# HP 7600 Series Plotter
# Models 240D and 240E
# Programmer's Reference

*C1600-90003*

HEWLETT
PACKARD

Manual Part Number: C1600-90001

Printed in U.S.A., June 1988

# HP 7600 Series Plotter Models 240D and 240E

# Programmer's Reference

*C1600-90003*

HEWLETT PACKARD

Manual Part Number: C1600-90001

Printed in U.S.A., June 1988

# 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.

This document contains proprietary information, which is protected by copyright. All rights are reserved. No part of the document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company. The information contained in this document is subject to change without notice.

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

# Excuse Us, Please!

Please make the following corrections in your *7600 Series Programmer's Reference* under the Encoded Polyline (PE) instruction. Delete the shaded text and use the underlined words to replace inaccurate text.

## Page 3-10

The **SYNTAX** appears as follows:

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

Delete *X,Y* . A value can be either a single number or a coordinate pair.

## Page 3-13

Steps 2 and 3 should read as follows:

2. After labeling (or any instruction which changes the current pen location and does not update it), 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).

## Page 3-14

Line 3Ø in the programming example appears as follows:

```
3Ø PRINT #1, "Input number of fractional decimal places in
             data"
```

You must delete the #1, (sending the statement to your display); otherwise, the program will crash and cause an error.

## Additional Information about the PE Instruction

After PE is executed, the pen remains in the current up/down position; the previous plotting mode (absolute or relative) is restored.

September 1988
Mfg. No. C1600-92001

# Printing History

New editions are complete revisions of the manual. Change sheets, which may be issued between editions, contain additional information. The dates on the title page change only when a new edition is published. Minor corrections that do not affect the function of the product may be made at reprint without a change to the print date.

Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual revisions.

| | |
|---|---|
| First Edition | June 1988 |

# How to Use This Documentation

This *Programmer's Reference* provides you with information about the Hewlett-Packard Graphics Language/2 (HP-GL/2) and how it is used with your plotter. Once you are familiar with the fundamental plotting concepts and HP-GL/2, use the manual as a detailed reference to specific questions. Use the alphabetical listing at the end of the Table of Contents to locate instructions quickly.

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.

## Organization

*This manual primarily describes how to program your plotter using HP-GL/2.* When operating your plotter in emulate mode as an HP 7586 pen plotter, you use HP-GL (the precursor to HP-GL/2). Chapter 13 covers HP-GL.

**Chapters 1 and 2** introduce fundamental HP-GL/2 programming and plotting concepts.

**Chapters 3–6** cover producing graphics in your plots, from simple lines to complex polygons.

**Chapters 7 and 8** provide information on labeling plots and using fonts.

**Chapter 9** explains techniques for changing the location, size, and orientation of your plots.

**Chapters 10–12** discuss device control. These chapters provide information on obtaining information from the plotter, using device-control instructions, and interfacing and handshaking.

**Appendix A** contains the information you'll need if you use your plotter in emulate mode.

**Appendix B** presents print-outs of the plotter's character sets.

**Appendix C** provides more complex example programs.

**Appendix D** is an alphabetical, quick-reference of all the instructions' syntax.

## Manual Terms and Conventions

The symbols and type styles used to represent the syntax requirements of HP-GL/2 and device-control instructions are discussed in Chapters 2 and 11, respectively.

Single, nonprinting ASCII characters (control codes) appear in bold type, for example, **ESC**.T.

Numbers are typed 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 both to the left and right of the decimal point (for example, 54 321. 123 45).

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.

# Table of Contents

1

# Before You Begin

Before you can write graphics programs for your plotter, you need to be familiar with your computer and a programming language such as BASIC, FORTRAN, or Pascal. You can use any computer that inputs and outputs ASCII* information to and from a peripheral device. **This manual assumes that you already have some experience with your programming language.**

This chapter discusses the following fundamental programming concepts necessary to begin plotting.

- Plotting with a raster plotter versus a vector pen plotter.
- Using your plotter's programming language, HP-GL/2, with your computer's programming language.
- Using the program examples in this manual.
- Maximizing your plotter's throughput.
- Hints for the novice programmer.

## Working with the Electrostatic Plotter

Although the plotter does not have physical pens, for the purpose of compatibility it has a 'logical' pen. 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 a physical pen. For the remainder of this manual, simply think of the plotter as having an actual, physical pen.

You will also see many references to 'vectors' in this manual. Although the plotter is a raster device, it uses a vector–based graphics language for simplicity and compatibility.

*American Standard Code for Information Interchange

## The Plotter's Programming Language

The plotter uses HP-GL/2, the next generation of Hewlett-Packard Graphics Language, the language used by Hewlett-Packard pen plotters. HP-GL/2 increases your plotter's throughput, performance, and future compatibility. Your plotter uses HP-GL/2 in HP-GL/2 mode; however, in 7586B emulate mode your plotter acts as an HP 7586B pen plotter and uses HP-GL. **This manual primarily describes how to use the plotter in HP-GL/2 normal mode.** The remainder of this chapter through Chapter 12 assumes that you are in normal mode. Appendix A, *Using Emulate Mode*, describes the differences between HP-GL and HP-GL/2. **Consult Appendix A when using emulate mode.**

HP-GL/2 is a collection of graphics instructions used to write your own programs for plots. The plotter also uses instructions to control certain plotter functions, called device-control instructions.

### HP-GL/2 Instructions

Each instruction is a two-letter mnemonic code designed to remind you of its function. For example, IN is the INitialize instruction, SP is the Select Pen instruction, and CI is the CIrcle instruction. Numbers, called *parameters*, are used with certain HP-GL/2 instructions to tell the plotter to complete (execute) the instruction in a particular way. A typical HP-GL/2 instruction appears as follows.

```
    SP1
```

The mnemonic tells the plotter to select a pen and the parameter specifies pen number one.

There are two types of HP-GL/2 instructions: graphics instructions and output instructions. Graphics instructions are presented in Chapters 2–9; output instructions in Chapters 10 and 11. Graphics instructions enable you to do the following.

- Define your plotting area and select certain settings, such as default parameters, plot size, and plot rotation.

- Define and draw your plot, including shapes such as lines, arcs, circles, polygons, rectangles, and characters.

Output instructions enable you to obtain information from the plotter, such as the current size of plotting area and error messages.

## Device-Control Instructions

Device-control instructions are used primarily with an RS-232-C* interface, and to some extent with HP-IB command sequences. Each device-control instruction is an escape sequence, such as the following.

```
ESC.B
```

Device-control instructions are discussed in Chapters 11 and 12. These instructions control advanced operations, such as changing and monitoring buffer sizes, handshaking, and verifying plotter operating conditions.

This manual gives full details on the capabilities of the plotter's HP-GL/2 and device-control instruction set. To communicate with the plotter, you must 'send' the instructions with their parameters in an appropriate output statement from your computer's language. The following section provides information and examples of how to send HP-GL/2 instructions with various programming languages.

# Using HP-GL/2 with Programming Languages

Before you can send instructions to the plotter, your computer and plotter must be connected and communicating. Refer to the *User's Guide* for interconnection information.

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 used primarily for the computer's monitor. Common languages used for plotting are BASIC, FORTRAN, Pascal and C.

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.

*All references to the 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.

If you are programming in a language other than BASIC, FORTRAN, or Pascal, you will have to make an extra effort 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 the plotter).

## Configuration Statements

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

```
    10  OPEN "COM1:192ØØ,N,8,1,RS,CS65535,DS,CD" AS #1
```

The BASIC configuration statement shown above is for the HP Vectra PC (also IBM PC, XT, and AT) RS-232-C interface. It OPENs COM1 (port 1) for output at **19200** baud, **N**o parity, **8** data bits, **1** stop bit, suppresses RTS (**RS**), waits 65535 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 listed below. Refer to the interconnection information in the User's Guide for additional examples.

*Microsoft is a trademark of Microsoft Corporation.

*Sample Configuration Statements*

| Computer | Language | Example |
|---|---|---|
| HP 3000<br>DEC VAX | FORTRAN<br>(RS-232-C) | `      WRITE (9,10)`<br>`10    FORMAT ("`*instruction*`")`<br>(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<br>(HP-IB) | `OUTPUT 705 ; ("`*instruction*`")`<br>(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<br>(HP-IB) | `WRITELN ('`*instruction*`')`<br>(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 HP-GL/2 instruction (*SP1*) is used as the instruction being sent.

| Computer | Language | Example |
|---|---|---|
| HP Touchscreen<br>HP Vectra PC<br>IBM PC/PC-XT/AT | GW BASIC | `PRINT #1, "SP1"` |
| HP 9000, Series 300<br>(HP-IB interface) | PASCAL | `WRITELN('SP1')` |
|  | HP BASIC | `OUTPUT 705; "SP1"` |
| HP 3000<br>DEC VAX | FORTRAN | `      WRITE(9,10)`<br>`10    FORMAT(X,3HSP1)` |

## Input Statements

The computer language output statements discussed in the preceding section are used to send information to the plotter. To get information *from* the plotter, you must first send an HP-GL/2 output statement to the plotter directing it to send information back to the computer; then you must use a computer language input statement to read the plotter's output. The following table lists a few computers and languages with their associated input statements. The HP-GL/2 output instruction (*OP;*) is used as the instruction being sent.

| Computer | Language | Example |
|---|---|---|
| HP Touchscreen<br>HP Vectra PC<br>IBM PC/PC-XT/AT | GW BASIC | `INPUT #1, A,B,C,D` |
| HP 9000, Series 300<br>(HP-IB interface) | PASCAL | `READLN(A,B,C,D);` |
|  | HP BASIC | `ENTER 705; A,B,C,D` |
| HP 3000<br>DEC VAX | FORTRAN | `      READ(9,40)`<br>`40    FORMAT(4I7)` |

## Using the Programming Examples

Examples are presented as complete programs written in a version of GW BASIC for the MS-DOS* 2.11 (or higher) operating system. Programs show the use of each instruction as well as its interaction with other instructions and the BASIC language.

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

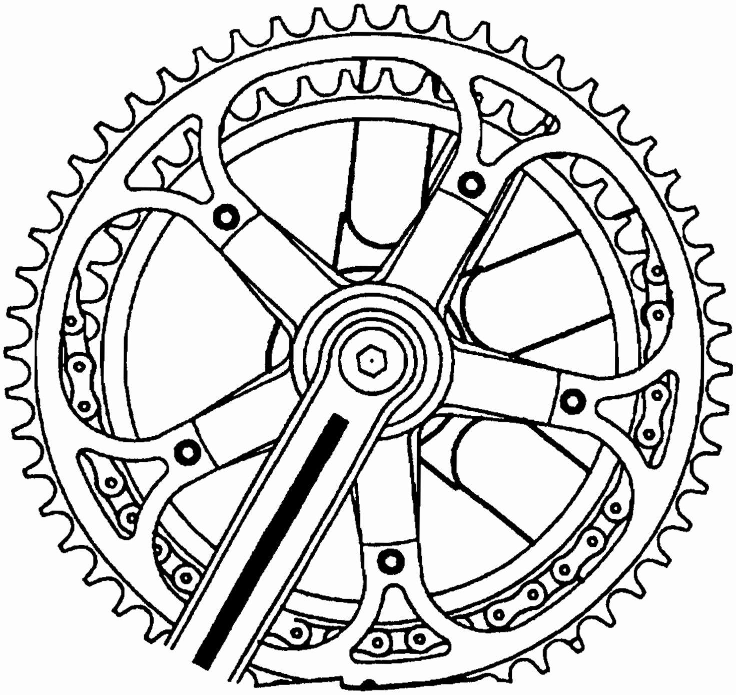
*If you are not using GW BASIC*, you may need to change some of the program statements as well as insert the proper configuration statement at the beginning of the program.

Below is a simple example of the way complete programs appear in this manual.

```
10 OPEN "COM1:96ØØ,N,8,1,RS,CS65535,DS,CD" AS #1
20 PRINT #1, "INPS5ØØØ,7ØØØPA2ØØØ,Ø"
30 PRINT #1, "SP1PDAAØ,Ø,45,25"
40 PRINT #1, "PU1Ø5Ø,1Ø6ØPDAAØ,Ø,-45,1Ø"
50 PRINT #1, "PU1ØØØ,ØPDAAØ,Ø,45PG;"
60 END
```

*GW BASIC and MS-DOS are trademarks of Microsoft, Inc.

*Illustrations of program art in this manual are not actual plotter output.* The illustrations were produced using a 300 dots-per-inch (dpi) laser printer. Your plotter's resolution, 406 dpi, gives superior output. The following photograph of actual plotter output gives you a sample of the quality you can expect from your plotter.



## Programming Statements You May Need to Change

You might need to change BASIC statements if you are not using the GW BASIC version used here, or if you are programming in another language. Your computer language documentation should tell you how to do this. You will likely need to change the following statements.

**PRINT #1** This statement sends the HP-GL/2 instructions, via an output file, to the plotter. Your computer might use a statement such as WRITE, WRITELN, OUTPUT, LPRINT, or simply PRINT. Here '1' corresponds to the file number in the 'OPEN' statement.

**INPUT #1** This statement causes information from the plotter to be read by the computer. Your computer might use a statement such as READ, READLN, or ENTER. Here '1' corresponds to the file number in the 'OPEN' statement.

**FOR...NEXT**
**X=3.14** FOR...NEXT are loop statements. X=3.14 is a variable assignment. Change these statements to whatever your language uses.

## Increasing Your Plotter's Throughput

The *throughput* of your plotter refers to the time it takes to complete a plot. You probably purchased this electrostatic plotter because of its throughput. To take full advantage of your plotter's significant speed, use a parallel interface, such as Centronics or HP-IB. For the best throughput with an RS-232-C interface, use 19 200 baud.

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

1. **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.

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

3. **Use the plotter's internal character set**, particularly for quick-check plots. Internal sets require only one byte per character. Refer to Chapters 7 and 8.

4. **Eliminate separators (semicolons) between instructions in your programs.** Eliminating separators in your programs can increase your plotter's throughput by approximately 10%. This is explained under *Understanding HP-GL/2 Syntax* in Chapter 2. (Note that PE, PG, and output instructions must still be followed by a semicolon.)

## Hints for the Novice Programmer

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

- Know your equipment, both the computer and 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 computer programming language. 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. Notice the syntax differences between the two versions of BASIC represented in the tables under *Output Statements* and *Input Statements* earlier in this chapter.

- Enter HP-GL/2 and device-control 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 a lowercase letter 'l' for the number one '1'. Substituting a semicolon for a comma or omitting a quotation mark can make the difference between success and failure.

- Begin every program with the proper configuration statement and an IN or DF instruction.

- End every program with a PG instruction.

To learn HP-GL/2, practice running the example programs provided with each instruction. Then try varying the parameter and combining the instructions in simple programs of your own.

# 2

# Setting Up Your HP-GL/2 Program

This chapter explains the information necessary for plotting and using HP-GL/2 instructions, as follows.

- Using HP-GL/2 instructions in the proper programming sequence.
- Establishing default conditions.
- Working with your plotter's coordinate system, units of measurement, and graphics limits.
- Scaling your plots.
- Using correct HP-GL/2 syntax.
- Understanding error messages.

The following instructions are described in this chapter.

| | |
|---|---|
| DF, | Default |
| FR, | Frame Advance |
| IN, | Initialize |
| IP, | Input P1 and P2 |
| IR, | Input Relative P1 and P2 |
| PG, | Advance Full Page |
| PS, | Plot Size |
| PW, | Pen Width |
| RP, | Replot |
| SC, | Scale |
| SP, | Select Pen |
| WU, | Pen Width Unit Selection |

## Outline of a Typical HP-GL/2 Program

When you write an HP-GL/2 program, there are certain instructions that must be placed correctly in your program if they are used. These instructions are listed in their required sequence and described briefly below.

- **IN** — The initialize (IN) instruction sets most of the plotter's instructions to their default conditions. If you use IN, it should be the first instruction in your program.

- **Other set-up instructions** — Any set-up instructions you plan to use usually come after IN and before any graphics instructions. These set-up instructions include the page size (PS), input P1 and P2 (IP), input P1 and P2 relative (IR), and scale (SC).

- **SP** — You must send a select pen (SP) instruction for your plot to be drawn. It must come after IN and before any graphics instructions. If you want to change the pen width unit selection (WU) or the pen width (PW), send these instructions *before* SP.

- **All graphics instructions** — All graphics instructions must come after the set-up instructions or your plot may not be drawn the way you want it.

- **PG** — You must send the advance full page (PG) instruction at the end of your program to tell the plotter to rasterize and draw your plot. You may send the replot (RP) instruction in place of or in addition to PG.

The following sections explain how to use these set-up instructions to establish default conditions, size your plot, scale your plot, set P1 and P2, select a pen and pen width, and advance the page.

## Establishing Default Conditions

Establish default conditions at the beginning of each program to prevent unexpected results. The parameters for many instructions remain in effect until you change them or the plotter is reset to default conditions. Always establish default conditions for each program unless you are certain you want to use these 'leftover' parameters.

There are four ways to establish default conditions: turning the plotter off and on, using the control-panel RESET button, or using the initialize (IN) or default (DF) instructions. Using the IN instruction and turning the plotter off and on both set the plotter to its factory defaults (IN doesn't default device-control instructions). This process is called *initialization*. The DF instruction is less powerful. Refer to the IN and DF instructions at the end of this chapter for a complete list of the default conditions each instruction establishes.

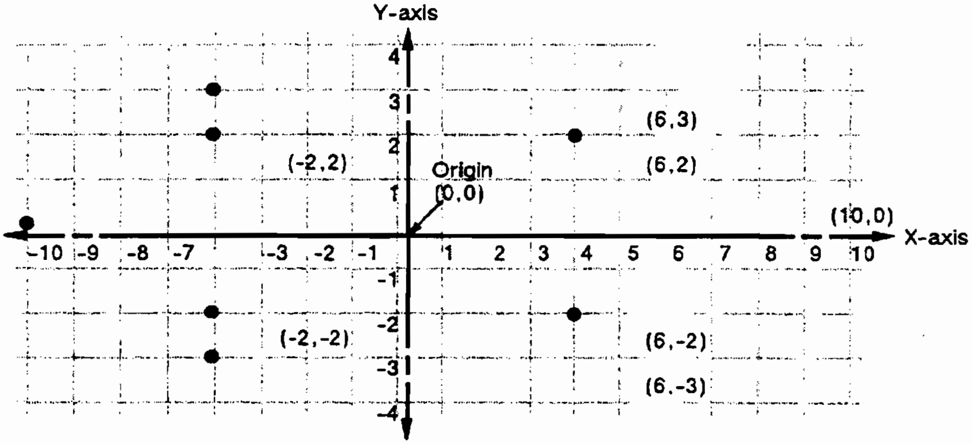
## Sizing Your Plot

Use the plot size (PS) instruction to set the size of your plot. If you are drawing small plots but always use the default plot size (roughly, 3 × 4 ft. on the E-size plotter and 2 × 3 ft. on the D-size plotter), you will waste a great deal of paper. PS also simplifies making very long plots, up to 50 feet. Be sure to place PS *before* any graphics instructions in your program, but after IN.

The following sections explain the plotter's coordinate system.

### The Plotter's Coordinate System

The plotter uses the Cartesian coordinate system. The Cartesian coordinate system is a grid formed by two perpendicular axes, usually called the X- and Y-axes. Refer to the following illustration. The intersection of the axes is called the *origin* of the system and has a location of (0,0).

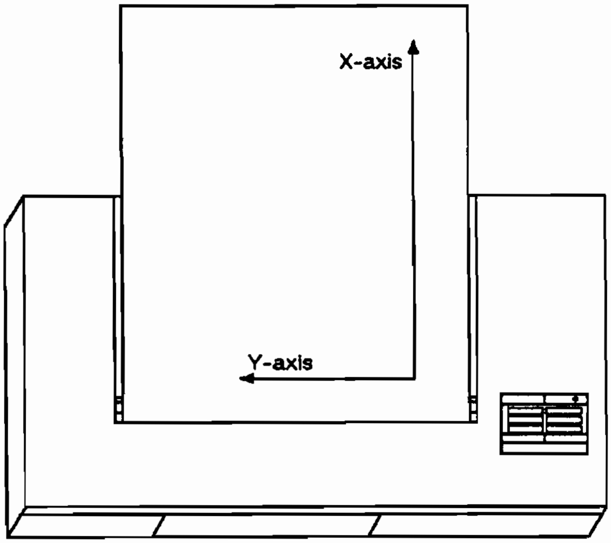


To locate any point on this grid (your plotter's plotting area), move from the origin a number of units along the X-axis, then move a number of units parallel to the Y-axis. The number of units you move is called a coordinate. Each point is designated by the combination of its X-coordinate and Y-coordinate, known as an X,Y coordinate pair. Positive X values are plotted to the right of the origin, and positive Y values are plotted above the origin.

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.

To specify a point to your plotter, you must always give a complete X,Y coordinate pair, the X-coordinate first and Y-coordinate second. The manual shows coordinate pairs in parentheses (X,Y) for clarity — do not use parentheses when in your program, just the numbers.

On your plotter, the X-axis is always parallel to the longest edge of your plot; the Y-axis is always parallel to the shortest edge of your plot (unless rotated by an RO instruction). The following illustration shows the default axis orientation, with the X-axis in the rollfeed direction.



## Units of Measurement

You can measure along the axes and express coordinates using two types of units: plotter units or user units.

### Plotter Units

A plotter unit (plu) is the smallest addressable move the plotter can make. Plotter units are whole numbers (integers). The following lists the equivalent measurements for plotter units, millimetres, and inches. Note that plotter units have a higher resolution than the plotter's 406 dpi. The plotter converts plotter units into equivalent pixel coordinates.

| | | |
|---|---|---|
| 1 plu | = | 0.025 mm or 0.00098 in. |
| 40 plu | = | 1 mm |
| 1016 plu | = | 1 in. |

### User Units

User units allow you to customize the coordinate system to your particular needs. For example, you could plot the moon cycle for the year by dividing the X-axis into 31 units for days of the month and the Y-axis into 12 units for months of the year. To mark a point on June 13, you would simply give the coordinate (13,6) rather than calculating the exact location in plotter units.

User units can represent months, years, dollars, francs, distances, temperatures, population, or whatever meets your requirements. The plotter internally converts user units to plotter units before locating the coordinates on paper. You establish user units using a process known as *scaling*. Scaling is explained later in this chapter.

User units can have fractional parts.

## Graphics Limits

The plotter has two types of graphics limits: hard-clip limits and soft-clip limits. These limits are boundaries of the plotting area. Any drawing that would extend beyond the limits is clipped (and not drawn) as you would clip a newspaper article.

### Hard-Clip Limits

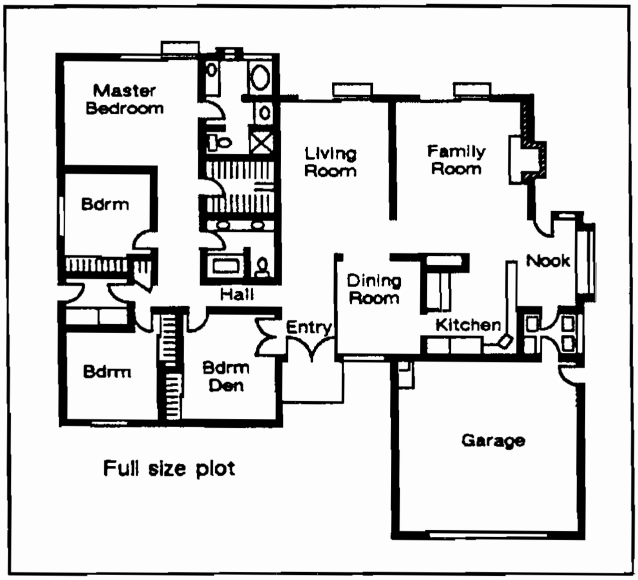
The hard-clip limits are your plotter's *physical* boundaries, the area available for plotting. The following table gives the default measurements of these limits, in plotter units.

| Plotter | Hard-Clip Limits |
|---|---|
| D/A1-size (Model 240D) | Default X-axis: 0 to 35 376<br>Default Y-axis: 0 to 24 000 |
| E/A0-size (Model 240E) | Default X-axis: 0 to 47 568<br>Default Y-axis: 0 to 35 840 |

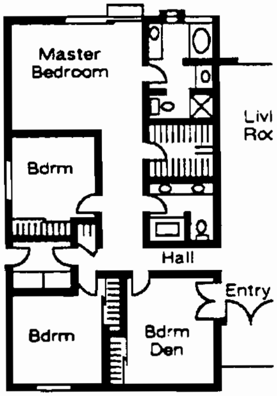
### Soft-Clip Limits

Soft-clip limits are *programmatically* set boundaries that temporarily restrict plotting to a specified area of the page. You can use soft-clip limits to ensure that nothing will be drawn beyond that portion of the page. Soft-clip limits are often referred to as *windows* because they allow you to draw attention to a particular set of data.

For example, look at the following floor plan.



After plotting the full floor plan, suppose that you decide to make a new plot showing only the sleeping areas, using the space on the right for text. To create the new plot, use the program for the full floor plan, adding soft-clip limits to restrict plotting to the left half of the plotting area. Finally, at the end of the program, add program lines to extend the soft-clip limits and plot the text.



You could drawn labels or another plot in this area

Soft-clip limits cause only this much of the previous plot to be drawn

Using soft-clip limits provides a great deal of flexibility with respect to where you plot data, how much data you plot, and reducing/enlarging plots. The input window (IW) instruction creates soft-clip limits. Refer to Chapter 9 for specifics on using soft-clip limits.

## Using Scaling

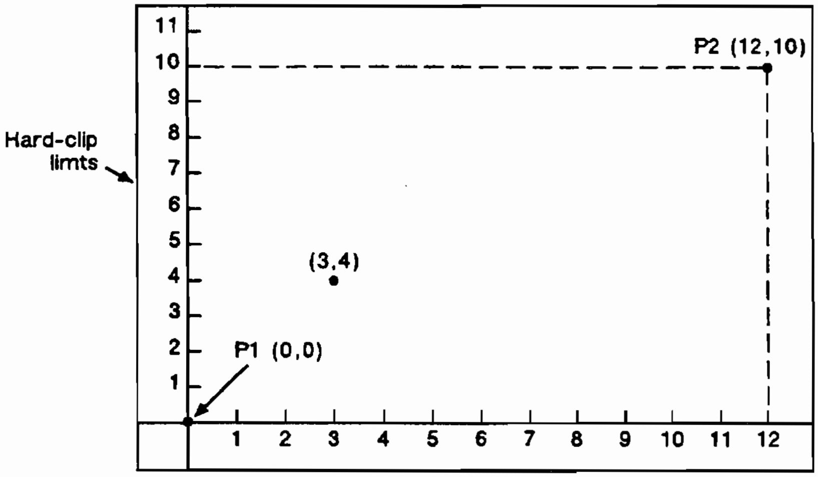
When you scale a plot, you define your own units of measurement which the plotter internally converts to plotter units. Scaling lets you plot in units that make sense to you and are easy for you to work with.

For example, you can scale your plot to divide the plotting area into 100 squares. As you plan you plot, you can think in terms of those 100 squares rather than in plotter units. Here's another use of scaling: since 400 plotter units equals 1 centimetre, you can establish this scaling to plot in user units of centimetres.

Scaling begins with the scaling points, P1 and P2. Think of P1 and P2 as two points marking opposite corners of a rectangle. You can make this rectangle any size and place it anywhere in the hard-clip limits depending on what plotter-unit coordinates you specify for P1 and P2. You set P1 and P2 using the input P1 and P2 (IP) or input relative P1 and P2 (IR) instruction.

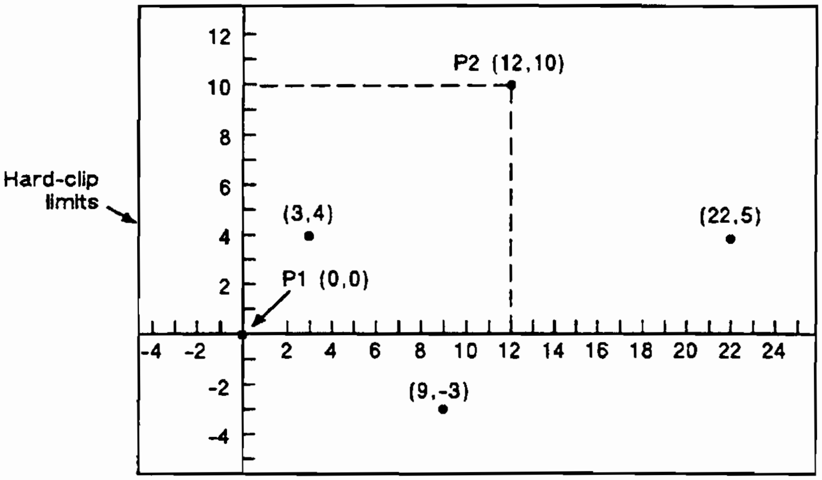
Next you use this rectangle to set up scaling for your plot. With the scale (SC) instruction you specify how many sections the rectangle should be divided into horizontally (the X-axis) and how many sections the rectangle should be divided into vertically (the Y-axis). Now you have created your user units.

