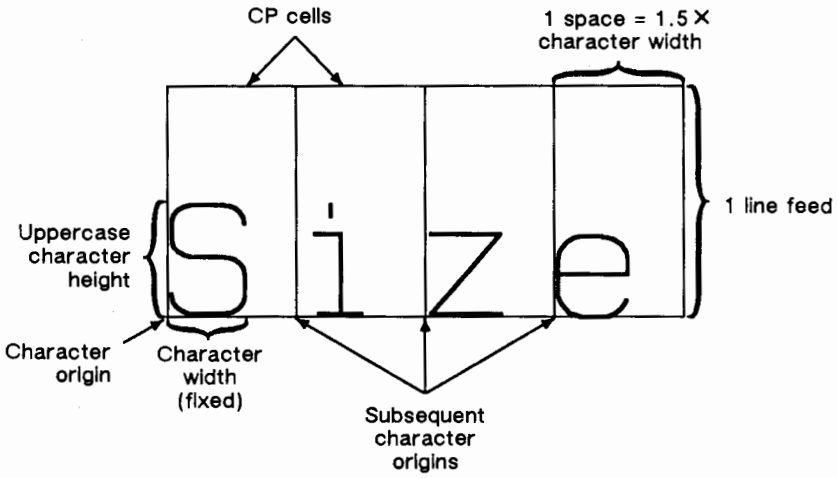


When you use the SI (Absolute Character Size) or SR (Relative Character Size) instructions to change the size of the characters or use the ES (Extra Space) instruction to add extra space around them, you alter the size of the CP cell.

Plotting with Fixed- and Variable-Space Fonts

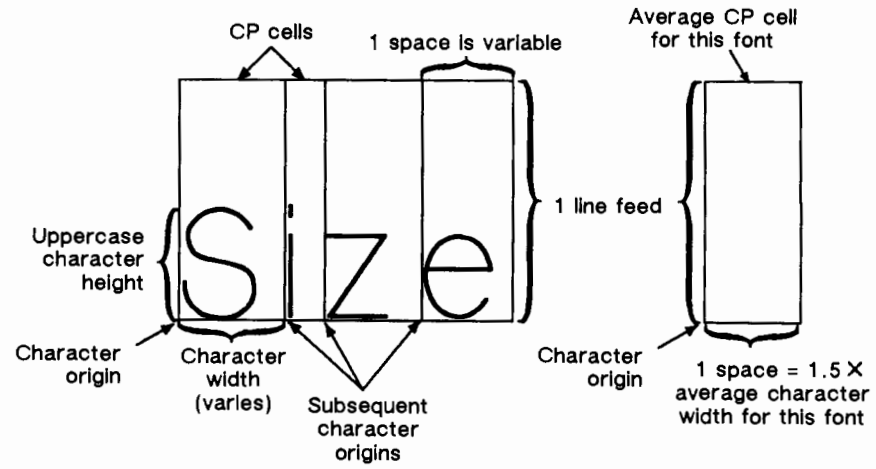
The variable-space fonts, by definition, use different widths for each letter. This variance produces some differences in the definition of the character plot cell and in the way some of the labeling instructions work with variable-space fonts. These differences are described in this section.

The following shows the difference between fixed- and variable-space fonts.



Fixed-Space Font

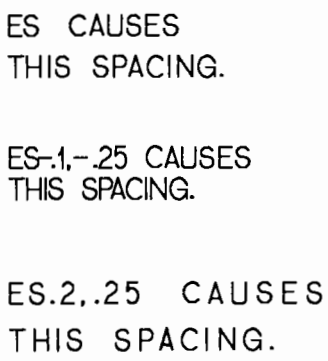
With variable-space fonts, the character plot cell width is defined as 1½ times the *average* character width. The actual space used by each character will vary according to the character's width. Refer to the following illustration.



Variable-Space Font

With variable-space fonts, the CP instruction and the ES instruction use the average character plot cell in computing lines and spaces. Both of the character size instructions (SI and SR) use the average character plot cell in calculating character size. Otherwise, these instructions behave the same as they do with fixed-space fonts.

Note that using ES with variable-space fonts can make them look as though they were fixed-space. Refer to the following illustration.



How Your Device Selects Fonts

The following summarizes the procedure that your device follows to select a font. The criteria are based on the parameters of the AD and SD instructions. The procedure is necessary because fixed-space and variable-space fonts use different criteria to determine text size. For fixed-space fonts, pitch determines size. The height parameter of the AD and SD instructions is only used to distinguish between fonts with equal pitch. For variable-space fonts, the height parameter of the AD and SD instructions determines text size; the pitch parameter is ignored. Your HP-GL/2 device performs the following steps in order.

1. If the specified *character set* does not exist, the plotter uses the user default set (set from the control panel, if any); otherwise, the plotter uses the factory default set.

2. If the specified *font spacing* (fixed/variable) is available, it is used; otherwise, the plotter uses the remaining spacing option.

   NOTE: A fixed-space scalable font will not be eliminated from consideration because of the selection of proportional font spacing. It wil be considered proportional for purposes of font selection.■

3. If the remaining fonts are proportionally spaced, *pitch* is ignored. For fixed-space fonts, if the specified pitch is not available, the plotter selects the next *greater* pitch. If no greater pitch is available, the plotter selects the closest available smaller pitch. (A pitch of 12 characters per inch [cpi] is greater than a pitch of 10 cpi; greater pitch means smaller characters.)

   NOTE: Any specified pitch is available for scalable fonts.■

4. The plotter selects the closest available height to the specified *height* parameter. The closest available height is in terms of absolute difference. For example, if the device has 6-, 8-, and 12-point fonts and the specified height is 10, both 8- and 12-point fonts are selected for the next selection criterion. All fonts with heights within a quarter-point of the specified height are considered to have the specified height.

   NOTE: Any specified height is available for scalable fonts.■

5. If the specified *posture* (upright/italic) is available in the remaining fonts, the plotter selects that posture; otherwise, this attribute is ignored.

6. If the specified *stroke weight* is available in the remaining fonts, the plotter selects that stroke weight.

   If the specified stroke weight is greater than or equal to 0 (zero) and is not available, the plotter selects the next thicker stroke weight. If no thicker stroke weight is available, the plotter selects the next thinner stroke weight.

   If the specified stroke weight is less than 0 (zero) and is not available, the plotter selects the next thinner stroke weight. If no thinner stroke weight is available, the plotter selects the next thicker stroke weight.

   NOTE: Any specified stroke weight is available for scalable fonts.■

7. If the specified *typeface* is available, the plotter selects that typeface; otherwise, the plotter ignores this attribute.

   NOTE: The stick fonts are typefaces 48, 49, and 50. The character set and posture selections must match an available stick font.■

8. If more than one font emerges after the above procedure, the location of fonts provides the following order of priority.

   - Downloaded bitmap soft fonts in ascending font ID order.
   - Downloaded scalable soft fonts in ascending font ID order.
   - Bitmap external cartridge fonts.
   - Scalable external cartridge fonts.
   - Bitmap internal fonts.
   - Scalable internal fonts.

9. If multiple fonts remain, a font in the specified orientation is selected. If none of the fonts is defined in the specified orientataion, automatic rotation is applied to one of the remaining fonts.

Using Variables in Labels

You may want to use numeric or string variables in your label instructions instead of writing the numbers or text in each instruction. The variables themselves are not enclosed in quotation marks because the computer program works with them—it substitutes the variable definition for the variable when it issues the instruction. Further, the variables are separated from the quotation marks by another punctuation mark, typically a comma or semicolon, called a delimiter. Note that the punctuation mark (delimiter) you use to separate the variable from the instruction determines the format that the number is printed in and can have a major effect on the appearance of your label.

Many computer languages use the quotation mark (single or double) to define the literal (actual) characters to be sent. The graphics instructions are enclosed in quotation marks because the computer program itself does nothing with them; since they're in quotation marks, it just sends them along to the plotter.

- **Commas** in BASIC typically cause numbers and text to be right-justified in a specific character field width. (The field width may be different for numbers and text.) For example, if the character field width is 15 characters, a 5-digit number will be indented 10 spaces to be printed in spaces 11 through 15.

- **Semicolons** in BASIC typically suppress the extra blank spaces and are recommended for use in labels. However, in many computer languages, numbers will still be printed with a blank space in front for the sign and one after for separation. Text will be printed with no spaces between the variable strings.

  Any blank spaces that you need within a label should be sent within the quotation marks as part of the character string.

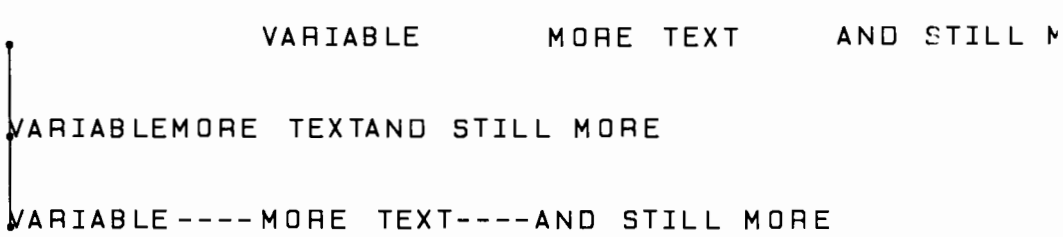
The following illustrations show the use of variables with a comma, a semicolon, and spaces within the quotation marks (hyphens are used for visual clarity). The vertical line indicates the current pen location when the Label instruction was issued. The first example uses numeric variables; the second uses string variables.

Numeric Variables

```
10  'Insert configuration statement here
20  PRINT #1, "INPS7000,5000SP1SI.187,.269PA1500,3000"
30  X=50
40  PRINT #1, "LB",X,X+1,X+2,+CHR$(3)
50  PRINT #1, "PA1500,2500"
60  Y=50
70  PRINT #1, "LB";Y;Y+1;Y+2;+CHR$(3)
80  PRINT #1, "PA1500,2000"
90  Z=50
100 PRINT #1, "LB";Z;"----";Z+1;"----";Z+2;+CHR$(3)
110 PRINT #1, "PA1500,3000WG20,0,360"
120 PRINT #1, "PD1500,2500WG20,0,360"
130 PRINT #1, "PD1500,2000WG20,0,360"
140 PRINT #1, "PG;"
150 END
```

NOTE: This version of GW-BASIC uses such a wide character field width with commas (line 60) that the "AND STILL MORE" is plotted off the right side of the page.■

```
10  'Insert configuration statement here
20  PRINT #1, "INPS5000,4000SP1SI.187,.269PA0,3000"
30  X$="VARIABLE"
40  Y$="MORE TEXT"
50  Z$="AND STILL MORE"
60  PRINT #1, "LB",X$,Y$,Z$,+CHR$(3)
70  PRINT #1, "PA0,2500"
80  PRINT #1, "LB"X$Y$Z$+CHR$(3)
90  PRINT #1, "PA0,2000"
100 PRINT #1, "LB"X$"----"Y$"----"Z$+CHR$(3)
110 PRINT #1, "PA0,3000WG20,0,360"
120 PRINT #1, "PD0,2500WG20,0,360"
130 PRINT #1, "PD0,2000WG20,0,360"
140 PRINT #1, "PG;"
150 END
```



Adding Carriage Returns and Line Feeds to Labels

Carriage returns and line feeds are nonprinting ASCII control characters. You can insert these formatting characters using a character string function like CHR$ or by producing them directly from the keyboard. The examples in this manual use the BASIC CHR$ function. On some computers, you can produce these ASCII control characters from the keyboard by pressing two keys simultaneously. Check your computer documentation for the availability of this option and for the appropriate combination of keys on your system. Here are some common ASCII control characters and their equivalents.

Control Character	CHR$ Function	Keyboard Equivalent
ETX (end-of-text)	CHR$(3)	**CONTROL** and **C**
LF (line feed)	CHR$(10)	**CONTROL** and **J**
CR (carriage return)	CHR$(13)	**CONTROL** and **M**

When you use a string function such as CHR$, you must separate it from the label string to prevent the plotter from drawing it. Separate it by closing the quotation marks, then inserting the string function. To ensure that the character produced by the string function is sent immediately after the label string without any extra space, you may need to link the label and the string function with a concatenation symbol such as **+**, **&**, **;**, or **,**. For example, in BASIC use: **...label" + CHR$(3)**. Use the symbol that your system and language require.

If you enter the character directly from the keyboard, you can enter the character within the label; you do not need to close quotes or use a concatenation symbol as you do with the CHR$ function.

The following instructions use string functions.

```
"LBLabel"+CHR$(3)

"LBLabel"+CHR$(13)+CHR$(10)+"return"+CHR$(3)
```

The following shows equivalent instructions using the keyboard. (The control codes **ETX**, **CR**, and **LF** are represented by $^{E}_{X}$, $^{C}_{R}$, $^{L}_{F}$, respectively; refer to your computer documentation for the characters that represent these control codes.)

```
"LBLabel^EX"

"LBLabel^CR^LFreturn^EX"
```

Both methods produce the following labels.

Moving to the Carriage Return Point

When you begin labeling, the current pen location is the carriage return point. When the plotter encounters a carriage return—**CR** or CHR$(13)—the pen moves to the carriage return point, adjusted up or down by any line feeds.

The following instructions update the current pen location and the carriage return point along with it: AA, AR, AT, DF, DI, DR, DV, IN, LO, PA, PE, PR, RO, and RT. A PD or PU instruction *with parameters* also updates the carriage return point. The CP instruction with a nonzero *lines* parameter updates the carriage return point's vertical location.

NOTE: If the plotter has a control panel, it may allow you to change the location of the pen, rotate the coordinate system, or initialize the plotter. Performing any of these functions from the control panel also updates the carriage return point.■

The LB instruction does *not* update the carriage return point to the current pen location, but continues labeling from the current pen location. This feature allows you to issue several label instructions that write one long label and still use a carriage return to get to the beginning of the entire label. However, when you embed carriage return characters in a label, you automatically position each portion of the label according to the label origin.

Control Characters

You can effectively use the following control characters in labels. All other control codes are ignored.

Decimal Value	ASCII Character	Associated Function
8	BS	Backspace
9	HP	Horizontal tab
10	LF	Line feed
13	CR	Carriage return
14	SO	Shift-out (equivalent to Select Alternate Font [SA] instruction)
15	SI	Shift-in (equivalent to Select Standard Font [SS] instruction)

NOTE: Character code values 0–31 and 128–159 are all defined as control codes in the 8-bit compatible character sets. In the 7-bit character sets, the character codes 128–255 are undefined. For 7-bit character sets, the plotter ignores characters 128–159 and prints spaces for characters 160–255.■

Enhancing Labels

While you can set most font attributes through the AD and SD instructions, you can further enhance your labels by changing such aspects as the character size and slant, the space between characters and lines, and the orientation and/or placement of the label on the page. To effectively use these enhancements, you should understand the properties of the character plot (CP) cell. Refer to *Working with the Character Plot Cell* earlier in this chapter.

Character Size and Slant

You can change the size and aspect ratio of the characters using the Absolute Character Size and the Relative Character Size (SI and SR) instructions. The Absolute Character Size instruction establishes the size of the uppercase "A" in centimeters and maintains this character size independent of the location of P1 and P2. The Relative Character Size instruction establishes the size of the uppercase "A" as a percentage of the distance between P1 and P2. Subsequent changes in the location of P1 and P2 will cause the character size to change. Changing the character size changes the size of the CP cell and proportionally changes the stroke weight used in labels (refer to AD and SD).

You can use the Character Slant (SL) instruction to slant the characters at a specified angle in either direction from the left vertical side of the CP cell. The CP cell is not altered.

Character Spaces and Text Lines

You can use the Extra Space (ES) instruction to insert extra spaces or lines between all characters or text lines in labels. For example, you could use ES to skip a space between every character (such as, "M E M O R A N D U M") or to skip a line between every line of text, double-spacing your text. You can also decrease the spacing between characters and lines.

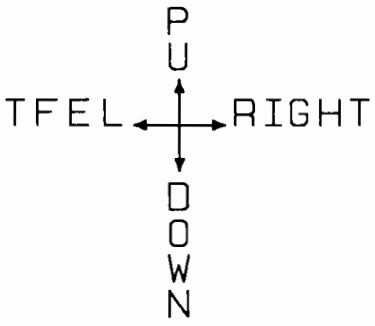
You can use the Character Plot (CP) instruction to move the pen a specific number of lines or spaces (CP cells) from the current pen location. Use the CP instruction, for example, to indent a label a certain number of spaces.

Label Orientation and Placement

You can place your labels anywhere on the page with any orientation you want. The Direction Absolute (DI) instruction specifies the angle at which you want the characters drawn, independent of the location of P1 and P2. The Direction Relative (DR) instruction specifies the angle at which you want the characters drawn as a function of the P1 and P2 distance; thus when you change P1 and P2, the label angle changes to maintain the same orientation.

DI and DR allow you to plot labels at any angle with the letters in their normal side-by-side orientation.

The Define Variable Text Path (DV) instruction allows you to specify the text path (right, left, up, or down) and the direction of line feeds with respect to the text path.



The Label Origin (LO) instruction greatly simplifies placing labels on a plot. Normally the first character origin is the current pen location when the Label instruction is issued. The LO instruction allows you to specify that the label be oriented from a point (in any direction you choose), offset distinctly, centered, and right- or left-justified from the current pen location. For example, the following shows four centered lines of text.

Lines of any length
can easily be
centered
without cumbersome calculations.

These lines use one (X,Y) coordinate pair, one LO instruction to center labels, and a carriage return and line feed after each line. An alternative method would involve calculating the length of the line in CP cells, dividing by 2, and using the CP instruction to "backspace" the required number of cells—and that's just the first line. Using LO saves you from calculating the number of characters sent to the plotter and allows you to take advantage of proportional fonts when the character widths are not known to your program.

Terminating Labels

LB tells the plotter to draw everything following the instruction, rather than interpret the characters as graphics instructions. Since the semicolon (;) is normally a text character, you must use a special "label terminator" to tell the plotter to return to interpreting characters as graphics instructions. You must use the label terminator, or the plotter will label the rest of your program on your plot.

The default label terminator is the nonprinting ASCII end-of-text character **ETX** (decimal code 3). In BASIC, this is accessed by the character string function CHR$(3). You can change the label terminator using the Define Label Terminator (DT) instruction.

AD, Alternate Font Definition

USE: Defines an alternate character set and its attributes: font spacing, pitch, height, stroke weight, and typeface. Use AD to set up an alternate character set that you can easily access when labeling. For designating a standard font, refer to the SD instruction.

SYNTAX: AD*kind,value... (,kind,value;)*
or
AD*(;)*

Parameter	Format	Functional Range	Default
kind	clamped integer	1 to 7	no default
value	clamped real	*kind*-dependent*	*kind*-dependent*

* Refer to the table following the parameter descriptions.

REMARKS: For pen plotters, the AD instruction defines a second character set to be used in labeling instructions. For electrostatic plotters and PCL devices, the AD instruction lets you define another font and its attributes. This instruction remains in effect until the plotter receives a DF, IN, or another AD instruction.

- **No Parameters** — Defaults the alternate font attributes. (The default font set of attributes is device-dependent, but the default font will always be a scalable font, not a bitmap font.)

- **Kind** — Specifies the attribute for which you are setting a value.

Kind	Attribute	Default Value	Description
1	Character set	277	Roman8
2	Font spacing	0	fixed spacing
3	Pitch	*media-dependent**	characters per inch
4	Height	*media-dependent**	font point size
5	Posture	*device-dependent*	upright or italic
6	Stroke weight	0	normal
7	Typeface	48	stick (fixed-vector)

* The default value for pitch is usually 9 cpi; for E-size media it is 5.9 cpi. The default character height value is usually 11.5; for E-size media it is 16.

- **Value** — Defines the characteristics of the attribute specified by the kind parameter.

The following tables list the kind parameters with their associated values. For kinds 1 (character set) and 7 (typeface), your plotter may support values other than those listed here. Refer to your user's guide or HP-GL/2 option manual for the attributes and values that are supported.

Kind 1: Character Set

The character set attribute defines the character set to be used as the alternate set. The values are listed in the order of the PCL identification. Note that your plotter may not support all of these character sets. Refer to the AD or SD instruction in your user or HP-GL/2 option documentation for a list of the character sets your device supports.

Kind 1: Character Set Values	Description	ISO Number
0 (and 277)	Roman8 (default)	—
1	Math-7	—
2	Line Draw-7	—
3	HP Large Characters	—
4	Norwegian v1	60
36	Norwegian v2	61
5	Roman Extensions	—
37	United Kingdom	4
6	French v1	25
38	French v2	69
7	HP German	—
39	German	21
263	Greek-8	—
8	Hebrew-7	—
264	Hebrew-8	—
9	Italian	15
202	Microsoft Publishing	—
234	DeskTop	—
330	PS Text	—
426	Ventura International	—
458	Ventura U.S.	—
11	JIS ASCII	14
43	Katakana	13
75	Chinese	57
267	Kana-8	—
299	Korean-8	—
12	Line Draw-7	—
44	HP Block Characters	—
76	Tax Line Draw	—
268	Line Draw-8	—
300	Ventura ITC Zapf Dingbats	—
332	PS ITC Zapf Dingbats	—
364	ITC Zapf Dingbats Series 100	—
396	ITC Zapf Dingbats Series 200	—
428	ITC Zapf Dingbats Series 300	—

(Continued)

Kind 1: Character Set Values	Description	ISO Number
13	Math-7	—
45	Tech-7 (DEC)	—
173	PS Math	—
205	Ventura Math	—
269	Math-8	—
14	ECMA-94 Latin 1 (8-bit version)	—
78	ECMA-94 Latin 2	—
174	ECMA-128 Latin 5	—
334	ECMA-113/88 Latin/Cyrillic	—
15	OCR-A	—
47	OCR-B	—
79	OCR-M	—
16	APL (typewriter-paired)	—
48	APL (bit-paired)	—
145	PC Line	—
18	Cyrillic ASCII	—
50	Cyrillic	—
114	PC Cyrillic	—
19	Swedish for names	11
51	HP Spanish	—
83	Spanish	17
115	Swedish	10
147	Portuguese	16
179	Portuguese	84
211	Spanish	85
243	HP European Spanish	—
275	HP Latin Spanish	—
531	HP-GL Download	—
563	HP-GL Drafting	—
595	HP-GL Special Symbols	—
20	Thai-8	—
276	Turkish-8	—
21	ANSI US ASCII	6
53	Legal	—
85	International Reference Version	2
181	HPL Language Set	—
245	OEM-1 (DEC Set)	—
277 (and 0)	Roman8 (default)	—
309	Windows	—
341	PC-8	—
373	PC-8 Denmark/Norway	—
405	PC-850	—
501	Pi Font	—
565	PC-852	—
22	Arabic (MacKay's Version)	—
278	Arabic-8	—

(Continued)

Kind 1: Character Set Values	Description	ISO Number
25	3 of 9 Barcode	—
57	Industrial 2 of 5 Barcode	—
89	Matrix 2 of 5 Barcode	—
153	Interleaved 2 of 5 Barcode	—
185	CODABAR Barcode	—
217	MSI/Plessey Barcode	—
249	Code 11 Barcode	—
281	UPC/EAN Barcode	—
505	USPS Zip	—
26	Not used	—

Kind 2: Font Spacing

The font-spacing attribute defines whether the spacing is fixed (all characters having equal width) or variable (each character occupying a space proportional to its size). Refer to *Plotting with Fixed- and Variable-Space Fonts* earlier in this chapter.