The following example will help you to understand the scaling 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 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).



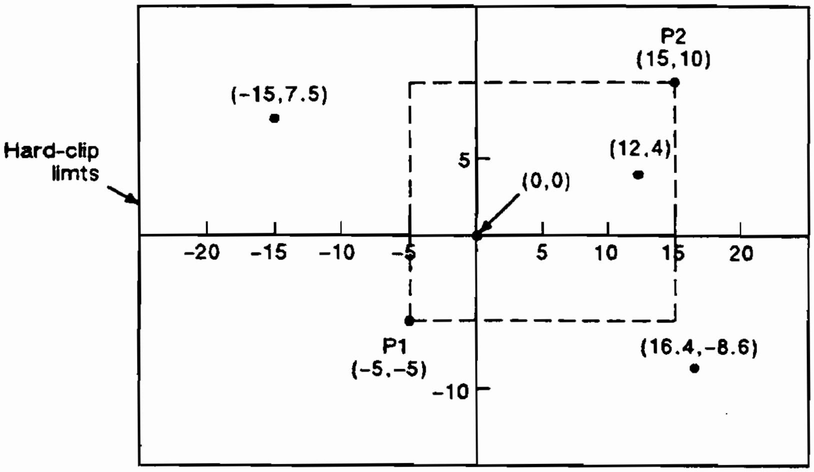
*User-Unit Scaling with Default P1 and P2*

If you move the locations of P1 and P2, the size of 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-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*

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*

The scaling instruction also allows you to specify whether your units will be of equal size on the X- and Y-axes (*isotropic* scaling) or unequal (*anisotropic* scaling). Refer to the scale (SC) instruction at the end of this chapter for more information on scaling plots.

## Selecting a Pen and Width

Although your plotter does not have physical pens, for the purpose of compatibility it has a 'logical' pen which you must select to draw your plot. Use the select pen (SP) instruction at the beginning of your plot.

You can change the width of the logical pen using the pen width (PW) instruction. Subsequent lines are drawn using the new width. Use PW to distinquish lines and enhance your plots. You may change widths as often as you like, and do not need to send an SP again.

Pen (line) widths can be measured either in millimetres or as a percentage of the diagonal distance from P1 to P2. Use the pen width unit selection (WU) instruction to specify how the width (a parameter of PW) is measured. Since using WU defaults all pen widths, with or without parameters, you should send WU *before* PW.

## Ending Your Program and Advancing the Page

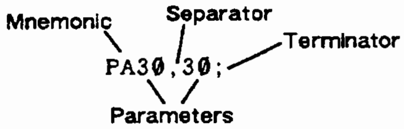
You must indicate the end of your plot before the plotter will rasterize and draw it. You can press the control-panel **PLOT** button or use the advance page instruction (PG) at the end of your program. (Sending PG is recommended since it saves you the trouble of walking over to the plotter.)

PG advances the plot size set by the PS instruction, or the default page size if there is no PS instruction.

You can use the replot (RP) instruction to make multiple copies of a plot. Since the plot is already stored in the plotter, using RP frees the computer while the copies are drawn. (For one copy, you can simply send PG a second time.)

## Understanding HP-GL/2 Syntax

HP-GL/2 instructions have four components: a mnemonic, parameter(s), separator(s), and a terminator. Refer to the following illustration of a typical HP-GL/2 instruction and description of its components.



```
PA30,30;
```

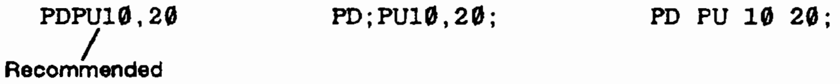
- **Mnemonic** — The two-letter mnemonic is designed to remind you of the instruction's function. The mnemonic can be upper- or lower-case.

- **Parameter(s)** — Some instructions have no parameters; for other instructions, parameters are optional.

- **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. In both HP-IB, Centronics, and RS-232-C interface configurations, HP-GL/2 instructions are terminated by the first letter of the next mnemonic or a semicolon. (Dropping the semicolon is recommended to enhance throughput.)

The following illustration shows the flexibility of the syntax. Each variation of the two-instruction sequence is permissible; however, the first method is recommended for your programming. 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.



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

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.

**NOTE:** Using the recommended syntax can increase your plotter's throughput by approximately 10%. ■

## 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 — For readability, the mnemonic is shown uppercase and separated from the parameters and/or terminator.

*parameters* — Parameters are shown in italic.

( ) — 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 optional and is shown in parentheses in most instruction syntax. Exceptions are the advance page (PG), encoded polyline (PE), and output instructions which *must* be terminated by a semicolon.

[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: RS-232-C — a carriage return (CR) and HP-IB — carriage return and line feed (CR LF). You can change the default terminator using the device-control instruction ESC.M, Set Output Mode. Refer to Chapter 10 for more information on output instructions.

**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* (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 omit the second parameter, you must also omit the third parameter, as shown. The plotter uses the most recently specified pattern length and mode. If you have not specified a length or mode since sending a default values (DF) or initialize (IN) instruction, the plotter uses the parameters' defaults.

```
    LT6
```

Do *not* send the following (the plotter would interpret 1 as the second parameter).

```
    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.

1. **Integer** — An integer from −67 108 863 ($-2^{26}+1$) to 67 108 863 ($2^{26}-1$)*. The plotter automatically rounds fractional parameters to the nearest integer. Using a number outside this range causes an error.

   Plotter units are always integer.

2. **Clamped Integer** — An integer from −32 767 ($-2^{15}+1$) 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. For example, the plotter treats all numbers between 32 767 and 67 108 863 as 32 767.

   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.

*Numbers within this range do not cause errors; however, the range exceeds the plotter's physical plotting area. Numbers greater than the plotter's hard-clip limits are not plotted.

3. **Real** — A number where the integer portion is from −67 108 863 ($-2^{26}+1$) to 67 108 863 ($2^{26}-1$)*, 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.

4. **Clamped Real** — A number where the integer portion is from −32 767.9999 ($-2^{15}+1$) 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. For example, the plotter treats all numbers between 32 767 and 67 108 863 as 32 767.

   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.

5. **Label** — Any sequence of characters. Refer to the label (LB) instruction in Chapter 6 for a complete description.

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 (i.e., 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. ■

*Numbers within this range do not cause errors; however, the range exceeds the plotter's physical plotting area. Numbers greater than the plotter's hard-clip limits are not plotted.

# Program Errors and Messages

If your plot is not drawn as you expected, you probably have an error in your program. There are two types of errors: those related to HP-GL/2 instructions and those related to device-control instructions. When you have an error, you can send either the output error (OE, Chapter 10) or the output extended error (ESC.E, Chapter 11) instruction to 'read' the error. The OE instruction sends back a number(s) representing an HP-GL error(s); ESC.E sends back a number(s) representing a device-control error(s).

## HP-GL/2 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 occurred.

*HP-GL/2 Errors*

| Error # | Meaning | Description |
|---|---|---|
| 1 | Unrecognized command | A mnemonic is incorrect or missing; an alphabetic character was specified in a parameter when a numeric character was expected. |
| 2 | Wrong number parameters | Too few or too many parameters; an incomplete X,Y coordinate pair. |
| 3 | Out-of-range or invalid parameter | A parameter is out of range. |
| 5 | Unknown character set | An unknown character set value used with AD or SD. |
| 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 | The dynamically allocated polygon and downloaded character buffer overflowed or the plotter's disc overflowed. |

The plotter ignores the instruction causing the error in the following cases.

- The instruction is not recognized.
- The instruction is missing a required parameter(s).
- The instruction has an out-of-range parameter(s).

The plotter *partially* ignores an instruction in the following cases.

- **The instruction is sent with more parameters than necessary** — The plotter ignores the extra parameters and executes the instruction as normal.
- **An PA, PD, PR, or PU instruction has a parameter exceeding the numeric range** — The plotter ignores out-of-range parameter and all subsequent parameters.
- **AD, LA, or SD instruction has a parameter exceeding the parameter range** — The plotter ignores the pair containing the out-of-range parameter and executes all other valid pairs.

The plotter ignores all plotting instructions when you move the pen to a position exceeding the plotter's numeric range, either with an single instruction or as a result of several instructions. (This is error 6, position overflow.)

When the plotter has an error 6, it ignores all plotting instructions until it receives an absolute instruction moving the pen back to a position within the plotter's numeric range.

## Device-Control Errors

The following table lists each error number, its meaning, and the probable cause of the error. Error number 0 indicates that no error has occurred.

*Device-Control Errors*

| Error # | Meaning | Description |
|---|---|---|
| 10 | Bad output request | (*RS-232-C only*) New output was requested before previous ouput was finished being transmitted. The new output will be ignored (thus causing the error). |
| 11 | Bad byte after ESC. | Invalid character received after first two characters (ESC.) in a device-control instruction. |
| 12 | Bad byte in I/O control | Invalid character received while parsing a device-control instruction. The parameter containing the invalid character and all following parameters are defaulted. |
| 13 | Bad parameter | One or more parameters are out of range. |
| 14 | Too many parameters | Too many parameters received. Additional parameters beyond the proper number are ignored; parsing of the instruction ends when a colon (normal termination) or the next ESC character (abnormal termiation) is received. |
| 15 | Bad transmission | (*RS-232-C only*) A parity error has been detected. |
| 16 | Buffer overflow | (*RS-232-C only*) The physical input buffer has overflowed. As a result, one or more characters have been lost; an HP-GL/2 error will probably occur. |
| 18 | Indeterminate error | Input error of indeterminate cause. |

## DF, Default Values

**USE:** Sets certain graphics functions to their factory default settings. 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 | Lower-left corner of hard-clip limits. |
| Alternate Font Definition | AD | Roman8, fixed spacing, fixed stick typeface. |
| Chord Tolerance Mode | CT | Degrees mode. |
| Absolute Direction | DI1,0 | Character direction parallel to X-axis. |
| Define Label Terminator | DT | ETX and printing mode. |
| Define Variable Text Path | DV | Text printed left to right with normal line feed. |
| Extra Space | ES | No extra space. |
| Fill Type | FT | Solid fill. |
| Input Window | IW | Hard-clip limits. |
| Line Attributes | LA | Butt ends, mitered joins, and miter limit=5. |
| Label Origin | LO1 | Standard labeling starting at current location. |

*(Table continues)*

*(Continued)*

| Function | Equivalent Instruction | Default Condition |
|---|---|---|
| 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 black. |
| Scale | SC | User-unit scaling off. |
| Standard Font Definition | SD | Roman8, fixed spacing, fixed stick typeface. |
| Character Size Absolute | SI | Width=0.285 cm<br>Height=0.375 cm |
| Character Slant | SL | No slant. |
| Symbol Mode | SM | Off. |
| Select Font | SS | Standard font. |
| Transparent Data | TD | Normal printing mode. |
| User-Defined Line Type | UL | Defaults all 8 line types. |

In addition, the carriage-return point for labeling is updated to the current pen position.

The following plotter conditions are *not* affected by a DF instruction.

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

**RELATED INSTRUCTIONS:** IN, Initialize

## FR, Frame Advance

**USE:** Advances the media to align adjacently drawn 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:** The length parameter of the plot size (PS) instruction determines the frame size. After each frame advance, the plotter-unit origin moves to the lower-left corner of the current frame. The P1 and P2 values adjust to reflect this change. The physical location of P1 and P2 are retained, but the logical location relative to the current origin changes.

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.

FR clears the polygon buffer.

**RELATED INSTRUCTIONS:** PS, Page Size

## IN, Initialize

**USE:** Resets most 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 I/0 or HP-GL/2 error condition without affecting handshake protocol.*

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| n | integer | 0 or 1 | no parameter |

**REMARKS:** The plotter implements the parameters as follows.

- **No Parameter** — Defaults all programmable HP-GL/2 features to the factory set conditions.

- **n** — Defaults all programmable HP-GL/2 features to the factory set conditions. Equivalent to (*IN*).

The IN instruction sets the plotter to the same conditions as the DF instruction, plus the following *additional* conditions.

- Raises the pen (*PU*).

- Returns pen location to lower-left corner of the hard-clip limits (*PA0,0*).

- Cancels plot rotation (*RO*).

- Sets P1 and P2 to the lower-left and upper-right corners, respectively, of the hard-clip limits (*IP*).

- Sets plot length to approximately 1½ times the paper width (*PS*).

- Sets pen width mode to metric, units are in millimetres (*WU*).

- Sets the pen width to 0.35 mm (*PW*).

- Clears HP-GL/2 errors.

In this manual, all program examples begin with (*IN*) to clear unwanted conditions from the previous program.

**RELATED INSTRUCTIONS:** DF, Default
OS, Output Status

## IP, Input P1 and P2

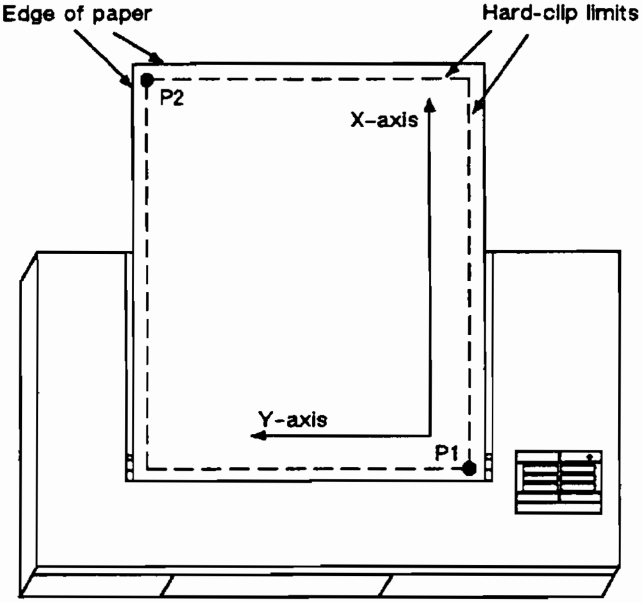
**USE:** Establishes new or default locations for the scaling points P1 and P2. P1 and P2 are used by the scale (SC) instruction to establish user-unit scaling. You can also use IP in advanced techniques such as plotting mirror images, enlarging/reducing plots, and enlarging/reducing relative character size or direction (refer to Chapters 7 and 9).

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates | integer | −67 108 863<br>to 67 108 863 | hard-clip limits* |

*The hard-clip limits, in plotter units, of the D-size plotter are (0,0) to (35 376,24 000); the hard-clip limts of the E-size plotter are (0,0) to (47 568,35 840).

**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** — Sets P1 and P2 to their default locations, 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. ■

- **X,Y Coordinates** — Specify the location of P1 and P2 in plotter units. If either coordinate of P1 equals the corresponding coordinate of P2, the coordinate(s) of P1 is incremented by 1 plotter unit.

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

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 located outside the hard-clip limits. Used carefully, the tracking function can be useful for preparing more than one equal-sized plot on one page. For an example, refer to *Drawing Equal-Sized Pictures on One Page*, in Chapter 8.

The locations of P1 and P2 interact with the commands DR, IW, LB, PW, RO, SC, SR, and WU. An IP instruction remains in effect until another IP instruction is executed, an IR instruction is executed, or the plotter is initialized.

**RELATED INSTRUCTIONS:** SC, Scale
RO, Rotate Coordinate System

**SPECIAL 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. IR can also be used in advanced techniques such as plotting mirror images, enlarging/reducing plots, and enlarging/reducing relative character size or direction (refer to Chapters 7 and 9).

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

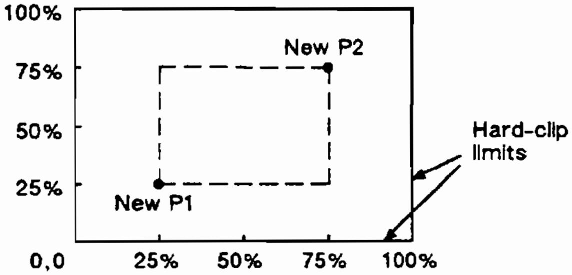
| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates | clamped real | –32 767.9999<br>to 32 767.9999 | 0,0,100,100% |

**REMARKS:** When P1 and P2 are set using IR, the scaled area is page-size independent.

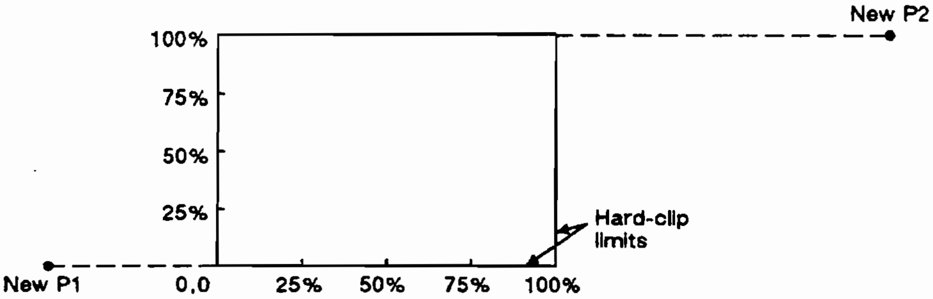
- **No Parameters** — Defaults P1 and P2 to the lower-left and upper-right corners of the hard-clip limits, respectively.

- **X,Y Coordinates** — Specify the location of P1 and P2 as percentages of the hard-clip limits. If either coordinate of P1 equals to the corresponding coordinate of P2, the coordinate(s) of P2 is incremented by 1 plotter unit.

  Sending the instruction *(IR25,25,75,75)* set P1 and P2 create an area half as high and half as wide as the hard-clip limits in the center of the page, as shown in the following illustration.



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



If you specify 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.

Specifying P2 is not required. P2 tracks P1; the P2 coordinates change so that the distances of X and Y between P1 and P2 remain the same. This tracking process can cause P2 to end up located outside the hard-clip limits. Used carefully, the tracking function can be useful for preparing more than one equal-sized plot on one page. For an example, refer to *Drawing Equal-Sized Pictures on One Page* in Chapter 9.

**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 plotter unit position is maintained. ■

The locations of P1 and P2 interact with the commands DR, IW, LB, PW, RO, SC, SR, and WU. An IR instruction remains in effect until another IR instruction is executed, an IP instruction is executed, or the plotter is initialized.

**RELATED INSTRUCTIONS:** SC, Scale
RO, Rotate Coordinate System

**SPECIAL ERRORS:**

| Condition | Error | Plotter Response |
|---|---|---|
| 1 or 3 parameters | 2 | ignores instruction |

## PG, Advance Page

**USE:** Terminates the plot being sent, then rasterizes and draws it, thus advancing the media one page-length and establishing the new pen location at the plotter-unit origin at the lower-left corner of the hard-clip limits. Refer to the plot size (PS) instruction to specify page length.

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

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| n | clamped integer | -32 767 to 32 767 | no parameter |

**REMARKS:** You must include a PG instruction at the end of your program for the plot to be drawn automatically. If you do not end with (*PG;*), you must press the control-panel **PLOT** button for the plot to be drawn.

- **No Parameters** — The PG instruction advances the page only if you have plotted on the current page.

- **n** — Any integer (within the plotter's range) included as a parameter in the PG instruction forces a page feed whether or not you have plotted on the media.

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 effect P1 and P2 values or plot rotation.

**RELATED INSTRUCTIONS:** PS, Plot Size

## PS, Plot Size

**USE:** Changes the size of the hard-clip limits and sets the X-axis along the longest edge of the plot. Use PS to simplify long-axis plotting or to minimize paper waste when drawing small plots.

**SYNTAX:** PS*length(,width;)*
or
PS*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| length | integer | 400 to 609 920 plu | machine dependent** |
| width | integer | machine dependent* | paper width |

* The width range for the D/A1-size plotter is 400 to 24 000 plu; the width range for the E/A0-size plotter is 400 to 35 840 plu.

** The default for the length parameter on the D/A1-size plotter is 35.4 in. (884 mm); the default length on the E/A0-size plotter is 47.8 in. (1189 mm).

**REMARKS:** Send PS, immediately *after* the IN instruction (IN defaults PS) and *before* any drawing instructions. (DF does not default PS.)

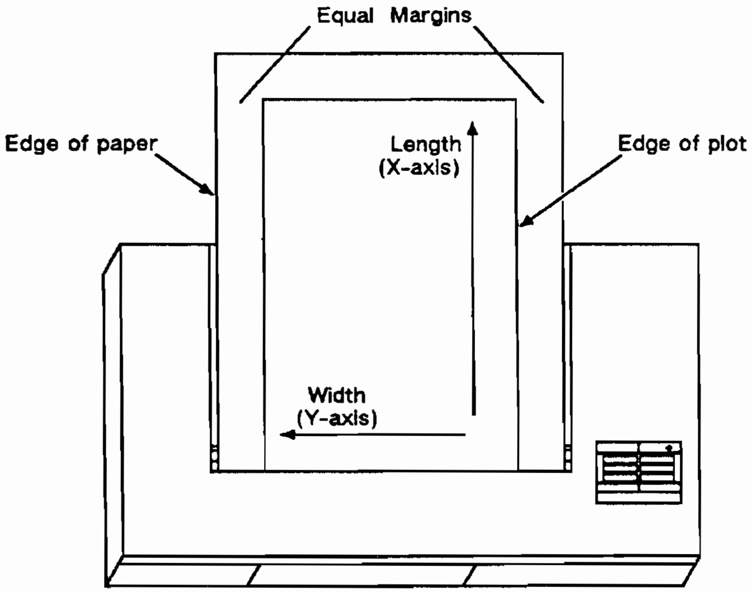
- **No Parameters** - Defaults the plot width to the paper width and the plot length to approximately 1$^1/_2$ times the width.

- **Length** — Establishes the new length, in plotter units, of the hard-clip limits. The length is always in the rollfeed direction of the media. The maximum length for a single plot is approximately 50 feet.

- **Width** — Establishes the new width, in plotter units, of the hard-clip limits. The width is always the horizontal direction.

PS automatically orients the coordinate system so that the X-axis is parallel to the longest edge of the plot. P1 and P2 are defaulted to the lower-left and upper-right corners of the hard-clip limits.

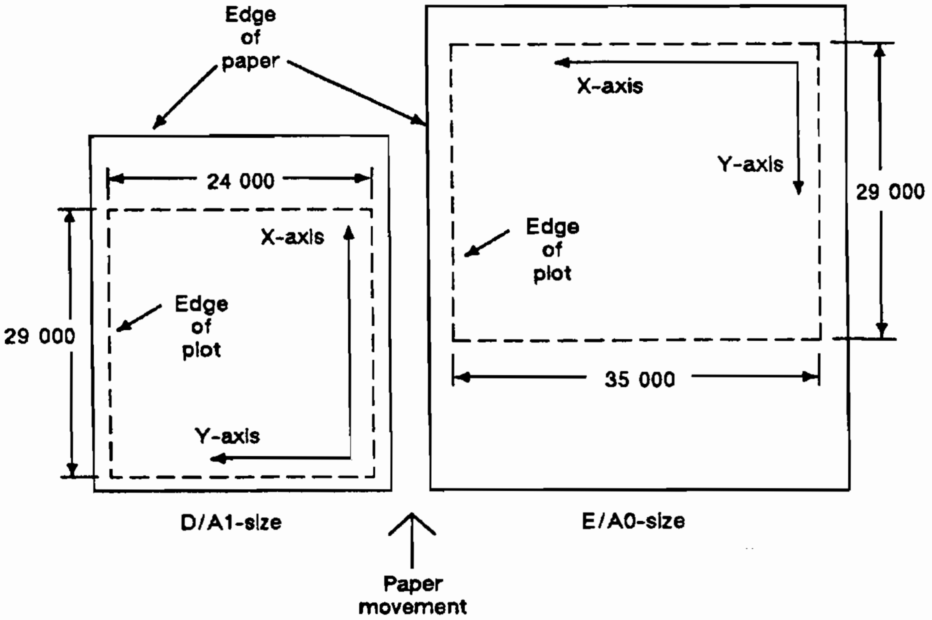
The following illustrates the direction of the X-axis when the length parameter is greater than or equal to the width parameter (i.e., the longest edge is in the rollfeed direction). When the width parameter is less than the paper width, the plot is centered, leaving equal margins on the left and right.



When both dimensions of a plot are less than or equal to the paper's width, specify the smaller dimensions as the length to conserve paper. The advance page (PG) instruction will advance the paper by the distance of the length parameter plus the necessary white space between plots (8.89 cm). The following illustration table lists standard small paper sizes and equivalent PS parameters.

| Standard Paper Sizes | Equivalent PS Parameters |
|---|---|
| English | |
| 8 ½ × 11 in. (A-size) | *(PS8636,11176)* |
| 11 × 17 in. (B-size) | *(PS11176,17272)* |
| 17 × 22 in. (C-size) | *(PS17272,22352)* |
| Metric | |
| 210 × 297 mm (A4-size) | *(PS8400,11880)* |
| 297 × 420 mm (A3-size) | *(PS11880,16800)* |
| 420 × 594 mm (A2-size) | *(PS16800,23760)* |

**NOTE:** Axis orientation always occurs *after any clipping imposed by machine hardware*. For example, the instruction (*PS29000,35000*) is oriented differently on D-size and E-size plotters. A D-size plotter automatically clips the width to the hard-clip limit of 24 000 plotter units; the resulting width is 24 000 and the length is 29 000. No autorotation takes place, as shown in the following illustration. On an E-size plotter, the length parameter is in the rollfeed direction but the X-axis is along the longest edge of the plotter. ■



*(PS29000,35000)*

If a rotate coordinate system (RO) instruction is sent after PS, the direction of the X-axis will change. The implementation of RO is relative to the autorotated position. PS sent after RO does not change the RO rotation, but does update the autorotation to the new longest side.

PS clears the polygon buffer.

| | |
|---|---|
| **RELATED INSTRUCTIONS:** | FR, Frame Advance<br>OH, Output Hard-Clip Limits<br>PG, Advance Full Page |

## PW, Pen Width

**USE:** Specifies a new width for the logical pen. Subsequent lines are drawn in this new width. Use PW to distinguish lines and enhance your plots.

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| width | clamped real | -32 767 to 32 767 | dependent* |
| pen | (ignore all but 0) | (ignore) | (ignore) |

*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:** You may change widths as often as you like, and do not need to send an SP again. If the pen is down when you change the width, the new width takes affect at the next pen down. If you use WU to change the units of the width parameter (metric or relative), send the WU *before* PW.

- **No Parameters** — Defaults the pen line width according to the current units set by WU: 0.35 mm if metric; 1% of the diagonal distance from P1 to P2 if relative.

- **Width** — Specifies the line width. Widths greater than 0.80 are set to the value specified. When specifying a width *less* than 0.80 mm, you are limited to the following widths. Lines of these widths always have round line caps and joins.

| Line Widths (mm) | Pixel Widths |
|---|---|
| 0.125 | 2 |
| 0.1875 | 3 |
| 0.25 | 4 |
| 0.3125 | 5 |
| 0.375 | 6 |
| 0.50 | 8 |
| 0.625 | 10 |
| 0.6875 | 11 |

**NOTE:** Plotter performance is best for line widths less than 0.80 mm. ■

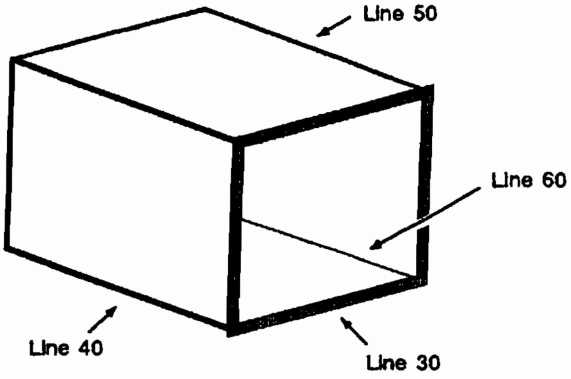
The width of a line less than or equal to 0.8 mm drawn on (parallel to) the soft-clip limit is not clipped. The width of a line greater than 0.8 mm drawn on (parallel to) the soft-clip limit *is* clipped, which can result in a line width that is less than 0.8 mm.

**NOTE:** Pen width does not 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.

A PW instruction remains in effect until another PW instruction or a WU instruction is executed. PW is *not* defaulted by the default values (DF) instruction.

**EXAMPLE:**