Kind 2: Font Spacing Values	Description
0	fixed spacing (default)
1	variable spacing

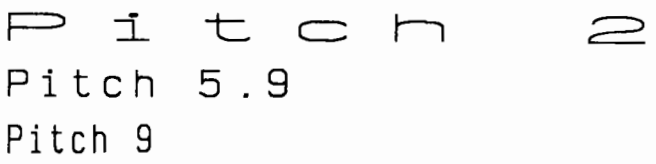
Kind 3: Pitch

The pitch attribute defines the number of characters per inch.

Kind 3: Pitch Values	Description
0 to 32 767.9999	*media-dependent**

* The default value is usually 9 cpi; for E-size media it is 5.9 cpi.

The following illustration shows text in three different pitches. Fixed-space fonts depend on pitch to determine character size. Proportional fonts ignore pitch. Note that with the AD instruction, you cannot create tall, skinny characters or short, wide characters; the character aspect ratio is preserved.



Kind 4: Height

The height attribute defines the font point size (i.e., the height of the character plot cell). Variable-space fonts depend on the height parameter to determine character size; fixed-space fonts depend on pitch for character size. Note that with the AD instruction, you cannot create tall, skinny characters or short, wide characters; the character aspect ratio is preserved.

Kind 4: Height Values	Description
0 to 32 767.9999	*media-dependent**

* The default character height value is usually 11.5; for E-size media it is 16.

The following illustration shows text in three different heights.

Kind 5: Posture

Posture defines the character's vertical position. The default posture is device-dependent (but usually upright).

Kind 5: Posture Values	Description
0	upright
1	italic

Kind 6: Stroke Weight

The stroke weight attribute defines the line width used in labels. The default stroke weight is device-dependent. Note that stroke weight is affected by changes in P1 and P2 locations when relative sizing is in effect; refer to the SR instruction.

Kind 6: Stroke Weight Values	Description
-7	very light
-3	light
0	normal
3	bold
7	very bold
9999	* (for stick font only)

* When the stick font (typeface 48) is selected, the value 9999 causes the stick font to be rendered in the current pen width.

The following illustration shows labels in five different stroke weights.

Kind 7: Typeface

The typeface attribute selects the font, which defines the style of the lettering. Characters drawn with arcs have a smoother, more rounded appearance than characters drawn with vectors. All HP-GL/2 devices support the stick fonts (48, 49, and 50). The drafting font is for use with the HP Drafting character set.

Kind 7: Typeface Values	Description
0	Line Printer or Line Draw
1	Pica
2	Elite
3	Courier
4	Helvetica
5	Times Roman
6	Letter Gothic
7	Script
8	Prestige
9	Caslon
10	Orator
11	Presentation
13	Serifa
14	Futura
15	Palatino
16	ITC Souvenir
17	Optima
18	ITC Garamond
20	Coronet
21	Broadway
23	Century Schoolbook
24	University Roman
27	ITC Korinna
28	Naskh (generic Arabic typeface)
29	Cloister Black
30	ITC Galliard
31	ITC Avant Garde Gothic
32	Brush
33	Blippo
34	Hobo
35	Windsor
38	Peignot
39	Baskerville
41	Trade Gothic
42	Goudy Old Style
43	ITC Zapf Chancery
44	Clarendon
45	ITC Zapf Dingbats
46	Cooper
47	ITC Bookman
48	Stick (default)
49	HP-GL Drafting
50	HP-GL fixed arc
51	Gill Sans
52	Univers
53	Bodoni
54	Rockwell
55	Melior

(Continued)

Kind 7: Typeface Values	Description
56	ITC Tiffany
57	ITC Clearface
58	Amelia
59	Park Avenue
60	Handel Gothic
61	Dom Casual
62	ITC Benguiat
63	ITC Cheltenham
64	Century Expanded
65	Franklin Gothic
68	Plantin
69	Trump Mediaeval
70	Futura Black
71	ITC American Typewriter
72	Antique Olive
73	Uncial
74	ITC Bauhaus
75	Century Oldstyle
76	ITC Eras
77	ITC Friz Quadrata
78	ITC Lubalin Graph
79	Eurostile
80	Mincho
81	ITC Serif Gothic
82	Signet Roundhand
83	Souvenir Gothic
84	Stymie
87	Bernhard Modern
89	Excelsior
90	Grand Ronde Script
91	Ondine
92	P.T. Barnum
93	Kaufman
94	ITC Bolt Bold
96	Helv Monospaced
97	Revue
101	Garamond (Stempel)
102	Garth Graphic
103	ITC Ronda
104	OCR-A
105	ITC Century
106	Englische Schreibschrift
107	Flash
108	Gothic Outline (URW)
109	Stencil (ATF)

(Continued)

Kind 7: Typeface Values	Description
110	OCR-B
111	Akzidenz-Grotesk
112	TD Logos
113	Shannon
114	ITC Century
152	Maru Gosikku
153	Gosikku (Kaku)
154	Socho
155	Kyokasho
156	Kaisho

The following examples show how to select each of the plotter's three fonts for the default character set (HP Roman8).

Fixed-vector `(AD7,48)`
Variable-arc `(AD2,1,7,5Ø)`
Fixed-arc `(AD2,Ø,7,5Ø)`

Related Instructions	Group
LB, Label SA, Select Alternate Font SD, Standard Font Definition SI, Absolute Character Size SR, Relative Character Size SS, Select Standard Font	*The Character Group*

CF, Character Fill Mode

USE: Specifies the way outline fonts will be rendered. It also changes the fill pattern for bitmap and stick fonts. If your plotter does not support outline fonts, this instruction is NOP'd (performs no operation); pen plotters generally do not support outline fonts.

SYNTAX: CF*fill mode(,edge pen;)*
or
CF*(;)*

Parameter	Format	Functional Range	Default
fill mode	clamped integer	0, 1, 2, or 3	0 (solid fill, edged)
edge pen	integer	*device-dependent*	0

REMARKS: For the following explanations note that bitmap and stick fonts cannot be edged.

- **No Parameters** — Defaults characters to solid fill with no edging. Equivalent to *CF0,0.*

- **Fill Mode** — Specifies how the plotter will render filled characters according to the following parameter values.

Mode	Fill Attribute	Edge Attribute
0	Solid fill using the current pen.	Edged with specified pen (or current pen if *edge pen* parameter is not specified).
1	No fill.*	Edged with specified pen (or current pen if *edge pen* parameter is not specified).
2	Uses current fill type.** If the fill pattern does not incorporate color information, the currently selected pen is used.	No edging (instruction ignores *edge pen* parameter if specified).
3	Uses current fill type.** If the fill pattern does not incorporate color information, the currently selected pen is used.	Edged with specified pen (or current pen if *edge pen* parameter is not specified).

* Characters are filled *only* if they cannot be edged.
** Refer to the FT instruction in chapter 7, *The Line and Fill Attributes Group.*

- **Edge Pen** — Specifies the pen number to use for edging. Edging width is device-dependent and varies with character size; it is not determined by the width of the pen specified as the edge pen.

Note that DI and DR do not cause rotation of fill patterns. Fill patterns remain fixed with respect to the current coordinate system. The CF instruction remains in effect until another CF instruction is executed, or the plotter is initialized or set to default conditions.

Related Instructions	Group
DI, Absolute Direction DR, Relative Direction	*The Character Group*
FT, Fill Type	*The Line and Fill Attributes Group*

CP, Character Plot

USE: Moves the pen the specified number of spaces and lines defined by the character plot cells from the current pen location. Use CP to position a label for indenting, centering, etc.

SYNTAX: CP*spaces,lines(;)*
or
CP*(;)*

Parameter	Format	Functional Range	Default
spaces	clamped real	*device-dependent*	no default
lines	clamped real	*device-dependent*	no default

REMARKS: The CP instruction includes an automatic pen up. When the instruction is completed, the original pen up/down position is restored. For more information on spaces, lines, and the character plot cell, refer to *Working with the Character Plot Cell* earlier in this chapter.

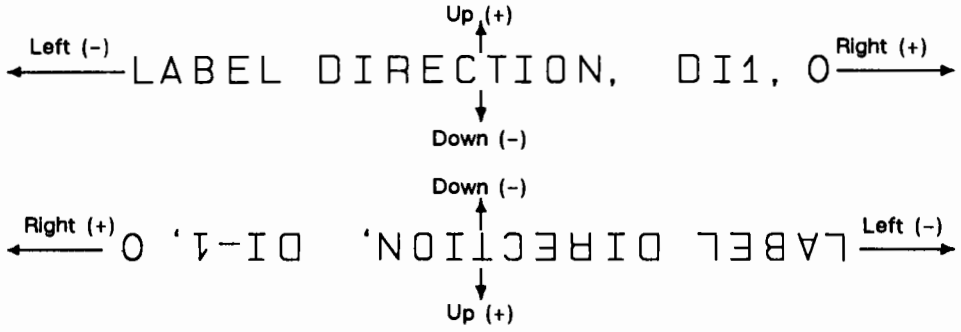
- **No Parameters** — Performs a carriage return and line feed (moves to the carriage return point and one line down).

- **Spaces** — Specifies the number of spaces (character plot cell widths) the pen moves relative to the current pen location. Positive values move the pen to the right of the current pen location; negative values move the pen to the left. Right and left are relative to current label direction. The width of the character plot cell is uniquely defined for each font; use ES to adjust the width.

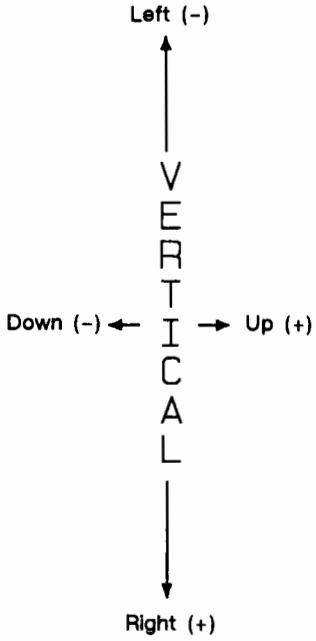
- **Lines** — Specifies the number of lines (character plot cell heights) the pen will move relative to the current pen location. Positive values move the pen up from the current pen location; negative values move the pen down (a value of -1 is equivalent to a normal line feed). Up and down are relative to the current label direction. The line feed distance is uniquely defined for each font; use ES to adjust the height.

  When you move the pen up or down a specific number of lines, the carriage return point shifts up or down accordingly.

The illustration below shows the effect of label direction on the positive and negative parameters.



The following illustration shows the direction of labeling with a vertical text path (refer to the DV instruction).



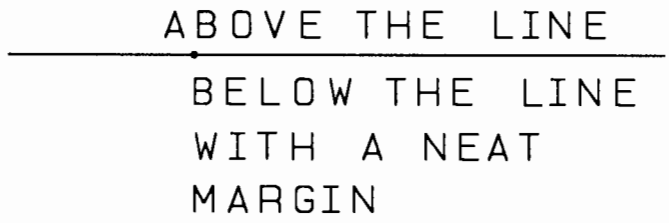
CP moves, pen up, from the current character origin to the next character origin. CP *affects the next label only*; you must issue new CP instructions for subsequent labels.

EXAMPLE: This example produces lettering along a line (but not directly on top of it) and aligns labels along a left margin.

The CP instruction in line 30 moves the next label 0.35 lines up so it will be drawn just above the line. The PA instruction in line 40 establishes a new carriage return point (the current pen location when the LB instruction is issued); the dot on the line marks the new carriage return point. Notice that the CP instruction without parameters (*CP;*) in line 60 performs the same function as the carriage return and line feed (*CHR$(13) + CHR$(10)*) in line 50.

NOTE: Although some program lines are shown on two lines to fit on this page, you should write them on just one line.■

```
10  'Insert configuration statement here
20  PRINT #1, "INPS7000,5000SP1PA5000,2500PD1500,2500PU"
30  PRINT #1, "CP5,.35LBABOVE THE LINE"+CHR$(3)
40  PRINT #1, "PA2500,2500WG20,0,360"
50  PRINT #1, "CP0,-.95LBBELOW THE LINE"+CHR$(13)+CHR$(10)+
      "WITH A NEAT"+CHR$(3)
60  PRINT #1, "CPLBMARGIN"+CHR$(3)
70  PRINT #1, "PG;"
80  END
```



Related Instructions	Group
DI, Absolute Direction DR, Relative Direction DV, Define Variable Text Path ES, Extra Space LO, Label Origin SI, Absolute Character Size SR, Relative Character Size	*The Character Group*

DI, Absolute Direction

USE: Specifies the slope or direction in which labels are drawn, independent of P1 and P2 settings. Use DI to change labeling direction when you are labeling curves in line charts, schematic drawings, blueprints, and survey boundaries.

SYNTAX: DI*run,rise(;)*
or
DI*(;)*

Parameter	Format	Functional Range	Default
run (or cos θ)	clamped real	*device-dependent*	1
rise (or sin θ)	clamped real	*device-dependent*	0

REMARKS: The DI instruction updates the carriage return point to the current location. While DI is in effect, with or without parameters, the label direction is not affected by changes in the locations of P1 and P2. However, the Define Variable Text Path (DV) instruction interacts with the DI (and DR) instructions, as explained later in this section.

- **No Parameters** — Defaults the label direction to absolute and horizontal (parallel to the X-axis). Equivalent to *(DI1,0)*.
- **Run, or Cos θ** — Specifies the X-component of the label.
- **Rise, or Sin θ** — Specifies the Y-component of the label.

  Together, the parameters specify the slope and direction of the label.

You can express the parameters as run and rise, or using the trigonometric functions cosine and sine according to the following relationship.

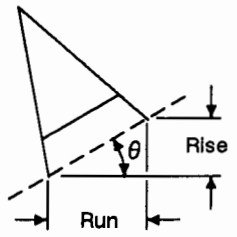
where: run and rise = number of measured units

θ = the angle measured in degrees

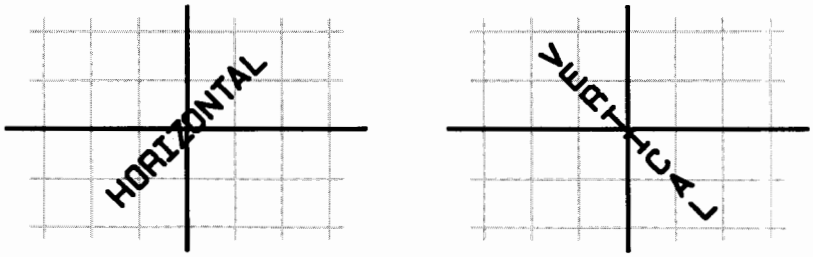
$$\frac{\sin\theta}{\cos\theta} = \frac{\text{rise}}{\text{run}}$$

$$\theta = \tan^{-1}\left(\frac{\text{rise}}{\text{run}}\right) \quad \text{and} \quad \tan\theta = \frac{\sin\theta}{\cos\theta}$$

Note that the run and rise determine the slope or angle of the baseline. Refer to the following illustration.

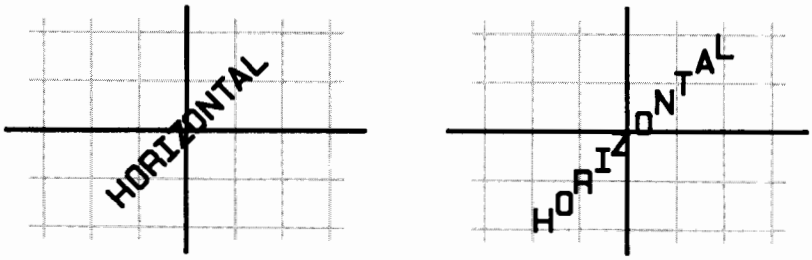


In horizontal mode (you haven't used the DV instruction), the run and rise appear to determine the slope of the entire label. However, if you have used the Define Variable Text Path (DV) instruction to label in a vertical path, the label appears to slant in the opposite direction even though the base of each letter is plotted on the same slope. The following illustration compares how labels plotted with the same run and rise parameters appear with horizontal (*DV0*) and vertical (*DV1*) text paths.



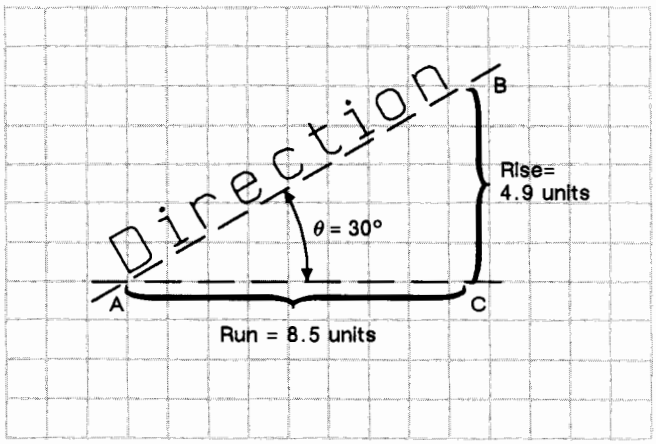
DV0 *DV1*

NOTE: *For devices that support the Dual-Context Extension* (see chapter 11): If a bitmap font is selected, the plotter draws each character in the label along the nearest perpendicular. In case of bisection, the angle is rounded down (e.g., 45° would round to 0°). Refer to the next illustration.■



Vector Font (SB0) *Bitmap Font (SB1)*

Suppose you want your label plotted in the direction shown in the following illustration. You can do this in one of two ways: measure the run and rise, or measure the angle.



(DI8.5,4.9)
or
(DI.866,.5)

To measure the run and rise, first draw a grid with the lines parallel to the X- and Y-axes. The grid units should be the same size on all sides, but their actual size is irrelevant. Then draw a line parallel to the label path (line AB in the illustration) and one parallel to the X-axis (line AC). The lines should intersect to form an angle.

Draw a line from point B along the Y-axis perpendicular to line AC. Count the number of grid units in line AC; this is the run. In the illustration above, the run is 8.5. Now, count the number of units along BC; this is the rise. In the illustration above, the rise is 4.9.

Your DI instruction using the run and rise would be (*DI8.5,4.9*).

If you know the angle (θ), you can use the trigonometric functions sine (SIN) and cosine (COS). In this example, θ = 30, COS 30 = 0.866, and SIN 30 = 0.5.

Your DI instruction using the SIN and COS would be (*DI.866,.5*). Whichever set of parameters you use, the label will be drawn as shown in the illustration above.

Note that the ratio of the SIN/COS parameters equals the ratio of the run/rise parameters. The ratio of the parameters to each other is more important than the actual values of the parameters.

Use the following table when you know the angle and want to specify the cosine and sine values. You can also use the SIN and COS functions available in most programming languages. The example at the end of this section shows both methods.

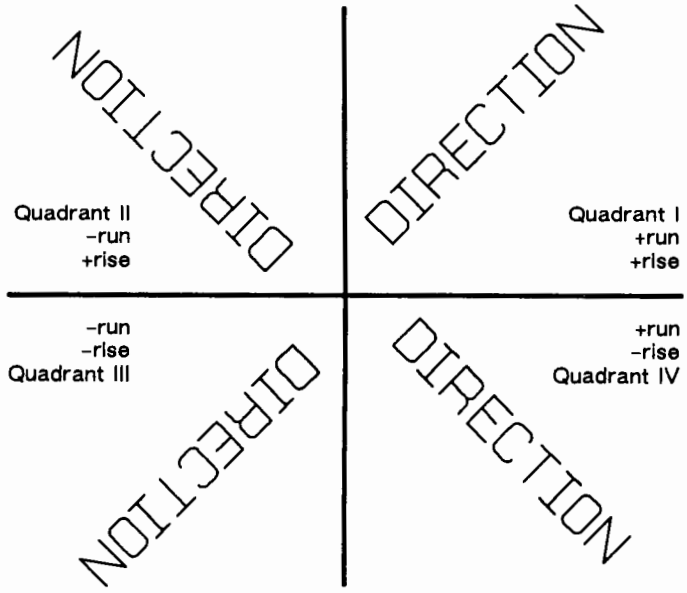
θ		Cosine	Sine
360	0	1	0
330	-30	0.87	-0.50
315	-45	0.71	-0.71
300	-60	0.50	-0.87
270	-90	0	-1
240	-120	-0.50	-0.87
225	-135	-0.71	-0.71
210	-150	-0.87	-0.50
180	-180	-1	0
150	-210	-0.87	0.50
135	-225	-0.71	0.71
120	-240	-0.50	0.87
90	-270	0	1
60	-300	0.50	0.87
45	-315	0.71	0.71
30	-330	0.87	0.50
0	-360	1	0

When using either method, at least one parameter must not be 0. The following table lists three common label angles produced by using 1s and 0s.

DI Instruction	Label Direction
DI 1,0	horizontal
DI 0,1	vertical
DI 1,1 or DI 0.7,0.7 (any parameters equal to each other)	45° angle

The relative size and sign of the two parameters determine the amount of rotation. If you imagine the current pen location to be the origin of a coordinate system for the label, you can see that the signs of the parameters determine which quadrant the label will be in.

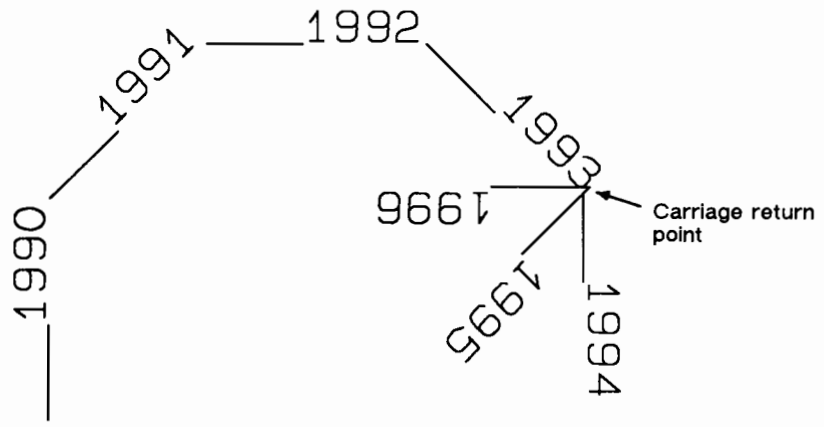
```
10  'Insert configuration statement here
20  PRINT #1, "INPS7000,5000SP1PA3500,2500"
30  PRINT #1, "DI1,1LB  DIRECTION"+CHR$(13)+CHR$(3)
40  PRINT #1, "DI1,-1LB  DIRECTION"+CHR$(13)+CHR$(3)
50  PRINT #1, "DI-1,-1LB  DIRECTION"+CHR$(13)+CHR$(3)
60  PRINT #1, "DI-1,1LB  DIRECTION"+CHR$(13)+CHR$(3)
70  PRINT #1, "PG;"
80  END
```



The DI instruction remains in effect until another DI or DR instruction is executed, or the plotter is initialized or set to default conditions.

EXAMPLE: This example illustrates the use of positive and negative parameters, the use of the BASIC COS and SIN functions, how the LB instruction updates the current pen location, and how DI updates the carriage return point.

```
10  'Insert configuration statement here
20  PRINT #1, "INPS7000,5000SP1"
30  PRINT #1, "PA3500,2500"
40  PRINT #1, "DI0,1LB___1990"+CHR$(3)
50  PRINT #1, "DI1,1LB___1991"+CHR$(3)
60  PRINT #1, "DI1,0LB___1992"+CHR$(3)
70  'Angle = 135 degrees. Must convert degrees to
80  '  radians.  Radians=Degrees*PI/180
90  '  Variables are integer.
100 PI=3.14593
110 A%=COS(315*(PI/180))
120 B%=SIN(315*(PI/180))
130 PRINT #1, "DI";A%;",";B%;"LB___1993"+CHR$(3)
140 C%=COS(270*(PI/180))
150 D%=SIN(270*(PI/180))
160 PRINT #1, "DI";C%;",";D%;"LB___1994"+CHR$(13)+CHR$(3)
170 E%=COS(225*(PI/180))
180 F%=SIN(225*(PI/180))
190 PRINT #1, "DI";E%;",";F%;"LB___1995"+CHR$(13)+CHR$(3)
200 G%=COS(-180*(PI/180))
210 H%=SIN(-180*(PI/180))
220 PRINT #1, "DI";G%;",";H%;"LB___1996"+CHR$(13)+CHR$(3)
230 PRINT #1, "PG;"
240 END
```



Related Instructions	Group
DR, Relative Direction DV, Define Variable Text Path LB, Label	*The Character Group*

ERRORS:

Condition	Error	Plotter Response
both parameters = 0 or number out of range	3	ignores instruction

DR, Relative Direction

USE: Specifies the direction in which labels are drawn, relative to the scaling points P1 and P2. *Label direction is adjusted when P1 and P2 change so that labels maintain the same relationship to the plotted data.* Use DR to change labeling direction when you are labeling curves.

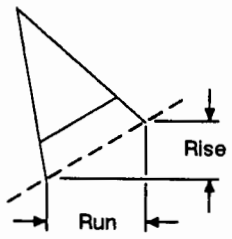
SYNTAX: DR*run,rise(;)*
or
DR*(;)*

Parameter	Format	Functional Range	Default
run	clamped real	*device-dependent*	1% of P2x - P1x
rise	clamped real	*device-dependent*	0% of P2Y - P1Y

REMARKS: The DR instruction updates the carriage return point to the current location. While DR is in effect, with or without parameters, the label direction is affected by changes in the location of P1 and P2. DR is also affected by the define variable text path (DV) instruction. Refer to the DI instruction earlier in this chapter for an explanation of this interaction.

- **No Parameters** — Defaults the label direction to relative and horizontal (parallel to the X-axis). Equivalent to *(DR1,0)*.
- **Run** — Specifies a percentage of the distance between P1x and P2x.
- **Rise** — Specifies a percentage of the distance between P1Y and P2Y.

The following illustrates the run and rise parameters.



With the DR instruction, the use of run and rise is somewhat different than with DI. Run is expressed as a percentage of the horizontal distance between P1 and P2; rise is a percentage of the vertical distance between P1 and P2.

$$\text{actual run} = \text{run parameter } (P2_X - P1_X)$$

$$\text{actual rise} = \text{rise parameter } (P2_Y - P1_Y)$$

To calculate the run and rise by measuring, refer to the next illustration. Think of the directional line on which the label rests as being parallel to a line beginning at the lower-left scaling point (usually P1) and intersecting the opposite side or the top edge of the graphics limits established by P1 and P2.

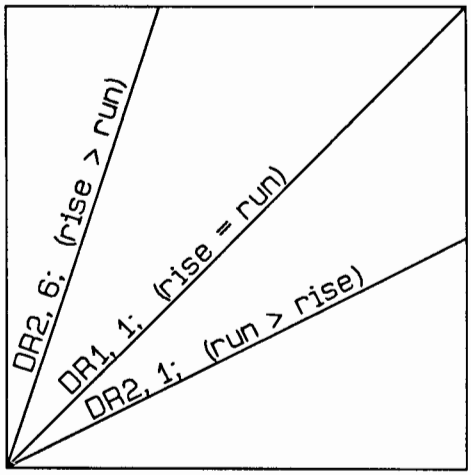
To calculate the angle of the label, use the run and rise parameters to form a fraction that is ≤ 1. It doesn't matter whether run or rise is the numerator. For example, if run = 4 and rise = 6, the fraction is $\frac{4}{6}$. If run = 6 and rise = 4, the fraction is still $\frac{4}{6}$.

The larger of the two terms determines whether the directional line intersects the top or side of the P1/P2 area as follows.

- If run = rise, the fraction equals 1 and the directional line extends from corner to corner of the P1/P2 area. (The exact corner is determined by the sign of the parameters.) Refer to the illustration below.

- If run > rise, the line intersects the *side* of the plotting area, a fraction of the way *up* toward the top scaling point. For example, if P1 is in the lower-left corner, run > rise, and the fraction is $\frac{1}{2}$, the directional line intersects the side $\frac{1}{2}$ of the way up toward P2. Refer to the illustration below.

- If run < rise, the line intersects the *top* of the plotting area, a fraction of the way *across* toward the right scaling point. For example, if P1 is in the lower-left corner, rise > run, and the fraction is $\frac{1}{3}$ ($\frac{2}{6} = \frac{1}{3}$), the directional line intersects the top $\frac{1}{3}$ of the way toward P2. Refer to the following illustration.

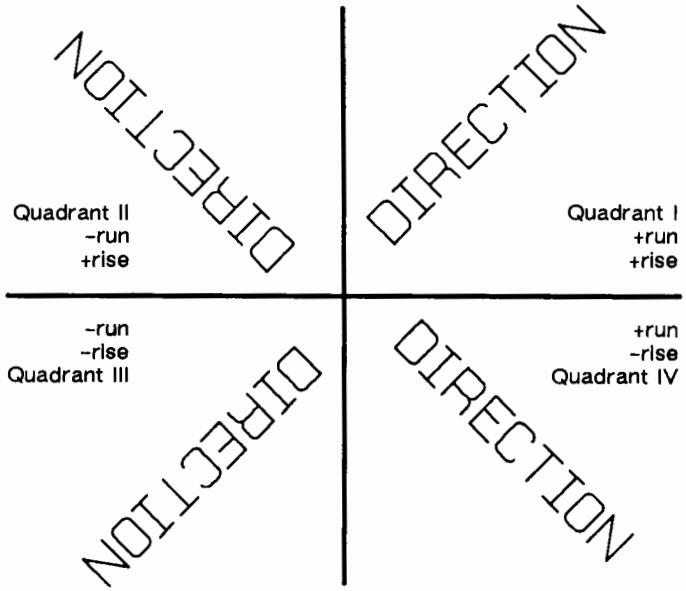


Negative parameters have the same effect on direction as described for the DI instruction.

At least one parameter must not be 0. The ratio of the parameters to each other is more important than the actual numbers. The table below lists three common label angles produced by using 1s and 0s.

DR Instruction	Label Direction
DR 1,0	horizontal
DR 0,1	vertical
DR 1,1 or DI 0.7,0.7 (any parameters equal to each other)	diagonal from P1 to P2

The relative size and sign of the two parameters determine the amount of rotation. If you imagine the current pen location to be the origin of a coordinate system for the label, you can see that the signs of the parameters determine in which quadrant the label will be.



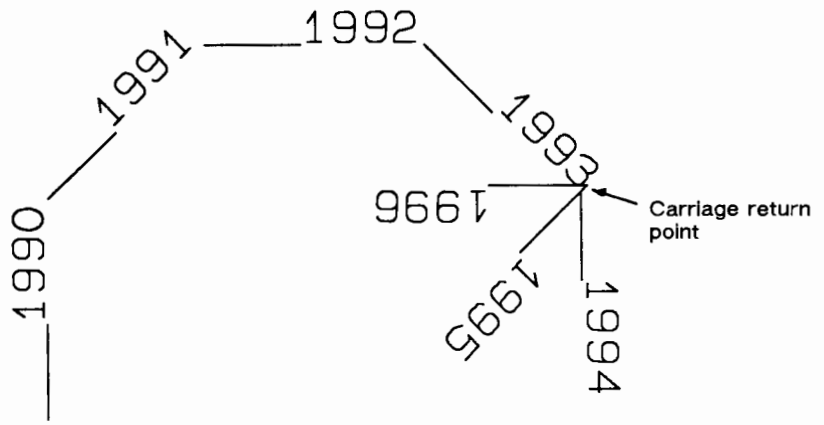
A DR instruction remains in effect until another DR or DI instruction is executed, or the plotter is initialized or set to default conditions.

EXAMPLE: This example illustrates the use of positive and negative parameters, how the LB instruction updates the current pen location, and how DR updates the carriage return point.

Note that this is the same example shown with the DI instruction. Changing the DI to DR and using the 1:0 ratio instead of COS and SIN are the only changes. However, if you plot them both and measure them, you'll discover that they are slightly different sizes. The size difference results from the DR instruction's use of the percentage of the P1/P2 distance.

NOTE: Labels begin at the current pen location and thus will be drawn parallel to the directional line, not necessarily on it.■

```
10  'Insert configuration statement here
20  PRINT #1, "INPS7000,5000SP1"
30  PRINT #1, "PA3500,2500"
40  PRINT #1, "DR0,1LB___1990"+CHR$(3)
50  PRINT #1, "DR1,1LB___1991"+CHR$(3)
60  PRINT #1, "DR1,0LB___1992"+CHR$(3)
70  PRINT #1, "DR1,-1LB___1993"+CHR$(3)
80  PRINT #1, "DR0,-1LB___1994"+CHR$(13)+CHR$(3)
90  PRINT #1, "DR-1,-1LB___1995"+CHR$(13)+CHR$(3)
100 PRINT #1, "DR-1,0LB___1996"+CHR$(13)+CHR$(3)
110 PRINT #1, "PG;"
120 END
```



Related Instructions	Group
DI, Absolute Direction DV, Define Variable Text Path LB, Label	*The Character Group*
IP, Input P1 and P2	*The Configuration/Status Group*

ERRORS:

Condition	Error	Plotter Response
both parameters = 0 or number out of range	3	ignores instruction

DT, Define Label Terminator

USE: Specifies the character to be used as the label terminator and whether it is printed. Use DT to define a new label terminator if your computer cannot use the default label terminator (**ETX**, decimal code 3).

SYNTAX: DT*label terminator(,mode)*;
or
DT;

NOTE: You *must* use a terminator (;) with this instruction.■

Parameter	Format	Functional Range	Default
label terminator	label	any character except **NULL**, **LF**, **ESC**, and **;** (decimal codes 0, 5, 27, and 59, respectively)*	**ETX** (decimal code 3)
mode	clamped integer	0 or 1	1 (nonprinting)

* If you specify decimal codes 0, 5, or 27, the plotter ignores the instruction; the label terminator is not changed.

REMARKS: The character immediately following DT is interpreted to be the new label terminator. You *must* terminate all Label (LB) instructions following a DT instruction with the specified label terminator.

- **No Parameters** — Defaults the label terminator to **ETX** (*not* a semicolon) and the mode to nonprinting (1).
- **Label Terminator** — Specifies the label terminator as the character immediately following the DT mnemonic. (If you use a space between the mnemonic and the label terminator parameter, the space becomes the label terminator.)
- **Mode** — Specifies whether the label terminator is printed.

  0 The label terminator prints if it is a printing character and performs its function if it is a control character.

  1 (Default.) The label terminator does not print if it is a printing character and does not perform its function if it is a control character.

A DT instruction remains in effect until another DT instruction is executed, or the plotter is initialized or set to default conditions.

EXAMPLE: The following program shows how to change the label terminator, as well as what happens to labels with different terminators.

NOTE: Although some program lines are shown on two lines to fit on this page, you should write them on just one line.■

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BINPS7000,5000SP1SC0,5000,0,5000"
30  PRINT #1, "SI.187,.269PA0,4500"
40  PRINT #1, "LBDefault: ETX terminates by"+CHR$(13)+
     CHR$(10)+CHR$(3)
50  PRINT #1, "LBperforming end-of-text function."+CHR$(3)
60  PRINT #1, "CPCPDT#,1;"
70  PRINT #1, "LBPrinting characters terminate,"+CHR$(13)+
     CHR$(10)+"#"
80  PRINT #1, "DT#,0;"
90  PRINT #1, "LBbut print only when print mode is on.#"
100 PRINT #1, "PUSP0PG;"
110 END
```

```
Default: ETX terminates by
performing end-of-text function.

Printing characters terminate,
but print only when print mode is on.#
```

Related Instruction	Group
LB, Label	*The Character Group*

DV, Define Variable Text Path

USE: Specifies either right, left, up, or down as the text path for labels and the direction of line feeds. Use DV to "stack" characters in a column.

SYNTAX: DV*path(,line;)*
or
DV*(;)*

Parameter	Format	Functional Range	Default
path	clamped integer	0, 1, 2, or 3	0 (horizontal)
line	clamped integer	0 or 1	0 (normal line feed)

REMARKS: The DV instruction determines the direction of the text path and the direction the carriage return point moves when a line feed is included in the label string.

- **No Parameters** — Defaults the text path to horizontal (not stacked) with normal line feed. Equivalent to *(DV0)*.

- **Path** — Specifies the location of each character with respect to the preceding character, relative to the labeling direction defined by the DI or DR instructions. The text path set by DV is not affected by changes to P1 and P2.

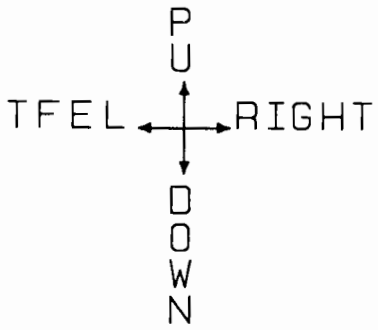
  0 0 degrees. (Right.) Each character begins to the right of the previous character. This is a horizontal text path (unless altered by DI or DR).

  1 -90 degrees. (Down.) Each character begins below the previous character. This is a vertical text path (unless altered by DI or DR).

  2 -180 degrees. (Left.) Each character begins to the left of the previous character. This is a horizontal text path (unless altered by DI or DR).

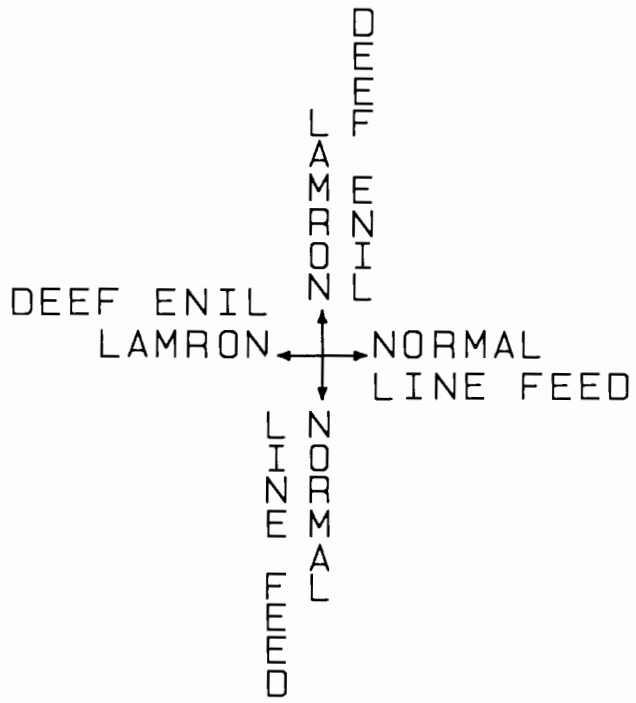
  3 -270 degrees. (Up.) Each character begins above the previous character. This is a vertical text path (unless altered by DI or DR).

  The following illustration shows the four text paths.

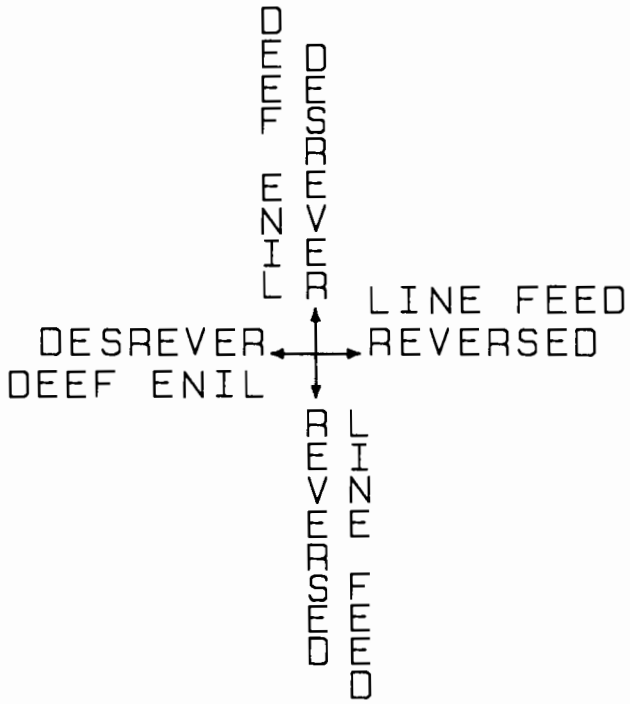


- **Line** — Specifies the location of each character with respect to the preceding character, relative to the labeling direction defined by the DI or DR instructions.

  0 **–90 degrees.** (Normal line feed.) Sets the direction of line feeds –90 degrees with respect to the text path.



1 **+90 degrees.** (Reversed line feed.) Sets the direction of line feeds +90 degrees with respect to the text path.



EXAMPLE: The following illustrates how line feeds and carriage returns affect vertical labels. Horizontal labels are shown for comparison. All terminators, line feeds, and carriage returns are sent using their ASCII decimal code equivalents with the CHR$ function.

```
10  'Insert configuration statement here
20  PRINT #1, "INPS7000,8000SP1PA1000,3000DV1"
30  'vertical
40  PRINT #1, "LBABC"+CHR$(10)+CHR$(13)+CHR$(3)
50  PRINT #1, "LBDEF"+CHR$(10)+CHR$(3)
60  PRINT #1, "LBGHI"+CHR$(3)
70  'horizontal
80  PRINT #1, "PA3000,3000DV0"
90  PRINT #1, "LBABC"+CHR$(10)+CHR$(13)+CHR$(3)
100 PRINT #1, "LBDEF"+CHR$(10)+CHR$(3)
110 PRINT #1, "LBGHI"+CHR$(3)
120 PRINT #1, "PG;"
130 END
```

Related Instructions	Group
DI, Absolute Direction DR, Relative Direction LO, Label Origin	*The Character Group*

ES, Extra Space

USE: Adjusts space between characters and lines of labels without affecting character size.

SYNTAX: ES*width(,height;)*
or
ES*(;)*

Parameter	Format	Functional Range	Default
width	clamped real	*device-dependent*	0
height	clamped real	*device-dependent*	0

REMARKS: An ES instruction remains in effect until another ES instruction is executed, or the plotter is initialized or set to default conditions.

- **No Parameters** — Defaults the amount of horizontal and vertical space between label characters or lines to no extra space. Equivalent to *(ES0,0)*.

- **Width** — Specifies increases (positive number) or decreases (negative number) in the space between characters. The width parameter is a fraction of the character plot cell width. For example, *ES.15* under default conditions causes the CP cell width to increase by 15 percent (1.15 × its current width). Character images are not distorted by ES. For maximum legibility, do not specify more than one extra space or subtract more than half a space.

- **Height** — Specifies increases (positive number) or decreases (negative number) in the space between lines. The length parameter is a fraction of the line feed height of the character plot cell. For maximum legibility, do not specify more than two extra lines or subtract more than half a line.

EXAMPLE:

```
10  'Insert configuration statement here
20  PRINT #1, "INPS8000,5000SP1PA2500,3200SI.187,.269"
30  PRINT #1, "ESLBES CAUSES"+CHR$(3)
40  PRINT #1, "CPLBTHIS SPACING."+CHR$(3)
50  PRINT #1, "PA2500,2500"
60  PRINT #1, "ES-.1,-.25LBES-.1,-.25  CAUSES"+CHR$(3)
70  PRINT #1, "CPLBTHIS SPACING."+CHR$(3)
80  PRINT #1, "PA2500,1800"
90  PRINT #1, "ES.2,.25LBES.2,.25  CAUSES"+CHR$(3)
100 PRINT #1, "CPLBTHIS SPACING."+CHR$(3)
110 PRINT #1, "PUSP0PG;"
120 END
```

```
ES CAUSES
THIS SPACING.

ES-.1,-.25 CAUSES
THIS SPACING.

ES.2,.25  CAUSES
THIS  SPACING.
```

Related Instructions	Group
CP, Character Plot LB, Label	*The Character Group*

LB, Label

USE: Plots text using the currently defined font. Use LB to annotate drawings or create text-only charts.

SYNTAX: LB *c . . . c* label terminator

Parameter	Format	Functional Range	Default
c . . . c	label	any character(s)	no default

REMARKS: The LB instruction includes an automatic pen down. When the instruction is completed, the original pen-up/down position is restored.

- **c . . . c** — Includes up to 256 ASCII characters (including control characters and the label terminator). Printing characters are drawn using the currently selected font. (Refer to the AD, SA, SD, and SS instructions for details on specifying and selecting fonts.)

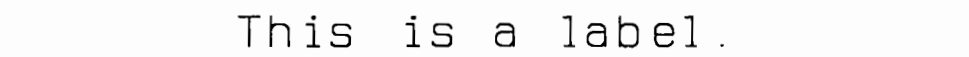
  You can include nonprinting characters such as the carriage return (**CR**, decimal code 13) and line feed (**LF**, decimal code 10). These characters invoke the specified function, but are not drawn.

  The label begins at the current pen location (unless altered by LO). After each character is drawn, the pen moves to the next character origin (refer to *Working with the Character Plot Cell* earlier in the chapter). The current pen location is not updated until the LB instruction is terminated.

- **Label Terminator** — Terminates the LB instruction. You *must* use the special label terminator (refer to the DT instruction) to tell the plotter to exit label mode. If you don't use the label terminator, everything following the LB mnemonic will be printed in the label, including other instructions. The default label terminator is the nonprinting end-of-text character **ETX** (decimal code 3). In BASIC, CHR$(3) accesses **ETX**. You can define a different terminator using the DT instruction.

EXAMPLE:

```
10  'Insert configuration statement here
20  PRINT #1, "INPS7000,5000SP1PA2500,2500"
30  PRINT #1, "LBThis is a label."+CHR$(3)
40  PRINT #1, "PG;"
50  END
```



Related Instructions	Group
AD, Alternate Font Definition CP, Character Plot SA, Select Alternate Font SD, Standard Font Definition SS, Select Standard Font DT, Define Label Terminator DI, Absolute Direction DR, Relative Direction DV, Define Variable Text Path LO, Label Origin SI, Absolute Character Size SR, Relative Character Size SL, Character Slant	*The Character Group*

LO, Label Origin

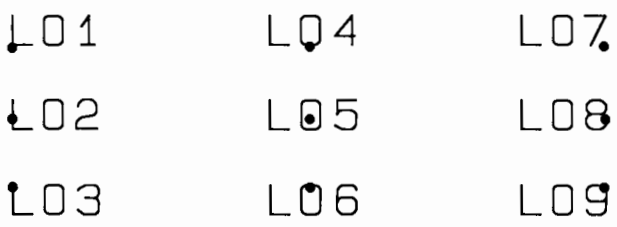
USE: Positions labels relative to current pen location. Use LO to center, left-justify, or right-justify labels. The label can be drawn above or below the current pen location and can also be offset by an amount equal to one-half the character's width or height.

SYNTAX: LO*position(;)*
or
LO*(;)*

Parameter	Format	Functional Range	Default
position	clamped integer	1 to 9, 11 to 19	1

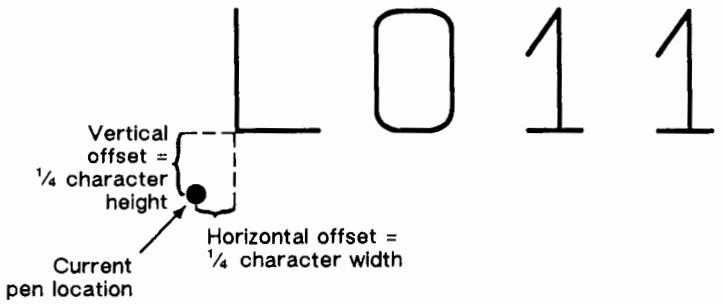
REMARKS: The plotter interprets the parameters as follows.

- **No Parameters** — Defaults the label origin. Equivalent to *(LO1)*.
- **Position** — Specifies how the label is oriented with respect to the current pen location. The following illustrates the different label origins. Each dot represents the current pen location.



The label positions LO 11 through LO 19 differ from LO 1 through LO 9 only in that the labels are offset from the current pen location.

The amount of offset is equal to one-fourth the character's point size (one-third [16 grid units] for stick fonts). The offset is shown in the following illustration.



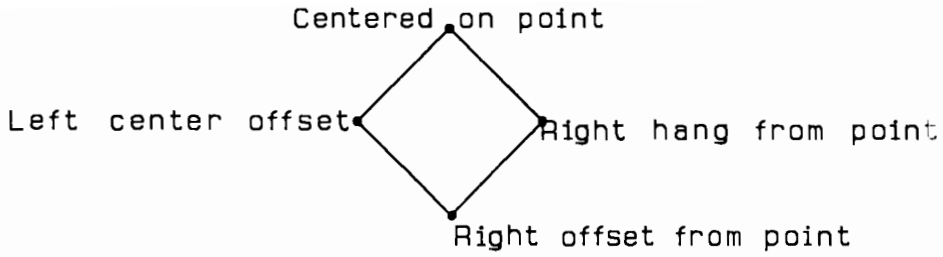
Thereafter, LO updates the carriage return point to the current location. When labeling, the current pen location (but not the carriage return point) is updated after each character is drawn and the pen automatically moves to the next character origin. If you want to return a pen to the current label's origin prior to the next label instruction, send a carriage return, CHR$(13), after the label but before the label terminator.

When you embed carriage return characters in a label, each portion of the label is positioned according to the label origin, just as if they were written as separate label instructions.

An LO instruction remains in effect until another LO instruction is executed, or the plotter is initialized or set to default conditions.

EXAMPLE:

```
10  'Insert configuration statement here
20  PRINT #1, "INPS7000,5000SP1SI.17,.26PA2500,2500"
30  PRINT #1, "PD2000,2000,2500,1500,3000,2000,2500,2500"
40  PRINT #1, "CI10,LO4LBCentered on point"+CHR$(3)
50  PRINT #1, "PU2000,2000CI10,LO18"
60  PRINT #1, "LBLeft center offset"+CHR$(3)
70  PRINT #1, "PU2500,1500CI10,LO13"
80  PRINT #1, "LBRight offset from point"+CHR$(3)
90  PRINT #1, "PA3000,2000CI10,LO3"
100 PRINT #1, "LBRight hang from point"+CHR$(3)
110 PRINT #1, "PG;"
120 END
```



Related Instructions	Group
DV, Define Variable Text Path LB, Label	*The Character Group*

SA, Select Alternate Font

USE: Selects the alternate font (already designated by the AD instruction) for subsequent labeling. Use SA to shift from the currently selected standard font to the designated alternate font.

SYNTAX: SA*(;)*

REMARKS: The SA instruction tells the plotter to draw subsequent labeling instructions using characters from the alternate character set previously designated by the AD instruction. The SA instruction is equivalent to using the shift-out control character (**SO**, decimal 14) within a label string.

The default-designated alternate font is Roman8 (set 0). The alternate font remains in effect until an SS instruction is executed, a shift-in control character (**SI**, decimal 15) is encountered, or the plotter is initialized or set to default conditions.

Related Instructions	Group
AD, Alternate Font Definition LB, Label SD, Standard Font Definition SS, Select Standard Font	*The Character Group*

SD, Standard Font Definition

USE: Defines the standard font and its attributes: font spacing, pitch, height, stroke weight, and typeface. For designating an alternate font refer to the AD instruction.

SYNTAX: SD*kind,value... (,kind,value;)*
or
SD*(;)*

Parameter	Format	Functional Range	Default
kind	clamped integer	1 to 7	no default
value	clamped real	*kind*-dependent*	*kind*-dependent*

* Refer to the table following the parameter descriptions.

REMARKS: This instruction remains in effect until the plotter receives a DF, IN, or another AD instruction.

- **No Parameters** — Defaults the standard font attributes. (The default font set of attributes is device-dependent, but the default font will always be a scalable font, not a bitmap font.)

- **Kind** — Specifies the attribute for which you are setting a value.

Kind	Attribute	Default Value	Description
1	Character set	277	Roman8
2	Font spacing	0	fixed spacing
3	Pitch	*media-dependent**	characters per inch
4	Height	*media-dependent**	font point size
5	Posture	*device-dependent*	upright or italic
6	Stroke weight	0	normal
7	Typeface	48	stick (fixed-vector)

* The default value for pitch is usually 9 cpi; for E-size media it is 5.9 cpi. The default character height value is usually 11.5; for E-size media it is 16.

- **Value** — Defines the characteristics of the attribute specified by the kind parameter.

The following tables list the kind parameters with their associated values. For kinds 1 (character set) and 7 (typeface), your plotter may support values other than those listed here. Refer to your user's guide or HP-GL/2 option manual for the attributes and values that are supported.

Kind 1: Character Set

The character set attribute defines the character set to be used as the standard set. The values are listed in the order of the PCL identification. Note that your plotter may not support all of these character sets. Refer to the AD or SD instruction in your user or HP-GL/2 option documentation for a list of the character sets your device supports.

Kind 1: Character Set Values	Description	ISO Number
0 (and 277)	Roman8 (default)	—
1	Math-7	—
2	Line Draw-7	—
3	HP Large Characters	—
4	Norwegian v1	60
36	Norwegian v2	61
5	Roman Extensions	—
37	United Kingdom	4
6	French v1	25
38	French v2	69
7	HP German	—
39	German	21
263	Greek-8	—
8	Hebrew-7	—
264	Hebrew-8	—
9	Italian	15
202	Microsoft Publishing	—
234	DeskTop	—
330	PS Text	—
426	Ventura International	—
458	Ventura U.S.	—
11	JIS ASCII	14
43	Katakana	13
75	Chinese	57
267	Kana-8	—
299	Korean-8	—
12	Line Draw-7	—
44	HP Block Characters	—
76	Tax Line Draw	—
268	Line Draw-8	—
300	Ventura ITC Zapf Dingbats	—
332	PS ITC Zapf Dingbats	—
364	ITC Zapf Dingbats Series 100	—
396	ITC Zapf Dingbats Series 200	—
428	ITC Zapf Dingbats Series 300	—

(Continued)

Kind 1: Character Set Values	Description	ISO Number
13	Math-7	—
45	Tech-7 (DEC)	—
173	PS Math	—
205	Ventura Math	—
269	Math-8	—
14	ECMA-94 Latin 1 (8-bit version)	—
78	ECMA-94 Latin 2	—
174	ECMA-128 Latin 5	—
334	ECMA-113/88 Latin/Cyrillic	—
15	OCR-A	—
47	OCR-B	—
79	OCR-M	—
16	APL (typewriter-paired)	—
48	APL (bit-paired)	—
145	PC Line	—
18	Cyrillic ASCII	—
50	Cyrillic	—
114	PC Cyrillic	—
19	Swedish for names	11
51	HP Spanish	—
83	Spanish	17
115	Swedish	10
147	Portuguese	16
179	Portuguese	84
211	Spanish	85
243	HP European Spanish	—
275	HP Latin Spanish	—
531	HP-GL Download	—
563	HP-GL Drafting	—
595	HP-GL Special Symbols	—
20	Thai-8	—
276	Turkish-8	—
21	ANSI US ASCII	6
53	Legal	—
85	International Reference Version	2
181	HPL Language Set	—
245	OEM-1 (DEC Set)	—
277 (and 0)	Roman8 (default)	—
309	Windows	—
341	PC-8	—
373	PC-8 Denmark/Norway	—
405	PC-850	—
501	Pi Font	—
565	PC-852	—
22	Arabic (MacKay's Version)	—
278	Arabic-8	—

(Continued)

Kind 1: Character Set Values	Description	ISO Number
25	3 of 9 Barcode	—
57	Industrial 2 of 5 Barcode	—
89	Matrix 2 of 5 Barcode	—
153	Interleaved 2 of 5 Barcode	—
185	CODABAR Barcode	—
217	MSI/Plessey Barcode	—
249	Code 11 Barcode	—
281	UPC/EAN Barcode	—
505	USPS Zip	—
26	Not used	—

Kind 2: Font Spacing

The font-spacing attribute defines whether the spacing is fixed (all characters having equal width) or variable (each character occupying a space proportional to its size). Refer to *Plotting with Fixed- and Variable-Space Fonts* earlier in this chapter.

Kind 2: Font Spacing Values	Description
0	fixed spacing (default)
1	variable spacing

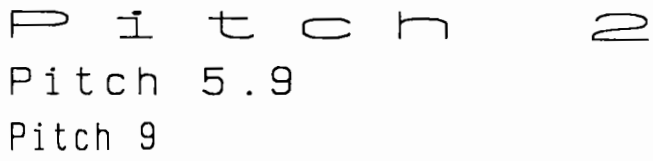
Kind 3: Pitch

The pitch attribute defines the number of characters per inch.

Kind 3: Pitch Values	Description
0 to 32 767.9999	*media-dependent**

* The default value is usually 9 cpi; for E-size media it is 5.9 cpi.

The following illustration shows text in three different pitches. Fixed-space fonts depend on pitch to determine character size. Proportional fonts ignore pitch. Note that with the AD instruction, you cannot create tall, skinny characters or short, wide characters; the character aspect ratio is preserved.



Kind 4: Height

The height attribute defines the font point size (i.e., the height of the character plot cell). Variable-space fonts depend on the height parameter to determine character size; fixed-space fonts depend on pitch for character size. Note that with the AD instruction, you cannot create tall, skinny characters or short, wide characters; the character aspect ratio is preserved.

Kind 4: Height Values	Description
0 to 32 767.9999	*media-dependent**

* The default character height value is usually 11.5; for E-size media it is 16.

The following illustration shows text in three different heights.

Kind 5: Posture

Posture defines the character's vertical position. The default posture is device-dependent (but usually upright).

Kind 5: Posture Values	Description
0	upright
1	italic

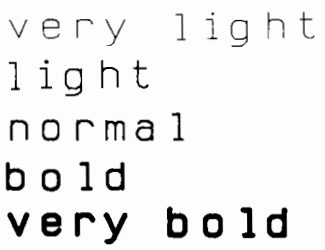
Kind 6: Stroke Weight

The stroke weight attribute defines the line width used in labels. The default stroke weight is device-dependent. Note that stroke weight is affected by changes in P1 and P2 locations when relative sizing is in effect; refer to the SR instruction.

Kind 6: Stroke Weight Values	Description
-7	very light
-3	light
0	normal
3	bold
7	very bold
9999	* (for stick font only)

* When the stick font (typeface 48) is selected, the value 9999 causes the stick font to be rendered in the current pen width.

The following illustration shows labels in five different stroke weights.



Kind 7: Typeface

The typeface attribute selects the font, which defines the style of the lettering. Characters drawn with arcs have a smoother, more rounded appearance than characters drawn with vectors. All HP-GL/2 devices support the stick fonts (48, 49, and 50). The drafting font is for use with the HP Drafting character set.

Kind 7: Typeface Values	Description
0	Line Printer or Line Draw
1	Pica
2	Elite
3	Courier
4	Helvetica
5	Times Roman
6	Letter Gothic
7	Script
8	Prestige
9	Caslon
10	Orator
11	Presentation
13	Serifa
14	Futura
15	Palatino
16	ITC Souvenir
17	Optima
18	ITC Garamond
20	Coronet
21	Broadway
23	Century Schoolbook
24	University Roman
27	ITC Korinna
28	Naskh (generic Arabic typeface)
29	Cloister Black
30	ITC Galliard
31	ITC Avant Garde Gothic
32	Brush
33	Blippo
34	Hobo
35	Windsor
38	Peignot
39	Baskerville
41	Trade Gothic
42	Goudy Old Style
43	ITC Zapf Chancery
44	Clarendon
45	ITC Zapf Dingbats
46	Cooper
47	ITC Bookman
48	Stick (default)
49	HP-GL Drafting
50	HP-GL fixed arc
51	Gill Sans
52	Univers
53	Bodoni
54	Rockwell
55	Melior

(Continued)

Kind 7: Typeface Values	Description
56	ITC Tiffany
57	ITC Clearface
58	Amelia
59	Park Avenue
60	Handel Gothic
61	Dom Casual
62	ITC Benguiat
63	ITC Cheltenham
64	Century Expanded
65	Franklin Gothic
68	Plantin
69	Trump Mediaeval
70	Futura Black
71	ITC American Typewriter
72	Antique Olive
73	Uncial
74	ITC Bauhaus
75	Century Oldstyle
76	ITC Eras
77	ITC Friz Quadrata
78	ITC Lubalin Graph
79	Eurostile
80	Mincho
81	ITC Serif Gothic
82	Signet Roundhand
83	Souvenir Gothic
84	Stymie
87	Bernhard Modern
89	Excelsior
90	Grand Ronde Script
91	Ondine
92	P.T. Barnum
93	Kaufman
94	ITC Bolt Bold
96	Helv Monospaced
97	Revue
101	Garamond (Stempel)
102	Garth Graphic
103	ITC Ronda
104	OCR-A
105	ITC Century
106	Englische Schreibschrift
107	Flash
108	Gothic Outline (URW)
109	Stencil (ATF)

(Continued)

Kind 7: Typeface Values	Description
110	OCR-B
111	Akzidenz-Grotesk
112	TD Logos
113	Shannon
114	ITC Century
152	Maru Gosikku
153	Gosikku (Kaku)
154	Socho
155	Kyokasho
156	Kaisho

The following examples show how to select each of the plotter's three fonts for the default character set (Roman8).

Fixed-vector `(AD7,48)`
Variable-arc `(AD2,1,7,5Ø)`
Fixed-arc `(AD2,Ø,7,5Ø)`

Related Instructions	Group
AD, Alternate Font Definition LB, Label SA, Select Alternate Font SI, Absolute Character Size SR, Relative Character Size SS, Select Standard Font	*The Character Group*

SI, Absolute Character Size

USE: Specifies the size of labeling characters in centimeters. Use SI to establish character sizing independent of P1 and P2.

SYNTAX: SI*width, height(;)*
or
SI*(;)*

Parameter	Format	Functional Range	Default
width	clamped real	*device-dependent*	dependent*
height	clamped real	*device-dependent*	dependent*

* Dependent on the current pitch and font height set by the AD or SD instructions. If set to default values, the width is 0.285 cm and the height is 0.375 cm.

REMARKS: While SI is in effect, with or without parameters, the size of characters of the currently selected font is not affected by changes in P1 and P2. An SI instruction remains in effect until another SI instruction is executed, an SR instruction is executed, or the plotter is initialized or set to default conditions.

- **No Parameters** — Defaults the character size to the attributes of the AD or SD instruction, depending on the font currently in use.

- **Width** — Specifies the width of the nominal character in centimeters. A negative width parameter mirrors labels in the right-to-left direction.

  The nominal character width is 0.5 of the point size (32 grid units for the stick font).

  NOTE: Changing character size also changes the width of the line used to draw stick font characters.■

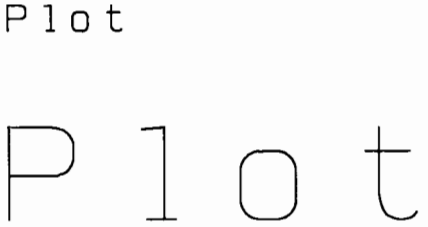
- **Height** — Specifies the cap height in centimeters. A negative height parameter mirrors labels in the top-to-bottom direction.

Note that in most languages the width of a letter is typically less than the height. If you set your characters to have a different "aspect ratio," they may look odd to your readers.

NOTE: If a bitmap font is selected, an SI may not be accurately executed. Labels will be rendered using the bitmap font that most closely approximates the character height or width specified by SI (character size is determined by height for proportional fonts and by width for fixed-space fonts). Bitmap fonts cannot be mirrored by using negative SI parameters.■

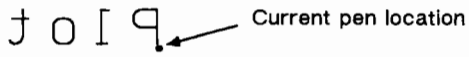
EXAMPLE: The following draws the word "Plot" in two sizes: the default and 1 cm wide by 1.5 cm high.

```
10  'Insert configuration statement here
20  PRINT #1, "INPS7000,5000SP1PA2500,3000"
30  PRINT #1, "LBPlot"+CHR$(3)
40  PRINT #1, "PA2500,2000SI1,1.5LBPlot"+CHR$(3)
50  PRINT #1, "PG;"
60  END
```



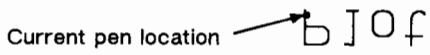
The following are examples of negative parameters producing mirror images of labels. A negative width parameter mirrors labels in the right-to-left direction.

```
"SI-.3,.45;LBPlot"+CHR$(3)
```



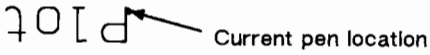
A negative height parameter mirrors labels in the top-to-bottom direction.

```
"SI.3,-.45;LBPlot"+CHR$(3)
```



Negative width and height parameters together mirror labels in both directions, causing the label to appear to be rotated 180 degrees.

```
"SI-.3,-.45;LBPlot"+CHR$(3)
```



Related Instructions	Group
AD, Alternate Font Definition DI, Absolute Direction DR, Relative Direction SD, Standard Font Definition SR, Relative Character Size	*The Character Group*

SL, Character Slant

USE: Specifies the slant at which labels are drawn. Use SL to create slanted text for emphasis, or to reestablish upright labeling after an SL instruction with parameters has been in effect.

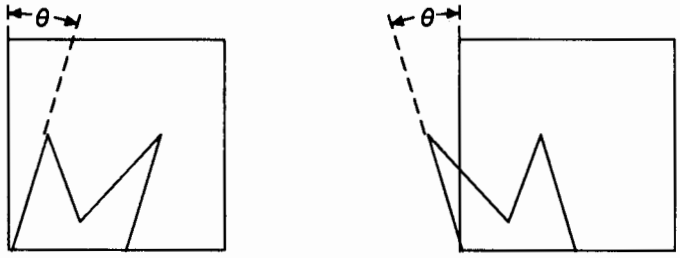
NOTE: The SL instruction has no effect when using bitmap fonts. Refer to the Scalable or Bitmap Fonts (SB) instruction in chapter 11, *The Dual-Context Extension*.■

SYNTAX: SL*tangent of angle(;)*
or
SL*(;)*

Parameter	Format	Functional Range	Default
tangent of angle	clamped real	*device-dependent*	0

REMARKS: The plotter interprets the parameters as follows.

- **No Parameter** — Defaults the slant to zero (no slant). Equivalent to (*SL0*).
- **Tangent of Angle** — Interpreted as an angle θ from vertical. The base of the character always stays on the horizontal, as shown in the following illustration.



Positive Slant *Negative Slant*

The SL instruction affects each character relative to an imaginary line beside the label. The direction or placement of the label on the plot does not affect the SL instruction; neither do the settings of P1 and P2. The DI and DR instructions do affect the slant direction, since the base of a character always stays on the baseline of the label.

You can specify the actual tangent value, or you can use the TAN function available on most computers. The following shows tangent values for selected angles.

θ	Tangent
0	0
-10	-0.18
-20	-0.36
-30	-0.58
-40	-0.84
-45	-1.00
0	0
10	0.18
20	0.36
30	0.58
40	0.84
45	1.00

An SL instruction remains in effect until another SL instruction is executed, or the plotter is initialized or set to default conditions.

EXAMPLE: The following shows you two methods for specifying the slant parameter. The first label uses a variable generated by the TAN function. The second label uses a tangent value given in the previous table.

NOTE: If you want to use the TAN function on your computer, check your computer documentation to see how your computer interprets angles. This version of GW-BASIC interprets angles as radians, so line 40 in this program converts radians to degrees.■

```
10  'Insert configuration statement here
20  PRINT #1, "INPS7000,5000SP1SI.7,1PA1000,1000"
30  PI=3.141593
40  A=TAN(20*(PI/180))
50  PRINT #1, "SL"A"LB+ Slant"+CHR$(3)
60  PRINT #1, "PA1000,300SL-.36LB- Slant"+CHR$(3)
70  PRINT #1, "PG;"
80  END
```

Related Instructions	Group
DI, Absolute Direction DR, Relative Direction LB, Label	*The Character Group*

SR, Relative Character Size

USE: Specifies the size of characters as a percentage of the distance between P1 and P2. Use SR to establish relative character sizes so that if the P1/P2 distance changes, the character sizes adjust to occupy the same relative amount of space.

SYNTAX: SR*width, height(;)*
or
SR*(;)*

Parameter	Format	Functional Range	Default
width	clamped real	*device-dependent*	0.75% of P2x – P1x
height	clamped real	*device-dependent*	1.5% of P2Y – P1Y

REMARKS: While SR is in effect, with or without parameters, the size of characters in the currently selected font is affected by changes in P1 and P2. An SR instruction remains in effect until another SR instruction is executed, an SI instruction is executed, or the plotter is initialized or set to default conditions.

- **No Parameters** — Defaults the relative character width to 0.75% of the X-axis distance (P2x – P1x) and the height to 1.5% of the Y-axis distance (P2Y – P1Y).

- **Width** — Sets the character width to the specified percentage of the distance between the X-coordinates of P1 and P2. A negative width parameter mirrors labels in the right-to-left direction.

  NOTE: Changing character size also changes the width of the line used to draw stick font characters.■

- **Height** — Sets the character height to the specified percentage of the distance between the Y-coordinates of P1 and P2. A negative height parameter mirrors labels in the top-to-bottom direction.

The character size you specify with SR is a *percentage* of (P2x – P1x) and (P2Y – P1Y). The plotter calculates the actual character width and height as follows:

$$\text{actual width} = \text{width parameter}/100 \times (P2x - P1x)$$

$$\text{actual height} = \text{height parameter}/100 \times (P2Y - P1Y)$$

For example, suppose P1 and P2 are located at (–6956,–4388) and (6956,4388), respectively. If you establish relative sizing and specify a width of 2 and a height of 3.5, the plotter determines the actual character size as follows:

$$\text{width} = 2/100 \times (6956 - (-6956)) = 278.24 \text{ plotter units, or } 0.695 \text{ cm}$$

$$\text{height} = 3.5/100 \times (4388 - (-4388)) = 307.16 \text{ plotter units, or } 0.768 \text{ cm}$$

If you changed P1 and P2 to (100,100) and (5000,5000), but didn't change the SR parameters, the character size would change as follows:

width = 2/100 × (5000 - 100) = 98 plotter units, or 0.245 cm

height = 3.5/100 × (5000 - 100) = 171.5 plotter units, or 0.429 cm

Note that in most languages the width of a letter is typically less than the height. If you set your characters to have a different "aspect ratio," they may look odd to your readers.

NOTE: Either negative SR parameters or switching the relative position of P1 and P2 will produce mirror images of labels. When P1 is in the lower left and P2 is in the upper right, the SR instruction gives the same mirroring results as the SI instruction. However, if you move P1 to the right of P2, characters are mirrored right-to-left; when you move P1 above P2, characters are mirrored top-to-bottom. When *both* of these situations occur (using negative parameters in the SR instruction with an unusual P1/P2 position), double mirroring may result in either direction, in which case *the two inversions cancel,* and lettering appears normal. If a bitmap font is selected, negative parameters or unusual P1/P2 locations do not cause mirroring.■

EXAMPLE: The following shows a label with character size relative to P1 and P2 (*SR*). Next, the locations of P1 and P2 are changed; then, the character size percentages are specified. Notice that the new character size has equal parameters of 2.5; because the P1/P2 area is square, the resulting characters are square.