```
10 'Insert configuration statement here
20 PRINT #1, "INPS5000,7000SP1PA3500,2500"
30 PRINT #1, "PW1.5PD4500,2800,4500,1800,3500,1500,
             3500,2500"
40 PRINT #1, "PW.8PD2300,2900,2300,1900,3500,1500"
50 PRINT #1, "PW.5PU2300,2900PD3300,3200,4500,2800"
60 PRINT #1, "PW.25PU4500,1800PD3500,2100"
70 PRINT #1, "PG;"
80 END
```



**RELATED INSTRUCTIONS:** SP, Select Pen
WU,Pen Width Unit Selection

**SPECIAL ERRORS:**

| Condition | Error | Plotter Response |
|---|---|---|
| pen number = 0 | 3 | ignores instruction |

## RP, Replot

**USE:** Draws multiple copies of plots. Since the plot is already stored in the plotter, using RP frees the computer while the copies are drawn.

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| n | clamped integer | 1 to 32 767 | 1 |

**REMARKS:** Use the RP instruction at the end of your plot either following or in place of the advance page (PG) instruction.

- n — Selects number of copies to be made.

  If the stored plot is not terminated by a PG instruction, RP terminates the plot and makes the requested number of copies. If the stored plot is terminated by a PG instruction, RP makes the requested number of copies in addition to the original copy.

The plotter ignores RP when printing the current page would produce no marks, i.e., the page is 'clean'.

**RELATED INSTRUCTIONS:** PG, Advance Page

## 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. 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.

**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 |
|---|---|---|---|
| $X_{min}$ | real | -67 108 863 to 67 108 863 | — |
| $X_{max}$ | real | -67 108 863 to 67 108 863 | — |
| $Y_{min}$ | real | -67 108 863 to 67 108 863 | — |
| $Y_{max}$ | real | -67 108 863 to 67 108 863 | — |
| type | clamped integer | 0 to 2 | 0 |
| left | clamped real | 0 to 100% | 50% |
| bottom | clamped real | 0 to 100% | 50% |
| $X_{factor}$ | clamped real | -32 767.9999<br>to 32 767.9999* | — |
| $Y_{factor}$ | clamped real | -32 767.9999<br>to 32 767.9999* | — |

*Excluding zero and values approaching zero

**REMARKS:** For a discussion of the basic concept of scaling, refer to *Using Scaling* earlier in this chapter.

There are three forms of scaling: anisotropic, isotropic, and point-factor. The type parameter tells the plotter which form you are using. Anisotropic and isotropic set up standard user-unit scaling. Point-factor scaling allows you to move the origin while maintaining a specific ratio of plotter units to user units.

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

FORMS 1 and 2: SC$X_{min}$,$X_{max}$,$Y_{min}$,$Y_{max}$ *(,type(,left,bottom;))*

These forms of scaling, anisotropic and isotropic, 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 and isotropic scaling.

- **$X_{min}$, $X_{max}$, $Y_{min}$, $Y_{max}$** — Represent the user unit X- and Y-axis ranges, respectively. As a result, the first and third parameters ($X_{min}$ and $Y_{min}$) are the coordinate pair that is mapped onto P1; the second and fourth parameters ($X_{max}$ and $Y_{max}$) are the coordinate pair that is mapped onto P2. For example, (*SC0,15,0,10*) specifies P1 as (0,0) and P2 and (15,10). This is different from the IP instruction, where the parameters are expressed as X,Y coordinate pairs rather than as ranges.

NOTE: $X_{min}$ cannot be set equal to $X_{max}$, and $Y_{min}$ cannot be set equal to $Y_{max}$. ■

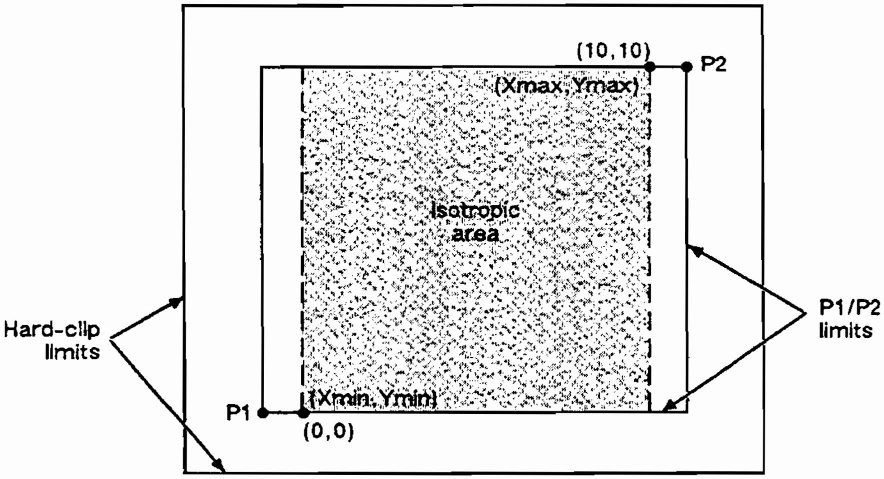
As their names suggest, you will normally want to specify $X_{min}$ smaller than $X_{max}$, and $Y_{min}$ smaller than $Y_{max}$. If you specify $X_{min}$ larger than $X_{max}$ and $Y_{min}$ larger than $Y_{max}$, your plot is drawn as a mirror image, reversed and/or upside down, depending on the relative positions of P1 and P2.

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 relative position and distance between P1 and P2 *after* an SC instruction is executed.

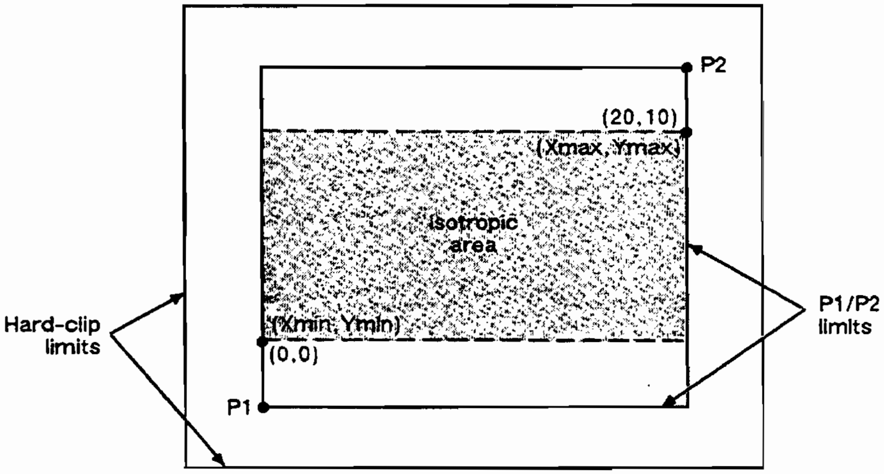
- **Type** — Specifies anisotropic or isotropic scaling.

  0 **Anisotropic scaling.** Allows a user unit to be different sizes along the X-axis and the Y-axis. Plotted shapes are distorted when you use anisotropic scaling. For example, a circle might be drawn as an ellipse (oval-shaped) rather than a circle.

  1 **Isotropic scaling.** Produces user units which are the same size on both the X- and Y-axes. The following illustrations show how the plotter adjusts the location of of ($X_{min}$,$Y_{min}$) and ($X_{max}$,$Y_{max}$) to create the largest possible isotropic area within the P1/P2 limits. (Remember the user units are always square regardless of the shape of the isotropic area.)



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



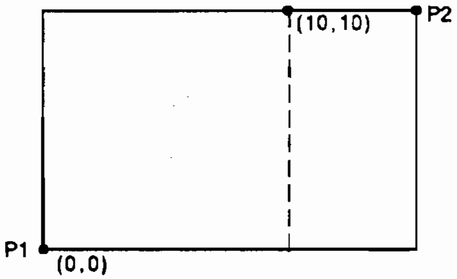
*(SC0,20,0,10,1)*

- **Left, Bottom** — Positions the isometric area in the P1/P2 limits. (These parameters are ignored when anisotropic scaling is specified.) 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.

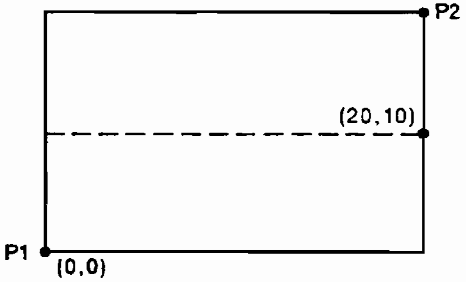
The isotropic area is normally centered with the unused space equally divided between left and right or top and bottom, as shown in the previous illustrations. Thus, the defaults of the left and bottom parameters are 50%.

*Although you must specify both parameters, the plotter applies only one: the left parameter applies when there is extra horzontal 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**

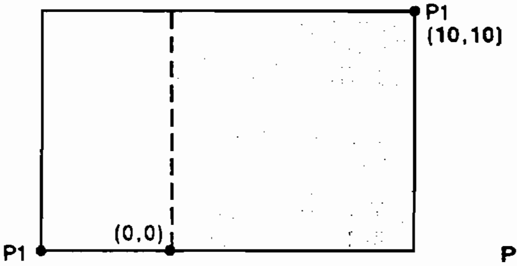


*(SC0,10,0,10,1,0,0)*

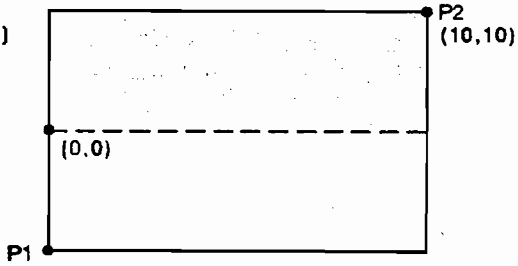


*(SC0,20,0,10,1,0,0)*

**Left, Bottom = 100,100**



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



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

**FORM 3:** SC*X*$_{\text{min}}$,*X*$_{\text{factor}}$,*Y*$_{\text{min}}$,*Y*$_{\text{factor}}$,*type(;)*

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

- **X$_{\text{min}}$,X$_{\text{factor}}$,Y$_{\text{min}}$,Y$_{\text{factor}}$** — Establish the user unit coordinates of P1 and the ratio of plotter to user units. X$_{\text{min}}$ and Y$_{\text{min}}$ are the user unit coordinates of P1. X$_{\text{factor}}$ sets the number of plotter units per user unit on the X-axis; Y$_{\text{factor}}$ 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.

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

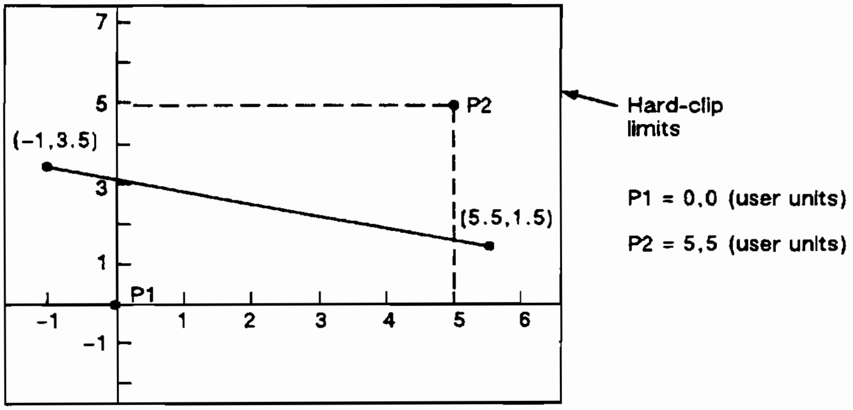
(*SC0,1,0,1,2*) moves the origin to P1 and establishes a one-to-one ratio of plotter to user units. This allows you to continue plotting in plotter units with the advantage of using real numbers.

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

(*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.

While scaling is on (after either form of the SC instruction has been executed), only those plotting instructions that can be issued in 'current units' are interpreted as user units; the instructions that can only be issued 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 that the SC parameters are mapped onto the current locations of P1 and P2. P1 and P2 do *not* represent a graphic 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. For example, you can plot from the point (-1,3.5) to the point (5.5,1.5) as shown in the following illustration.



**RELATED INSTRUCTIONS:** IP, Input P1 and P2

**SPECIAL 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 $Y_{min}=Y_{max}$<br>or<br>number out of range | 3 | ignores instruction |
| $X_{factor}=0$ or $Y_{factor}=0$ | 3 | ignores instruction |

## SP, Select Pen

**USE:** Selects the plotter's 'logical' pen for subsequent plotting. An SP instruction *must* be included in the beginning of each program for the plotter to draw.

**SYNTAX:** SP*pen number(;)*
or
SP*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| pen number | integer | 0 to 67 108 863 | 0 |

**REMARKS:** Although your plotter does not have physical pens, for the purpose of compatibility it has a 'logical' pen which you must select to draw your plot.

- **No Parameters** — Cancels pen selection; subsequent plotting instructions are not drawn. Equivalent to (*SP0*).

- **Pen Number** — Selects the plotter's 'logical' pen. The plotter will not draw unless an SP is sent.

  **0** Cancels pen selection; subsequent plotting instructions are not drawn. Not required at end of program.

  1 Selects the pen. (Since there is only one logical pen, numbers greater than 1 are interpreted as 1.)

Use the pen width (PW) instruction to change the line width. You may change widths as often as you like, and do not need to send an SP again.

**RELATED INSTRUCTIONS:** PW, Pen Width
WU,Pen Width Unit Selection

## 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 | Range | Default |
|---|---|---|---|
| type | clamped integer | 0 or 1 | 0 (metric) |

**REMARKS:** Since using WU, with or without parameters, defaults all pen widths you should send WU *before* PW (which sets a new pen width).

- **No Parameters** — Defaults all pen widths according to the current units set by WU: 0.35 mm if metric; 1% of the diagonal distance from P1 to P2 if relative.

- **Type** — Specifies how the width parameter of the pen width (PW) instruction is interpreted.

  **0** **Metric.** Interprets the pen width parameter in millimetres.

  **1** **Relative.** Interprets the pen width parameter as a percentage of the diagonal distance between P1 and P2.

A WU instruction remains in effect until another WU instruction is executed, or the plotter is initialized. WU is *not* defaulted by the default values (DF) instruction.

**RELATED INSTRUCTIONS:** PW, Pen Width
SP, Select Pen

3

# Drawing Lines

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

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

The following instructions are described in this chapter.

PA, Plot Absolute
PD, Pen Down
PE, Polyline Encoded
PR, Plot Relative
PU, Pen Up

## Pen Position and Location

Although your plotter does not have physical pens, for the purpose of compatibility it has a 'logical' pen to which pen instructions apply. Terminology such as pen position and location refer to this logical pen.

### Pen Position

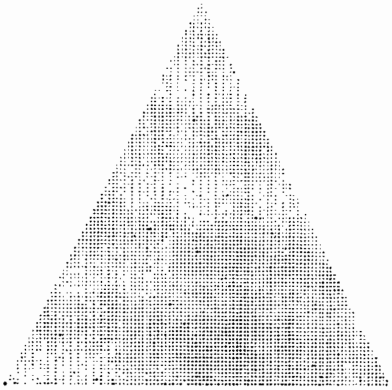
Pen position refers to whether the pen is up or down. Use the pen up (PU) instruction to raise the pen and move to the desired plotting location. Use the pen down (PD) instruction to lower the pen and begin drawing. You must be aware of the pen's position (up or down) to avoid drawing stray lines between 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 instructions in the following list have an automatic PD instruction and return the pen to its previous up/down position after execution.

The instructions in the following list have an automatic pen down instruction and return the pen to its previous up/down position after execution.

| | |
|---|---|
| CI, | Circle |
| EA, | Edge Rectangle Absolute |
| ER, | Edge Rectangle Relative |
| EW, | Edge Wedge |
| FP, | Fill Polygon |
| LB, | Label |
| RA, | Fill Rectangle Absolute |
| RR, | Fill Rectangle Relative |
| SM, | Symbol Mode |
| WG, | Fill Wedge |

**NOTE:** Whenever the plotter receives a pen down instruction, it produces a dot at the next coordinate. (This simulates the dot created when a pen plotter lowers its pen to the media.) 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. Refer to the following illustration.



For best results, include a pen up (PU) instruction before any instruction with an automatic pen down. ■

The definition of the instruction will tell you whether it has an automatic pen down. If part of your plot isn't drawn, make sure your program uses the PD instruction before the affected instructions. When you turn on the plotter or return it to default conditions using the initialize (IN) instruction, the pen position is up.

### Pen Location

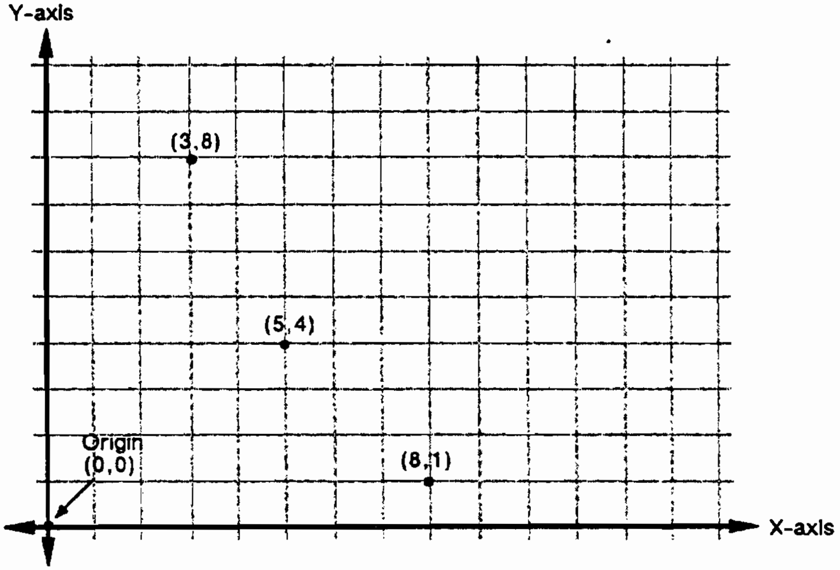
Pen location refers to the X,Y coordinates of the current plotting location (the point at which the next plotting instruction will be drawn.) Whenever a plotting instruction is completed, the pen location is updated to the current point. The next instruction begins at the current pen location. Use the pen up (PU) instruction to move the pen and change the current pen location.

Some instructions do not update the current pen location. For example, after the circle (CI) instruction finishes drawing a circle, it returns to the previous pen location. The definition of each instruction will tell you whether the current pen location is updated or restored.

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.

## Absolute and Relative Movement

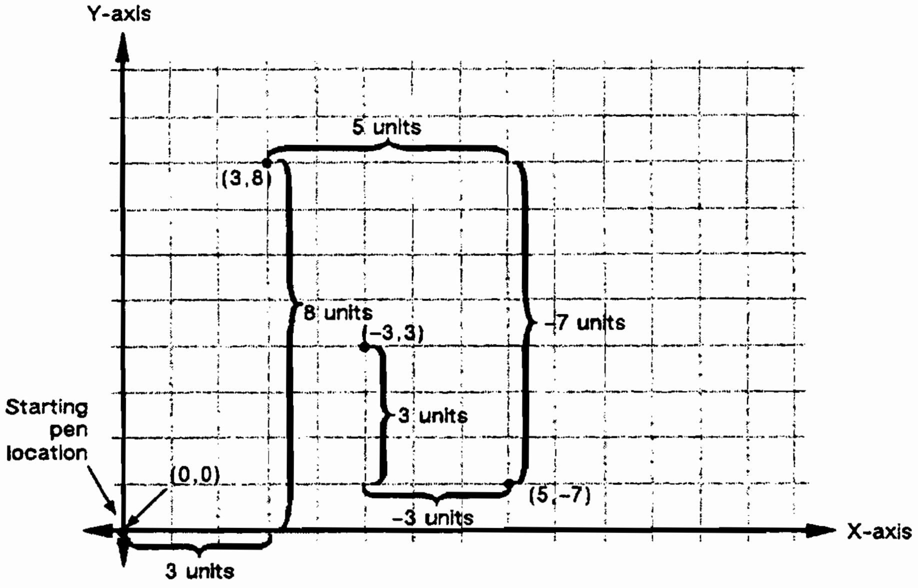
The plot absolute (PA) and plot relative (PR) instructions allow you to set whether you want to draw using absolute or relative movement. Absolute movement uses X,Y coordinates to specify an exact, fixed point relative to the origin (0,0). In the following illustration the coordinates (3,8), (5,4), and (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*

Relative movement uses X,Y *increments* to specify the number of units the pen moves from its current pen location. As with absolute coordinates, the units can be user units or plotter units. All instructions that use relative increments include 'relative' in their name (except the PE instruction).

For example, 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 left 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 use the edge relative rectangle (ER) instruction (which draws from the current pen location to the increments given) to draw the box without having to calculate the absolute coordinates of the opposite corner.

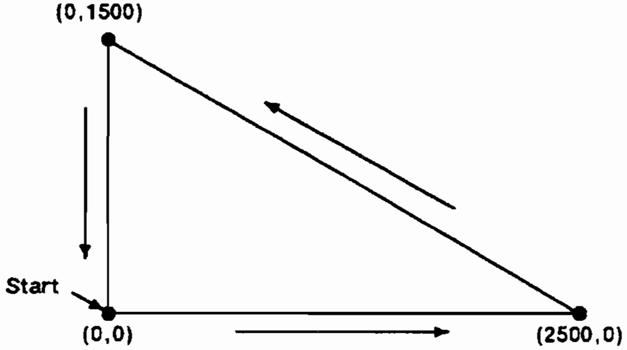
**NOTE:** Relative increments add to the current pen location. The plotter automatically converts the new relative location to absolute coordinates and updates the current pen location. Using relative coordinates increases throughput. ■

## Drawing Lines

You can draw lines between two points (X,Y coordinate pairs) using the PD instruction and a series of absolute and/or relative coordinate pairs. (If one (or both) coordinate pair falls outside the plotting area (soft- 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, "INPS5000,7000PA2000,0"
30 PRINT #1, "SP1PDAA0,0,45,25"
40 PRINT #1, "PU1050,1060PDAA0,0,-45,10"
50 PRINT #1, "PU1000,0PDAA0,0,45PG;"
60 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 can quadruple your plotter's throughput, particularly over an RS-232-C interface. Use PE in place of PA, PD, PR, and PU.

## PA, Plot Absolute

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

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates | current units | -67 108 863 to 67 108 863 | — |

**REMARKS:** The plotter interprets the parameters as follows.

- **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 use the polygon mode (PM) instruction, the X,Y coordinate enter the polygon buffer (are not drawn).

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

**RELATED INSTRUCTIONS:**
PE, Polyline Encoded
PR, Plot Relative

**SPECIAL ERRORS:**

| Condition | Error | Plotter Response |
|---|---|---|
| odd number of coordinates | 2 | ignores last coordinate |

## PD, Pen Down

**USE:** Lowers the plotter's 'logical' pen and draws subsequent graphics instructions.

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates/ increments | current units | −67 108 863 to 67 108 863 | — |

**REMARKS:** This instruction emulates a pen plotter which must lower the pen to draw lines on the media.

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

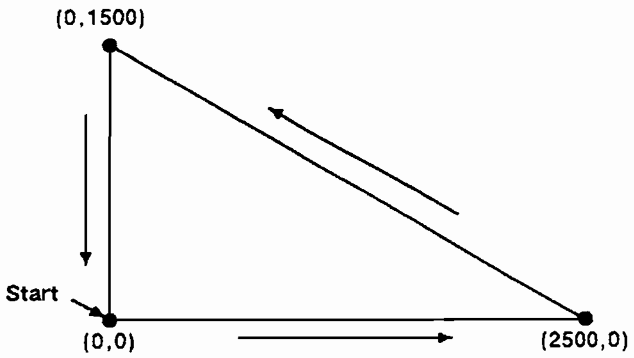
- **X,Y Coordinates/Increments** — Draws (in current units) to the point specified. You can specify as many X,Y coordinate pairs as you want. When you include more than one coordinate pair, the plotter draws to each point in the order given.

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

  Whether the PD 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.

**EXAMPLE:**

```
10 'Insert configuration statement here
20 PRINT #1, "INPS5000,7000SP1"
30 PRINT #1, "PA0,0PD2500,0,0,1500,0,0PG;"
40 END
```



**RELATED INSTRUCTIONS:**

PA, Plot Absolute
PE, Polyline Encoded
PR, Plot Relative
PU, Pen Up

**SPECIAL ERRORS:**

| Condition | Error | Plotter Response |
|---|---|---|
| odd number of coordinates | 2 | ignores last coordinate |

## PE, Encoded Polyline

**USE:** Reduces the size of your file by representing vectors in base 64 or base 32. Using PE can quadruple your throughput, particularly with an RS-232-C interface.

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

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| flag | character | ':', '<', '>', '=', or '7' | — |
| value | character | flag dependent* | — |
| X,Y coordinates | character | –67 108 863 to 67 108 863 plotter units** | — |

*Refer to the table following the parameter description.

**PR and PE have extended ranges of $-2^{31}$ to $2^{31}-1$ plotter units. 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 PE instruction incorporates PA, PR, PD, PU, and SP. Lines are drawn using the current line types and attributes, in current units. The plotter moves to all points with the pen down unless a pen-up flag precedes the point. Relative mode and base 64 are assumed. In parameter value data, all spaces, delete characters, and control characters are ignored, as well as characters 128–160 and 255.

- **Flag** — Indicates how subsequent values should be interpreted. Flags are ASCII characters. The plotter disregards the eighth bit of a flag, i.e., a character code of 61 and a character code of 189 both send a '=' (the absolute flag). *Flag values are encoded in the same manner as coordinate data.*

  : **Select Pen.** Indicates that the subsequent value is the desired pen number. A PE command without pen select defaults to 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 *bits* contained in the coordinate data. Defaults to zero.

- = **Absolute.** Indicates that the subsequent coordinate pair is defined by absolute coordinates.

NOTE: Using absolute and real coordinates reduces the efficiency of the PE instruction. *Relative coordinates produce the most compact data stream*. For best results, scale your plots so that you use only integer coordinates. ■

- 7 **Seven Bit.** Indicates that all subsequent coordinate pair values should be interpreted in 7-bit mode. Once you send a seven-bit flag, base 32 is used and eighth bits are ignored for the remainder of the command.

- **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 a coordinate pairs. These values are encoded as discussed under the next parameter (but without fraction adjustment).

The following table lists the functional ranges for the value parameters.

| Flag | | Functional Range |
|---|---|---|
| : | Select Pen | 0 to 67 108 863 |
| < | Pen Up | — |
| > | Fractional Data | –26 to 26 |
| = | Absolute | — |
| 7 | Seven Bit | — |

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

  When you use the symbol mode (SM) instruction, PE draws the specified symbol at each X,Y coordinate. When you use the polygon mode (PM) instruction, the X,Y coordinate enters the polygon buffer (are not drawn).

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

*Encoding Number Ranges*

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

A generic algorithm for encoding a number is provided below. Assume x is the number to be encoded.

1. **Fraction adjustment.** Multiply x by $2^n$, where 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 is negative, multiply the absolute value of x by two and add one. This sets the sign bit.

```
if $(x \geq 0)$
    $x = 2 \times abs(x)$
else
    $x = 2 \times abs(x) + 1$
```

4. **Convert the number to base 64 or 32.** Use base 64 if your system sends 8 bits without parity. Use base 32 is your system sends 7 bits with parity (seven flag is sent).

5. **Encode the data.** Encode each base 64 or 32 digit into the ASCII character range, as described below, starting with the least significant digit, and output each character as it is encoded. The most significant digit uses a different range than the low order digits. 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)
   ```

When using PE (in the default relative mode), the computer program does not know the current pen location after printing a label (normally, the current pen location is updated to the end of the label.) 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 changes the current pen location and does not update it), set lost to false.