```
10  'Insert configuration statement here
20  PRINT #1, "INPS7000,8000SP1IP-6956,-4388,6956,4388"
30  PRINT #1, "SRPA0,2700LBRELATIVE LABEL SIZE"+CHR$(3)
40  '
50  PRINT #1, "IP0,0,5500,5500PA0,2000"
60  PRINT #1, "LBNEW P1 AND P2 CHANGE LABEL SIZE"+CHR$(3)
70  '
80  PRINT #1, "PA0,1000SR2.5,2.5"
90  PRINT #1, "LBNEW SR INSTRUCTION"+CHR$(3)+"CP"
100 PRINT #1, "LBCHANGES LABEL SIZE"+CHR$(3)
110 PRINT #1, "PG;"
120 END
```

RELATIVE LABEL SIZE

NEW P1 AND P2 CHANGE LABEL SIZE

NEW SR INSTRUCTION CHANGES LABEL SIZE

Related Instructions	Group
DI, Absolute Direction DR, Relative Direction IP, Input P1 and P2 IR, Input Relative P1 and P2 SI, Absolute Character Size	*The Character Group*

SS, Select Standard Font

USE: Selects the standard font (already designated by the Standard Font Definition [SD] instruction) for subsequent labeling. Use SS to shift from the currently selected alternate font to the designated standard font.

SYNTAX: *SS(;)*

REMARKS: The SS instruction tells the plotter to draw subsequent labeling instructions using characters from the standard font designated by the SD instruction. The SS instruction is equivalent to using the shift-in control character (**SI**, decimal 15) within a label string.

The default-designated standard font is Roman8 (set 0). This font is in effect when the plotter is initialized or set to default conditions. The SS instruction remains in effect until an SA instruction is executed.

Related Instructions	Group
AD, Alternate Font Definition LB, Label SA, Select Alternate Character Set SD, Standard Font Definition	*The Character Group*

TD, Transparent Data

USE: Specifies whether control characters perform their associated function or print their character when labeling. Use TD to access printable characters that in normal mode function only as control characters.

SYNTAX: TD*(mode;)*
or
TD*(;)*

Parameter	Format	Functional Range	Default
mode	clamped integer	0 or 1	0 (normal)

REMARKS: For a list of control characters and their associated functions, refer to *Control Characters* earlier in this chapter.

- **No Parameters** — Defaults the labeling mode to normal. Equivalent to *(TD0)*.
- **Mode** — Selects the normal or transparent mode for labeling.

  0 **Normal.** Control characters with an associated functionality perform their function and do not print.

  1 **Transparent.** All characters print and perform no other function (except the currently defined label terminator, which terminates the label). The plotter prints a space for nonprinting or undefined characters.

Transparent data mode must be enabled to access printable characters whose character code has an associated functionality in normal mode. For example, the left arrow in the PC-8 character set has a character code of 27. In normal mode, a character code of 27 is interpreted as an escape character; in transparent data mode, a character code of 27 causes an arrow to be printed.

Related Instructions	Group
AD, Alternate Font Definition DT, Define Label Terminator LB, Label SA, Select Alternate Font SD, Standard Font Definition SS, Select Standard Font	*The Character Group*

9

The Technical Graphics Extension

Most HP-GL/2 devices support the instructions in the Technical Graphics Extension. These instructions give added flexibility when your applications require technical functionality. The information in this chapter enables you to achieve the following results.

- Specify chord tolerance for arcs, circles, and wedges.
- Design and download user-defined characters.
- Turn on/off your plotter's automatic cutter.
- Advance media for long-axis plotting.
- Write a message to your plotter's control panel.
- Ask the plotter to output information to the computer.
- Specify a user-defined page size.
- Set the quality level of your output.
- Sort your plot data for faster throughput.
- Set the pen speed on pen plotters.

The following instructions are described in this chapter.

Instruction	Summary
BP, Begin Plot	Indicates the beginning of a plot.
CT, Chord Tolerance Mode	Allows you to adjust the number of chords in an arc or circle by indicating how chord tolerance is specified.
DL, Download Character	Allows you to design characters and store them in a buffer for repeated use.*
EC, Enable Cutter	Turns on/off a plotter's automatic cutter.*
FR, Frame Advance	Advances the media to the next frame for long-axis plotting.
MC, Merge Control	Controls pixel color when two or more graphics intersect.
MG, Message	Writes a message to the plotter's front-panel display.*
MT, Media Type	Indicates the type of media loaded in the plotter.
NR, Not Ready	Takes the plotter offline.
OE, Output Error	Outputs any program errors (for debugging).
OH, Output Hard-Clip Limits	Outputs the hard-clip limit coordinates.
OI, Output Identification	Outputs a plotter identification string.
OP, Output P1 and P2	Outputs the P1/P2 coordinate locations.
OS, Output Status	Outputs plotter status information.
PS, Plot Size	Sets the hard-clip limits to a given size.
QL, Quality Level	Indicates the level of quality for plotting.
ST, Sort	Indicates the types of sorting the plotter can use.
VS, Velocity Select	Sets the pen speed for all or selected pens.

* Device-specific.

The following is a summary of the parameter formats and their *minimum* ranges. Your device may support a greater range than listed here.

Parameter Format	Minimum Ranges
Integer	2^{23} to $2^{23}-1$ (-8 388 608 to 8 388 607)
Real	2^{23} to $2^{23}-1$ (-8 388 608.000 0 to 8 388 607.999 9)
Clamped Integer	2^{15} to $2^{15}-1$ (-32 768 to 32 767)
Clamped Real	2^{15} to $2^{15}-1$ (-32 768.000 0 to 32 767.999 9)

Defining a Picture

A picture is defined as the portion of an HP-GL/2 data stream optionally begun with a Begin Plot (BP) instruction and terminated with an Advance Full Page (PG) instruction. This picture is divided into two portions: the picture header and the picture body. The picture header contains the instructions that set up the picture to be plotted. The picture body contains the instructions that actually cause the plotter to mark on the paper.

The Picture Header State

The plotter enters the picture header state on power-up and after receiving a BP, PG, or RP instruction. The following instructions, if used, must be issued in the picture header state. These are Merge Control (MC), Plot Size (PS), and Quality Level (QL). Each of these instructions affects the entire picture. If received after the picture header information, these instructions will generate errors and their functionality will be ignored.

NOTE: Precede the instructions IP (Input P1 and P2), IR (Input Relative P1 and P2), IW (Input Window), and/or AC (Anchor Corner) with the PS (Plot Size) instruction. PS restores these instructions to their defaults.■

The Picture Body State

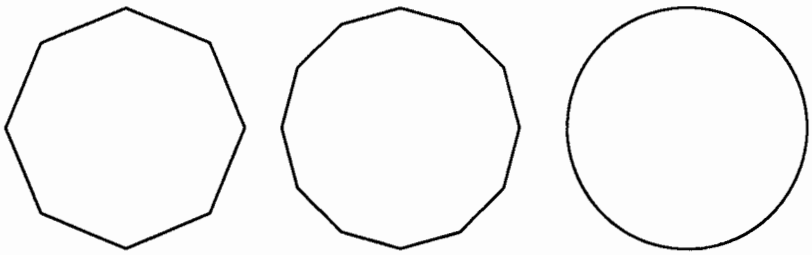
The picture header state is terminated when any instruction is received that would result in marks being made on the page. The instructions that indicate the beginning of the picture body state are as follows.

Picture Body Instructions	Group
AA, Arc Absolute AR, Arc Relative AT, Absolute Arc Three Point PA, Plot Absolute PD, Pen Down PE, Polyline Encoded PR, Plot Relative PU, Pen Up* RT, Relative Arc Three Point	*The Vector Group*
CI, Circle EA, Edge Rectangle Absolute EP, Edge Polygon ER, Edge Rectangle Relative EW, Edge Wedge FP, Fill Polygon PM, Polygon Mode Instruction RA, Fill Rectangle Absolute RR, Fill Rectangle Relative WG, Fill Wedge	*The Polygon Group*
CP, Character Plot LB, Label	*The Character Group*

* With parameters.

Chords and Chord Tolerance

To draw curves, the plotter draws a series of straight lines, called *chords*, to represent each arc segment. The apparent smoothness of the curve depends on the number of chords used to draw it; the more chords, the smoother the shape appears. The following illustration shows circles drawn with different numbers of chords.



The number of chords in any curved shape is determined by the chord tolerance—the allowable deviation from a smooth curve. Many instructions allow you to set the chord tolerance you want. Chord tolerance can be specified in two ways: as an angle in degrees or as the maximum distance the arc drawn may deviate from the true arc. These two methods are called chord angle and deviation distance. The chord angle is the default method. Use the Chord Tolerance Mode (CT) instruction to select the method you want to use.

NOTE: Do *not* use an adaptive line type when drawing circles, arcs, wedges, or polygons. The plotter will attempt to draw the complete pattern in every chord; there are 72 chords in a circle using the default chord angle.■

The Downloadable Set and User-Defined Characters

If you need special label characters or symbols that are not included in any of the character sets, you can design your own character or even an entire character set using the Download Character (DL) instruction.

The DL instruction allows you to define up to 94 characters to be stored in a buffer for repeated use during a plot. The character plot cell used by the DL instruction is always a fixed-space character cell, regardless of the spacing in the font currently selected.

Obtaining Plotter Output

When the plotter receives an output instruction, it responds by making the information available in the form of an output response. If you want to retrieve this information, your computer must read the output response.

Most languages use an input statement such as ENTER, INPUT#, READ, or READLN to read the output response. When you read the output response, be sure to specify the correct number and type of variable(s) the programming language will require to store the output response. For example, BASIC requires that a character string variable be in the form *A$*, and numeric variables be in the form *A*. The examples in this chapter use these conventions.

Refer to your programming language documentation for the correct input statement to use and the correct format for numeric and character string variables.

NOTE: Output instructions *must* be terminated with a semicolon.■

When an output response includes more than one piece of information, read each piece of information whether or not you need it. This ensures that the response is cleared from the output buffer. Read each piece of information into a separate variable. For example, the Output Hard-clip Limits (OH) instruction outputs four integers; your input statement might resemble the following.

```
INPUT #1, A,B,Y,Z
```

For Centronics Users

The Centronics interface supports data transmission in one direction only. However, output instructions will perform other expected operations, such as setting or clearing status bits.

For HP-IB Users

When using an HP-IB configuration, the response to an output instruction is sent after the output instruction is completely processed *and* the input buffer is completely empty. For example, the following sends the Output Error (OE) instruction and then other HP-GL/2 instructions before reading the output response.

```
"OE;PMØCI5ØØPM2FP"
```

In this example, the plotter does not respond to the Output Error (OE) instruction until the circle is completely filled.

The plotter signals the end of its output response with an output response terminator, noted in this manual by [TERM]. For HP-IB users, the output response terminator is a carriage return followed by a line feed (**CR LF**).

NOTE: Be sure the plotter is not set to **LISTEN ONLY.** Otherwise, the plotter will not send an output response and your program will halt. Refer to your product documentation for more information regarding HP-IB addressing.■

For RS-232-C Users

The plotter outputs information according to the handshake protocol. Your computer documentation should specify whether or not delays are required.

The plotter signals the end of its output response with an output response terminator, noted in this manual by [TERM]. When using the RS-232-C interface, the default output response terminator is a carriage return (**CR**).

Using Output Instructions

Use the following procedures for sending output instructions.

1. Send the output instruction to the plotter as you do other HP-GL/2 instructions. Note that all of the output instructions use terminators and no parameters.

2. Read the plotter's output response immediately using an input statement appropriate to your language, keeping in mind the number and type of variable(s).

Don't send multiple output instructions and then try to read the responses sequentially. This often leads to intermittent timing problems that are dependent on what the computer and plotter are currently doing.

Identifying the Plotter and Its Functions

When you have more than one peripheral (such as plotters and printers) connected to your computer, it may be necessary to have the plotter output its identification so that you know it is online. Use the Output Identification (OI) instruction to output the plotter ID to the computer.

Obtaining Error Information

Use the Output Error (OE) instruction for error retrieval. The OE instruction outputs the error number that corresponds to the first HP-GL/2 error the plotter receives. Use this instruction to identify errors by number when debugging a program. In addition, these errors set the error bit of the status byte in the response to Output Status (OS) instruction.

Obtaining Status Byte Information

The eight-bit status byte stores information about plotter operating conditions. You can use the Output Status (OS) instruction to learn the status of the plotter's current operating conditions. Each condition is assigned a bit number (from 0 to 7) and a corresponding decimal value. (The conditions, bit numbers, and corresponding decimal values are shown in the description for the OS instruction.)

You can obtain the value of the status byte by reading the response to the OS instruction or executing an HP-IB serial poll of the plotter.

Summary of Output Responses

The following table summarizes the output responses generated by HP-GL/2 output instructions. Use this table when programming in languages that require you to specify the variable type and maximum number of digits to be stored as variables (as in FORTRAN).

Note in the following table that numeric ranges do not include the sign of the response. For example, if a five-digit response is a negative value, a minus sign precedes the five digits. The minus sign does not replace a digit.

Instruction	Parameters Returned*	Type	Range
OE	Error number	Integer	1 digit
OH	XLL, YLL XUR, YUR	Integers	$\leq$ 6 digits each (plotter units)
OI	Model number	String	up to 30 characters
OP	P1X, P1Y P2X, P2Y	Integers	$\leq$ 8 digits each
OS	Status number	Integers	$\leq$ 3 digits

* In addition to these parameters, the output terminator [TERM] is always sent at the end of output, and commas are sent to separate parameters.

BP, Begin Plot

USE: Places the plotter into picture header state and indicates the beginning of a new plot.

SYNTAX: BP*(kind,value...(,kind,value);)*
or
BP*(;)*

Parameter	Format	Functional Range	Default
kind	clamped integer	1 to 4	no default
value	*kind*-dependent	*kind*-dependent	*kind*-dependent

REMARKS: BP performs the equivalent of PG and IN. It begins a new plot regardless of whether the previous plot was terminated. It makes the previous plot unavailable (that is, a BP followed by an RP [Replot] instruction produces nothing).

- **No Parameters** — Defaults all "values."
- **Kind,Value** — Defined as follows.

Kind	Format	Value
1	newstring	**Picture name:** Defaults to device-dependent name, such as a sequence number or the time of day if a clock is available. The parameter format is newstring (enclosed in quotation marks). This value can be used by devices that implement control-panel spool queue manipulation to identify the plot for the user. (The string parameter is assumed to be in the same language as for the control panel on power-up. That is, if you power up the machine in Spanish, the plotter assumes the picture name is encoded in the Spanish character set.)
2	clamped integer	**Number of copies:** Default is 1. When the plotter receives a Replot (RP) instruction, only the additional number of copies specified by RP are made. This is regardless of the number of copies specified (and already printed) by BP. For example, *BP2,3;"(picture name)"RP2"* results in a total of five copies of the plot, not nine.
3	clamped integer	**File disposition code:** Controls replot capability. **0** Enables replot; saves the file if room exists (default). **1** Destroys the file after printing (the file is not saved in memory or on disk). Also, this value prevents retrieval of the file through the RP instruction or the plotter's control panel.

(Continued)

Kind	Format	Value
4	clamped integer	**Render last plot if unfinished.** **0** No (default). **1** Yes.

Using BP in a Dual-Context Environment

If you entered HP-GL/2 mode with **ESC**%#B (where # is 0 or positive), the device ignores the PG functionality of the BP instruction. The instruction is mapped to IN regardless of the *kind,value* parameters.

When you enter HP-GL/2 mode with **ESC**%-1B, BP behavior is dependent on the device technology, as follows.

If the device is a buffered raster device, the default behavior of BP discards the previous plot if it has not yet been printed with either PG or RP. The instruction *BP4,1* allows overriding the default behavior by generating a PG.

For a pen plotter (or any device that plots as data is received), a *BP4,1* is always ignored and a BP instruction always results in a conditional page advance (if the page has been plotted on, the page is advanced).

Related Instruction	Group
IN, Initialize	*The Configuration and Status Group*

CT, Chord Tolerance Mode

USE: Determines whether the chord tolerance parameter of the AA, AR, AT, CI, EW, RT, and WG instructions is interpreted as a chord angle in degrees or as a deviation distance in current units. This instruction changes the way the number of chords is determined. This instruction is defaulted by DF or IN.

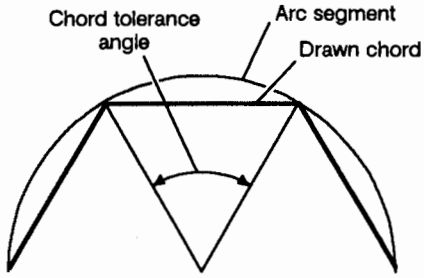
SYNTAX: CT*mode(;)*
or
CT*(;)*

Parameter	Format	Functional Range	Default
mode	clamped integer	0 or 1	0

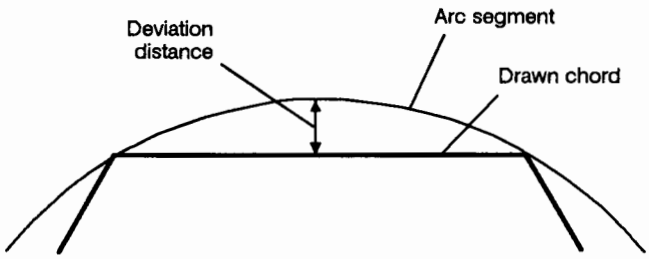
REMARKS: A plotted circle or arc actually consists of a series of straight line segments, or chords, that represent arc segments. Increasing the number of chords increases the smoothness of the circle (but uses more of the plotter's disk space). Chord tolerance is defined as the acceptable deviation from a smooth circle and can be established as either a chord angle or a deviation distance.

- **No Parameter** — Defaults the chord tolerance to a chord angle. Equivalent to (*CT0*).
- **Mode** — Specifies the following types of chord tolerance.

  0 Sets the chord tolerance mode to chord angle. A chord angle specifies, in degrees, the maximum angle created when lines from each end of the chord intersect the center point of the circle. When chord tolerance mode is specified as chord angle, a circle or arc will always have the same number of chords, regardless of its size.



1 Sets chord tolerance mode to deviation distance. Deviation distance specifies, in current units, the maximum distance between the chord and the arc segment it represents. When you specify a deviation distance, the number of chords in a circle will vary with its size.



Note that when you change from the default mode of chord angle to deviation distance, the chord angle parameter of the arc instructions (AA, AR, AT, and RT), the wedge instructions (EW and WG), and the Circle instruction (CI) will be interpreted as *current units*, not *degrees*.

Related Instructions	Group
AA, Arc Absolute AR, Arc Relative AT, Absolute Arc Three Point CI, Circle RT, Relative Arc Three Point	*The Vector Group*
EW, Edge Wedge WG, Fill Wedge	*The Polygon Group*

DL, Download Character

USE: Allows you to design characters and store them in a buffer for repeated use. Use DL whenever you want to create characters or symbols not included in the plotter's character sets.

SYNTAX: DL*character number((,up),X,Y...(,up),X,Y;)*
or
DL*character number(;)*
or
DL*(;)*

Parameter	Format	Functional Range	Default
character number	clamped integer	33 to 126	—
up	clamped integer	-128	—
X,Y coordinates	clamped integer	-127 to 127 primitive grid units	—

REMARKS: After you designate (SD or AD) and select (SS or SA) the HP-GL/2 Download character set (character set 531), use the DL instruction to create characters vector-by-vector, one character per command. Once defined with DL, characters can be used in the LB instruction. They can also be used in Symbol Mode (SM), but will not be centered unless defined that way in the character grid. All text attributes (size, slant, direction, and label origin) apply to downloadable characters. The characters you define in the downloadable set have fixed spacing.

- **No Parameters** — Clears all characters in the downloadable character set from the buffer. All characters are undefined.

- **Character Number** — Specifies the decimal value (33–126) of the character. If the character number has been previously defined, the new definition overwrites the old one.

  When not followed by additional parameters (*DLcharacter number*), clears the corresponding character from the downloadable set. Clearing a character means that it is undefined; referring to that character in a label string results in a space.

- **Up (-128)** — When used, this up flag indicates that the next coordinate pair defines a move with the pen up; subsequent moves are made with the pen down. (Pen up is the default for the first coordinate pair.)

- **X,Y Coordinates** — Absolute coordinates ranging from -127 to 127; drawn on a 32- × 32-unit grid. After the first pair, which always defines a pen-up move, all coordinates define moves with the pen down (unless preceded by an up flag).

The number of parameters following the character number parameter is not restricted. Your plotter can download 94 characters, with at least 20 points (coordinate pairs) in each character.

Defining a Downloadable Character

The plotter allocates space in the downloadable character buffer as needed. The DL font uses a fixed overhead of 206 bytes (consumed when the first character is downloaded), plus 2 overhead bytes per defined character. The points in a DL character average 1½ bytes each.

1. Design the character in absolute units on a 32- × 32-unit grid in a 48- × 64-unit cell.

   NOTE: The origin (0,0) is in the lower-left corner of the grid. This is the same grid used for the fixed-vector character sets in the plotter. The area occupied by a 32- × 32-unit grid approximates the size of an uppercase *A*. The downloaded character may extend outside this grid to ±127 units on each axis.■

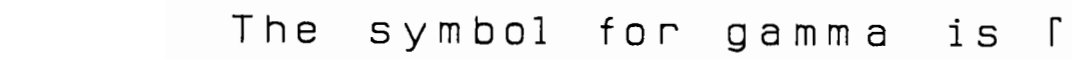
2. Assign a character number (decimal code) to the downloadable character.

3. Designate the starting point with the first X,Y coordinate pair (this will always be a pen-up move).

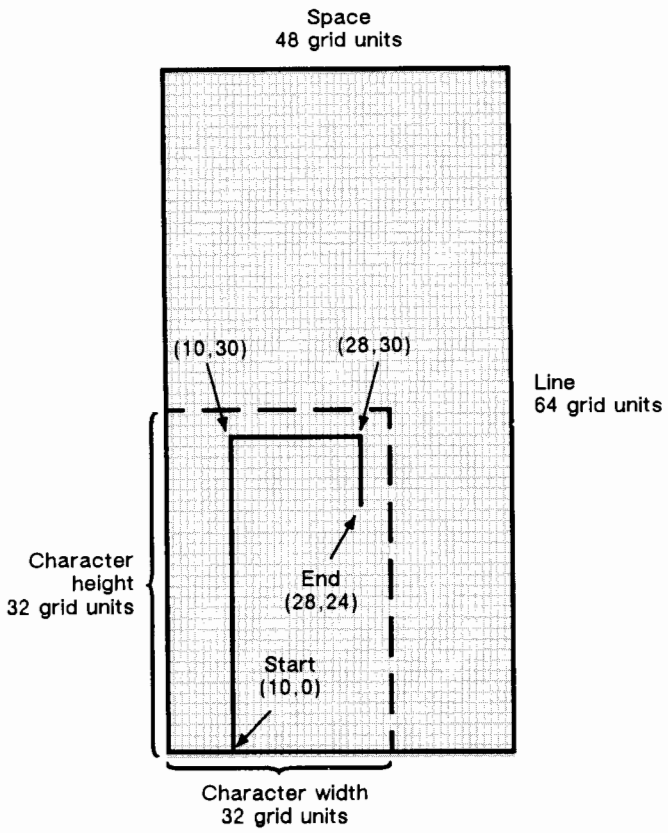
4. Specify the vectors of the character using absolute X,Y coordinates.

EXAMPLE: The program example defines downloadable character 65 as a gamma symbol. Note in line 70 that the shift-out control character (decimal code 14) is used to switch to the alternate font.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPINPS5000,7000SP1"
30  PRINT #1, "AD1,531,3,3"
40  PRINT #1, "DL65,10,0,10,30,28,30,28,24"
50  PRINT #1, "PA300,300"
60  PRINT #1, "LB The symbol for gamma "+CHR$(3)
70  PRINT #1, "LBis "+CHR$(14)+"A"+CHR$(3)
80  PRINT #1, "PUSP0PG;"
90  END
```



The next illustration shows how the gamma symbol fits in its character grid.