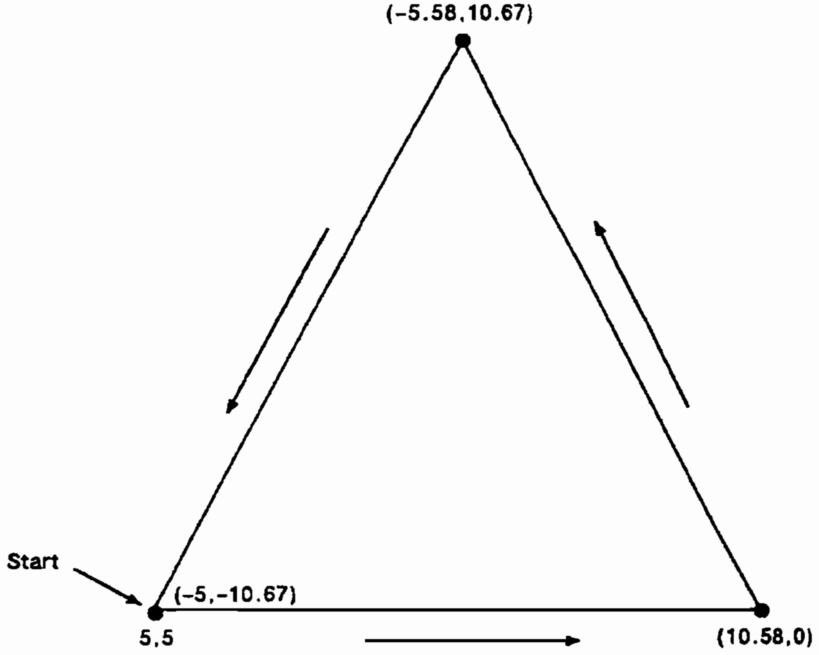
3. If lost = true at the beginning or the PE instruction, use an absolute flag for the next coordinate only (the coordinates will automatically return to integer).

4. Set lost to false.

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

**EXAMPLE:** The following program converts relative real coordinates to base 64.

```
10 'Insert configuration statement here
20 PRINT #1, "INPS6000,8000SC1,20,1,20,1SP1PU5,5"
30 PRINT #1, "Input number of fractional decimal places in
                data"
40  INPUT F
50 'calculate number of fractional binary bits
60 F = F * 3.33
70 F = INT(F)
80 A = F
90 IF F >= 0 THEN F = 2*ABS(F) ELSE F = 2*ABS(F)+1
100 F = 191+F
110 PRINT #1, "PE>"+CHR$(F)
120 'convert coordinate data to base 64
130 FOR J = 1 to 6
140    READ C
150    C = C * (2^A)
160    C = INT(C)
170    IF C >= 0 THEN C = 2*ABS(C) ELSE C = 2*ABS(C)+1
180    WHILE C >= 64
190    PRINT #1, CHR$(63+(C MOD 64))
200    C = C\64
210    WEND
220    C = 191+C
230    PRINT #1, CHR$(C)
240 NEXT J
250 PRINT #1, ";"
260 PRINT #1 "PG;"
270 DATA 10.58,0,-5.58,10.67,-5,-10.67
280 END
```


(-5.58,10.67)
Start
(-5,-10.67)
5,5
(10.58,0)

## 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 | −67 108 863 to 67 108 863* | — |

*PR and PE have extended ranges of $2^{31}$ to $2^{31}-1$ plotter units. 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.

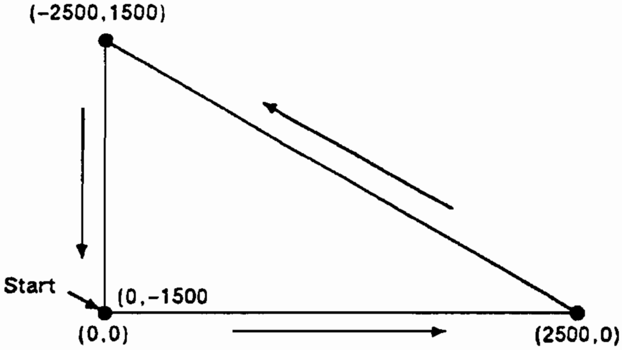
- **X, Y Increments** — Specify incremental moves relative to the current pen location. When you include more than one increment pair, the pen moves to each point in the order given (relative to the previous point), using the current pen up/down position. If the pen is up, PR moves the pen to the point; 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. When you use the polygon mode (PM) instruction, the X,Y coordinate enter the polygon buffer (are not drawn).

  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, "INPS5000,7000SP1"
30 PRINT #1, "PA0,0PDPR2500,0,-2500,1500,0,-1500"
40 PRINT #1, "PG;"
50 END
```



**RELATED INSTRUCTIONS:** PA, Plot Absolute
PE, Polyline Encoded

**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:** Moves to subsequent points without drawing. Use PU to move between points without drawing a connecting line.

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

| Parameter | Format | Range | Default |
|---|---|---|---|
| X,Y coordinates/ increments | current units | -67 108 863 to 67 108 863 | — |

**REMARKS:** This instruction emulates a pen plotter which must raise the pen to prevent drawing stray lines on the media.

- 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, PA draws the specified symbol at each X,Y coordinate. When you use the polygon mode (PM) instruction, the X,Y coordinate enter the polygon buffer (are not drawn).

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

  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:**
PA, Plot Absolute
PD, Pen Down
PE, Polyline Encoded
PR, Plot Relative

**SPECIAL ERRORS:**

| Condition | Error | Plotter Response |
|---|---|---|
| odd number of coordinates | 2 | ignores last coordinate |

4

# Drawing Shapes

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

- Draw rectangles.
- Draw circles, arcs, and wedges.
- Control the smoothness of circles, arcs, and wedges.

The following instructions are described in this chapter.

| | |
|---|---|
| AA, | Arc Absolute |
| AR, | Arc Relative |
| AT, | Absolute Arc Three Point |
| CI, | Circle |
| CT, | Chord Tolerance Mode |
| EA, | Edge Rectangle Absolute |
| ER, | Edge Rectangle Relative |
| EW, | Edge Wedge |
| RT, | Relative Arc Three Point |

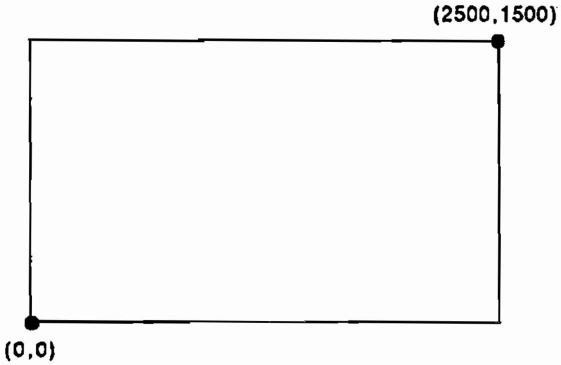
## Drawing Rectangles

You can draw a rectangle by outlining (edging) the defined area using the edge rectangle absolute (EA) or edge rectangle relative (ER) instructions. (You can also create shaded (filled) rectangles, covered in the next chapter.)

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.

The following simple program uses EA to draw a rectangle.

```
10 'Insert configuration statement here
20 PRINT #1, "INPS5000,7000SP1"
30 PRINT #1, "PA0,0EA2500,1500"
40 PRINT #1, "PG;"
50 END
```

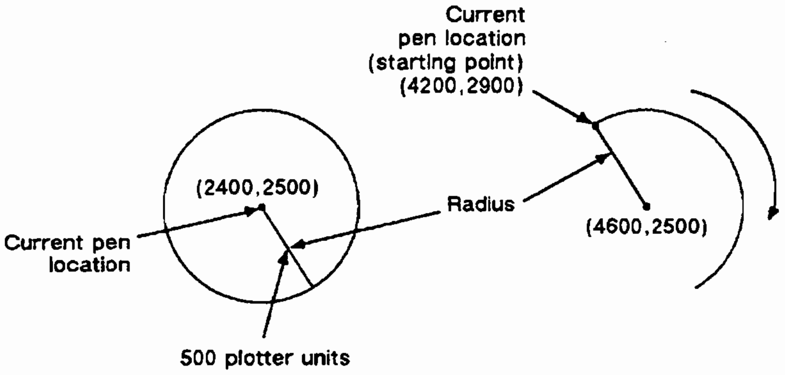


## Drawing Circles and Arcs

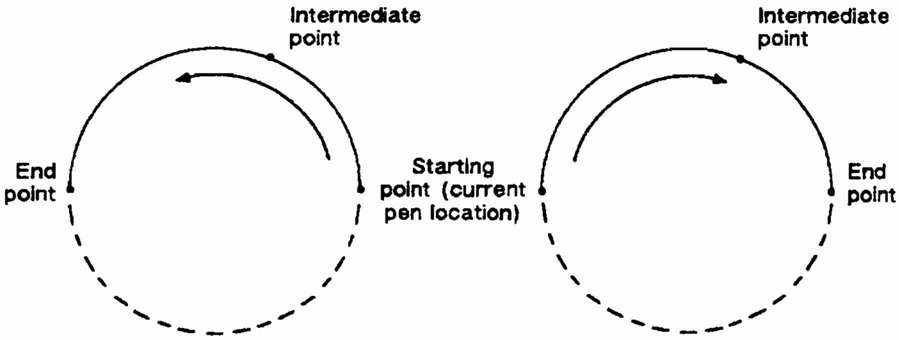
The circle (CI) instruction uses your current pen position as the center of the circle; you specify the radius of the circle. The arc absolute (AA) and arc relative (AR) instructions use a similiar method for drawing arcs: Your current pen location becomes one end of the arc; you specify the center point (setting the radius) and the number of degrees through which you want the arc drawn.

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

```
10 'Insert configuration statement here
20 PRINT #1, "INPS5000,7000SP1PA2400,2500"
30 PRINT #1, "CI500PA4200,2900PDAA4600,2500,-180"
40 PRINT #1, "PG;"
50 END
```



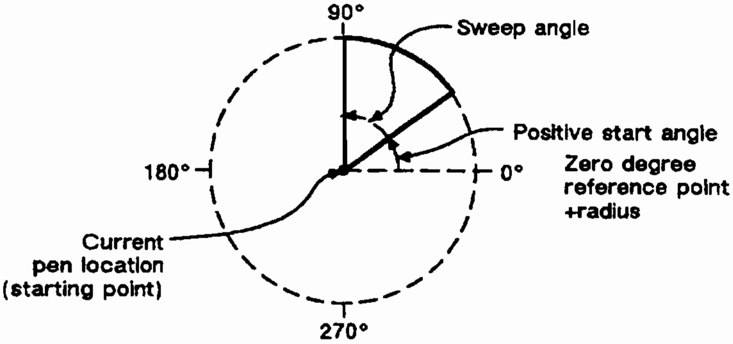
You can also draw arcs using the absolute arc three point (AT) and relative arc three point (RT) instructions. These instructions use three known points (your current pen location plus two points you specify) to calculate a circle and draw the appropriate arc segment of its circumference. The arc is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



## Drawing Wedges

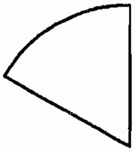
A wedge is a section of a circle. Wedges are commonly used to draw pie charts. You can draw a wedge by outlining (edging) the defined area using the edge wedge (EW) instruction. (You can also create shaded (filled) wedges, covered in the next chapter.)

The wedge instruction uses 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) determines the location of a 'zero-degree' reference point. The start angle is the number of degrees from the zero 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. The following illustration shows the different parameters of a wedge with a positive radius.


90°
Sweep angle
Positive start angle
180°
0°
Zero degree reference point +radius
Current pen location (starting point)
270°

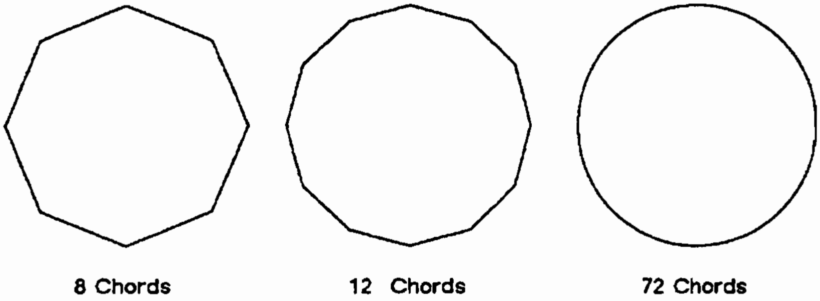
The following simple program draws a wedge using the EW instruction.

```
10 ´Insert configuration statement here
20 PRINT #1, "INPS5ØØØ,7ØØØSP1"
3Ø PRINT #1, "PA25ØØ,35ØØEW6ØØ,9Ø,6Ø"
4Ø PRINT #1, "PG;"
5Ø END
```



## Chords and Chord Tolerance

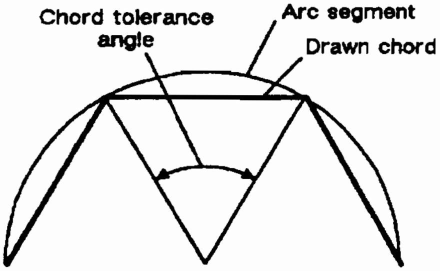
To draw curves the plotter draws a series of straight lines, called *chords*, to represent each arc segment. The apparent smoothness of the curve depends on the number of chords used to draw it; the more chords, the smoother the shape appears. The following illustration shows circles drawn with different numbers of chords.



The number of chords is determined by the chord tolerance, which 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. Use the chord tolerance mode (CT) instruction to select the method you want to use.

### Chord Angle

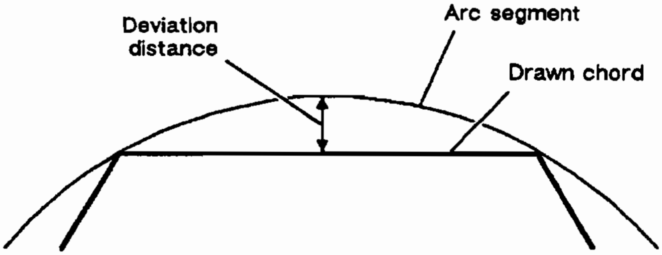
The chord angle method specifies, in degrees, the maximum angle created when lines from each end of the chord intersect the center point of the circle.



When you specify the chord tolerance as a chord angle, the curved shape will always have the same number of chords, regardless of its size. For example, a circle drawn with the default chord angle of 5 degrees will always have 72 chords. One result is that a large circle will appear less smooth than a smaller circle having the same chord angle.

## 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 plotter adjusts the number of chords in the curved shape to maintain the deviation distance. A small circle drawn with a specific deviation distance has fewer chords than a large circle drawn with the same deviation distance.

# AA, Arc Absolute

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

**SYNTAX:** AA*X center,Y center,sweep angle(,chord tolerance;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y center | current units | −67 108 863 to 67 108 863 | — |
| sweep angle | clamped real | −360 to 360 degrees | — |
| chord tolerance | | | |
| chord angle* | clamped real | 0.5 to 90 degrees | 5 degrees |
| chord deviation | current units | 0 to 67 108 863 | 5 degrees |

*Chord angle is the default interpretation of chord tolerance.

**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 drawing the arc, the pen location remains at the end of the arc.

- **X,Y Center** — Specify the absolute location 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.)

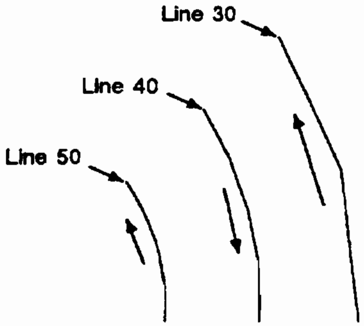
  Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off. If current scaling is not isotropic, the arc drawn is elliptical rather than circular.

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

- **Chord Tolerance** — Specifies the chord tolerance used to draw the arc. The default is a chord angle of 5 degrees. Refer to the *Chords and Chord Tolerance* discussion at the beginning of this chapter or the chord tolerance (CT) instruction for information on setting and determining chord tolerance.

**EXAMPLE:**

```
10 'Insert configuration statement here
20 PRINT #1, "INSP1PS5000,7000PA2000,0"
30 PRINT #1, "PDAA0,0,45,25"
40 PRINT #1, "PU1050,1060PDAA0,0,-45,10"
50 PRINT #1, "PU1000,0PDAA0,0,45PG;"
60 END
```



**RELATED INSTRUCTIONS:**

| | |
|---|---|
| AR, | Arc Relative |
| AT, | Absolute Arc Three Point |
| CI, | Circle |
| CT, | Chord Tolerance Mode |
| LA, | Line Attributes |
| LT, | Line Type |
| PW, | Pen Width |
| RT, | Relative Arc Three Point |

## AR, Arc Relative

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

**SYNTAX:** AR*X increment,Y increment,sweep angle(,chord tolerance;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y increments | current units | -67 108 863 to 67 108 863 | — |
| sweep angle | clamped real | -360 to 360 degrees | — |
| chord tolerance | | | |
| chord angle* | clamped real | 0.5 to 90 degrees | 5 degrees |
| chord deviation | current units | 0 to 67 108 863 | 5 degrees |

*Chord angle is the default interpretation of chord tolerance.

**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 drawing the arc, the pen location remains at the end of the arc.

- **X,Y Increments** — Specify 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.)

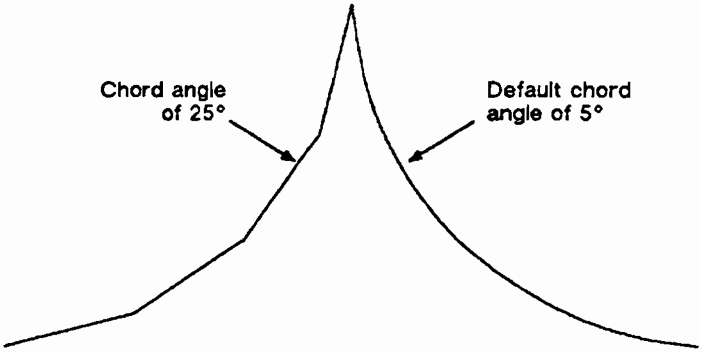
  Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off. If current scaling is not isotropic, the arc drawn is elliptical rather than circular.

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

- **Chord Tolerance** — Specifies the chord tolerance used to draw the arc. The default is a chord angle of 5 degrees. Refer to the *Chords and Chord Tolerance* discussion at the beginning of this chapter or the chord tolerance (CT) instruction for information on setting and determining chord tolerance.

**EXAMPLE:**

```
10 'Insert configuration statement here
20 PRINT #1, "INSP1PS5000,7000PA1500,1500PD"
30 PRINT #1, "AR0,2000,80,25AR2000,0,80"
40 PRINT #1, "PG;"
50 END
```



**RELATED INSTRUCTIONS:**

AA, Arc Absolute
AT, Absolute Arc Three Point
CT, Chord Tolerance Mode
LA, Line Attributes
LT, Line Type
PW, Pen Width
RT, Relative Arc Three Point

# AT, Absolute Arc Three Point

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

**SYNTAX:** AT*X inter,Y inter,X end,Y end(,chord tolerance;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y increments | current units | -67 108 863 to 67 108 863 | — |
| sweep angle | clamped real | -360 to 360 degrees | — |
| chord tolerance<br>chord angle* | clamped real | 0.5 to 90 degrees | 5 degrees |
| chord deviation | current units | 0 to 67 108 863 | 5 degrees |

***Chord angle is the default interpretation of chord tolerance.**

**REMARKS:** The AT instruction uses the current pen location and two specified points to calculate a circle and draw the approprate 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 end points. After drawing the arc, the pen location remains at the end of the arc.

- **X,Y Inter** — Specify 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 end point.

  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.

- **X,Y End** — Specify the absolute location of the end point of the arc.

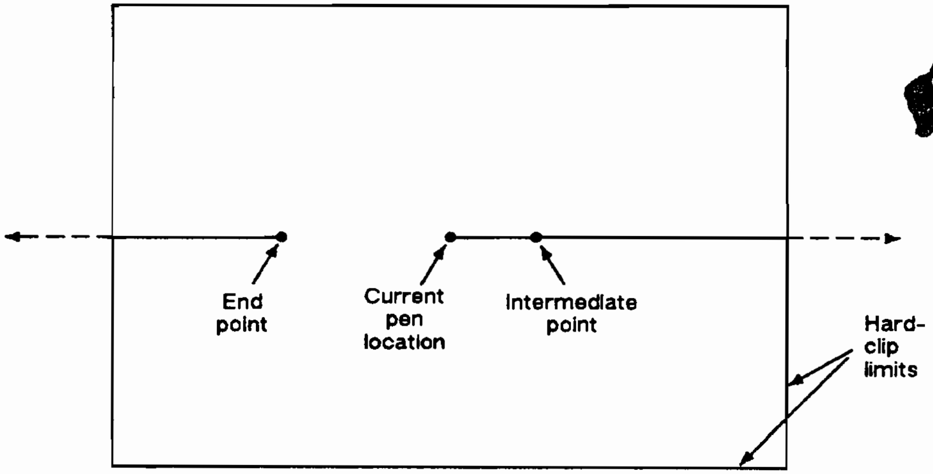
  Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off. If current scaling is not isotropic, the arc drawn is elliptical rather than circular.

- **Chord Tolerance** — Specifies the chord tolerance used to draw the arc. The default is a chord angle of 5 degrees. Refer to *Chords and Chord Tolerance* discussion at the beginning of this chapter.

If the arc is defined by three identical points, the instruction draws a dot.

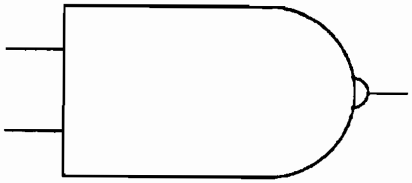
If the intermediate point is the same as either the current pen position or the end point, a line is drawn between the current pen position and the end point. If the current pen position is the same as the end point, a circle is drawn, with its diameter being the line from the current pen position to the intermediate point.

If the current pen position, intermediate point, and end point are collinear, a straight line is drawn. If the intermediate point does not lie between the current pen location and the end point, two lines are drawn, one from the current pen location and the other from the end point, leaving a gap between them. Refer to the following illustration. Both lines extend to the hard-clip limits or current window.



**EXAMPLE:**

```
10 'Insert configuration statement here
20 PRINT #1, "INPS5000,7000SP1PA1000,100PD2500,100"
30 PRINT #1, "PU650,1150PD1000,1150PU650,450PD1000,450"
40 PRINT #1, "PU1000,100PD1000,1500PD2500,1500AT3200,800,
              2500,100"
50 PRINT #1, "PU3200,900PDAT3300,800,3200,700PU3300,800
              PD3500,800"
90 PRINT #1, "PG;"
100 END
```



**RELATED INSTRUCTIONS:**

AA, Arc Absolute
AR, Arc Relative
CT, Chord Tolerance Mode
LA, Line Attributes
LT, Line Type
PW, Pen Width
RT, Relative Arc Three Point

## CI, Circle

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

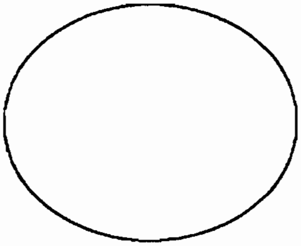
**SYNTAX:** CI*radius(,chord tolerance;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| radius | current units | -67 108 863 to 67 108 863 | — |
| chord tolerance<br>chord angle* | clamped real | 0.5 to 90 degrees | 5 degrees |
| chord deviation | current units | 0 to 67 108 863 | 5 degrees |

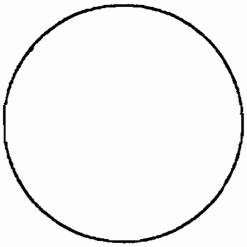
*Chord angle is the default interpretation of chord tolerance.

**REMARKS:** The CI instruction includes an automatic pen down. When a CI instruction is received, the pen lifts, moves from the center of the circle (the current pen location) to the starting point on the circumference, lowers the pen, draws the circle, then returns with the 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. 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 (72 in the default chord angle mode). Always use isotropic scaling in plots that draw circles; anisotropic scaling may produce an ellipse. Refer to the discussion of scaling in Chapter 2 and the scale (SC) instruction description for more information.



Anisotropic scaling



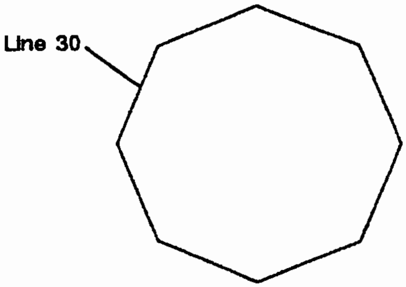
Isotropic scaling

- **Radius** — Measured from the current pen location. Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off.

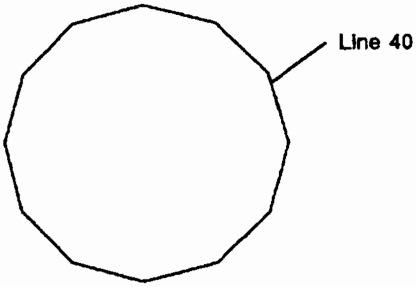
- **Chord Tolerance** — Specifies the chord tolerance used to draw the arc. The default is a chord angle of 5 degrees. Refer to the *Chords and Chord Tolerance* discussion at the beginning of this chapter or the chord tolerance mode (CT) instruction for information on setting and determining chord tolerance.

**EXAMPLE: Effects of Chord Angle on Circle Smoothness**

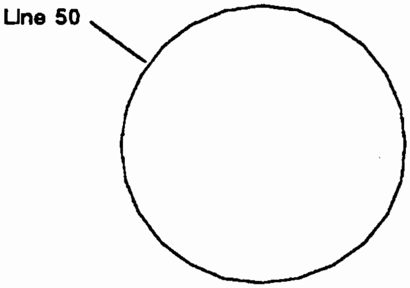
```
10 'Insert configuration statement here
20 PRINT #1, "INPS6000,7000SP1SC-3000,3000,-2000,2000,1"
30 PRINT #1, "PA-1700,2000CI750,45"
40 PRINT #1, "PA300,2000CI750,30"
50 PRINT #1, "PA-1700,-200CI750,15"
60 PRINT #1, "PA300,-200CI750PG;"
70 END
```



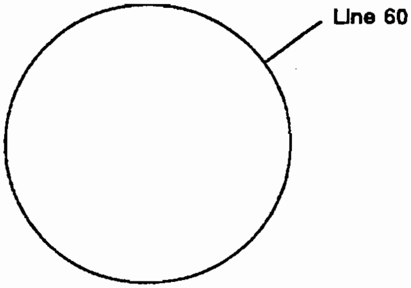
45-Degree chord angle



30-Degree chord angle



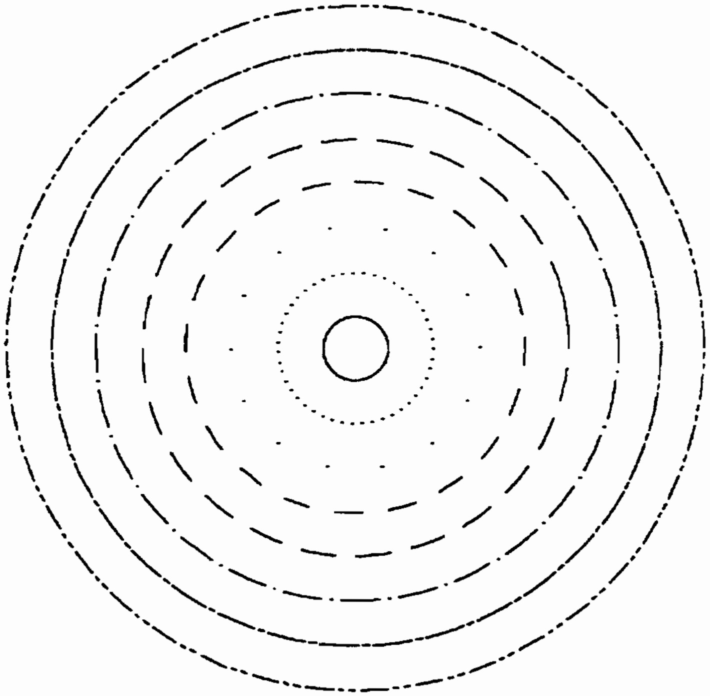
15-Degree chord angle



5-Degree chord angle

### Drawing Circles with Different Radii and Line Types