Related Instructions	Group
AD, Alternative Font Definition LB, Label SA, Select Alternate Character Set SD, Standard Font Definition SS, Select Standard Character Set	*The Character Group*

ERRORS:

Condition	Error	Plotter Response
Y-coordinate missing	2	ignores instruction
up flag follows X-coordinate	2	ignores instruction
parameter out of range	3	ignores instruction
X,Y data out of range	3	ignores instruction
buffer overflow	7	ignores instruction

EC, Enable Cutter

USE: Enables or disables the automatic cutter function on your plotter. Not all devices have an automatic cutter.

SYNTAX: EC(*n*;)
or
EC(;)

Parameter	Format	Functional Range	Default
n	clamped integer	*device-dependent*	enabled

REMARKS: If the cutter is enabled, cutting is done after each PG and RP instruction. Rollfeed devices that do not have a cutter may draw a line where the paper should be cut.

- **No parameter** — Enables the cutter function.
- **n** — Disables the cutter function. This parameter can be any number in the plotter's range.

FR, Frame Advance

USE: Advances the media to align adjacent frames, forming the equivalent of a long-axis plot. The plotter treats each frame as a separate window and plots only the data falling within that frame. (Using the Plot Size [PS] instruction for long-axis plotting is simpler and faster than than FR.)

SYNTAX: FR(;)

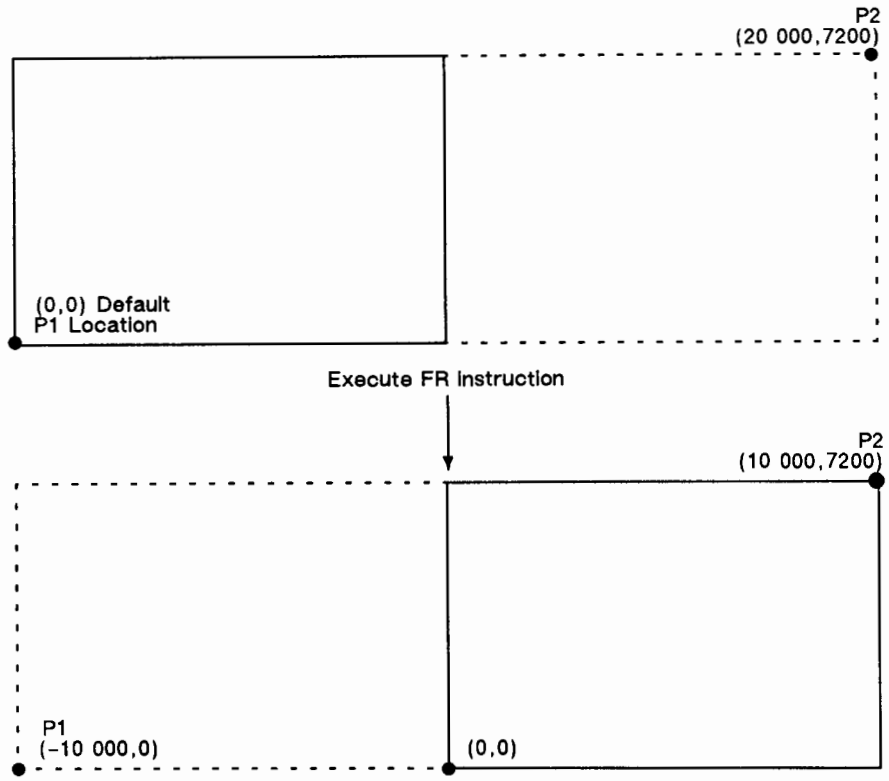
REMARKS: FR updates the current pen location to the new plotter-unit origin, leaves the pen in the up position, and clears the polygon buffer.

The length parameter of the Plot Size (PS) instruction determines the frame size. When *length* ≥ *width*, the orientation is reverse landscape and each FR instruction extends the plot in the direction of the positive X-axis. When *length* < *width*, the orientation is reverse portrait and each FR instruction extends the plot in the direction of the positive Y-axis.

The length advanced with an FR instruction is shorter than that of a PG instruction; the margins between the plotter areas are deleted so that the frames share a common edge.

After each frame advance, the plotter-unit origin moves to the lower-left corner of the new frame. The physical locations of P1 and P2 are retained, but the logical locations relative to the current origin change.

For example, in the following, the P1/P2 values are input as *IP0,0,20000,7200*. After sending FR, the logical values of P1 and P2 on the new frame become (-10 000,0) and (10 000,7200), respectively.



The Anchor Corner (AC) and window retain their coordinate values after each frame advance. If specified in user units, they maintain their physical locations with respect to P1 and P2; if specified in plotter units, they shift their physical locations with the plotter-unit origin. If you have not sent an AC or IW instruction, each frame advance defaults the anchor corner and window with respect to the hard-clip limits of the current frame.

FR is ideal when used with rollfeed plotters. When using single-sheet plotters, frames will share a common edge only when complete frames will fit on the current sheet. If multiple frames will not fit, the current page is printed and ejected. The next sheet loaded will contain the new frame. Any plotter that can store an entire multiframe plot can print multiple copies using either the BP or RP instructions.

Related Instructions	Group/Extension
PS, Plot Size	*The Technical Graphics Extension*
IW, Input Window	*The Configuration and Status Group*
AC, Anchor Corner	*The Line and Fill Attributes Group*

MC, Merge Control

USE: Provides control over the color of those pixels where two or more graphics intersect on the page. This instruction is for raster devices only; it performs no function on pen plotters. When used, this instruction must appear with the picture header information.

SYNTAX: MC*mode(;)*
or
MC*(;)*

Parameter	Format	Functional Range	Default
mode	clamped integer	0 or 1	0 (off)

REMARKS: The entire plot must be done with a single setting of this instruction. You cannot turn this feature on, off, then on again within the same plot. Therefore, this instruction is ignored in the picture body state.

- **No parameter** — Defaults merge control to off. Equivalent to *MC0*.
- **Mode** — Specifies one of the following modes.

  0 **Merge control off** — Each graphic replaces the pixels at the destination location; overwrites the pixels without merging (default).

  1 **Merge control on** — Plotter creates a merge color at the intersection of the vectors.

The merged color is a result of a device-dependent table lookup algorithm in the plotter. The table lookup functions on a pixel-by-pixel basis and applies to all graphics done with HP-GL/2 instructions: vectors, text, polygon fill (including raster patterns).

MG, Message

USE: Writes a message to the display on a plotter's control panel. If the plotter does not have a control panel with a display, the instruction is ignored.

SYNTAX: MG(*message;*)
or
MG(*;*)

Parameter	Format	Functional Range	Default
message	newstring	any character	no default

REMARKS: Character codes in the message are interpreted in the character set enabled at power-up for display of control panel menu commands and error messages. The character set selected for labeling does not affect the display of this instruction.

- **No parameter** — Clears the display.

- **Message** — Long text lines automatically wrap to the next line, if one exists. Once the end of the display is reached, however, the message does not wrap and overwrite the beginning of the display.

  You can use the following control codes within your message. All other control codes are ignored. Some computers allow you to enter these characters from the keyboard. The keyboard equivalents are given if your computer allows this.

Control Character	CHR$ Function	Keyboard Equivalent
BS (backspace)	CHR$(8)	**CONTROL** and **H**
LF (line feed)	CHR$(10)	**CONTROL** and **J**
CR (carriage return)	CHR$(13)	**CONTROL** and **M**

  If your plotter uses one display for displaying messages and control-panel menus, the plotter switches between the MG message and the control-panel menu (for user responses). The plotter restores the MG message after the user responds to the control-panel menu. The plotter retains the message until it is overwritten or cleared by another MG instruction or the plotter is initialized.

MT, Media Type

USE: Indicates the type of media loaded in the plotter.

SYNTAX: MT*type(;)*
or
MT*(;)*

Parameter	Format	Functional Range	Default
type	clamped integer	0 to 5	0 (paper)

REMARKS: Your device uses this information according to its plotting or printing technology. Your HP-GL/2 device might change resolution, drop volume, alter plot/print speed, change from bidirectional printing to unidirectional printing, or respond in some other manner as necessary. The actual changes made will take into account both the MT and QL (Quality Level) instructions, internal knowledge of the pen type, and any control-panel setups.

- **No parameter** — Defaults media type to paper. Equivalent to *MT0*.
- **Type** — Specifies the type of media according to the following parameter values.

  0 Paper (default).

  1 Transparency.

  2 Vellum.

  3 Polyester film (such as Mylar).

  4 Translucent paper.

  5 Special paper.

All devices will recognize the types listed, though several of the types may be treated in the same manner (for example, 1 and 3).

NR, Not Ready

USE: Takes the plotter offline for a specified amount of time. This can enable you to set control-panel conditions before starting your plot.

SYNTAX: NR(*timeout;*)

Parameter	Format	Functional Range	Default
timeout	clamped integer	*device-dependent*	0

REMARKS: Place this instruction at the beginning of a plot after the IN instruction. The timeout feature of this instruction lets the user set up pen groupings, media registration, pen force and acceleration, etc. The plotter restores its online status and resumes plotting when either the timeout elapses or the user puts the device back online (whichever comes first).

- **No parameter** — Defaults timeout to zero (no timeout). Same as *NR0*.
- **Timeout** — Specifies in seconds how long a user may need to perform any control-panel setup operations or media loading. The plotter will go online and resume plotting when either the timeout elapses or the user puts the plotter back online (whichever happens first).

NOTE: On plotters that cannot perform a page advance, the PG instruction places the plotter in a "not ready" state. The plotter will remain offline until new media is loaded or the necessary user interaction occurs.■

OE, Output Error

USE: Outputs a number corresponding to the HP-GL/2 error received by the plotter since the most recent OE, BP, or IN instruction, or power-up. Use OE for debugging programs. (Do *not* use on networks or Centronics interfaces.)

SYNTAX: OE;

NOTE: You *must* use a terminator (;) with output instructions.■

Parameter	Response	Format	Range
none	error number	clamped integer	0 to 7

REMARKS: The OE instruction outputs an integer (within the range 0 to 7) corresponding to the first HP-GL/2 error (if any) that occurred. The plotter outputs only the first error. If you suspect more than one error, place the instruction in as many locations in your program as necessary. The following table defines the error numbers.

HP-GL/2 Error Number	Meaning
0	No error.
1	Unrecognized instruction (or not recognized in current state).
2	Wrong number of parameters.
3	Out-of-range or invalid parameter.
4	Not used.
5	Not used.
6	Position overflow.
7	Buffer overflow/out of memory.

After outputting the error number, the plotter sets its internal error number to zero and clears bit 5 of the status byte, indicating that no error has occurred since the last OE.

EXAMPLE: Note that your computer may use different statements and format to read input from a peripheral. Use the statements your computer requires. In the following, line 30 contains two errors.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "SP1PA1000,1000,20EDOE;"
40  INPUT #1, A
50  PRINT A
60  END
```

first error (wrong number of parameters)

second error (unrecognized instruction)

Plotter response: 2 [TERM]

By referring to the table, you know that error 2 indicates the wrong number of parameters. Once the first error is corrected, run the program again to find the other HP-GL/2 error.

OH, Output Hard-Clip Limits

USE: Outputs the X,Y coordinates of the current hard-clip limits to the computer. Use OH to determine the plotter-unit dimensions of the area in which plotting can occur. (Do *not* use on networks or Centronics interfaces.)

SYNTAX: OH;

NOTE: You *must* use a terminator (;) with output instructions.■

Parameter	Response	Format	Range
none	X_{LL},Y_{LL},X_{UR},Y_{UR}	integer	hard-clip limits

REMARKS: The coordinates are always expressed in plotter units and represent the lower-left (X_{LL},Y_{LL}) and upper-right (X_{UR},Y_{UR}) corners of the hard-clip limits. After sending the OH instruction, have your program immediately read the plotter's output response.

EXAMPLE: The following program outputs the plotter identification. Note that your computer may use different statements and format to read input from a peripheral. Use the statements your computer requires.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "OH;"
40  INPUT #1, A,B,Y,Z
50  PRINT A,B,Y,Z
60  END
```

Related Instruction	Extension
PS, Plot Size	*The Technical Graphics Extension*

OI, Output Identification

USE: Outputs the plotter's identifying order number. This information is useful in a remote operating configuration (where several plotters are connected to the computer) to determine which plotter model is online, or when software needs the plotter's model number. (Do *not* use on networks or Centronics interfaces.)

SYNTAX: OI;

NOTE: You *must* use a terminator (;) with output instructions.■

Parameter	Response	Format	Range
none	plotter ID	character string	up to 30 characters

REMARKS: The plotter outputs its order number and letter as a character string.

EXAMPLE: The following program outputs the plotter identification. Note that your computer may use different statements and format to read input from a peripheral. Use the statements your computer requires.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "OI;"
40  INPUT #1, A$
50  PRINT A$
60  END
```

OP, Output P1 and P2

USE: Outputs the X,Y coordinates (in plotter units) of the current scaling points P1 and P2 to the computer. Use OP to help compute the number of plotter units per user unit when scaling is on. OP can also be used with the Input Window (IW) instruction to programmatically set the window to P1 and P2. (Do *not* use on networks or Centronics interfaces.)

SYNTAX: OP;

NOTE: You *must* use a terminator (;) with output instructions.■

Parameter	Response	Format	Range
none	P1x,P1y,P2x,P2y	integer	*device-dependent*

* Note that P2 tracks P1 and can be outside your plotter's range.

REMARKS: The P1/P2 coordinates are output as plotter units. After sending the OP instruction, have your program immediately read the plotter's output response.

Upon completion of output, bit position 1 of the status word is cleared (refer to the Output Status [OS] instruction).

EXAMPLE: Note that your computer may use different statements and format to read input from a peripheral. Use whatever is required by your computer. The following reads output from the plotter and prints the information on the computer's screen.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "OP;"
40  INPUT #1, A,B,Y,Z
50  PRINT A,B,Y,Z
60  END
```

Related Instructions	Group/Extension
OS, Output Status PS, Plot Size	*The Technical Graphics Extension*
IP, Input P1 and P2 IR, Input Relative P1 and P2	*The Configuration/Status Group*

OS, Output Status

USE: Outputs the decimal value of the status byte. Use OS when debugging a program. (Do *not* use on networks or Centronics interfaces.)

SYNTAX: OS;

NOTE: You *must* use a terminator (;) with output instructions.■

Parameter	Response	Format	Range
none	status number	clamped integer	0 to 255

REMARKS: On execution of the OS instruction, the internal 8-bit status byte is converted to an ASCII integer between 0 and 255 and output to your computer. Instruct your computer to read the output response; then refer to the following table and find the *largest decimal value* that can be subtracted from the output response. The condition corresponding to the decimal value has been met.

Continue subtracting the largest possible decimal value from the remainder of the output response. Each time you subtract a decimal value, the corresponding condition has been met. Continue this process until the remainder is zero.

Decimal Value	Meaning	Bit Number
1	Pen is down.	0
2	P1 or P2 newly established; cleared by OP.	1
4	Not used (bit always set to 0).	2
8	Initialized; cleared by OS.	3
16	Ready for data, buffer empty (bit always set to 1).	4
32	Error; cleared by OE.	5
64	Not used (bit always set to 0).	6
128	Not used (bit always set to 0).	7

On power-up, the status byte is 26, the sum of 16 (ready for data), 8 (initialized), and 2 (P1/P2 newly established). On execution of OS, bit position 3 is cleared and the status byte is 18.

EXAMPLE: The following outputs the numeric representation of the status byte. Note that your computer may use different statements and format to read input from a peripheral. Use whatever is required by your computer.

```
10  'Insert configuration statement here
20  PRINT #1, CHR$(27)+"%-1BBPIN"
30  PRINT #1, "OS;"
40  INPUT #1, S
50  PRINT S
60  END
```

Related Instructions	Group/Extension
OE, Output Error OP, Output P1 and P2 PS, Plot Size	*The Technical Graphics Extension*
IN, Initialize IP, Input P1 and P2 IR, Input Relative P1 and P2	*The Configuration/Status Group*

PS, Plot Size

USE: Sets the hard-clip limits to a given size. Use PS to simplify long-axis plotting or to minimize paper waste when drawing small plots. This is especially useful with plotters using rollfeed media. When used, this instruction must appear in the picture header information.

SYNTAX: PS(*length(,width);*)
or
PS*(;)*

Parameter	Format	Functional Range	Default
length	integer	*device-dependent*	hard-clip limits*
width	integer	*device-dependent*	hard-clip limits

* The default length for single-sheet media is the hard-clip limits; for rollfeed media it is approximately 1.5 times the media width.

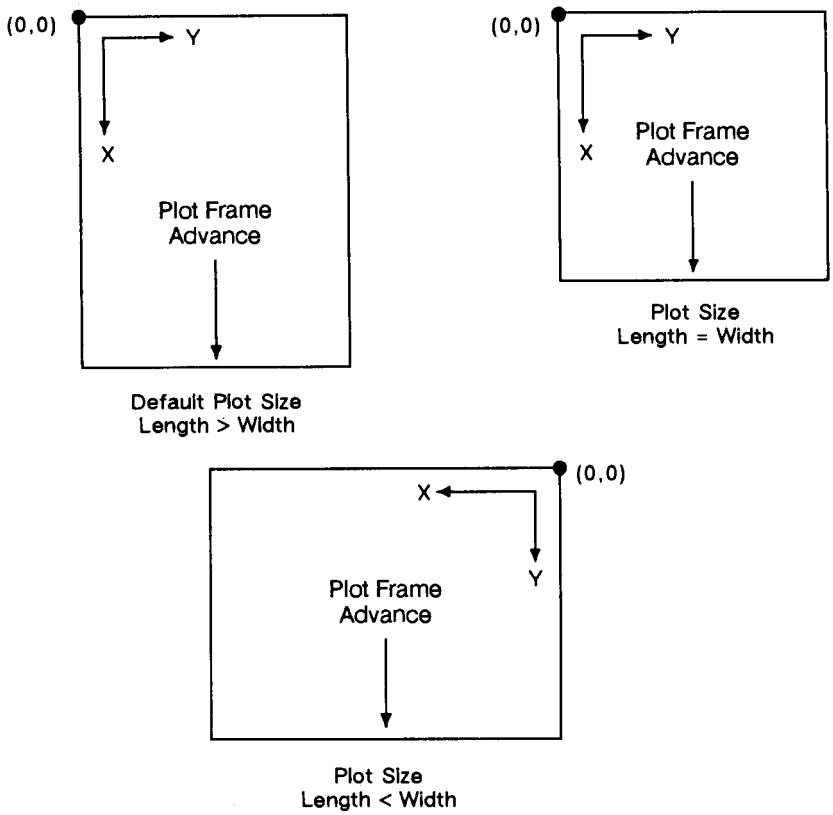
REMARKS: Send PS immediately *after* the BP or IN instruction (IN defaults PS if no marks have been made on the paper) and *before* any drawing instructions. (DF does not default PS.) Note that the entire plot must be done with a single PS instruction. You cannot change the plot size in the middle of a plot.

- **No Parameters** — Defaults the plot length and width to the hard-clip limits (the maximum printable area).

- **Length** — Establishes the new length, in plotter units, of the hard-clip limits. The length always corresponds to the direction of the plot frame advance. Refer to the documentation for your plotter or HP-GL/2 option for the maximum length for a single plot. If the length is zero, the plotter ignores the instruction.

- **Width** — Establishes the new width, in plotter units, of the hard-clip limits. The width is always the horizontal direction. If the width is zero, the plotter ignores the instruction.

  If you specify a plot size larger than your media's maximum plotting area, your plot size will be clamped to the hard-clip limits. The following illustrate how PS orients the default coordinate system according to the length and width after clamping. The default origin defines the lower-left corner of the hard-clip limits and is always located on the side of the plot frame opposite the next plot frame.



If either length or width is less than zero, the instruction is ignored.

PS defaults P1 and P2 to the lower-left and upper-right corners of the hard-clip limits, defaults the size of the user-defined window and the location of the anchor corner, and clears the polygon buffer. Any previous rotation (RO) is then added to the default X-axis orientation. Current scaling is applied to the new physical area. The plotter also updates the current pen location to the lower-left corner of the new plotting area.

If a Rotate Coordinate System (RO) instruction is sent after PS, the direction of the X-axis changes. The implementation of RO is relative to the autorotated position. PS sent after RO does not change the RO rotation, but does update the X-axis to the new longer side.

The Advance Page (PG) instruction will advance the paper by the distance of the length parameter plus the necessary white space between plots. The following lists standard paper sizes and equivalent PS parameters.

Measurement	Standard Paper Sizes	Equivalent PS Parameters*
English	8½ × 11 in. (A-size)	PS8900,7350
	11 × 17 in. (B-size)	PS15 000,9 850
	17 × 22 in. (C-size)	PS21 050,15 000
	22 × 34 in. (D-size)	PS32 300,21 050
Metric	210 × 297 mm (A4-size)	PS9 600,7 100
	297 × 420 mm (A3-size)	PS14 550,10 600
	420 × 594 mm (A2-size)	PS22 450,14 550
	594 × 840 mm (A1-size)	PS31 400,22 450

* These plot sizes are based on 16 mm margins on three sides of the media, the fourth margin being 40 mm. These values are also based on loading C-size media horizontally in the plotter, with all other media being loaded vertically. Refer to your plotter manual for media-loading instructions.

NOTE: When combining the Plot Size and Scale instructions and accurate scaling is a *must*, note the following. If you specify a plot size larger than your media's maximum plotting area, your plot size will be reduced to the hard-clip limits and your user units will be smaller than you intended. To correct, add an IP instruction so that the P1/P2 area is equal to the intended size of your PS instruction (IP0,0,PS*length*PS*width*). This moves P2 off the page, but guarantees accurate scaling. Place the IP instruction between the PS and SC instructions.■

Note that IN defaults this instruction when the plotter is in the picture header state; DF does not default PS.

Related Instructions	Group/Extension
FR, Frame Advance OH, Output Hard-Clip Limits	*The Technical Graphics Extension*
PG, Advance Full Page	*The Configuration/Status Group*

ERRORS:

Condition	Error	Plotter Response
PS received when plotter is in picture body state	1	ignores instruction
length or width ≤ 0	3	ignores instruction

QL, Quality Level

USE: Sets "draft" or "final" mode for your output. Use QL on raster devices to optimize your toner usage when final quality is not necessary.

SYNTAX: QL*(quality level;)*
or
QL*(;)*

Parameter	Format	Functional Range	Default
quality level	clamped integer	0 to 100	*device-dependent*

REMARKS: The entire plot must be done with one quality level setting. You cannot change quality levels in picture body state.

- **No Parameter** — Device-dependent. Each plotter will make different use of this instruction depending on its hardware technology and firmware implementation. The result will be seen in different speed/quality trade-offs. For example, a pen plotter primarily varies pen speed, but may vary acceleration as well. An electrostatic plotter might vary paper speed, resolution, or rasterization algorithms. Refer to the manual for your product or HP-GL/2 option for the number of quality levels and the effects the various levels have on the output.

- **Quality Level** — Specifies the level of quality for your plot from 0 (draft quality) to 100 (presentation or final quality). The number of quality levels supported is device-dependent. When quality level is zero (draft), a plotter may not implement Merge Control (MC) or may limit it to vectors only.