```
10 'Insert configuration statement here
20 PRINT #1, "INPS5000,7000SP1SC-75,75,-75,75,1"
30 PRINT #1, "PA0,0LTCI5LT0CI-12LT1CI19"
40 PRINT #1, "LT2CI-26LT3CI33LT4CI-40"
50 PRINT #1, "LT5CI47LT6CI54PG;"
60 END
```



**RELATED INSTRUCTIONS:**

CT, Chord Tolerance Mode
LA, Line Attributes
LT, Line Type
PW, Pen Wdith
SC, Scale
WG, Fill Wedge

## 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.

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

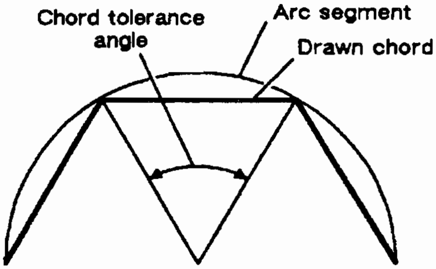
| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| n | integer | 0 or 1 | 0 |

**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 disc 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.

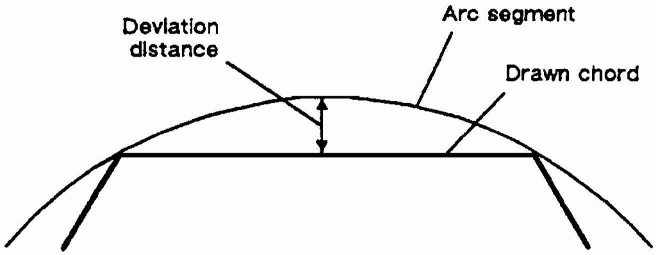
- **No Parameter** — Defaults the chord tolerance to a chord angle. Equivalent to (*CT0*).

- **n** — 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 the circle will vary with its size.



| | |
|---|---|
| **RELATED INSTRUCTIONS:** | AA, Arc Absolute<br>AR, Arc Relative<br>AT, Absolute Arc Three Point<br>CI, Circle<br>EW, Edge Wedge<br>RT, Relative Arc Three Point<br>WG, Fill Wedge |

## EA, Edge Rectangle Absolute

**USE:** Defines and outlines a rectangle using absolute coordinates. Use EA when drawing charts or schematics that require rectangles.

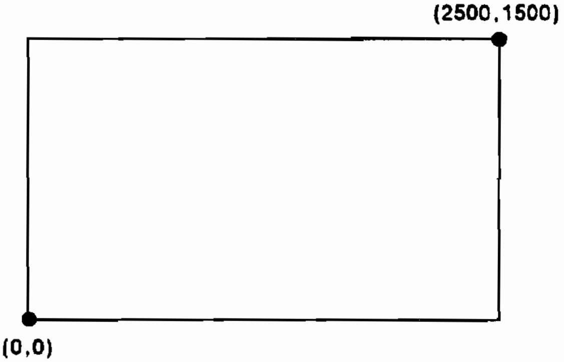
**SYNTAX:** EA*X*,*Y*(;)

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates | current units | −67 108 863 to 67 108 863 | — |

**REMARKS:** The EA instruction defines and edges a rectangle using the current pen, line type, line attributes, and absolute plotting. 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.

  **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 either two diagonally opposite corners. ■

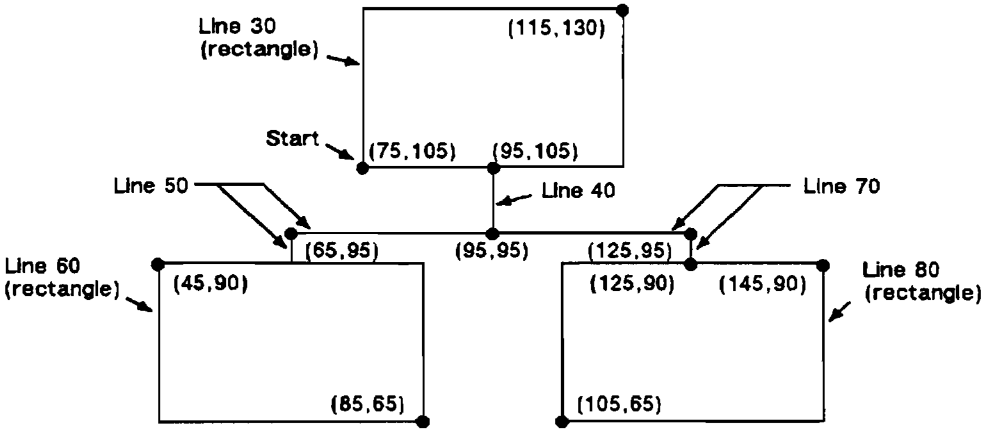


The only difference between the EA instruction and the RA instruction is that the EA instruction produces an outlined rectangle; and RA, a filled one.

The EA instruction clears the polygon buffer and then uses it to define the rectangle before drawing. Refer to *Using the Polygon Buffer* in Chapter 6 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, "INPS6000,8000SC0,150,0,150,1SP1"
30 PRINT #1, "PA75,105EA115,130"
40 PRINT #1, "PA95,105PD95,95"
50 PRINT #1, "PD65,95,65,90"
60 PRINT #1, "PU45,90EA85,65"
70 PRINT #1, "PU95,95PD125,95,125,90"
80 PRINT #1, "PU145,90EA105,65PG;"
90 END
```



**RELATED INSTRUCTIONS:**

RA, Fill Rectangle Absolute
ER, Edge Rectangle Relative
RR, Fill Rectangle Relative

**SPECIAL ERRORS:**

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

## ER, Edge Rectangle Relative

**USE:** Defines and outlines a rectangle using relative coordinates. Use ER when drawing charts or schematics that require rectangles.

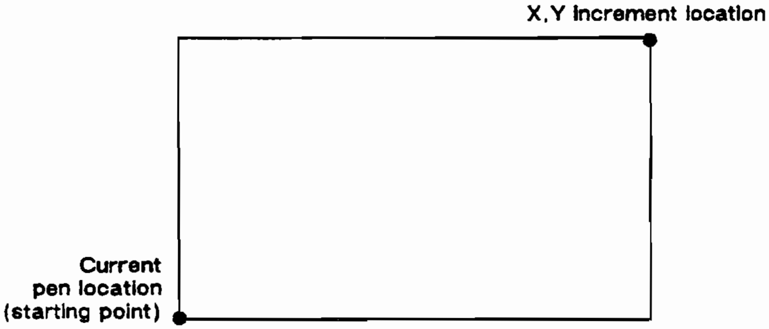
**SYNTAX:** ER*X,Y(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y increments | current units | −67 108 863 to 67 108 863 | — |

**REMARKS:** The ER instruction defines and edges a rectangle using the current pen, line type, line attributes, and relative coordinates. 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.

  **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 either diagonally opposite corners. ■

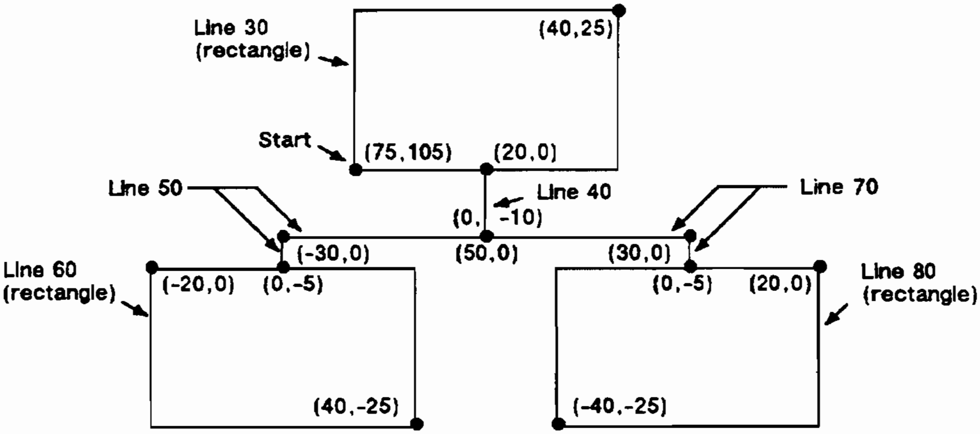


The only difference between the ER instruction and the RR instruction is that the ER instruction produces an outlined rectangle; and RR, a filled one.

The ER instruction clears the polygon buffer and then uses it to define the rectangle before drawing. Refer to *Using the Polygon Buffer* in Chapter 6 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, "INPS6000,8000SC0,150,0,150,1SP1"
30 PRINT #1, "PA75,105ER40,25"
40 PRINT #1, "PR20,0PD0,-10"
50 PRINT #1, "PD-30,0,0,-5"
60 PRINT #1, "PU-20,0ER40,-25"
70 PRINT #1, "PU50,5PD30,0,0,-5"
80 PRINT #1, "PU20,0ER-40,-25PG;"
90 END
```



**RELATED INSTRUCTIONS:**

EA, Edge Rectangle Absolute
RA, Fill Rectangle Absolute
RR, Fill Rectangle Relative

**SPECIAL ERRORS:**

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

## EW, Edge Wedge

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

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

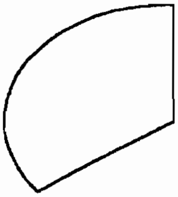
| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| radius | current units | −67 108 863 to 67 108 863 | — |
| start angle | clamped real | −360 to 360 degrees | — |
| sweep angle | clamped real | −360 to 360 degrees | — |
| chord tolerance | | | |
| chord angle* | clamped real | 0.5 to 90 degrees | 5 degrees |
| chord deviation | current units | 0 to 67 108 863 | 5 degrees |

*Chord angle is the default interpretation of chord tolerance.

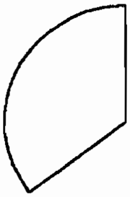
**REMARKS:** The EW instruction defines and edges a wedge using the current pen, line type and attributes. The EW instruction includes an automatic pen down. When the instruction is completed, the original pen location and up/down position are restored.

The only difference between the EW instruction and the WG instruction is that the EW instruction produces an outlined wedge; and 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 description for more information.



Anisotropic scaling



Isotropic scaling

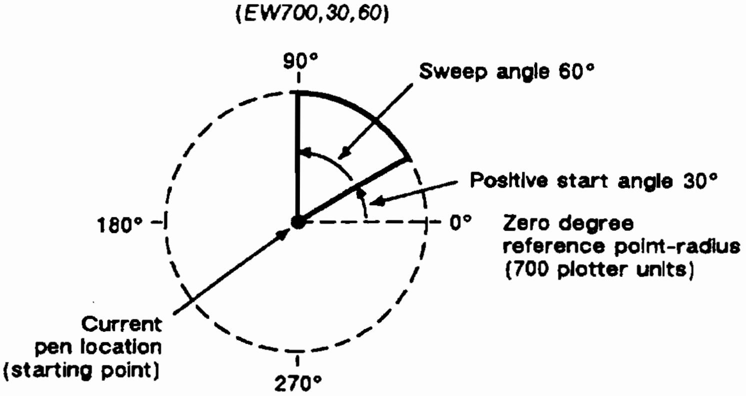
- **Radius** — Specifies the distance from the current pen location to the start of the wedge's arc. Since the wedge is a portion of a circle, this parameter is the radius of the circle. It specifies the distance from the current pen location (which becomes the center of the circle), to any point on 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 of the radius determines the location of the zero-degree reference point. The illustration following the parameter descriptions shows 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 (the direction from the positive X-axis toward the positive Y-axis) from the zero-degree reference point; a negative start angle positions the radius clockwise from the zero-degree reference point.

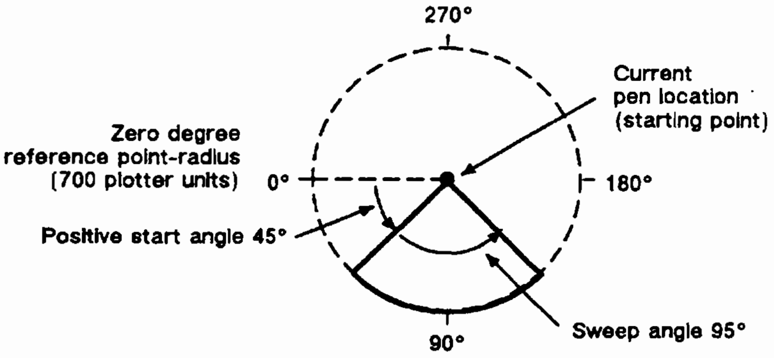
- **Sweep Angle** — Specifies the number of degrees through which the arc is drawn. A positive sweep angle draws the arc counterclockwise; a negative sweep angle draws the arc clockwise. If you specify a sweep angle greater than 360 degrees, a 360-degree angle is used.

- **Chord Tolerance** — Specifies the chord tolerance used to draw the arc. The default is a chord angle of 5 degrees. Refer to the *Chords and Chord Tolerance* discussion in Chapter 4 or the chord tolerance mode (CT) instruction for information on setting and determining chord tolerance.



*Positive Radius*

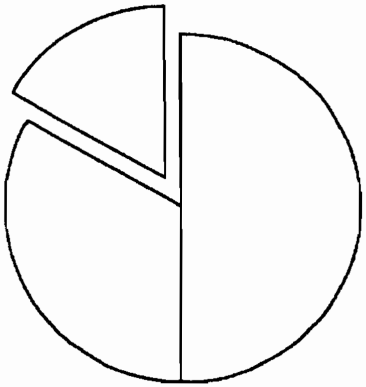
*(EW-700,45,95)*



*Negative Radius*

**EXAMPLE:**

```
10 'Insert configuration statement here
20 PRINT #1, "INPS5ØØØ,7ØØØSP1SC-3ØØØ,3ØØØ,-2ØØØ,2ØØØ,1"
3Ø PRINT #1, "PAØ,Ø"
4Ø PRINT #1, "EW-1ØØØ,9Ø,18Ø"
5Ø PRINT #1, "EW-1ØØØ,33Ø,12Ø"
6Ø PRINT #1, "PR-6Ø,11Ø"
7Ø PRINT #1, "EW-1ØØØ,27Ø,6ØPG;"
8Ø END
```



**RELATED INSTRUCTIONS:**

CT, Chord Tolerance Mode
LA, Line Attributes
LT, Line Type
PW, Pen Width
SC, Scale
WG, Fill Wedge

**SPECIAL ERRORS:**

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

## RT, Relative Arc Three Point

**USE:** Draws an arc segment, using relative coordinates, from a starting point through an intermediate point to an end point. 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 tolerance;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y incr inter | current units | −67 108 863 to 67 108 863 | — |
| X,Y incr end | current units | −67 108 863 to 67 108 863 | — |
| chord tolerance | | | |
| chord angle* | clamped real | 0.5 to 90 degrees | 5 degrees |
| chord deviation | current units | 0 to 67 108 863 | 5 degrees |

*Chord angle is the default interpretation of chord tolerance.

**REMARKS:** The RT instruction uses the current pen location and two specified points to calculate a circle and draw the approprate 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 end points. After drawing the arc, the pen location remains at the end of the arc.

- **X,Y Incr Inter** - Specify the location of an intermediate point of the arc in relative increments. The arc is drawn clockwise or counterclockwise, as necessary, so that it passes through the intermediate point before the end point.

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.

- **X,Y Incr End** - Specify the location of the end point of the arc in relative increments.

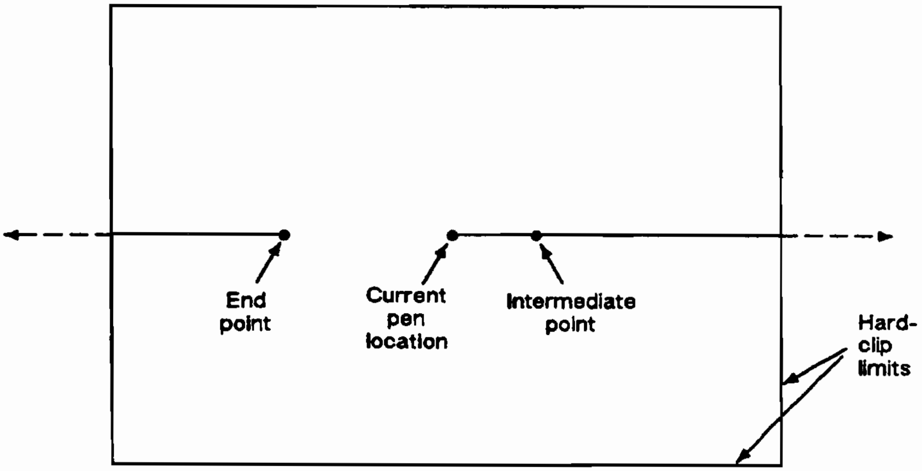
  Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off. If current scaling is not isotropic, the arc drawn is elliptical rather than circular.

- **Chord Tolerance** - Specifies the chord tolerance used to draw the arc. The default is a chord angle of 5 degrees. Refer to *Chords and Chord Angles* discussion at the beginning of this chapter.

If the arc is defined by three identical points, the instruction draws a dot.

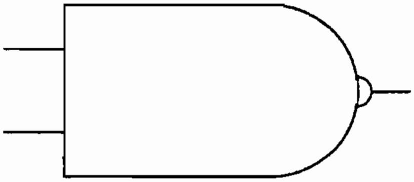
If the intermediate point is the same as either the current pen position or the end point, a line is drawn between the current pen position and the end point. If the current pen position is the same as the end point, a circle is drawn, with its diameter being the line from the current pen position to the intermediate point.

If the current pen position, intermediate point, and end point are collinear, a straight line is drawn. If the intermediate point does not lie between the current pen location and the end point, two lines are drawn, one from the current pen location and the other from the end point, leaving a gap between them. Refer to the following illustration. Both lines extend to the hard-clip limits or current window.



**EXAMPLE:**

```
10 ´Insert configuration statement here
20 PRINT #1, "INPS5ØØØ,7ØØØSP1PA1ØØØ,1ØØPRPD15ØØ,Ø"
30 PRINT #1, "PU-185Ø,1Ø5ØPD35Ø,ØPU-35Ø,-7ØØPD35Ø,Ø"
40 PRINT #1, "PUØ,-35ØPDØ,15ØØPD15ØØ,ØRT7ØØ,-75Ø,Ø,-15ØØ"
50 PRINT #1, "PU7ØØ,85ØPDRT1ØØ,-1ØØ,Ø,-2ØØPU1ØØ,1ØØPD2ØØ,Ø"
90 PRINT #1, "PG;"
100 END
```



**RELATED INSTRUCTIONS:**

AA, Arc Absolute
AR, Arc Relative
AT, Absolute Arc Three Point
CT, Chord Tolerance Mode
LA, Line Attributes
LT, Line Type
PW, Pen Width

5

# Enhancing Your Plots

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

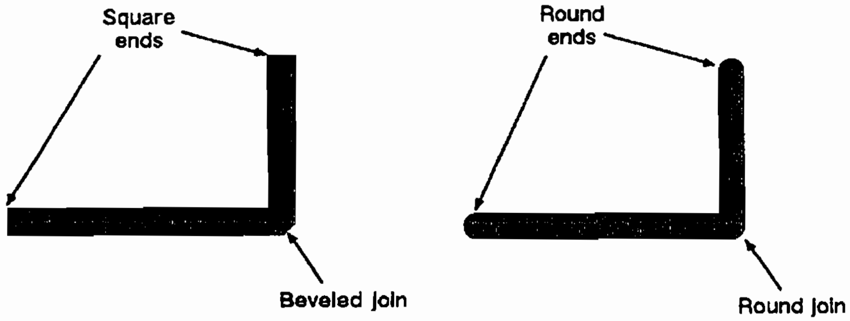
- Enhance your plots with line types.
- Enhance your plots with fill types.
- Position fill type patterns.

The following instructions are described in this chapter.

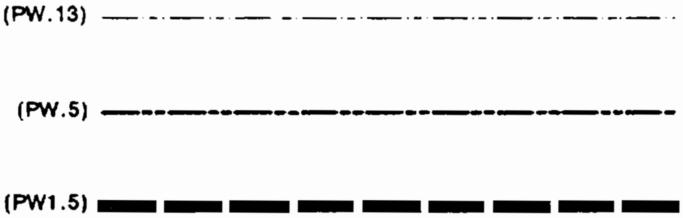
| | |
|---|---|
| AC, | Anchor Corner |
| FT, | Fill Type |
| LA, | Line Attributes |
| LT, | Line Type |
| RA, | Fill Rectangle Absolute |
| RF, | Raster Fill Definition |
| RR, | Fill Rectangle Relative |
| UL, | User-Defined Line Type |
| WG, | Fill Wedge |

## Using Line Attributes and Types

You can change the appearance of the lines you draw by using the line attribute (LA) and line type (LT) instructions. The line attribute 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 the lines and line types you draw by using the pen width (PW) instruction.

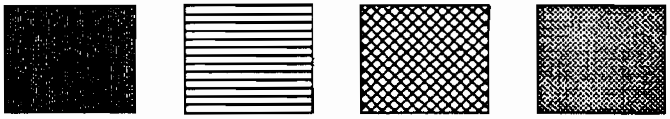


Once you specify a line type and line attributes, all lines created by the following instructions are drawn using the new line type and attributes. Line types and their interactions with fill patterns are discussed later in this chapter.

| | |
|---|---|
| AA, | Arc Absolute |
| AR, | Arc Relative |
| AT, | Absolute Arc Three Point |
| CI, | Circle |
| EA, | Edge Rectangle Absolute |
| EP, | Edge Polygon |
| ER, | Edge Rectangle Relative |
| EW, | Edge Wedge |
| FP, | Fill Polygon |
| PA, | Plot Absolute |
| PD, | Pen Down |
| PE, | Polyline Encoded |
| PR, | Plot Relative |
| RA, | Fill Rectangle Absolute |
| RR, | Fill Rectangle Relative |
| RT, | Relative Arc Three Point |
| WG, | Fill Wedge |

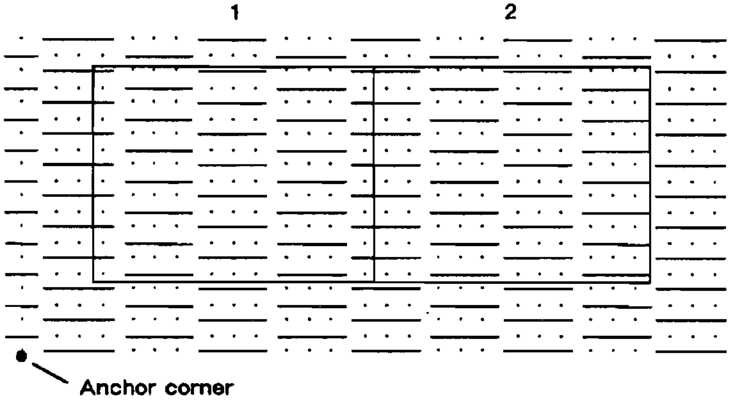
## Using Fill Types

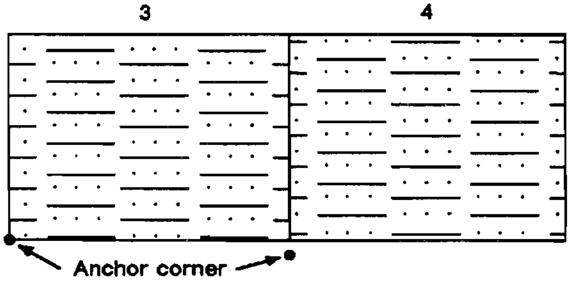
Using the fill type (FT) instruction adds detail to your plots and increases their visual effectiveness. You can fill any closed geometric shape that you've drawn using fill rectangle absolute (RA), fill rectangle relative (RR), fill wedge (WG), or fill polygon (FP) instructions. Fill types can be solid, parallel lines (hatched), cross-hatched, or shaded as shown in the following illustration. You can also create your own fill type using the raster fill definition (RF) instruction.



When you use hatched or cross-hatched 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. Note that solid and shaded fill types use less of your plotter's disc space than hatched and cross-hatched fill types.

All fill types have an *anchor corner,* the starting point of the fill's pattern. Its default location is in the lower-left corner of the hard-clip limits. Conceptually, the fill type replicates out from the anchor corner in the X- and Y-directions, as shown in the following illustration. Figures are filled by that portion of the fill type resident to the area, refer to rectangles 1 and 2.





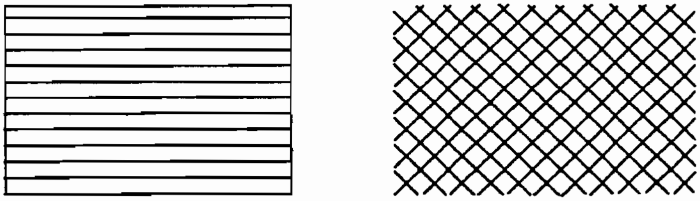
Use the anchor corner (AC) instruction to position the fill type to the figure. Rectangle 3 has an anchor corner set in its the lower-left corner. Rectangle 4 has an anchor corner set below the lower-left corner to alter the pattern's position and provide contrast to the adjacent figure.

## Filling Rectangles

Use the fill rectangle absolute and relative instructions (RA and RR) to fill rectangles. Like the edge rectangle instructions, the fill rectangle absolute and relative instructions start at the current pen location; you specify the diagonally opposite corner. The rectangular shape is formed by the fill pattern.

If you use an open fill type, you may want to also edge (or outline) the rectangle for a cleaner edge. The following program draws two filled rectangles: one edged and one not.

```
10 ´Insert configuration statement here
20 PRINT #1, "INPS5000,7000SP1PA0,0FT3RR1500,1000"
30 PRINT #1, "ER1500,1000PR2000,0FT4,100,45RR1500,1000PG;"
40 END
```

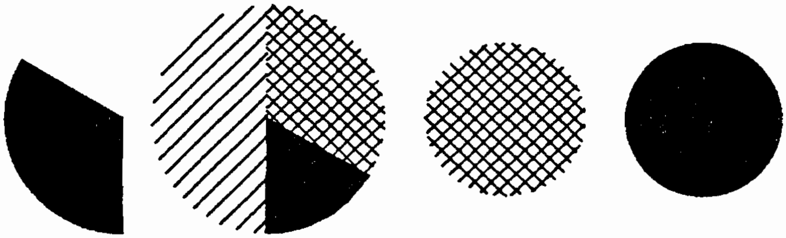


## Filling Wedges and Circles

Use the fill wedge (WG) instruction to fill wedges and circles. Like the edge wedge instruction, the fill wedge instruction starts at the current pen location; you specify the radius, the start angle, and the sweep angle. To fill a circle, simply specify a 360-degree sweep angle.

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

```
10 ´Insert configuration statement here
20 PRINT #1, "INPS5000,7000SP1"
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, "PA4500,2500FTWG400,0,360PG;"
90 END
```



## 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 | -67 108 863 to 67 108 863 | — |

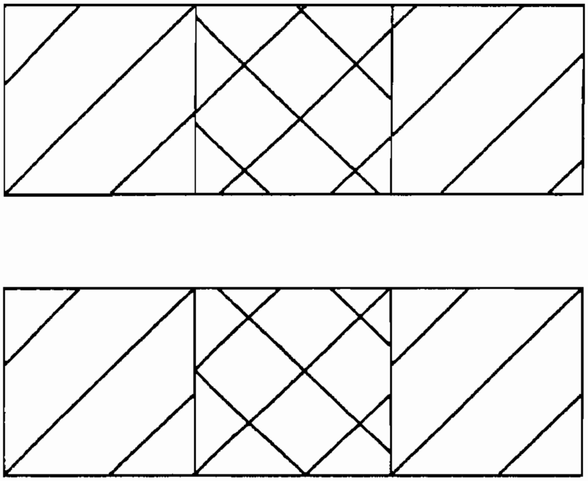
**REMARKS:** The 'anchor corner' is the point at which any fill pattern starts. Setting the anchor corner guarantees that a corner point of the selected fill pattern will be at the specified coordinate, aligned vertically and horizontally.

- **No Parameters** — Defaults the anchor corner to the lower-left corner of the hard-clip limits. Equivalent to *(AC0,0)*.

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

**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, "INPS5000,7000SP1PA3000,3000"
30 PRINT #1, "FT3,400,45RR1000,1000ER1000,1000"
40 PRINT #1, "PR1000,0FT4RR1000,1000ER1000,1000"
50 PRINT #1, "PR1000,0FT3RR1000,1000ER1000,1000"
60 PRINT #1, "PA3000,1500AC3000,1500RR1000,1000ER1000,1000"
80 PRINT #1, "PA4000,1500AC4000,1500FT4RR1000,1000
               ER1000,1000"
90 PRINT #1, "PA5000,1500AC5000,1500FT3RR1000,1000
               ER1000,1000"
100 PRINT #1, "PC;"
110 END
```



**RELATED INSTRUCTIONS:**

FT, Fill Type
RA, Fill Rectangle Absolute
RF, Raster Fill Type
RR, Fill Rectangle Relative
WG, Fill Wedge

## FT, Fill Type

**USE:** Selects the shading pattern used to fill polygons (FP), rectangles (RA or RR), or wedges (WG). Use FT to enhance plots with solid fill, parallel lines (hatching), cross-hatching, 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 and 2 | clamped real | type dependent* | type dependent* |

*** Refer to the table following the parameter descriptions.**

**REMARKS:** There are six forms of fill type. The type parameter tells the plotter which form you are using. The plotter uses the current settings when optional parameters are omitted.

- **No Parameters** — Defaults the fill type to solid fill. Equivalent to (*FT1*).

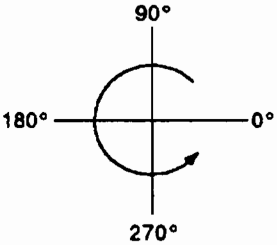
- **Type** — Selects the fill pattern. The table below lists the parameter values and corresponding fill types.

- **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 and 2 | solid bidirectional | ignored | ignored |
| 3 | hatched (parallel lines) | spacing of lines | angle of lines |
| 4 | cross-hatched | spacing of lines | angle of lines |
| 10 | shading | shading level | ignored |
| 11 | user-defined raster | raster-fill index | ignored |

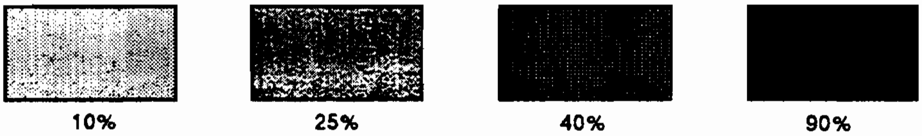
For fill types 3 and 4, the option1 parameter specifies the distance between the lines in the fill. This distance is specified in current units measured along the X-axis. Option1 must be a positive number (if zero, then solid fill is used). The default spacing is 1% of the diagonal distance from P1 to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

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 X-axis, as shown in the following illustration (0 and 180 are horizontal; 90 and 270 are vertical). The first set of lines for cross-hatched fill types are drawn at the specified angle and the next set drawn at that angle plus 90 degrees.



Types 3 and 4 use the current line type defined by the line type, width, and attributes.

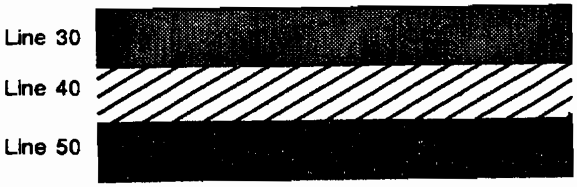
For fill type 10, the option1 parameter specifies the level of shading. The level is specified as a percentage*. The following illustration shows four levels of shading and the specified percentage.



For fill type 11, the option1 parameter selects the corresponding user-defined raster fill. Refer to the raster fill type (RF) instruction. If you have not issued a RF instruction, the plotter will use solid fill.

**EXAMPLE:**

```
10 'Insert configuration statement here
20 PRINT #1, "INPS5ØØØ,7ØØØSP1PA2ØØØ,2ØØØ"
3Ø PRINT #1, "FTRR25ØØ,3ØØER25ØØ,3ØØ"
4Ø PRINT #1, "PRØ,3ØØFT3,8Ø,3ØRR25ØØ,3ØØ"
5Ø PRINT #1, "PRØ,3ØØFT1Ø,36RR25ØØ,3ØØER25ØØ,3ØØ"
6Ø PRINT #1, "PG;"
7Ø END
```



**RELATED INSTRUCTIONS:**

LA, Line Attributes
LT, Line Type
PT, Pen Thickness
PW, Pen Width
RF, Raster Fill Definition

*The plotter uses 16 predefined levels of shading equally divided between 0 and 100%. Values are rounded to the nearest predefined level.

## LA, Line Attributes

**USE:** Specifies how line ends and joins (corners) are phyisically shaped. Use LA when drawing lines greater than 0.8 mm to define their appearance.

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

| Parameter | Format | Range | Default |
|---|---|---|---|
| kind | clamped integer | 1 through 3 | 1 |
| value | clamped integer | kind dependent* | kind dependent* |

* Refer to the table following the parameter descriptions.

**REMARKS:** 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. 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 specifed 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 | 0 to 32 767.9999** | 5 (default, refer to description under *Miter Limit*) |

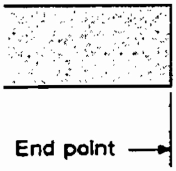
*Lines with a width of 0.80 mm or less always have rounded caps, regardless of the current attribute setting.

**Values less than 1.1 are set to 1.1, but do not cause an error.

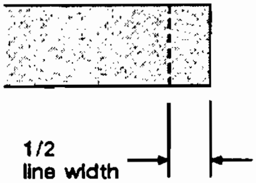
**NOTE:** Labels are always drawn with triangular ends and joins. ■

## Line Ends

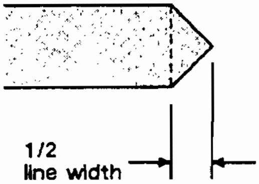
The value you specify for line ends determines how the ends of line segments are shaped. The following illustration describes the four types of line ends.



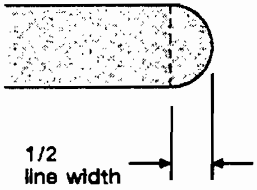
**Butt ends (1)** Terminate at the end point.



**Square ends (2)** Terminate one half line width beyond the end points.



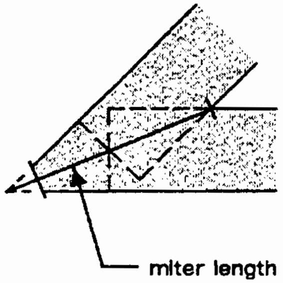
**Triangular ends (3)** Terminate one half line width beyond the end points.



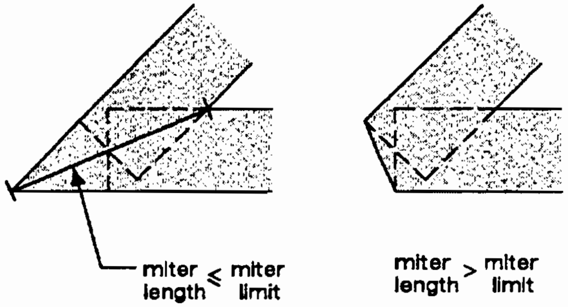
**Round ends (4)** Terminate in semicircular having a diameter equal to the current line width. Round ends use more disc space than other ends.

## Line Joins

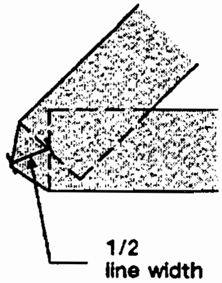
The value you specify for the line joins attribute determines how intersecting line ends (corners) are shaped. The following illustration describes the five types of line joins. If the first and last points of a series of lines are the same, they join according to the current line join and miter limit.



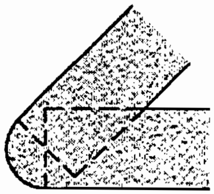
**Mitered join (1)**
Formed by two lines extending from the outer edge of each vector until they meet. The miter limit applies to this join.



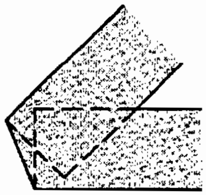
**Mitered/beveled join (2)** Formed by two lines extending from the outer-edge of each vector until they meet. If the miter length exceeds the miter limit, a beveled join is used.



**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.

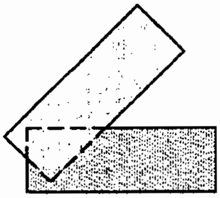


**Rounded join (4)**
Formed by an arc with a diameter equal to the current line width. Round joins use more disc space than other joins.

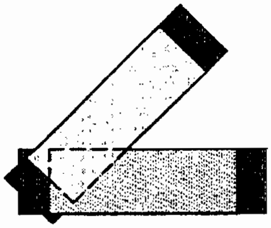


**Beveled join (5)**
Formed by a line connecting the outer edge of one vector to the outer edge of the other vector.

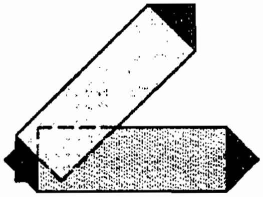
When you select 'no join', the currently selected line ends for the two lines merely overlap. Refer to the following illustration.



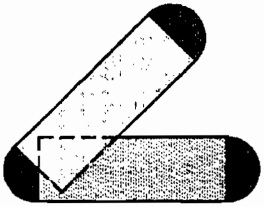
No Join
Butt End



No Join
Square End



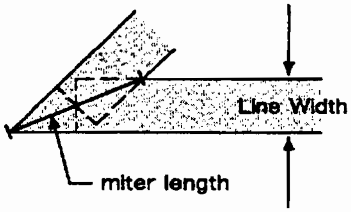
No Join
Triangular End



No Join
Round End

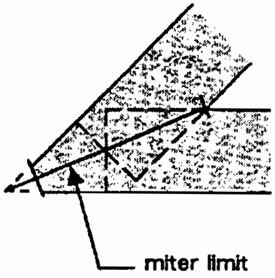
## Miter Limit

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.

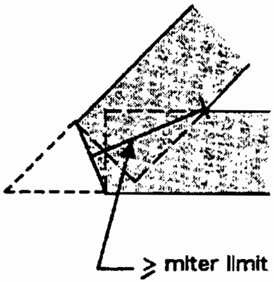


$$\text{Miter Limit} = \frac{\text{Miter Length}}{\text{Line Width}}$$

When the miter 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

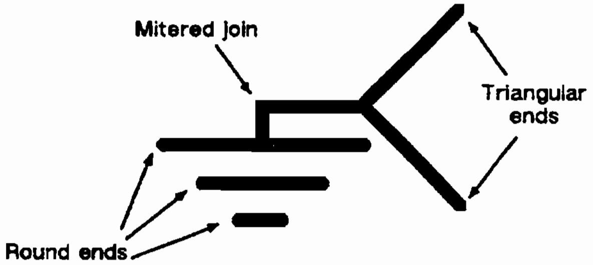


Maximum clipped mitered join (beveled join)

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, "INPS5000,7000SP1PA4000,3000"
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, "PG;"
90 END
```



**RELATED INSTRUCTIONS:**

LT, Line Type
PW, Pen Width
UL, User-Defined Line Type

## LT, Line Type

**USE:** Specifies the line pattern to be used when drawing lines. Use LT to distinguish lines and enhance your plot.

**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 set by the line attribute (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.

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

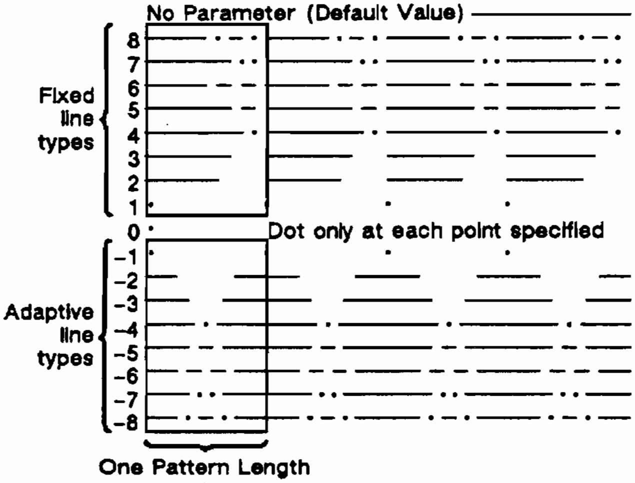
- **Line Type** — Produces the corresponding line pattern. Line patterns can be of fixed or adaptive type.

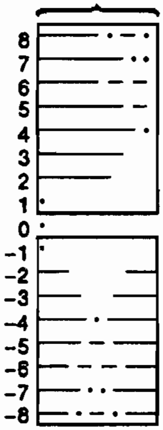
  **Positive line types (1 — 8)** are fixed line types and use the specified pattern length to draw lines. Any unused part of the pattern (residue) is carried over into the next line, or CI, EA, EP, ER, EW, FP, PM, RA, RR, or WG instruction. The current residue is cleared by a CP, DF, IN, IP, IR, LB, LT (except (*LT*) and (*LT99*)), PG, PU, SC, and UL.

  **A zero line type (0)** draws only a dot at the X,Y coordinates for 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 line types. The pattern length is automatically adjusted so that each line contains one or more complete patterns.

Line patterns are composed of alternate pen down and pen up moves which are percentages of the pattern length (the first percentage is always pen down). The following illustration shows first the line type patterns, then gives the pattern percentages.





Pattern Percentages

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 8 | 50 | 10 | 0 | 10 | 10 | 10 | 0 | 10 | |
| 7 | 70 | 10 | 0 | 10 | 0 | 10 | | | |
| 6 | 50 | 10 | 10 | 10 | 10 | 10 | | | |
| 5 | 70 | 10 | 10 | 10 | | | | | |
| 4 | 80 | 10 | 0 | 10 | | | | | |
| 3 | 70 | 30 | | | | | | | |
| 2 | 50 | 50 | | | | | | | |
| 1 | 0 | 100 | | | | | | | |
| 0 | | | | | | | | | |
| -1 | 0 | 100 | 0 | | | | | | |
| -2 | 25 | 50 | 25 | | | | | | |
| -3 | 35 | 30 | 35 | | | | | | |
| -4 | 40 | 10 | 0 | 10 | 40 | | | | |
| -5 | 35 | 10 | 10 | 10 | 35 | | | | |
| -6 | 25 | 10 | 10 | 10 | 10 | 10 | 25 | | |
| -7 | 35 | 10 | 0 | 10 | 0 | 10 | 35 | | |
| -8 | 25 | 10 | 0 | 10 | 10 | 10 | 0 | 10 | 25 |

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. ■

If you are currently plotting with a solid line type (*LT*), you can use (*LT99*) to return to plotting with the previous line type. The plotter ignores (*LT99*) if you are currently plotting with a nonsolid line type. Sending a DF, IN, IP, IR, PG, SC, or UL instruction while plotting with a solid line type clears the previous line type and a subsequent (*LT99*) has no effect.

- **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 millimetres. You must specify a length greater than zero or the plotter ignores the instruction. If you don't specify a length, the plotter uses the last value specified.

- **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.

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

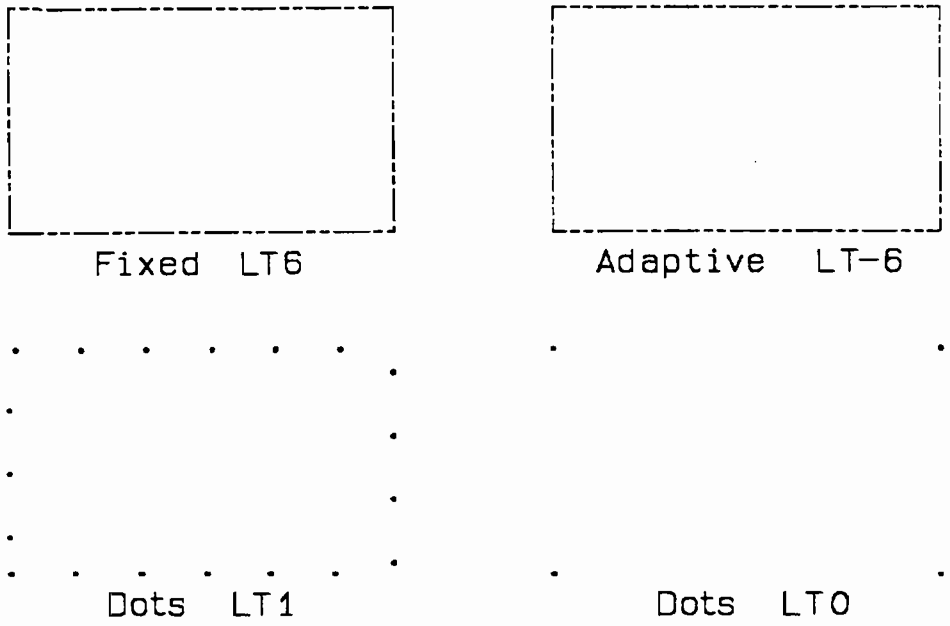
  When specified as a percentage, the pattern length changes along with changes in P1 and P2.

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

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

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

**EXAMPLE:**



**RELATED INSTRUCTIONS:**

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

## RA, Fill Rectangle Absolute

**USE:** Defines and fills a rectangle using absolute coordinates. Use RA to fill rectangular shapes in plots. To outline a rectangle using absolute coordinates, use the EA instruction.

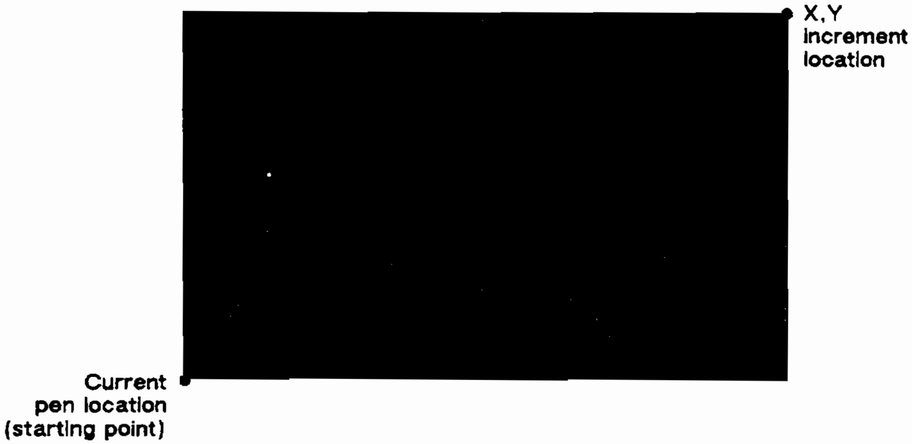
**SYNTAX:** RA*X,Y(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates | current units | -67 108 863 to 67 108 863 | — |

**REMARKS:** The RA instruction defines and fills a rectangle using the current pen, the current line and fill types, and absolute plotting. 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.

  **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 either two opposite corners. ■

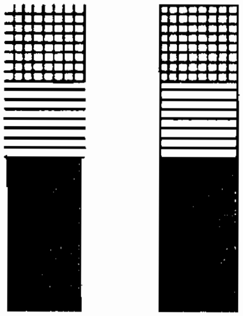


The only difference between the RA instruction and the EA instruction is that the RA instruction produces a filled rectangle; and EA, an outlined one.

The RA instruction clears the polygon buffer and then uses it to define the rectangle before drawing. Refer to *Using the Polygon Buffer* in Chapter 6 for more information.

**EXAMPLE:** This program uses RA with three different fill types (refer to the FT instruction) to create rectangles such as those you might use in a PERT chart. The rectangles in the right bar are edged using the EA instruction.

```
10  'Insert configuration statement here
20  PRINT #1, "INPS5000,7000SP1PA400,400RA800,1200"
30  PRINT #1, "PA400,1200FT3,50RA800,1600"
40  PRINT #1, "PA400,1600FT4RA800,2000"
50  PRINT #1, "PA1200,400FTRA1600,1200EA1600,1200"
60  PRINT #1, "PA1200,1200FT3,50RA1600,1600EA1600,1600"
70  PRINT #1, "PA1200,1600FT4RA1600,2000EA1600,2000"
80  PRINT #1, "PG;"
90  END
```



**RELATED INSTRUCTIONS:**

EA, Edge Rectangle Absolute
EP, Edge Polygon
ER, Edge Rectangle Relative
FT, Fill Type
LT, Line Type
RF, Raster Fill Definition
RR, Fill Rectangle Relative

**SPECIAL ERRORS:**

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

## RF, Raster Fill Definition

**USE:** Defines a rectangular pattern that may be used as area fill. Use RF to create your own fill types.

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

| Parameter | Format | Range | Default |
|---|---|---|---|
| index | clamped integer | 1 to 8 | 1 (solid) |
| width | clamped integer | 8, 16, 32, or 64 | — |
| height | clamped integer | 8, 16, 32, or 64 | — |
| pen number | integer | 0 to 67 108 863 | — |

**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 uses the pattern in effect when rasterization begins. ■

- **No Parameters** - Defaults all raster fill patterns to solid fill.
- **Index** - Specifies the index number of the pattern being defined. Eight patterns can exist concurrently.

  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 two, from eight to 64 pixels.

- **Pen Number** - Represents a pixel in the pattern being defined and indicates its color.

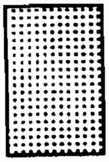
  **0** White.

  **>0** Black.

  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 (zero). Patterns are printed in rows parallel to the X-axis.

**EXAMPLE:**

```
10  'Insert configuration statement here
20 A$=",Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø"
3Ø B$=",Ø,Ø,Ø,Ø,Ø,1,1,1,1,1,1,Ø,Ø,Ø,Ø,Ø"
4Ø PRINT #1, "INPS5ØØØ,7ØØØSP1PA35ØØ,25ØØ"
5Ø PRINT #1, "RF2,16,16"
6Ø   FOR I = 1 TO 5
7Ø   PRINT #1, A$
8Ø   NEXT I
9Ø  FOR M = 1 TO 6
1ØØ  PRINT #1, B$
11Ø  NEXT M
12Ø  FOR I = 1 TO 5
13Ø  PRINT #1, A$
14Ø  NEXT I
15Ø PRINT #1, ";"
16Ø PRINT #1, "FT11,2RR5ØØ,8ØØEP"
2ØØ PRINT #1, "PG;"
21Ø END
```



**RELATED INSTRUCTIONS:** FT, Fill Type

## RR, Fill Rectangle Relative

**USE:** Defines and fills a rectangle using relative coordinates. Use RR to fill rectangular shapes in plots. To outline a rectangle using relative coordinates, use the ER instruction.

**SYNTAX:** RR*X,Y(;)*

| Parameter | Format | Range | Default |
|---|---|---|---|
| X,Y increments | current units | -67 108 863 to 67 108 863 | — |

**REMARKS:** The RR instruction defines and fills a rectangle using the current pen, the current line and fill types, and relative plotting. 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.

  **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 two opposite corners. ■

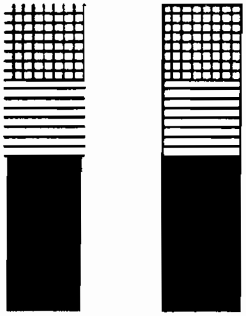
X,Y
Increment
location
Current
pen location
(starting point)

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

The RR instruction clears the polygon buffer and then uses it to define the rectangle before drawing. A rectangle requires enough buffer space to hold five points. The default buffer size is sufficient. Refer to *Using the Polygon Buffer* in Chapter 5 for more information.

**EXAMPLE:** This program uses RR with three different fill types (refer to the FT instruction) to create rectangles such as those you might use in a bar chart. The rectangles in the right bar are edged using the ER instruction.

```
10 'Insert configuration statement here
20 PRINT #1, "INPS5000,7000SP1PA400,400RR400,800"
30 PRINT #1, "PR0,800FT3,50RR400,800"
40 PRINT #1, "PR0,400FT4RR400,400"
50 PRINT #1, "PA1200,400FTRR400,800ER400,800"
60 PRINT #1, "PR0,800FT3,50RR400,400ER400,400"
70 PRINT #1, "PR0,400FT4RR400,400ER400,400"
80 PRINT #1, "PG;"
90 END
```



**RELATED INSTRUCTIONS:**

EA, Edge Rectangle Absolute
ER, Edge Rectangle Relative
RA, Fill Rectangle Absolute

**SPECIAL ERRORS:**

| Condition | Error | Plotter Response |
|---|---|---|
| polygon buffer overflow | 7 | ignores excess data |

## UL, User-Defined Line Type

**USE:** Creates line types by specifying gap patterns.

**SYNTAX:** UL index *(,gap1,...gap20;)*
or
UL*(;)*

| Parameter | Format | Range | Default |
|---|---|---|---|
| index | clamped integer | \|1 through 8\| | — |
| gaps | clamped real | 0 to 32 767.9999 | default line types |

**REMARKS:** The UL instruction allows you to define and store your own line types. The instruction does not itself select a line type. Use the LT instruction to select the pattern you've defined using UL.

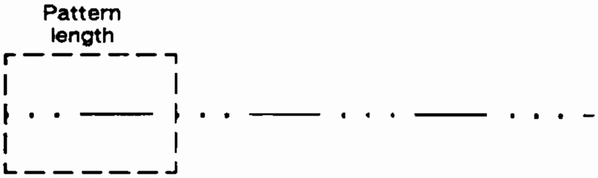
- **No Parameters** — Defaults all line types, refer to the LT instruction.
- **Index** — Identifies the line type to be redefined. Specifying an index without gap parameters sets the line type identified by the index to its default pattern.

  The index 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.
- **Gaps** — Specify alternate pen-down and pen-up stretches in the line type pattern; odd numbered gaps are pen-down moves, even numbered gaps are pen-up moves. The first gap is a pen-down move. Gap values are converted to percentages of the LT instruction's pattern length parameter.

  A maximum of 20 gaps are allowed for each user-defined line type. Gap values must be positive; a gap value of zero produces a dot if specified for an odd numbered gap. The sum of the gap parameters must be greater than zero.

**EXAMPLE:**

```
10 'Insert configuration statement here
20 PRINT #1, "INPS5000,7000SP1PA4000,3000"
30 PRINT #1, "UL8,0,15,0,15,0,15,40,15"
40 PRINT #1, "LT8,10PU2000,2500PD5000,2500"
80 PRINT #1, "PG;"
90 END
```



**RELATED INSTRUCTIONS:** LT, Line Type

**SPECIAL ERRORS:**

| Condition | Error | Plotter Response |
|---|---|---|
| sum of gap parameters equals zero | 3 | ignores instruction |

## 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 tolerance;)*

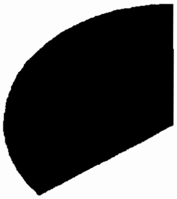
| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| radius | current units | –67 108 863 to 67 108 863 | — |
| start angle | clamped real | –360 to 360 degrees | — |
| sweep angle | clamped real | –360 to 360 degrees | — |
| chord tolerance | | | |
| chord angle* | clamped real | 0.5 to 90 degrees | 5 degrees |
| chord deviation | current units | 0 to 67 108 863 | 5 degrees |

*Chord angle is the default interpretation of chord tolerance.

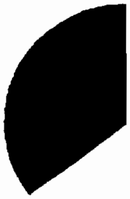
**REMARKS:** The WG instruction defines and fills a wedge using the current pen, fill type, and line types. The WG instruction includes an automatic pen down. When the instruction is completed, the original pen location and up/down position are restored.

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

Always use isotropic scaling in any plot that draws wedges. Refer to the discussion of scaling in Chapter 2 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.

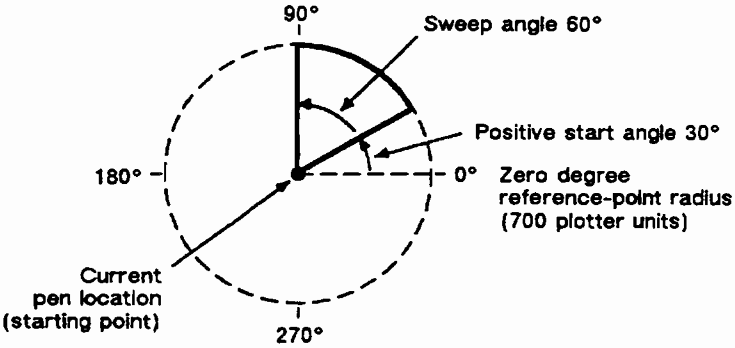
The radius is interpreted in current units: as user units when scaling is on; as plotter units when scaling is off. The sign of the radius determines the location of the zero-degree reference point. The illustration following the parameter descriptions shows the location of the zero-degree reference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

- **Start Angle** — Specifies the beginning point of the arc as the number of degrees from the zero-degree reference point. A positive start angle positions the radius counterclockwise (the direction from the positive X-axis toward the positive Y-axis) from the zero-degree reference point; a negative start angle positions the radius clockwise from the zero-degree reference point.

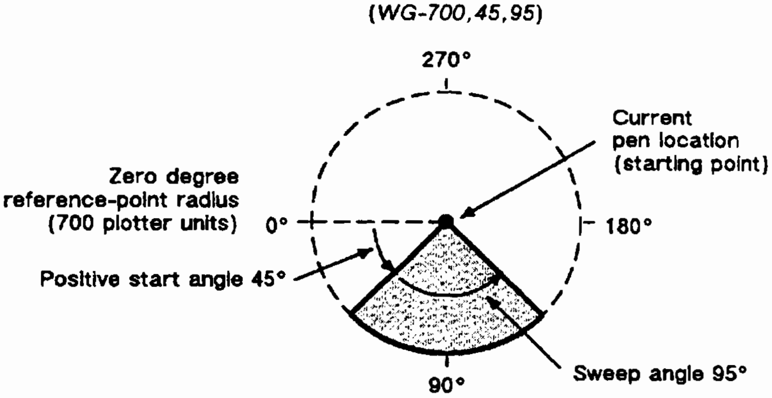
- **Sweep Angle** — Specifies the number of degrees through which the arc extends. A positive sweep angle defines the arc counterclockwise; a negative sweep angle defines the arc clockwise. If you specify a sweep angle greater than 360 degrees, a 360-degree angle is used.

- **Chord Tolerance** — Specifies the chord tolerance used to define the arc. The default is a chord angle of 5 degrees. Refer to the *Chords and Chord Tolerance* discussion at the in Chapter 4 or the chord tolerance mode (CT) instruction for information on setting and determining chord tolerance.

*(WG700,30,60)*


90°
Sweep angle 60°
Positive start angle 30°
180°
0°
Zero degree reference-point radius (700 plotter units)
Current pen location (starting point)
270°

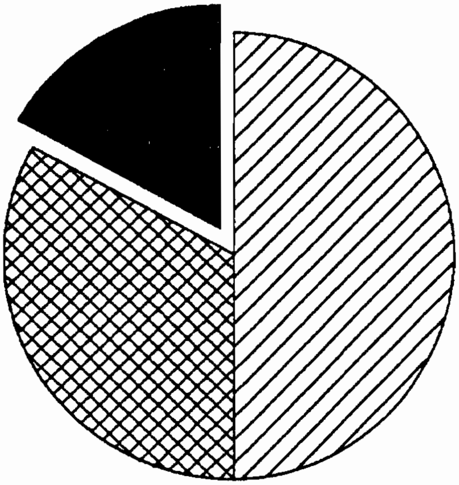
*Positive Radius*


(WG-700,45,95)
270°
Current pen location (starting point)
Zero degree reference-point radius (700 plotter units)
0°
180°
Positive start angle 45°
Sweep angle 95°
90°

*Negative Radius*

**EXAMPLE:**