  A plotter with only one quality level will ignore this instruction. A plotter with two or more levels will support at least 0 and 100 and map any other value to one of the supported values. Mapping to another value may occur either through *rounding* to the nearest supported value (for example, a plotter with only two levels would treat QL60 the same as QL100) or through *thresholding* (for example, anything over 80 is mapped to 100).

  For a pen plotter, QL is primarily a speed control. The pen speed set by the VS instruction or the plotter's control panel is the maximum speed, which corresponds to *QL0*. If the pen speed is not set by VS or from the control panel, the plotter determines the maximum speed based on its knowledge of media type (from the MT instruction) and pen type (from the carousel). Minimum speed (corresponding to *QL100*) is linearly interpolated between the media and pen types.. The default level for pen plotters is usually *QL0* (maximum speed).

ERRORS:

Condition	Error	Plotter Response
QL received when plotter is in picture body state	1	ignores instruction

ST, Sort

USE: Specifies how the plotter sorts vectors for plotting.

SYNTAX: ST*(switches;)*
or
ST*(;)*

Parameter	Format	Functional Range	Default
switches	clamped integer	-1, 0, and any sum of 1, 2, and 4 (1-7)	*device-dependent*

REMARKS: The sorting switches are device-dependent and vary according to hardware technology, buffer size, and plotting/CPU speed trade-offs.

- **No parameter** — Defaults the sorting to a device-dependent switch. Refer to the manual for your product or HP-GL/2 option to determine what switches are supported and which is the default.

- **Switches** — Determines sorting as follows.

  -1 Turns on all sorting (or optimal combination).

  0 Turns off all sorting; plots in the order received.

  1 Pen sorting: vectors are sorted by pen color rather than in the order in which they were received.

  2 Endpoint swap (bidirectional plotting): swaps successive vector endpoints to minimize pen-up moves.

  4 Geographic sorting: sorts all vectors by their geographic area before moving to another area.

VS, Velocity Select

USE: Specifies pen speed. Use VS on pen plotters to optimize pen life and line quality for each pen and media combination. Increase line quality and create a slightly thicker line on any media by slowing the pen speed.

SYNTAX: VS*(pen velocity(,pen number;))*
or
VS(;)

Parameter	Format	Functional Range	Default
pen velocity	clamped integer	*device-dependent**	*device-dependent*
pen number	clamped integer	*device-dependent***	all pens

* Your plotter's fastest pen velocity may be greater or less than 60 cm/s. Refer to the manual for your plotter or HP-GL/2 option for more information.

** Your plotter carousel may handle a different number of pens, or if you are using a raster device, you may have a different number of logical pens you can use. Refer to the manual for your plotter or HP-GL/2 option for more information.

REMARKS: Use the VS instruction to increase the pen life and line quality produced by your pens. Slowing pen speed increases line quality.

The VS instruction remains in effect until another VS instruction is received or the plotter is initialized or set to default conditions.

- **No Parameters** — Sets the speed for all pens to the default value. Refer to the manual for your product or HP-GL/2 option for more information.

- **Pen Velocity** — Specifies the pen speed in centimeters per second (cm/s). You can increment the pen speed by 1 cm/s. The selected velocity applies only when the pen is down. Pen-up speed is device-dependent.

- **Pen Number** — Applies the pen speed to a specific pen. When this parameter is omitted, the velocity applies to all pens.

The following table lists the velocity range and recommended speed for each pen type.

Pen Type	Recommended Velocity Range (cm/s)	Recommended Plotting Speed (cm/s)
fiber-tip paper	5 to 80	40
fiber-tip transparency	5 to 15	10
disposable drafting	10 to 30	20
refillable drafting	5 to 15	15

Note that your plotter may let you set pen speed from the control panel. However, the pen speed you specify using the control-panel buttons may apply to all pens. Refer to the user documentation for your product for more information.

ERRORS:

Condition	Error	Plotter Response
pen velocity parameter greater than plotter maximum	none	executed at plotter default
pen number greater than 8	none	ignores instruction
more than 2 parameters	2	ignores extra parameter
number is ≤ 0	3	ignores instruction

10

The Palette Extension

This extension defines a block of instructions you can implement along with the HP-GL/2 kernel instruction set to satisfy the functionality of plotters with special palette features. This extension is especially useful with raster devices. If you have access to a pen plotter and a raster device, you will want to define a palette to get the most from your HP-GL/2 program on the raster device, while maintaining pen plotter functionality.

The following instructions are described in this chapter.

Instruction	Summary
CR, Set Color Range for Relative Color Data	Establishes the range for RGB data.
NP, Number of Pens	Sets the size of the HP-GL/2 palette.
PC, Pen Color Assignment	Assigns colors to specific pens.
SV, Screened Vectors	Selects the type of screening to apply to vectors and area fill.
TR, Transparency Mode	Specifies whether an overlaying white area is transparent or opaque.

The following is a summary of the parameter formats and their *minimum* ranges. Your device may support a greater range than listed here.

Parameter Format	Minimum Ranges
Integer	2^{23} to $2^{23}-1$ (-8 388 608 to 8 388 607)
Real	2^{23} to $2^{23}-1$ (-8 388 608.000 0 to 8 388 607.999 9)
Clamped Integer	2^{15} to $2^{15}-1$ (-32 768 to 32 767)
Clamped Real	2^{15} to $2^{15}-1$ (-32 768.000 0 to 32 767.999 9)

Defining Your Palette

It is good practice to define your palette just after initialization and before you begin sending graphics data to the plotter. Once defined, this information should not need updating. Defining a palette includes the following.

- Indicating the size of the palette (NP instruction).
- Assigning a color to each pen in the palette (PC instruction).
- Specifying the width for each pen in the palette (PW instruction*).

For maximum flexibility, you will want to allow the user to specify values for each of these areas.

For pen plotters, it is the responsibility of the user to put the proper pen (color and width) into the appropriate carousel stall. Also for pen plotters, the maximum number of logical pens is 32. Your pen plotter may support logical pens beyond the number of physical pens in your plotter. If your plotter holds a maximum of 8 physical pens in its carousel, the plotter performs a modulo function on any logical pen over 8 so that the plotter selects the appropriate physical pen from the carousel.

Effect of a Color Palette on Monochrome Devices

Monochrome devices will produce varying levels of grayscale area fills and lines for pens other than black and white in the color palette. However, the grayscale is unpredictable when drawing a raster pattern with a colored pen. For example, yellow will be converted to a light shade of gray; red will be converted to a darker shade of gray. A red pen used in a 10% raster fill pattern may produce an area fill more dense than expected.

* The PW instruction is described in chapter 7, *The Line and Fill Attributes Group*.

CR, Set Color Range for Relative Color Data

USE: Establishes the range for RGB (red/green/blue) data. This instruction maps current pen colors to a numeric range while leaving the current palette colors themselves unchanged. Pen plotters ignore this instruction.

On pen plotters:

SYNTAX: CR(*black-ref red,white-ref red, black-ref green, white-ref green, black-ref blue, white-ref blue;*)
or
CR(;)

Parameter	Format	Functional Range	Default
b-ref (red, green, blue)	clamped real	*device-dependent*	0
w-ref (red, green, blue)	clamped real	*device-dependent*	255

REMARKS: This instruction is returned to its defaults by the IN instruction.

- **No Parameters** — Defaults the black color references (*black-ref*) to 0 (zero) and white-references (*white-ref*) to 255.

- **Black-ref, White-ref** — Sets color to a range defined by a black- and white-reference value for each primary color (red, green, and blue). The first two parameters set the black- and white-references (respectively) for red; the second pair for green; and the final pair for blue.

  For example, the instruction *CR0,63,0,63,0,63* sets the white-reference to 63,63,63 and the black-reference to 0,0,0. The value for a medium blue would equal 0,0,31.

  And the instruction *CR4,63,0,127,0,31* sets the white-reference to 63,127,31 and the black-reference to 4,0,0. Medium blue, in this case, would be 4,0,15.

Using a black-reference of 0,0,0 and white-reference of 100,100,100 makes it easy to specify colors as percentages of their black- and white-reference values.

If the black-reference value for any primary is equal to the white-reference value for the same primary, the instruction is ignored.

ERRORS:

Condition	Error	Plotter Response
1 to 5 parameters	2	ignores parameters
more than 7 parameters	2	ignores extra parameters

NP, Number of Pens

USE: Establishes the size (number of pens) of the HP-GL/2 palette.

SYNTAX: NP*n(;)*
or
NP(;)

Parameter	Format	Functional Range	Default
n	clamped integer	*device-dependent**	*device-dependent***

* Parameter must be a power of 2.
** Default palette size for monochrome raster devices is 2. Default palette size for color raster devices is 8.

REMARKS: The palette is an array of logical pens, each having an associated color value and an associated width. Define pen colors through the Pen Color Assignment (PC) instruction, described later in this chapter. Establish pen widths through the PW and WU instructions (refer to chapter 7, *The Line and Fill Attributes Group*).

- **No Parameters** — Defaults the palette size for monochrome raster devices to 2 pens; for color raster devices to 8 pens.

- **n** — Sets the size of the palette as a power of 2. If *n* is not a power of 2, the plotter uses the next larger power of 2. The maximum value of *n*, while device-dependent, is equal to or greater than the number of distinct colors the plotter can produce. If *n* is greater than the maximum, the plotter uses the maximum-size palette.

  If the current pen is outside the range of the new palette size, the plotter applies a modulo function to select a pen number within the range of the new palette.

This instruction is usually defaulted by the IN instruction. If you are using your plotter in a dual-context mode (available on some raster devices only) and you entered HP-GL/2 context using **ESC**%2B or **ESC**%3B, then the PCL palette in effect when the context was entered is restored.

ERRORS:

Condition	Error	Plotter Response
$n < 2$	3	ignores instruction

PC, Pen Color Assignment

USE: Assigns colors to specific pens.

SYNTAX: PC*(pen(,red,green,blue);)*
or
PC*(pen;)*
or
PC(;)

Parameter	Format	Functional Range	Default
pen	integer	determined by NP	no default
red,green,blue	clamped real	determined by CR	(see table below)

REMARKS: PC remains in effect until another PC instruction assigns new values for a specific pen or all pens in the palette, or until the plotter is initialized by the IN instruction. The first 8 pens for color devices default to the colors in the table below, even when the palette is larger than 8 pens. The remaining pen colors are device-dependent. For a monochrome device, pen 0 defaults to white; all remaining pen colors default to black.

- **No Parameters** — Defaults the colors of all pens as follows.

Number of Pens In Palette	Pen Number	Color
2 (NP2)	0	White
	1	Black
4 (NP4)	0	White
	1	Black
	2	Red
	3	Green
8 (NP8)	0	White
	1	Black
	2	Red
	3	Green
	4	Yellow
	5	Blue
	6	Magenta
	7	Cyan

- **Pen** — Specifies the number of the pen being defined. When you specify only the pen number (and no RGB values), the pen assumes the color as specified in the table above. For example, *PC3* defaults pen 3 to green.

- **Red,Green,Blue** — Specify the primary component values for the specified pen. (Refer to the CR instruction earlier in this chapter for a description of the range associated with the RGB values.)

If a red, green, or blue parameter is outside of the color range defined in the CR instruction, the value is clamped to the color range limits.

EXAMPLE: The following line assigns red to pen 1, green to pen 2, and blue to pen 3 using the default color range.

```
PC1,255,Ø,ØPC2,Ø,255,ØPC3,Ø,Ø,255
```

ERRORS:

Condition	Error	Plotter Response
2 or 3 parameters	2	ignores instruction
more than 4 parameters	2	ignores extra parameters
out-of-range pen parameter*	3	ignores instruction

* The range for the pen parameter is defined by the size of the palette. Refer to the NP instruction.

SV, Screened Vectors

USE: Selects the type of screening (area fill) to be applied to vectors (lines, hatch patterns [fill types 3 and 4], arcs, circles, and edges of polygons, rectangles, and wedges). SV does not apply to solid fill types, stroked characters, or edges of characters.

SYNTAX: SV*screen type(,option1(,option2);)*
or
SV(;)

Parameter	Format	Functional Range	Default
screen type	clamped integer	0, 1, or 2	0
option1, option2	clamped integer	*screen type*-dependent	*screen type*-dependent

REMARKS: All screening patterns use the current anchor corner (refer to the AC instruction in chapter 7, *The Line and Fill Attributes Group*). This instruction is defaulted by DF and IN.

- **No Parameters** — Defaults to no screening (solid vectors) and sets previously specified options for other screen types to their defaults. Equivalent to *SV0*.
- **Screen Type** — Determines the screen type as follows.

  0 No screening. (The plotter ignores the *option1* and *option2* parameters.)

  1 Shaded fill.

  2 User-defined raster fill (refer to the RF instruction in chapter 7, *The Line and Fill Attributes Group*).

- **Option1, Option2 —** Dependent on the screen type as follows.

  For Screen Type 1 (Shaded Fill): Option1 specifies the shading percentage and is a value between 0 and 100. Option2 is ignored.

  For Screen Type 2 (User-defined Fill): Option1 specifies the index of the pattern being selected. If the pattern was defined using only 1s and 0s (refer to the RF instruction in chapter 7, *The Line and Fill Attributes Group*), it can be printed using the current pen. If the pattern includes pen numbers other than 0 and 1, option2 is ignored.

  Option2 is a Boolean specifying whether or not the current color should be applied. When the pattern was defined using only 1s and 0s, and option2 is "1" (true), the color of the current pen is applied to the 1s pixels in the pattern. If option2 is "0" (false) or is not present, the 1s pixels are printed in the color of pen number 1.

For user-defined fill types, use RF to set up your raster pattern and *SV2,index* to select screening for vector fill.

NOTE: When screening is in effect, the quality of vectors may degrade as line width decreases or when colors requiring super-pixeling are used. Results below the metric default line width are device-dependent.■

If you specify a screen type only (you do not specify option1 or option2), the plotter assumes the previous option1 and/or option2 values.

Related Instructions	Group
AC, Anchor Corner FT, Fill Type RF, Raster Fill Definition WU, Pen Width Unit Selection	*The Line and Fill Attributes Group*

TR, Transparency Mode

USE: Defines how the white areas of graphics images are plotted by turning the transparency mode on or off. The IN and DF instructions return TR to its default.

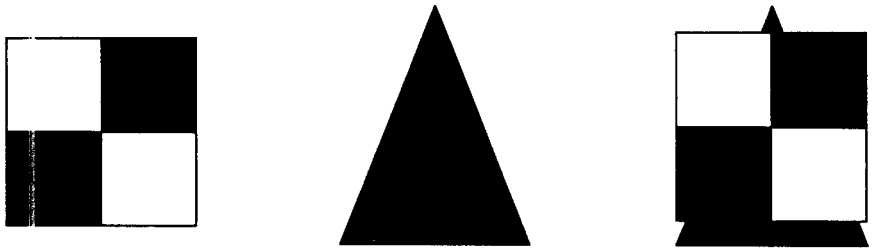
SYNTAX: TR*(n)*;
or
TR;

Parameter	Format	Functional Range	Default
n	clamped integer	0 or 1	1 (on)

REMARKS: White is defined as the white-reference specified by the CR instruction (described earlier in this chapter). This instruction is defaulted by the DF and IN instructions.

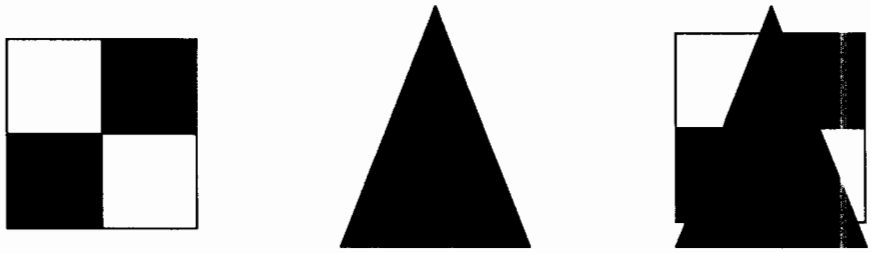
- **No Parameter —** Defaults transparency mode to on. This is equivalent to *TR1;*.
- **n —** Turns transparency mode on or off as follows.

  0 Turns transparency mode off; overlaying white areas become opaque.



Transparency Mode Off

1 Turns transparency mode on; overlaying white areas are transparent. The white areas do not obscure other graphics images.



Overlaying image | Previously defined image | Final image

Transparency Mode On

ERRORS:

Condition	Error	Plotter Response
$n \geq 2$	3	ignores instruction
2 or more parameters	2	ignores extra parameters

11

The Dual-Context Extension

This extension is implemented by devices that support both Hewlett-Packard's printer command language (PCL) and HP-GL/2. This extension gives you the flexibility to switch between PCL and HP-GL/2 data on devices that provide both text and vector graphics publishing features.

Along with instructions in the typical HP-GL/2 format, this extension includes instructions in the PCL format. This chapter assumes you have some familiarity with PCL instruction format.

The following instructions are described in this chapter.

Instruction	Summary
ESC%#A, Enter PCL Mode	Instructs the device to interpret subsequent instructions as PCL commands.
ESCE, Reset	Restores the raster device to certain defaults.
FI, Primary Font Selection by ID	Designates a font as the primary font.
FN, Secondary Font Selection by ID	Designates a font as the secondary font.
SB, Scalable or Bitmap Fonts	Specifies the type of font that may be used in labeling.

Using Dual-Context PCL Instructions

The following two Dual-Context Extension instructions are in the PCL format.

- **ESC**%#A
- **ESC**E

These instructions are PCL escape sequences because they use the escape control code (decimal code 27) to begin the sequence. **ESC**E is a two-character escape sequence. It requires no parameters. In the **ESC**%#A instruction, the symbol "#" represents the parameter variable for a numeric value.

The **ESC**%-1B instruction, which we recommend you use at the beginning of your HP-GL/2 programs, is also a PCL instruction (but it is not considered a Dual-Context Extension instruction). This instruction puts your device in HP-GL/2 mode. All subsequent data is interpreted as HP-GL/2 information.

The **ESC**%#A instruction, besides entering PCL mode, controls the translation of pen location and palette information to PCL equivalents. In the PCL format, the pen location is called the current active position (CAP). The CAP can be moved explicitly anywhere within the logical page by using PCL control codes and escape sequence. Refer to the *HP PCL Programmer's Guide* for more information on PCL programming.

After **ESC**E, the CAP is "floating" until printable data or a command affecting the CAP is received. This means the CAP moves in accordance with PCL orientation, top margin, left margin, and line-spacing instructions. Once printable data or an instruction affecting CAP is received, however, the CAP is "fixed" (it no longer is affected by changes to orientation, margins, or spacing).

Modifications to HP-GL/2 Instructions in Dual-Context Mode

The functionality of the following HP-GL/2 instructions is modified when used in a dual-context environment.

Instruction	Modification
AC, Anchor Corner	An IN, DF, or **ESC**E instruction places the anchor point at the lower-left corner of the picture frame relative to the current coordinate system.
DF, Default Values	AC sets to the lower-left corner of the picture frame, relative to the current coordinate system. IW sets the user window equal to the PCL frame window.

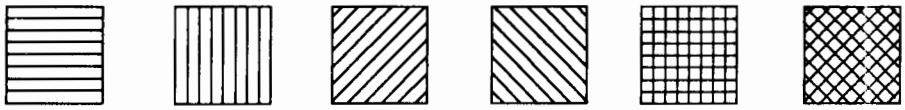
(Continued)

Instruction	Modification
FT, Fill Type	Additional fill types (21 and 22) are imported from PCL. Fill type 21 is a predefined patterned fill. *Option1* specifies the pattern type. A fill type is specified using a value between 1 and 6. *Option2* is ignored. Fill type 22 selects the PCL user-defined fill specified using the **ESC***c#W instruction. *Option1* specifies the PCL ID of the user-defined fill pattern. *Option2* is ignored. If *option1* is invalid for any reason (e.g., the pattern was deleted), the plotter selects a solid fill in the current color.
IN, Initialize	If HP-GL/2 mode was entered with **ESC**%2B or **ESC**%3B, IN restores the palette brought in from PCL mode.
IP, Input P1 and P2	IP sets P1 to the lower-left corner of the PCL picture frame (as viewed from the current HP-GL/2 orientation) and P2 to the opposite corner.
IR, Input Relative P1 and P2	The references to the hard-clip limits are replaced by the PCL picture frame. The default locations for P1 and P2 are the lower-left and upper-right corners of the picture frame, as viewed from the current orientation.
IW, Input Window	IW defaults the window to the PCL picture frame. The maximum printable area is the intersection of the user-defined window, the PCL picture frame, the PCL logical page, and the hard-clip limits. If parameters are specified in plotter units, they are scaled by the picture frame scaling factor; that is, the ratio of the PCL picture frame size to the HP-GL/2 plot size.
PG, Advance Full Page	PG is ignored in PCL mode because it could cause undesirable results when importing plots. Use a form feed (FF) character in PCL mode to eject the page. A PCL reset, page length, page size, orientation, or input cassette control instruction causes a conditional page eject. A page eject caused by a PCL instruction does not affect the HP-GL/2 cursor location.
PM, Polygon Mode	**ESC**E is recognized in polygon mode.
PW, Pen Width	Metric widths are scaled by the ratio of the size of the PCL picture frame to the HP-GL/2 plot size. For example, if the HP-GL/2 plot size is twice as large as the PCL picture fame, *WUPW.3* sets the width of vectors to 0.15 mm.

(Continued)

Instruction	Modification
RO, Rotate Coordinate System	Rotations are relative to the default HP-GL/2 coordinate system as defined for PCL. P1 or P2 may be rotated outside the current picture frame; they can be relocated to the lower-left and upper-right corners of the picture frame by sending an IP or IR instruction.
RP, Replot	RP is ignored in PCL; a page eject can only be accomplished from the PCL context (use a form feed [FF] character).
SC, Scale	When scaling is off, current units are as follows: (plotter units × (PCL picture fame size/HP-GL/2 plot size)) *Left* and *bottom* parameters are relative to the PCL picture frame, as viewed from the current orientation. The directional implications of *left* and *bottom* assume the default P1/P2 orientation.
SI, Absolute Character Size	The plotter adjusts the *width* and *height* parameters (in centimeters) by the picture frame scaling factor.
SP, Select Pen	When the HP-GL/2 context imports the current PCL palette and the number of bits per index is >1 (set by **ESC***r#U), the PCL palette entry with an index of 0 defines the color of pen 1; the palette entry with an index of 1 defines the color of pen 2. In general, the palette entry with an index of *n* defines the color of pen *n* + 1.
TD, Transparent Data	The sequence **ESC**E (ASCII values 27 and 69, respectively) is a special case. In normal mode within a label, **ESC**E resets the device and changes to the PCL environment. In transparent mode, **ESC**E will be printed (it will not perform a reset).
WU, Pen Width Unit Selection	If an HP-GL/2 plot size is specified, metric units are adjusted by the current PCL picture frame scaling factor.
SV, Screened Vectors	Additional screen types (21 and 22) are imported from PCL. Screen type 21 is a predefined patterned fill. *Option1* specifies the pattern type. A screen type is specified using a value between 1 and 6. *Option2* is ignored. Fill type 22 selects the PCL user-defined fill specified using the **ESC***c#W instruction. *Option1* specifies the PCL ID of the user-defined fill pattern. *Option2* is ignored. If *option1* is invalid for any reason (e.g., the pattern was deleted), the plotter selects a solid fill in the current color.

The following shows the additional patterns available for the FT and SV instructions (as described in the previous table).



ESC%#A, Enter PCL Mode

USE: Causes the device to start interpreting data as PCL commands. This instruction is ignored if it is received when the device is not in HP-GL/2 mode.

SYNTAX: **ESC%#A**

NOTE: You *must* use the capital "A" terminator with this instruction.■

Parameter	Functional Range	Default
#	0 to 3*	0

* Values other than 0, 1, 2, or 3 are interpreted as 0.