```
10  ´Insert configuration statement here
20  PRINT #1, "INPS5ØØØ,7ØØØSP1SC-3ØØØ,3ØØØ,-2ØØØ,2ØØØ,1"
3Ø  PRINT #1, "PAØ,ØFT3,75,45WG-1ØØØ,9Ø,18Ø"
35  PRINT #1, "EW-1ØØØ,9Ø,18Ø"
4Ø  PRINT #1, "FT4,6Ø,45WG-1ØØØ,33Ø,12Ø"
45  PRINT #1, "EW-1ØØØ,33Ø,12Ø"
5Ø  PRINT #1, "PR-6Ø,11ØFT1,Ø,Ø"
6Ø  PRINT #1, "WG-1ØØØ,27Ø,6Ø"
65  PRINT #1, "EW-1ØØØ,27Ø,6Ø"
7Ø  PRINT #1, "PG;"
8Ø  END
```



**RELATED INSTRUCTIONS:**

CT, Chord Tolerance Mode
EP, Edge Polygon
EW, Edge Wedge
FT, Fill Type
LA, Line Attributes
LT, Line Type
PW, Pen Width
SC, Scale

**SPECIAL ERRORS:**

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

6

# Using Polygon Mode

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

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

The following instructions are described at the end of the chapter.

| | |
|---|---|
| EP, | Edge Polygon |
| FP, | Fill Polygon |
| PM, | Polygon Mode |

## Drawing Polygons

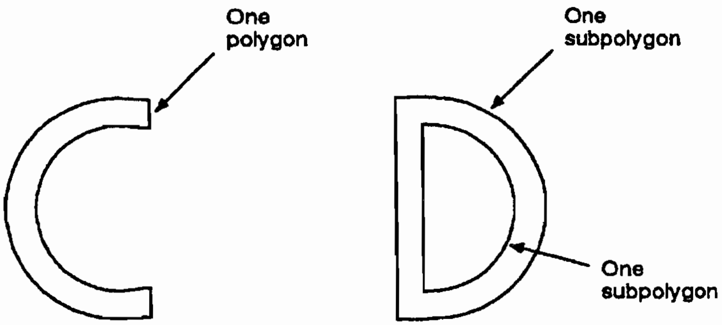
A polygon is a closed sequence of connected line segments (which may cross each other). Drawing polygons requires the use of *polygon mode*. The polygon mode (PM) instruction tells the plotter to store subsequent instructions and coordinates in the polygon buffer before plotting the shape. (Rectangles, circles, and wedges are exceptions since they have their own drawing instructions).

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.

| | |
|---|---|
| PM1/PM2, | Polygon Mode |
| PA, | Plot Absolute |
| PE, | Polyline Encoded |
| PR, | Plot Relative |
| PU, | Pen Up |
| PD, | Pen Down |
| AA, | Arc Absolute |
| AR, | Arc Relative |
| AT, | Absolute Arc Three Point |
| CI, | Circle |
| CT, | Chord Tolerance Mode |
| RT, | Relative Arc Three Point |

## 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.

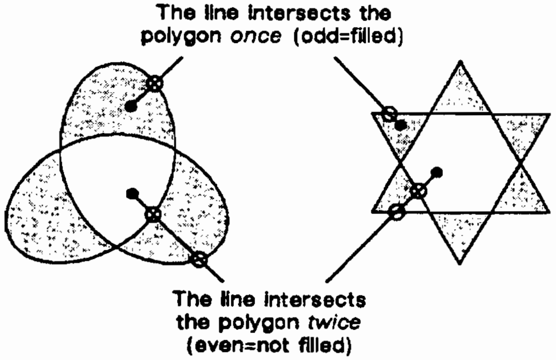
To create a series of subpolygons (e.g., the letter D), move the pen to the starting location of the first subpolygon, then enter polygon mode. Define the outer shape of the letter D using the appropriate instructions and coordinates, then enter subpolygon mode. Define the inner shape of the D, then exit polygon mode. Now draw the subpolygons using either the edge polygon (EP) or fill polygon (FP) instruction.

For more information on entering and exiting polygon mode, refer to the polygon mode (PM) instruction at the end of this chapter. Note that you can define points with the pen up or down. However, the EP instruction only draws 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.

You can use all of the output instructions while in polygon mode.* The plotter does not store the output instructions in the polygon buffer; they are executed immediately. If you initialize the plotter while in polygon mode, the plotter exits polygon mode and begins executing the next instructions. You must exit polygon mode to execute other HP-GL/2 instructions.

## 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.


The line intersects the polygon *once* (odd=filled)
The line intersects the polygon *twice* (even=not filled)

*Single polygon* *Two subpolygons*

* The output instructions are OE, OH, OI, OP, and OS. (Refer to Chapters 9 and 10.)

### 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. This can change your current pen location and the placement of the circle in your polygon resulting 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. ■

## 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 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

The plotter has 16 900 bytes of available memory. The plotter has two buffers: the polygon buffer and the downloadable character buffer (discussed in Chapter 8). The plotter's available memory is allocated to the downloadable character buffer as needed, the rest goes to the polygon buffer.

## Approximating Polygon Buffer Use

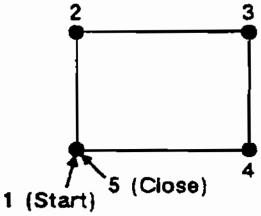
You can use the following formula to estimate how much buffer space a polygon consumes. Each point in a polygon uses 8 bytes. (If your program contains downloadable characters, refer to the download character (DL) instruction in Chapter 8 to estimate how many bytes are being used by the downloadable character buffer. Subtract this amount from the available memory, 16 900 bytes, to determine the size of the polygon buffer.)

# of points in polygon × 8 = buffer space consumed by polygon

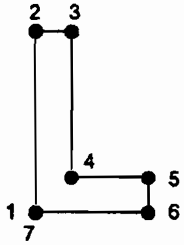
For example, dividing 16 900 by 8 equals 2112.5. If you allow some extra for fill types and such, you can estimate that a polygon with up to 1500 points easily fits in the polygon buffer. The following explains how to count the number of points in a polygon.

### *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.

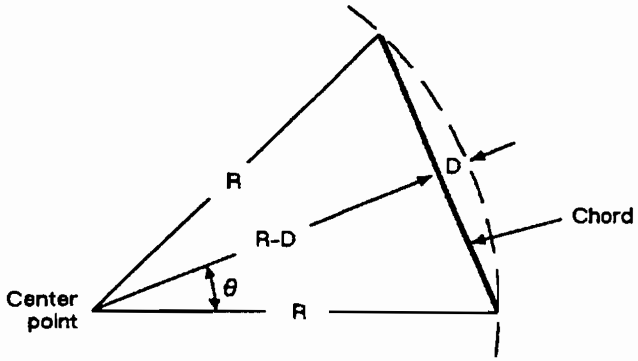
$$\# \text{ of points} = \frac{\text{arc angle (degrees)}}{\text{chord angle (degrees)}} + 1$$

Using this formula, a full circle with the default chord angle of 5 degrees consists of 73 points (360/5 + 1 = 73) and a 45-degree arc with an 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 one: 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 above.

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.

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

Because you want the angle that created the whole chord, not the bisected chord, the chord angle, then, is twice $\theta$. 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 when the radius is 1000 and the deviation distance is 20, the angle $\theta$ 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.

## 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 rectangle and wedge instructions (EA, ER, EW, RA, RR, and WG).

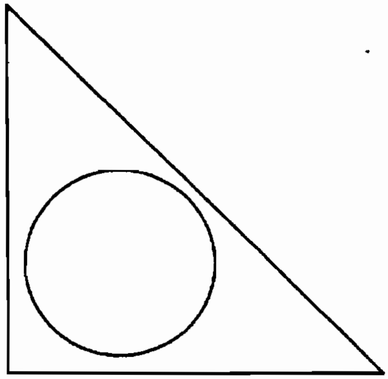
**SYNTAX:** EP(;)

**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 only edges 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, "INPS5000,7000SP1PA2000,0"
30  PRINT #1, "PM0PD0,2000,0,0,2000,0PM1"
40  PRINT #1, "PU600,600CI500PM2"
50  PRINT #1, "EPPG;"
70  END
```



**RELATED INSTRUCTIONS:**

EA, Edge Rectangle Absolute
ER, Edge Rectangle Relative
EW, Edge Wedge
LA, Line Attributes
LT, Line Types
PM, Polygon Mode

## 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).

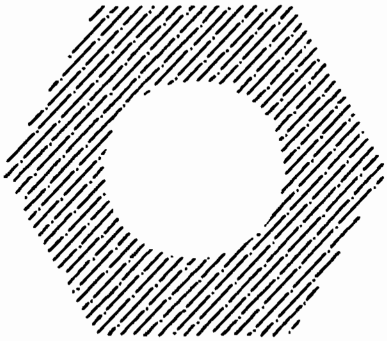
**SYNTAX:** FP(;)

**REMARKS:** The FP instruction fills any polygon that is currently in the polygon buffer. This includes wedges and rectangles defined using the EA, ER, EW, WG, RA, and RR 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 both with the pen down and the pen up. 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.

```
10  'Insert configuration statement here
20  PRINT #1, "INPS6ØØØ,8ØØØSP1PA15ØØ,15ØØ"
30  PRINT #1, "PMØCI1ØØØ,6ØPA15ØØ,15ØØCI5ØØPM2"
40  PRINT #1, "LT4FT3,5Ø,45"
50  PRINT #1, "FPPG;"
60  END
```



**RELATED INSTRUCTIONS:**

EA, Edge Rectangle Absolute
EP, Edge Polygon
ER, Edge Rectangle Relative
EW, Edge Wedge
FT, Fill Type
LA, Line Attributes
LT, Line Type
PM, Polygon Mode
PW, Pen Width

**SPECIAL ERRORS:**

| Condition | Error | Plotter Response |
|---|---|---|
| previous PM, RA, RR, or WG instruction overflowed the polygon buffer | 7* | ignores instruction |

*Error actually occurs when buffer overflows. It is possible to define and edge a polygon, but not be able to fill it.

## PM, Polygon Mode Instruction

**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, and 2 | — |

**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 Parameters** — Clears the polygon buffer and enters polygon mode. Equivalent to (*PM0*).

- **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 paragraphs 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 lists these instructions.

| | |
|---|---|
| AA, | Arc Absolute |
| AR, | Arc Relative |
| AT, | Absolute Arc Three Point |
| CI, | Circle |
| CT, | Chord Tolerance Mode |
| DF, | Default Values |
| IN, | Initialize |
| PA, | Plot Absolute |
| PD, | Pen Down |
| PE, | Polyline Encoded |
| PM1/PM2, | Polygon Mode |
| PR, | Plot Relative |
| PU, | Pen Up |
| RT, | Relative Arc Three Point |

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

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

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 buffer, but are executed immediately. You must exit polygon mode to execute other graphics instructions.

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 only draws 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 defined.

NOTE: For best results, define points with the pen up if you plan to fill the polygon since FP has an automatic pen down, refer to *Pen Position* in Chapter 3. ■

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 can end up with an unexpected polygon by not closing it yourself.

### (PM1)

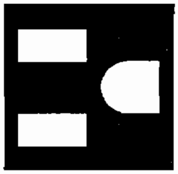
Use (*PM1*) to close the current polygon (or subpolygon) and remain in polygon mode. When you use (*PM1*), the point after (*PM1*) becomes the first point of the next subpolygon. This move is *not* used as a boundry when filling a polygon with FP. When drawing the polygon, the pen will always move to 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. For best results, include a pen up (PU) instruction *before* a (*PM2*). Refer to *Pen Position* in Chapter 3.

**EXAMPLE:** The following draws the surface area of a 3-prong receptacle as a series of subpolygons, fills and edges it using the FP and EP instructions, respectively.

```
10  'Insert configuration statement here
20  PRINT #1, "INPS6000,7000SP1PA2000,2000"
30  PRINT #1, "PM0"
40  PRINT #1, "PD3000,2000,3000,3000"
50  PRINT #1, "PD2000,3000,2000,2000"
60  PRINT #1, "PM1"
70  PRINT #1, "PD2080,2160,2480,2160,2480,2340,2080,2340,
                2080,2160"
80  PRINT #1, "PM1"
90  PRINT #1, "PD2080,2660,2480,2660,2480,2840,2080,2840,
                2080,2660"
100 PRINT #1, "PM1"
110 PRINT #1, "PD2920,2340,2920,2660,2720,2660"
120 PRINT #1, "AA2720,2500,180PD2920,2340"
130 PRINT #1, "PM2FPEPPG;"
140 END
```



**RELATED INSTRUCTIONS:** EP, Edge Polygon
FP, Fill Polygon

**SPECIAL ERRORS:**

| Condition | Error | Plotter Response |
|---|---|---|
| invalid instruction used in polygon mode | 1 | ignores illegal instruction |
| buffer overflow | 7 | ignores overflowing points |

# 7

# Labeling Your Plots

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

- Draw and position labels.
- Work with the character plot cell.
- Change label size, slant, and direction.
- Use variables in labels.

The following instructions are described in this chapter.

| | |
|---|---|
| CP, | Character Plot |
| DI, | Absolute Direction |
| DR, | Relative Direction |
| DT, | Define Label Terminator |
| DV, | Define Variable Text Path |
| ES, | Extra Space |
| LB, | Label |
| LO, | Label Origin |
| SI, | Absolute Character Size |
| SL, | Character Slant |
| SM, | Symbol Mode |
| SR, | Relative Character Size |
| TD, | Transparent Data |

## Using and Terminating Labels

Use labels to create text charts or to 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.

These techniques are illustrated using the plotter's default character set (Roman8). All of the instructions in this chapter are valid with the plotter's other character sets and fonts. Chapter 8 discusses the plotter's other character sets and fonts.

## Terminating Labels

The label (LB) instruction tells the plotter to draw everything following the instruction, rather than interpreting 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.

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 must use the label terminator, or the plotter will label the rest of your program on your plot. You can change the label terminator using the define label terminator (DT) instruction. Read the following section for more information on inserting ASCII control characters in your labels.

## Carriage Returns and Line Feeds

You can also insert carriage returns and line feeds in your labels. Carriage returns and line feeds are also 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.

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 your system and language requires.

Check the table in Appendix B for the complete list of the plotter's responses to ASCII control characters. Here are some common CHR$ equivalents.

| | |
|---|---|
| CHR$(3) | produces **ETX** (end-of-text) |
| CHR$(10) | produces **LF** (line feed) |
| CHR$(13) | produces **CR** (carriage return) |

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 combinations and their equivalents.

| | |
|---|---|
| CONTROL and C | produces ETX |
| CONTROL and J | produces LF |
| CONTROL and M | produces CR |

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)
```

or

```
"LBLabel"+CHR$(13)+CHR$(10)+"return"+CHR$(3)
```

The following shows equivalent instructions using the keyboard.

"LBLabel$^{E}{}_{X}$"

"LBLabel $^{C}{}_{R}{}^{L}{}_{F}$return$^{E}{}_{X}$"

Both methods produce the following labels.

```
Label          Label

               return
```

## The Carriage Return Point

The current pen location (before being adjusted by a prior LB instruction) is the carriage return point. If the plotter encounters a carriage return (CR, CHR$(13), or *CP*), the pen moves to the carriage return point, adjusted down by any line feeds.

The carriage return point is updated to the current pen location by these instructions: AA, AR, AT, CP, DF, DI, DR, DV, IN, LO, PA, PE, PR, RO, and RT. A PR or PU instruction with parameters also updates the carriage-return point. The LB instruction does *not* update the carriage return point, just 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.

## 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. 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.

Variables 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.

- **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.

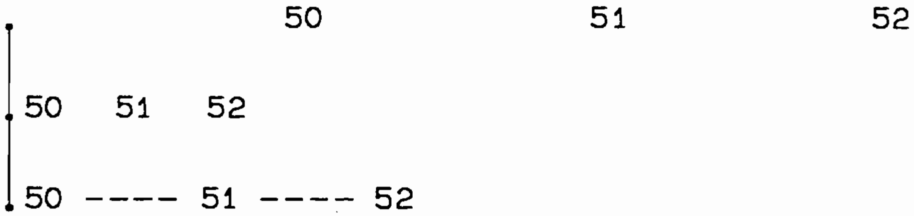
- **Semicolons** (in BASIC) typically suppress the extra blank spaces and are recommended for use in labels. However, in many computer languages, numbers will still be printed with a blank space in front for the sign and one after for separation. (Refer to the table of control characters in Appendix B for the backspacing control characters.) 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.

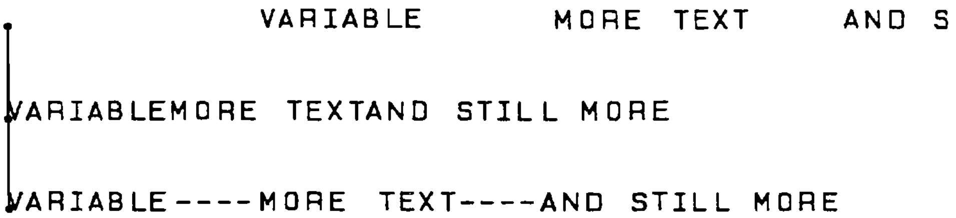
### Numeric Variables

```
1Ø  'Insert configuration statement here
2Ø  PRINT #1, "INPS5ØØØ,7ØØØSP1SI.187,.269PA15ØØ,3ØØØ"
3Ø  X=5Ø
4Ø  PRINT #1, "LB",X,X+1,X+2,+CHR$(3)
5Ø  PRINT #1, "PA15ØØ,25ØØ"
6Ø  Y=5Ø
7Ø  PRINT #1, "LB";Y;Y+1;Y+2;+CHR$(3)
8Ø  PRINT #1, "PA15ØØ,2ØØØ"
9Ø  Z=5Ø
1ØØ PRINT #1, "LB";Z;"----";Z+1;"----";Z+2;+CHR$(3)
11Ø PRINT #1, "PA15ØØ,3ØØØWG2Ø,Ø,36Ø"
115 PRINT #1, "PD15ØØ,25ØØWG2Ø,Ø,36Ø"
12Ø PRINT #1, "PD15ØØ,2ØØØWG2Ø,Ø,36Ø"
13Ø PRINT #1, "PG;"
14Ø END
```



**NOTE:** This version of GW-BASIC uses such a wide character field width with commas that the 'AND STILL MORE' is plotted off the right side of the page. ■

```
10  'Insert configuration statement here
20  PRINT #1, "INPS4ØØØ,5ØØØSP1SI.187,.269PAØ,3ØØØ"
3Ø  X$="VARIABLE"
4Ø  Y$="MORE TEXT"
5Ø  Z$="AND STILL MORE"
6Ø  PRINT #1, "LB",X$,Y$,Z$,+CHR$(3)
7Ø  PRINT #1, "PAØ,25ØØ"
8Ø  PRINT #1, "LB"X$Y$Z$+CHR$(3)
9Ø  PRINT #1, "PAØ,2ØØØ"
1ØØ PRINT #1, "LB"X$"----"Y$"----"Z$+CHR$(3)
11Ø PRINT #1, "PAØ,3ØØØWG2Ø,Ø,36Ø"
115 PRINT #1, "PDØ,25ØØWG2Ø,Ø,36Ø"
12Ø PRINT #1, "PDØ,2ØØØWG2Ø,Ø,36Ø"
13Ø PRINT #1, "PG;"
14Ø END
```



```
               VARIABLE       MORE TEXT     AND S

VARIABLEMORE TEXTAND STILL MORE

VARIABLE----MORE TEXT----AND STILL MORE
```

## 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** — Roman8 fixed-space vector font (character set 0 or 277). Refer to Chapter 8 for information about the various plotter character sets.
- **Label terminator** — ASCII end-of-text character **ETX** (decimal code 3). Refer to the DT instruction.
- **Label starting point** — Current pen location. Refer to the LO instruction.
- **Character size** — Width=0.285 cm, height=0.375 cm. Refer to the AD, SD, SI, and SR instructions.
- **Label direction** — Horizontal. Refer to the DI, DR, and DV instructions.

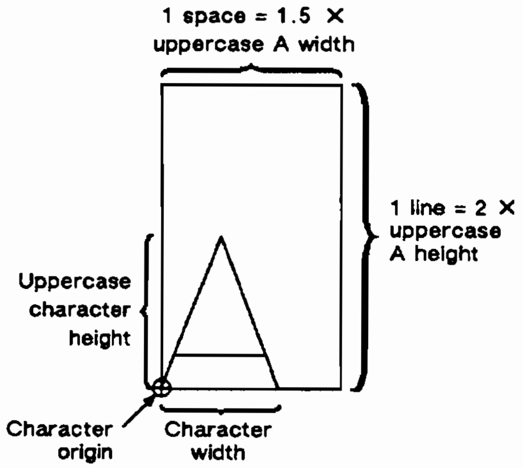
- **Space between characters and lines** — Normal. Refer to the ES instruction.

- **Character slant** — None (vertical). Refer to the SL instruction.

## Working with the Character Plot Cell

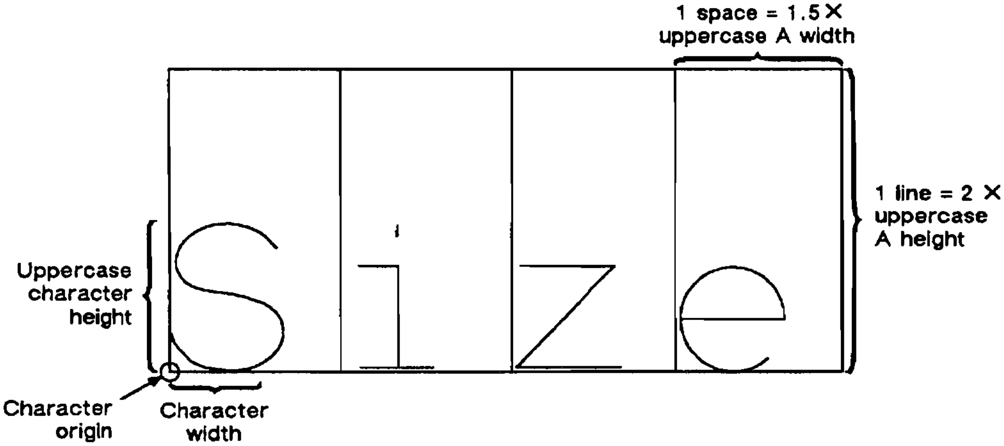
The character plot (CP) cell is the basis for almost all positioning and movement of labels. Think of the CP cell as a rectangular area around a character that includes blank areas above and to the right of the character. In a fixed-space font (such as the default font being discussed in this chapter), the CP cell is the same size for every character. (Refer to Chapter 8 for information on the CP cell in variable-space fonts.)

The CP cell is based on the uppercase 'A'. The CP cell height corresponds to a text line and is twice the height of the A. The CP cell width corresponds to a character space and is one and one-half times the width of the A. If you use the SI or SR instructions to change the size of the characters or use the ES instruction to add extra space around them, you will alter the size of the CP cell.



The current pen location when the label (LB) instruction is issued becomes the 'character origin' of the first CP cell (unless you have used the label origin (LO) instruction to alter the origin). After drawing the first character, the pen moves to the character origin of the next CP cell and draws the next character. This continues until the end of the label unless an embedded control character (for example, a carriage return) is encountered.

The character origin is the lower left corner of the CP cell. It is *not* necessarily the lower left corner of the character. Since the CP cell is based on an uppercase A, some of the lowercase characters begin slightly to the right of the left edge of the CP cell in order to maintain a reasonable appearance. Observe the location of the lowercase 'i' in the word 'Size' in the illustration below.


1 space = 1.5 X uppercase A width
1 line = 2 X uppercase A height
Uppercase character height
Character origin
Character width
Size

## Enhancing Labels

You can 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. Before using these enhancements, be sure you understand the character plot (CP) cell described previously.

### Character Size and Slant

You can change the size of the characters using the absolute character size and the relative character size (SI and SR) instructions. The absolute character size (SI) instruction establishes the size of the uppercase A in centimetres and maintains this character size independent of the location of P1 and P2 or the media size. The relative character size (SR) 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 line width 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.

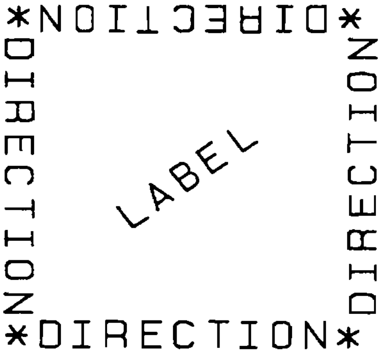
## Character Spaces and Text Lines

You can use the extra space (ES) instruction to automatically put extra spaces or lines between all characters or lines. 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.

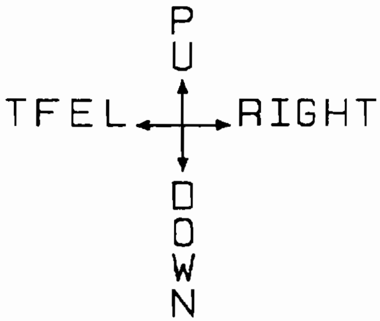
You can use the character plot (CP) instruction to move the pen a specific number of lines or spaces (CP cells) from the current pen location. Use the CP instruction, for example, to indent a label a certain number of spaces.

## Label Orientation and Placement

You can place your labels anywhere on the page in almost any orientation you desire. 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. Refer to the following illustration.



Using 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 offset slightly (in any direction you desire), offset distinctly, centered, and right- or left-justified from the current pen location. For example, the following illustration shows four centered lines of text.

Lines of any length
can easily be
centered
without cumbersome calculations.

This plot was programmed with 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 two, and using the CP instruction to 'backspace' the required number of cells — and that's just the first line.

The LO instruction also works well for offsetting a label from a data point so that the point is not obscured. Many of the plotted examples in this manual use the LO instruction to label X,Y coordinate points and to offset the label slightly from a line.

## CP, Character Plot

**USE:** Moves the pen the specified number of 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 | −32 767.9999 to 32 767.9999 | — |
| lines | clamped real | −32 767.9999 to 32 767.9999 | — |

**REMARKS:** The CP instruction includes an automatic pen up. When the instruction is completed, the original pen up/down position are 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 one line down and returns to the carriage-return point).

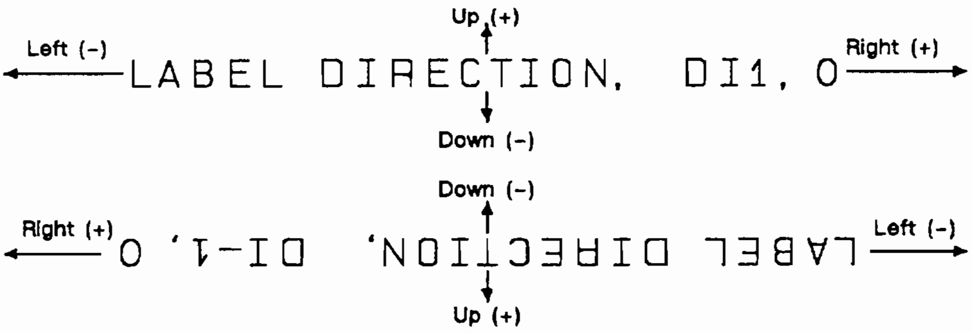
- **Spaces** — Specifies the number of character plot cell widths (spaces) the pen moves relative to the current pen location. Positive values specify the number of spaces the pen will move to the right of the current pen position; negative values specify the number of spaces the pen moves to the left. 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.

  **NOTE:** If you are using a variable-space font, the average character plot cell will be used. Refer to Chapter 7 for information about variable-space fonts. ■

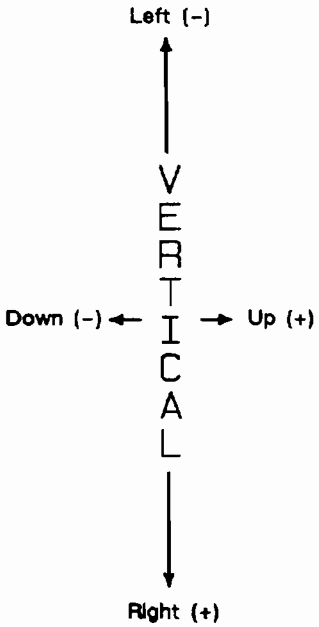
- **Lines** — Specifies the number of character plot cell heights (lines) the pen will move relative to the current pen location. Positive values specify the number of lines the pen will move up from the current pen position; negative values specify the number of lines the pen moves down (a value of −1 is equivalent to a line feed). Up and down are relative to the current label direction. The height of the character plot cell 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).

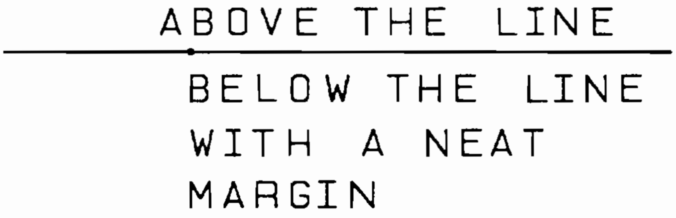


CP moves pen up with respect to the current character origin. CP *only affects the next label*, you must issue new CP instructions for subsequent labels.

**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.