REMARKS: This instruction, besides entering PCL mode, affects only the CAP and palette. It does not affect any other PCL variables.

- **No Parameters** — Enters PCL mode. Equivalent to **ESC%0A**. Note that the HP-GL/2 palette does *not* become the default PCL palette.

- **#** — Enters PCL mode and sets logical pen location and palette according to the following parameter values.

  0 Specifies that the new PCL CAP and palette should retain the previous PCL CAP and palette.

  1 Sets the PCL CAP to the current HP-GL/2 pen location. If the current HP-GL/2 pen location is outside the bounds of the PCL logical page, the new PCL CAP will be the nearest point on the logical page boundary.

  2 Sets the PCL palette equivalent to the current HP-GL/2 palette.

  3 Sets the PCL CAP and palette to the current HP-GL/2 pen location and palette.

When the value is either 2 or 3 and the currently selected color index is larger than the HP-GL/2 palette being imported, the device performs a modulo function to obtain an index into the smaller palette.

ESCE, Reset

USE: Resets the device in HP-GL/2 mode as well as in PCL mode. After **ESC**E, the device should remain online to prevent either the loss of data or any disruption of established I/O communication.

SYNTAX: **ESC**E

REMARKS: This instruction sets the following conditions.

- Prints any partial pages of data that may have been received.
- Resets all programmable features to their default values. (Programmable features set from the control panel are reset to their control-panel defaults.)
- Deletes all temporary fonts.
- Returns the device to the PCL parsing mode (**ESC**%ØA). As this indicates, this instruction is recognized in HP-GL/2 mode, also.
- Initializes the device (sends the HP-GL/2 IN instruction).

ESCE is a valid HP-GL/2 terminator and incorporates all of the functionality of IN. In addition, it defaults the following.

PCL palette
logical page orientation
picture frame
picture frame anchor point
HP-GL/2 plot size
HP-GL/2 palette

NOTE: The **ESC**E is a special case when you are in normal label mode (TD0); it causes the device to reset and enter PCL mode. When your device is in transparent mode (TD1), **ESC**E does not perform its reset function. Refer to the TD instruction in chapter 8, *The Character Group*.■

FI, Primary Font Selection by ID

USE: Designates any font that has been assigned a *font ID* as the primary (standard) font. *Font ID*s are assigned in the PCL environment.

SYNTAX: FI*font ID*

Parameter	Functional Range	Default
fontID	0 – 32 767	no default

REMARKS: The *font ID* is a nonnegative parameter. If the designated font is present, the primary font attributes are set to those of the designated font. If the designated font is proportionally spaced, the pitch attribute is not changed.

If you are using scalable fonts (SB0), and a bitmap font is selected by ID using FI, SB0 is changed to SB1.

FN, Secondary Font Selection by ID

USE: Designates any resident font that has been assigned in a *font ID* as the secondary (alternate) font. *Font ID*s are assigned in the PCL environment.

SYNTAX: FN*font ID*

Parameter	Functional Range	Default
fontID	0 – 32 767	no default

REMARKS: The *font ID* is a nonnegative parameter. If the designated font is present, the secondary font attributes are set to those of the designated font. If the designated font is proportionally spaced, the pitch attribute is not changed.

If you are using scalable fonts (SB0), and a bitmap font is selected by ID using FI, SB0 is changed to SB1.

SB, Scalable or Bitmap Fonts

USE: Specifies the type of font to use in subsequent labeling. This instruction is ignored in devices that do not support both scalable and bitmap fonts.

SYNTAX: SB*(n;)*
or
SB(;)

Parameter	Format	Functional Range	Default
n	clamped integer	0 or 1	0 (scalable fonts)

REMARKS: This instruction is defaulted by the DF and IN instructions. The SB instruction takes effect immediately, changing both the standard and alternate fonts to be scalable or bitmap, as requested.

- **No Parameter** — Defaults to scalable fonts. Equivalent to SB0.
- **n** — Determines the type of font according to the following parameter values.

  0 Scalable fonts only.

  1 Bitmap fonts allowed. Also, all fonts will be subject to the same bitmap limitations; that is, limited character fill, no slant, limited direction and size, and character-clipped as opposed to bit-clipped.

Scalable fonts are computationally more complex than bitmap fonts, but they respond more accurately to some HP-GL/2 instructions. Bitmap fonts may offer a wider variety of typefaces on some devices for plots that do not use the size, direction, slant, or character fill instructions.

The choice of scalable or bitmap fonts can affect the performance of the following HP-GL/2 kernel instructions.

Affected Instructions	Limitation
CF	Bitmap characters cannot be edged.
DI, DR	Bitmap characters can be printed only with orthogonal orientations. Refer to the DI instruction in chapter 8, *The Character Group*, for an illustration of direction instructions with bitmap fonts.
SI, SR	Sizes of bitmap fonts are approximate only.
SL	Slant is ignored for bitmap fonts.

12

The Digitizing Extension

This extension is available on pen plotters only. Check your product's documentation to determine whether this extension is supported. The information in this chapter discusses the instructions used in digitizing, along with the methods and procedures for digitizing and verifying the entry of a point.

You can use your plotter to digitize as well as plot graphics. "Digitizing" is moving the pen or digitizing sight to a point on the plotting surface, entering the point, and sending the X,Y coordinates of that point to the computer. This lets you recreate a drawing from hardcopy, instead of redesigning it from scratch.

The following instructions are described in this chapter.

Instruction	Summary
DC, Digitize Clear	Clears and exits digitize mode.
DP, Digitize Point	Enters digitize mode.
OD, Output Digitized Point and Pen Status	Outputs the coordinate location of the digitized point.

The Digitizing Procedure

Although you can use a pen for digitizing, we recommend you use a digitizing sight. Refer to your product's documentation for information on using a digitizing sight. Digitize with the sight in the pen-down position for the highest accuracy.

NOTE: To avoid smearing ink on the tip of the digitizing sight or damaging the sight, do not load the digitizing sight directly into the pen carousel.■

When you are ready to digitize your plot, use the Digitize Point (DP) instruction to enter digitize mode. Refer to your pen plotter documentation to enter the digitized point from the plotter's control panel. After entering the point, use the Output Digitized Point (OD) instruction to send the X,Y coordinates of the point and the pen status (up/down) back to your computer. The Digitize Clear (DC) instruction clears and exits digitize mode.

Obtaining Plotter Output

When the plotter receives an output instruction, it responds by making the information available in the form of an output response. If you want to retrieve this information, your computer must read the output response.

Most languages use an input statement such as ENTER, INPUT#, READ, or READLN to read the output response. When you read the output response, be sure to specify the correct number and type of variable(s) the programming language will require to store the output response. For example, BASIC requires that a character string variable be in the form *A$*, and numeric variables be in the form *A*. The examples in this chapter use these conventions.

Refer to your programming language documentation for the correct input statement to use and the correct format for numeric and character string variables.

NOTE: Output instructions *must* be terminated with a semicolon.■

When an output response includes more than one piece of information, read each piece of information whether or not you need it. This ensures that the response is cleared from the output buffer. Read each piece of information into a separate variable. For example, the Output Hard-clip Limits (OH) instruction outputs four integers; your input statement might resemble the following.

```
INPUT #1, A,B,Y,Z
```

For Centronics Users

The Centronics interface supports data transmission in one direction only. Therefore, digitizing is not possible when using a Centronics interface.

For HP-IB Users

When using an HP-IB configuration, the response to an output instruction is sent after the output instruction is completely processed *and* the input buffer is completely empty. For example, the following sends the Output Error (OE) instruction and then other HP-GL/2 instructions before reading the output response.

```
"OE;PMØCI5ØØPM2FP"
```

In this example, the plotter does not respond to the Output Error (OE) instruction until the circle is completely filled.

The plotter signals the end of its output response with an output response terminator, noted in this manual by [TERM]. For HP-IB users, the output response terminator is a carriage return followed by a line feed (**CR LF**).

NOTE: Be sure the plotter is *not* set to **LISTEN ONLY.** Otherwise, the plotter will not send an output response and your program will halt. Refer to your product documentation for more information regarding HP-IB addressing.■

For RS-232-C Users

The plotter outputs information according to the handshake protocol. Your computer documentation should specify whether or not delays are required.

The plotter signals the end of its output response with an output response terminator, noted in this manual by [TERM]. When using the RS-232-C interface, the default output response terminator is a carriage return (**CR**).

Using Output Instructions

Use the following procedures for sending output instructions.

1. Send the output instruction to the plotter as you do other HP-GL/2 instructions. Note that all of the output instructions use terminators and no parameters.
2. Read the plotter's output response immediately using an input statement appropriate to your language, keeping in mind the number and type of variable(s).

Don't send multiple output instructions and then try to read the responses sequentially. This often leads to intermittent timing problems that are dependent on what the computer and plotter are currently doing.

Using the OD Instruction

Use the following procedures for sending output instructions.

1. Send the OD instruction to the plotter as you do other HP-GL/2 instructions.
2. Read the plotter's output response immediately using an input statement appropriate to your language, keeping in mind the number and type of variable(s).
3. Output digitized points one at a time. Digitize the point, then output the point to the computer. Continue in this manner until all points on your plot are digitized.

Digitizing with the Plotter

Familiarize yourself with the Digitizing Extension instructions later in this chapter before reviewing the digitizing methods here. When digitizing, you must make sure that a point has been entered before attempting to retrieve that point. The following subsections show you the methods for digitizing and retrieving points.

Manual Digitizing

The manual method is the easiest digitizing method to understand. It is not efficient, however, when you want to enter many points, or when user interaction during program execution is not possible. The following steps detail a typical program using the manual method.

1. Put the plotter in digitizing mode by sending a DP instruction to the plotter.
2. Have the program display or print a message on the computer screen prompting you to enter a point.
3. Cause the program to pause until instructed to continue. Using the BASIC INPUT statement and entering an empty string when you are ready to continue will work on some systems. Some versions of BASIC use statements such as STOP, WAIT, or PAUSE.
4. Move the digitizing sight (or pen) to the desired point using the control-panel cursor-control buttons. Complete final positioning with the sight (or pen) down. Press the appropriate button on your pen plotter's control panel (refer to your plotter documentation).
5. Cause the program to resume. The way you resume program execution depends on the statement you used to halt the program. If you used an INPUT statement in step 3, press the **RETURN** key on the computer.
6. Output the digitized information to the computer using the Output Digitized Point (OD) instruction. Have your computer read the information (the X,Y coordinates and the pen position). Then take the necessary steps to process the digitized data.

Using this method, you do not need to monitor the status byte because the program does not proceed to the OD instruction until you enter a point and cause the program to resume.

Example: Digitizing Using the Manual Method

The following program digitizes a single point and displays the coordinates and pen status.

NOTE: Your computer may read input from the plotter differently than shown here. Refer to your computer documentation.■

```
10  'Insert configuration statement here
20  PRINT #1, "DP;"
30  PRINT "Press plotter's ENTER button to digitize point."
40  PRINT:PRINT "Press computer's RETURN key to continue."
50  INPUT N$
60  PRINT #1, "OD;"
70  INPUT #1, X,Y,P
80  PRINT X,Y,P
90  END
```

Monitoring the Status Byte

The second digitizing method monitors bit position 2 of the plotter's status byte, which is set when a digitized point is available. Refer to the Output Status (OS) instruction in chapter 9, *The Technical Graphics Extension*, for more information.

There are a variety of ways to monitor bit position 2, depending on the instructions available in your computer. If there are instructions in your programming language to check bits directly, the third least significant bit (lsb) should be checked for the occurrence of a 1.

If no bit operations are available, the status byte can be operated on arithmetically to check for the availability of a digitized point. Executing successive divisions by a power of 2 and checking the answer for an odd or even integer is a common way of monitoring bits without converting the number to binary form. The following steps detail a program using this method.

1. Send a DP instruction to the plotter.

2. Have the program display or print a message on the computer screen prompting you to enter a point.

3. Send an OS instruction followed by a loop dividing the status value and checking for a final odd or even integer.

   When you press the control-panel button to enter a digitized point, the X,Y coordinates and the pen status information are stored and bit position 2 is set. The status value increments by the value of bit 2; your division yields the odd integer, allowing the program to continue.

4. Send an OD instruction. Read and display the X,Y coordinates and pen status. Then take the necessary steps to process the digitized data.

Example: Digitizing by Monitoring the Status Byte

The following sequence of BASIC instructions checks the proper bit of the status byte. In line 60, use your computer's BASIC read statement to read the status byte into a variable called STATUS. In lines 70 and 80, you must use an integer statement (e.g., INT) that truncates, not rounds. Note that your computer system may require BASIC statements other than those presented here (refer to your programming language documentation).

```
10  'Insert configuration statement here
20  PRINT #1, "DP;"
30  PRINT "Press plotter's ENTER button to digitize point."
40  PRINT #1, "OS;"
50  INPUT #1, STATUS
60  STATUS = INT(STATUS/4)
70  IF STATUS = INT (STATUS/2)*2 THEN 40
80  PRINT #1, "OD;"
90  INPUT #1, X,Y,P
100 PRINT X,Y,P
110 END
```

Example: Digitizing Many Points

When the computer is used to monitor bit position 2, it may not process the data points immediately. Allocate space for the total number of points to be digitized. Then, establish a loop to process the total number of points and call a subroutine each time to check that a point has been entered.

When prompted to enter a point in the following example, use the cursor keys to move the digitizing sight to each location, then press the appropriate control-panel button that enters digitized points. After digitizing the program's 25 points, all coordinates will display on the computer's screen.

```
10  'Insert configuration statement here
20  DIM X(25),Y(25),P(25)
30  FOR C=1 TO 25
40     PRINT #1, "DP;"
50     PRINT "Enter point ";C
60     GOSUB 140
70     PRINT #1, "OD;"
80     INPUT #1, X(C),Y(C),P(C)
90  NEXT C
100 FOR C=1 TO 25
110    PRINT X(C),Y(C),P(C)
120 NEXT C
130 END
140 'Check bit 2 for digitized point
150 PRINT #1, "OS;"
160 INPUT #1, STATUS
170 STATUS=INT(STATUS/4)
180 IF STATUS=INT(STATUS/2)*2 THEN 150
190 RETURN
```

DC, Digitize Clear

USE: Terminates digitize mode. For example, if you are using an interrupt routine in a digitizing program to branch to another plotting function, use DC to clear the digitize mode immediately after branching.

SYNTAX: DC;

REMARKS: When the plotter receives the DC instruction, it terminates digitize mode. It may also turn off any control-panel light (awaiting input of digitized point) and reactivate automatic pen lift. Refer to your pen plotter's documentation.

ERRORS:

Condition	Error	Plotter Response
1 or more parameters	2	ignores parameters

DP, Digitize Point

USE: Places the plotter in digitize mode. It may also turn on a control-panel light to indicate it is waiting for you to enter a point. Use the OD instruction to obtain the coordinates of a point on a plot.

SYNTAX: DP;

REMARKS: This instruction suppresses any automatic pen storage and lift that may be implemented in the plotter. That is, you have full control of the pen holder while the plotter is in digitize mode.

Use the control-panel cursor-control buttons to move the digitizing sight to the appropriate location, lower the sight, then press the appropriate control-panel button (usually the **ENTER** button). Digitizing a point (by pressing the control-panel **ENTER** button) sets bit position 2 of the status byte, indicating a digitized point is available for output. Use the OD instruction to *retrieve* the X,Y coordinates of the point and the pen-up/down position. You can display them on your computer screen or write them to a file.

ERRORS:

Condition	Error	Plotter Response
1 or more parameters	2	ignores parameters

OD, Output Digitized Point and Pen Status

USE: Outputs the X,Y coordinates and pen position associated with the last digitized point. Use this instruction after the DP instruction to return the coordinates of the digitized point to your computer.

SYNTAX: OD;

NOTE: You *must* use a terminator (;) with output instructions.■

Parameter	Response	Format	Range
none	X,Y,pen status	X,Y: current units pen status: integer	*device-dependent* 0 (up) or 1 (down)

REMARKS: The ranges of the X,Y coordinates are the hard-clip limits of the plotter. The pen position is either 0 (up) or 1 (down).

After sending the OD instruction, have your program read the plotter's output response. The timing of output depends on the interface you are using.

Executing an OD instruction clears bit position 2 of the status byte. Refer to the OS (Output Status) instruction in chapter 9, *The Technical Graphics Extension*.

ERRORS:

Condition	Error	Plotter Response
1 or more parameters	2	ignores parameters

Glossary

Absolute Movement — Moving to a point, the location of which is specified with respect to the coordinate system origin.

Address — Specifies the plotter's location on the HP-IB (IEEE-488) interface cable (bus).

Anisotropic Scaling — A ratio of user units along the X- and Y-axes such that their size is unequal. Anisotropic scaling can cause circles to be drawn as ellipses.

ASCII — American Standard Code for Information Interchange. An 8-bit code that uses 7 bits to represent character data such as letters, punctuation, symbols, and control characters. Bit 8 can be used for parity.

ASCII Control Character — A nonprinting ASCII character (decimal codes 0–32 and 127) that starts, modifies, or stops a device function. Control functions affect data processing, transmission, or interpretation.

Baseline — The imaginary line on which a line of text rests.

BASIC — Beginner's All-purpose Symbolic Instruction Code; a programming language which uses common English words.

Baud Rate — For an RS-232-C interface, the data transmission rate between a computer and a peripheral (bits per second).

Bitmap — A region of memory treated as a rectangular array of pixels.

Buffer — A part or parts of computer or device memory where data is held until it can be processed. Usually refers to a memory area reserved for I/O operations.

Bus — Short for HP-IB (IEEE-488) interface.

Byte — Eight bits; the size of a computer word. Used by ASCII binary code to represent alphanumeric characters.

Cap Height — Distance from the baseline to the top of a capital letter.

Carriage Return Point — The point to which the pen moves when the device receives a carriage return character in label mode.

Character Plot Cell — *CP cell.* A rectangular area defined by the character plot cell width and a line feed.

Character Plot Cell Width — Distance from the beginning of one character to the beginning of the next.

Chord — A straight line joining two points on an arc or on the circumference of a circle.

Chord Angle — The maximum angle that determines the number of chords in a circle or an arc. The greater the number of chords, the smoother the circle or arc.

Communication — Data exchange between two or more devices.

Configuration — The way in which computer equipment and software are interconnected and set up to operate as a system.

Configuration Switches — The switches on the device that configure the plotter for software.

CPI — Characters per inch. See Pitch.

Default — A value or condition that is assumed if no other value or condition is specified.

DPI — Dots per inch, the plotter's resolution of raster images on the media.

Driver — Configuration data used by software to control input and output between the computer and a peripheral device (e.g., a plotter).

Escape Sequence — A string of characters, beginning with the escape character (decimal 27 in ASCII). All HP-GL device-control instructions are escape sequences.

Handshake — RS-232-C communication between a computer and the plotter about the availability of I/O buffer space. A handshake ensures correct and complete data transfer.

Hard-Clip Limits — The boundaries of the printing area beyond which the device cannot print.

Hatch — A fill pattern of parallel lines.

Height — Measurement of a font, usually in points. A point is $1/72$ inch.

HP-GL/2	Hewlett-Packard's vector graphics language standard.
HP-IB	Short for Hewlett-Packard Interface Bus. Hewlett-Packard's version of IEEE Standard 488-1978 for interfacing programmable devices (e.g., computers, plotters, and printers).
IEEE 488-1978 Interface	A parallel interface standardized by Electronic Industries Association Standard 488-1978.
Initialize	To set plotter conditions to known default values.
Isotropic Scaling	A ratio of user units along the X- and Y-axes such that the sizes of the units are equal.
Join	The point where two line segments connect.
Label Terminator	The final character in a label string. The default label terminator is the ASCII character **ETX**.
Literal String	When using BASIC, any sequence of letters, numbers, and symbols enclosed by quotation marks. The plotter can only accept literal strings or a specific set of ASCII control characters.
Lost Mode	Refers to the state when a coordinate is specified that exceeds the internally representable number range of the device.
Miter Join	Two connecting thick lines that are extended until they meet at an angle, as in a picture frame.
Orientation	Specifies the presentation of graphics on a page, such as portrait or landscape.
P1	A scaling point the plotter uses that generally specifies the location of the plotting area's lower-left corner.
P2	A scaling point the plotter uses that generally specifies the location of the plotting area's upper-right corner.
Parallel Interface	An interface type in which a separate line is used for each data bit in a byte or word and all bits are transferred simultaneously.
Parity	An error-checking method for information transfer between a computer and a peripheral device. Parity is used to check the accuracy of binary data.
Pen Status	The pen's up/down position.
Peripheral (device)	A device separate from, but used with, a computer. For example, a printer or plotter.

Picture Body — That part of the program that would result in marks being drawn on the media.

Picture Header — That part of the program beginning after a PG or RP instruction. The MC, PS, and QL instructions affect the entire picture and are restricted to this area of the program.

Pitch — The number of characters printed in a horizontal inch. Pitch only applies to fixed-spaced fonts since the number of characters per inch varies for variable-spaced fonts.

Pixel — Picture element that is the smallest element within a bitmap to be assigned color or intensity.

Point — Unit of measurement equal to $1/72$ inch.

Polygon — A closed figure of arbitrary shape.

Polygon Buffer — That portion of the device's memory used for storing the points of a defined polygon.

Raster — A matrix of pixels, where each pixel is defined by a bit. A bit that is "on" will print a dot on the paper. A bit that is "off" will leave the area blank.

Relative Movement — Moving to a point, the location of which is specified with respect to the current pen location.

Resolution — Image sharpness. Addressable resolution is the smallest move you can specify programmatically. Mechanical resolution (pen plotters) is the smallest mechanical move the plotter can make.

RS-232-C Interface — A serial interface standardized by the Electronic Industries Association Standard RS-232-C.

Scale — To divide the plotting area into units convenient for your application.

Scaling Points — Points assigned the user-unit values specified in the HP-GL Scale (SC) instruction. These points, also known as P1 and P2, define opposite corners of a rectangular area.

Soft-Clip Limits — User-defined limits beyond which the device will not print. Also referred to as a window.

Subpolygon — A polygon defined to be part of another polygon.

Wedge — A closed sector of a circle or an ellipse.

Window — See Soft-Clip Limits.

Subject Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

The HP-GL/2 Reference Guide

A HANDBOOK FOR PROGRAM DEVELOPERS

HP-GL, the long-time industry standard graphics language, is now even better. HP-GL/2 is Hewlett-Packard's "next generation" vector graphics language for hardcopy devices.

The new language allows you to access the full functionality of HP-GL/2 plotters. These functions

- allow data compaction for reduced plot sizes and transmission times 4-5 times faster than previous HP-GL files
- let you specify a plot size up to 50 feet and send all the data at once, or a small plot size that does not waste media on rollfeed plotters
- include sophisticated imaging capabilities, allowing programmatic control over
  - pen colors
  - line widths
  - line patterns
  - fill types
  - line caps and joins
  - line intersections and overlays
  - context switching for merged vector and rastor data

The HP-GL/2 Reference Guide: A Handbook for Progam Developers gives you the ability to write one HP-GL/2 driver that will support *all* HP-GL/2 plotters. The generic functional approach to the language, combined with a complete index, directs you to the graphics information you need whether your application is business presentations, electrical engineering, mechanical engineering, architectural engineering, integrated circuit board design, or mapping.

The HP-GL/2 Reference Guide also includes tips to ensure your HP-GL/2 program yields successful results on pen plotters, electrostatic plotters, or raster devices with HP-GL/2 capability.

Corporate and Professional Publishing Group

Addison-Wesley Publishing Company

5959-9733

9 780201 563085 90000

ISBN 0-201-56308-8

56308