```
10 ´Insert configuration statement here
2Ø PRINT #1, "INPS7ØØØ,8ØØØSP1PA5ØØØ,25ØØPD15ØØ,25ØØPU"
3Ø PRINT #1, "CP5,.35LBABOVE THE LINE"+CHR$(3)
4Ø PRINT #1, "PA25ØØ,25ØØWG2Ø,Ø,36Ø"
5Ø PRINT #1, "CPØ,-.95LBBELOW THE
                LINE"+CHR$(13)+CHR$(1Ø)+"WITH A
                NEAT"+CHR$(3)
6Ø PRINT #1, "CPLBMARGIN"+CHR$(3)
7Ø PRINT #1, "PG;"
8Ø END
```



**RELATED INSTRUCTIONS:**

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

## DI, Absolute Direction

**USE:** Specifies the 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 | −32 767.9999 to 32 767.9999 | 1 |
| rise (or sin θ) | clamped real | −32 767.9999 to 32 767.9999 | 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 location 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.

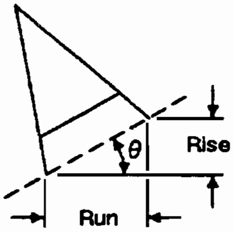
- **No Parameters** — Defaults the label direction to absolute and horizontal (parallel to X-axis). Equivalent to (*DI1,0*).

- **Run or Cos θ** — Specifies the X-component of the label.

- **Rise or Sin θ** — Specify the Y-component of the label.

  Together, the parameters specify the slope and direction of the label.

You can express the parameters in measured units as rise and run, or using the trigonometric functions cosine and sine according to the following relationships.



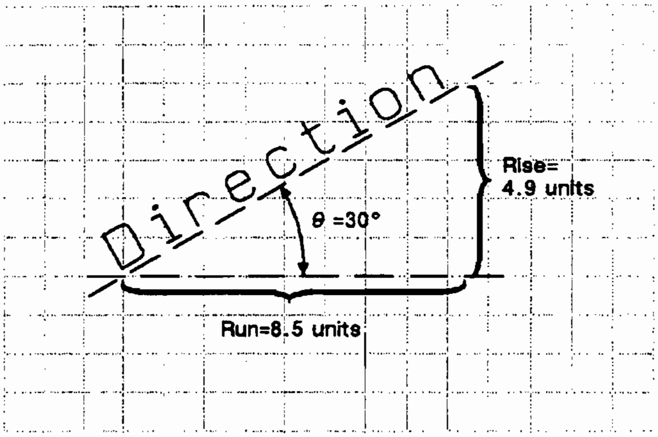
where: run and rise = number of measured units

$\theta$ = the angle measured in degrees

$$\frac{\sin\theta}{\cos\theta} = \frac{\text{rise}}{\text{run}}$$

$$\theta = \tan^{-1}\left(\frac{\text{rise}}{\text{run}}\right) \quad \text{and} \quad \tan\theta = \frac{\sin\theta}{\cos\theta}$$

Suppose you want your label plotted in the direction shown in the following illustration. You can do this by measuring the run and rise or by measuring 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-axis. 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 and one parallel to the X-axis. The lines should intersect to form an angle.

Select a point on the open end of your angle (where another line would create a triangle). On the line parallel to the X-axis, count the number of grid units from the intersection of the two lines to your selected point. This is the run. In the illustration above, the run is 8.5. Now, count the number of units from your selected point along a perpendicular line that intersects the line along the label. 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 in the same direction as shown in the previous illustration.

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 |
|---|---|---|
| 0 | 1 | 0 |
| -30 | 0.87 | -0.50 |
| -45 | 0.71 | -0.71 |
| -60 | 0.50 | -0.87 |
| -90 | 0 | -1 |
| -120 | -0.50 | -0.87 |
| -135 | -0.71 | -0.71 |
| -150 | -0.87 | -0.50 |
| -180 | -1 | 0 |
| -210 | -0.87 | 0.50 |
| -225 | -0.71 | 0.71 |
| -240 | -0.50 | 0.87 |
| -270 | 0 | 1 |
| -300 | 0.50 | 0.87 |
| -315 | 0.71 | 0.71 |
| -330 | 0.87 | 0.50 |
| -360 | 1 | 0 |

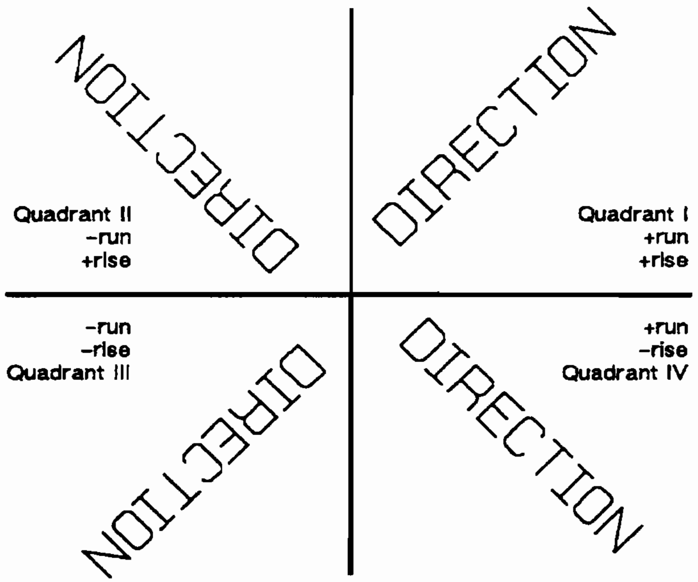
| θ | Cosine | Sine |
|---|---|---|
| 0 | 1 | 0 |
| 30 | 0.87 | 0.50 |
| 45 | 0.71 | 0.71 |
| 60 | 0.50 | 0.87 |
| 90 | 0 | 1 |
| 120 | -0.50 | 0.87 |
| 135 | -0.71 | 0.71 |
| 150 | -0.87 | 0.50 |
| 180 | -1 | 0 |
| 210 | -0.87 | -0.50 |
| 225 | -0.71 | -0.71 |
| 240 | -0.50 | -0.87 |
| 270 | 0 | -1 |
| 300 | 0.50 | -0.87 |
| 315 | 0.71 | -0.71 |
| 330 | 0.87 | -0.50 |
| 360 | 1 | 0 |

***When using either method***, at least one parameter must not be zero. The ratio of the parameters to each other is more important than the actual numbers. The following table lists three common label angles produced by using 1's and 0's.

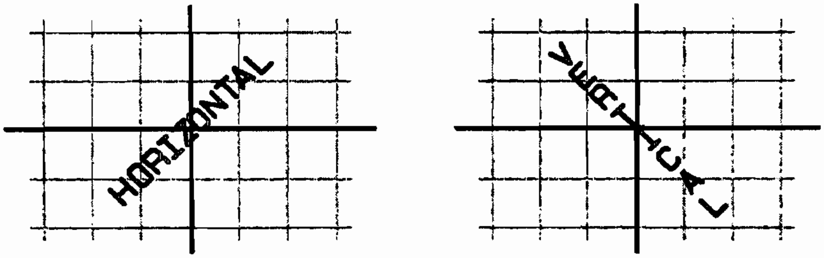
| DI Instruction | Label Direction |
|---|---|
| DI 1,0 | horizontal |
| DI 0,1 | vertical |
| DI 1,1 or DI 0.7,0.7<br>(any parameters equal to each other) | 45-degree angle |

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, "INPS5000,7000SP1PA3500,2500"
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
```



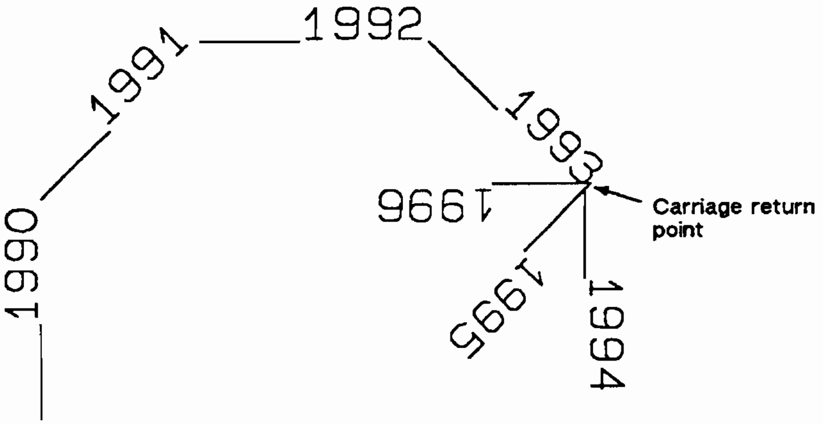
When plotting with a horizontal (Right) text path, the run and rise appear to determine the slope of the entire label. However, if you have used the define variable text path (DV) instruction to select vertical (Down) text path, the label appears to slant in the opposite direction even though the base 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.



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, "INPS5ØØØ,7ØØØSP1"
30  PRINT #1, "PA35ØØ,25ØØ"
40  PRINT #1, "DIØ,1LB___1987"+CHR$(3)
50  PRINT #1, "DI1,1LB___1988"+CHR$(3)
60  PRINT #1, "DI1,ØLB___1989"+CHR$(3)
70  'Angle = 135 degrees. Must convert degrees to
80  '  radians.  Radians=Degrees*PI/18Ø
90  '  Variables are integer.
100 PI=3.14593
110 A%=COS(315*(PI/18Ø))
120 B%=SIN(315*(PI/18Ø))
130 PRINT #1, "DI"A%","B%"LB___199Ø"+CHR$(3)
140 C%=COS(27Ø*(PI/18Ø))
150 D%=SIN(27Ø*(PI/18Ø))
160 PRINT #1, "DI"C%","D%"LB___1991"+CHR$(13)+CHR$(3)
170 E%=COS(225*(PI/18Ø))
180 F%=SIN(225*(PI/18Ø))
190 PRINT #1, "DI"E%","F%"LB___1992"+CHR$(13)+CHR$(3)
200 G%=COS(-18Ø*(PI/18Ø))
210 H%=SIN(-18Ø*(PI/18Ø))
220 PRINT #1, "DI"G%","H%"LB___1993"+CHR$(13)+CHR$(3)
230 PRINT #1, "PG;"
240 END
```



**RELATED INSTRUCTIONS:**

DR, Relative Direction
DV, Define Variable Text Path
LB, Label

**SPECIAL 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.

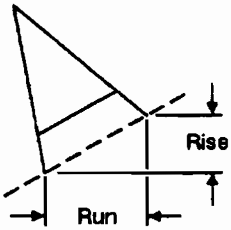
**SYNTAX:** DR*run,rise(;)*
or
DR*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| run | clamped real | -32 767.9999<br>to 32 767.9999 | 1% of P2X-P1X |
| rise | clamped real | -32 767.9999<br>to 32 767.9999 | 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, as explained later in this section.

- **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.

You can define the parameters of run and rise as shown in the following illustration.



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.

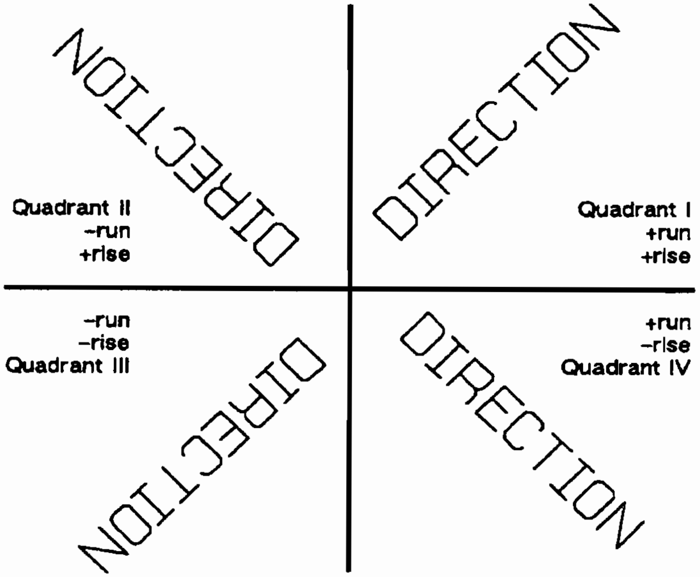
$$\text{run} = \text{run}(P2X - P1X)$$

$$\text{rise} = \text{rise}(P2Y - P1Y)$$

At least one parameter must not be zero. 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 ones and zeros.

| DR Instruction | Label Direction |
|---|---|
| DR 1,0 | horizontal |
| DR 0,1 | vertical |
| DR 1,1 or DR 0.7,0.7<br>(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.

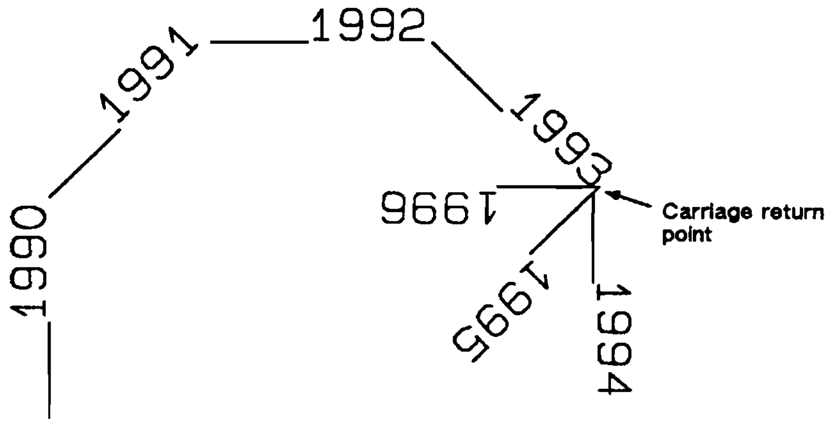


A DR instruction remains in effect until another DR or DI instruction is executed, or the plotter is initialized or set to default conditions.

**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:** 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, "INPS5ØØØ,7ØØØSP1"
3Ø  PRINT #1, "PA35ØØ,25ØØ"
4Ø  PRINT #1, "DRØ,1LB___199Ø"+CHR$(3)
5Ø  PRINT #1, "DR1,1LB___1991"+CHR$(3)
6Ø  PRINT #1, "DR1,ØLB___1992"+CHR$(3)
7Ø  PRINT #1, "DR1,-1LB___1993"+CHR$(3)
8Ø  PRINT #1, "DRØ,-1LB___1994"+CHR$(13)+CHR$(3)
9Ø  PRINT #1, "DR-1,-1LB___1995"+CHR$(13)+CHR$(3)
1ØØ PRINT #1, "DR-1,ØLB___1996"+CHR$(13)+CHR$(3)
11Ø PRINT #1, "PG;"
12Ø END
```



**RELATED INSTRUCTIONS:**

DI, Absolute Direction
DV, Define Variable Text Path
LB, Label

SPECIAL ERRORS:

| Condition | Error | Plotter Response |
|---|---|---|
| both parameters = 0<br>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 term(,mode;)*
or
DT*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| label terminator | label | any character except NULL, LF, ENQ, and ; (decimal codes 0, 5, 27, and 59 respectively) | ETX<br>(decimal code 3) |
| mode | clamped integer | 0 or 1 | 0 (printing) |

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 Parameter** — Defaults the label terminator to ETX (*not* a semicolon) and the mode to printing (0).

- **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** (Default) The label terminator prints if it is a printing character and performs its function if it is a character code.

  1 The label terminator does not print if it is a printing character and does not perform its function if it is a character code.

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 plotted labels with different terminators in the default mode.

**NOTE:** Although some program lines are shown on two lines to fit on this page, you should write them on just one line. ■

```
1Ø  'Insert configuration statement here
2Ø  PRINT #1, "INPS5ØØØ,7ØØØSP1SCØ,5ØØØ,Ø,5ØØØ"
3Ø  PRINT #1, "SI.187,.269PAØ,45ØØ"
4Ø  PRINT #1, "LBDefault control character
                 ETX"+CHR$(13)+CHR$(1Ø)+CHR$(3)
5Ø  PRINT #1, "LBterminates by performing
                 end-"+CHR$(13)+CHR$(1Ø)+CHR$(3)
6Ø  PRINT #1, "LBof-text function."+CHR$(3)
7Ø  PRINT #1, "CPCPDT#"
8Ø  PRINT #1, "LBPrinting characters termi
                 nate,"+CHR$(13)+CHR$(1Ø)+"#"
9Ø  PRINT #1, "LBbut are also printed.#"
1ØØ PRINT #1, "CPCPDT"+CHR$(13)+"
11Ø PRINT #1, "LBControl characters termi
                 nate"+CHR$(1Ø)+CHR$(13)
12Ø PRINT #1, "LBand perform their function."+CHR$(13)
13Ø PRINT #1, "PG;"
14Ø END
```

```
Default control character ETX
terminates by performing end-
of-text function.

Printing characters terminate,
#but are also printed.#

Control characters terminate
and perform their function.
```

**RELATED INSTRUCTIONS:** LB, Label

## DV, Define Variable Text Path

**USE:** Specifies either right, left, up, or down as the text path for subsequent labels and the direction of line feeds. Use DV to 'stack' characters in a column.

**SYNTAX:** DVpath(,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 that the current location moves after each character is drawn (text path) and the direction that the carriage-return point moves when a line feed is included in the label string.

- **No Parameter** — 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 in P1 and P2.

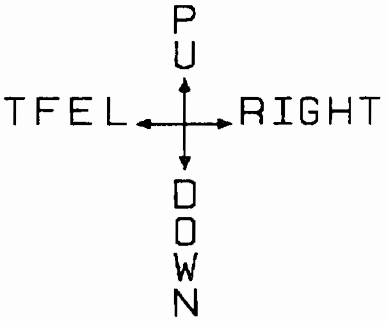
  **0** **0 degrees.** (Right) Each character begins to the right of the previous character. This is a horizontal text path (unless altered by DI or DR.)

  **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.)

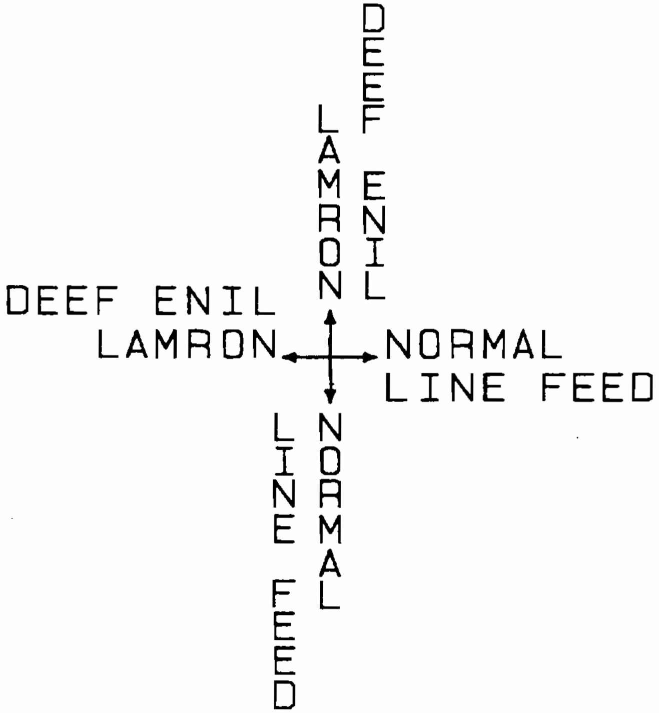
  **3** **–270 degrees.** (Up) Each character begins above the previous character. This is a vertical text path (unless altered by DI or DR.)

The following illustration shows the four text paths.

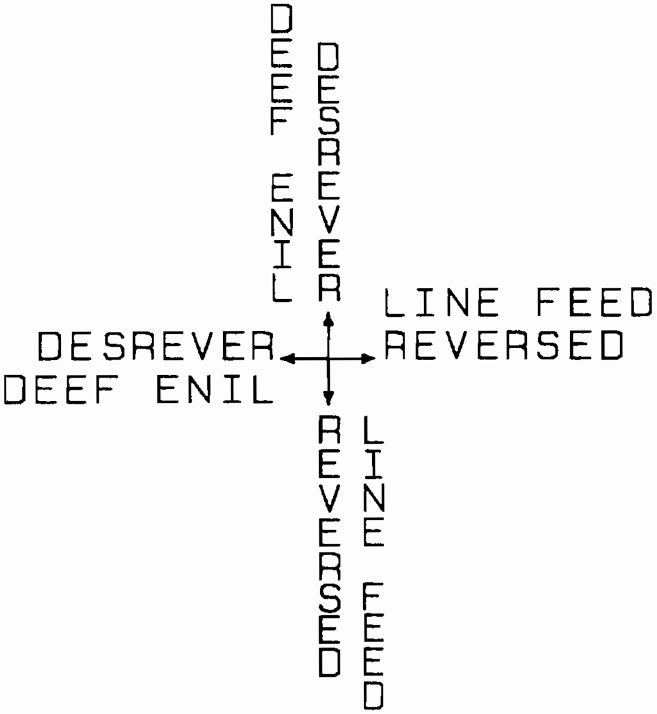


- **Line** — Specifies the location of each character with respect to the preceding character, relative to the labeling direction defined by the DI or DR instructions.

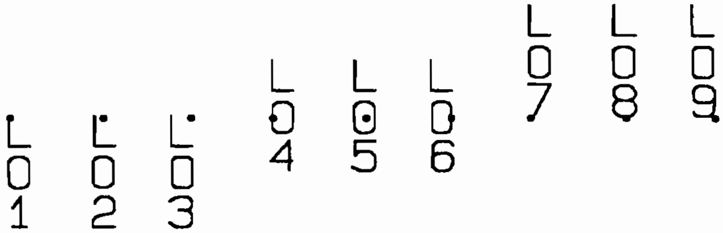
  **0** **–90 degrees.** (Normal line feed) Sets the direction of line feeds –90 degrees with respect to the text path.



0 **+90 degrees.** (Reversed line feed) Sets the direction of line feeds +90 degrees with respect to the text path.



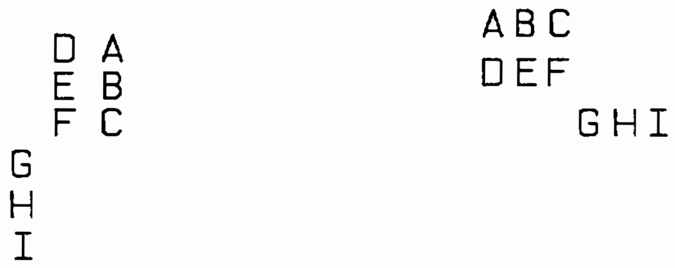
The DV instruction rotates the label origins set using the LO instruction. The following illustration shows some of the LO instruction label origins for a down text path.



**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, as follows.

| | |
|---|---|
| CHR$(3) | produces **ETX** (end-of-text) |
| CHR$(10) | produces **LF** (line feed) |
| CHR$(13) | produces **CR** (carriage return) |

```
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:**

DI, Absolute Direction
DR, Relative Direction
LO, Label Origin

## 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 | −32 767.9999 to 32 767.9999 | 0 |
| height | clamped real | −32 767.9999 to 32 767.9999 | 0 |

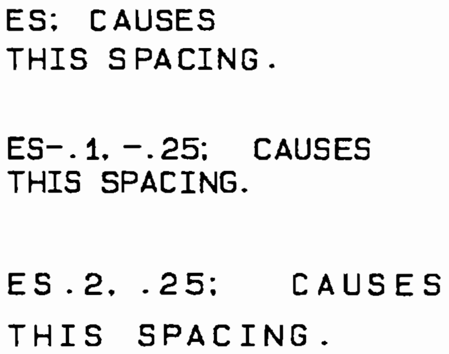
**REMARKS:** The plotter interprets the parameters as follows.

- **No Parameters** — Defaults the spaces and lines between characters to no extra space. Equivalent to *(ES0,0)*.
- **Spaces** — Specifies increases (positive number) or decreases (negative number) in the space between characters as a fraction of the 'space' character. For maximum legibility, do not specify more than one extra space or subtract more than half a space.
- **Lines** — Specifies increases (positive number) or decreases (negative number) in the space between lines as a fraction of the 'space' character. For maximum legibility, do not specify more than two extra lines or subtract more than half a line.

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

**EXAMPLE:**

```
10  ´Insert configuration statement here
20  PRINT #1, "INPS5000,8000SP1PA2500,3200SI.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, "PG;"
120 END
```



```
ES: CAUSES
THIS SPACING.


ES-.1,-.25:  CAUSES
THIS SPACING.


ES.2,.25:  CAUSES

THIS SPACING.
```

**RELATED INSTRUCTIONS:**

CP, Character Plot
LB, Label

## 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 | — |

**REMARKS:** The LB instruction includes an automatic pen down. When the instruction is completed, the original pen up/down position are restored. Labels are always drawn with triangular line ends and joins.

- **c...c** — Includes up to 150 ASCII characters. Printing characters are drawn using the currently selected character set. (Refer to Chapter 8 for information on selecting alternate character sets.)

  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. Refer to Appendix B for a list of ASCII characters.

  The label begins at the current pen location, (unless altered by LO). After each character is drawn, the pen location is updated to be the next character origin (refer to *Working With the Character Plot Cell* earlier in the chapter.)

- **Label Terminator** — Terminates the LB instruction. You *must* use the special label terminator (refer to the DT instruction) to tell the plotter to exit from the 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**. If you wish, you can define a different terminator using the DT instruction.

**EXAMPLE:**

```
10  'Insert configuration statement here
20  PRINT #1, "INPS5000,7000SP1PA2500,2500"
30  PRINT #1, "LBThis is a label."+CHR$(3)
40  PRINT #1, "PG;"
50  END
```

This is a label.

**RELATED INSTRUCTIONS:**

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

## 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.

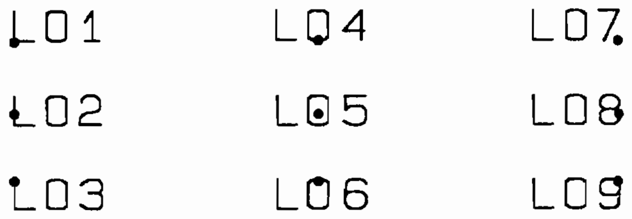
**SYNTAX:** LO*position(;)*
or
LO*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| position | clamped integer | 1 to 9 or 11 to 19 | 1 |

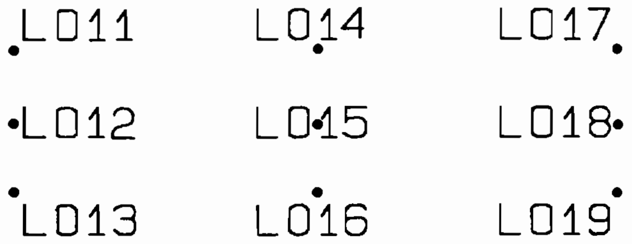
**REMARKS:** The plotter interprets the parameters as follows.

- No Parameters — Defaults the label origin. Equivalent to (*LO1*).

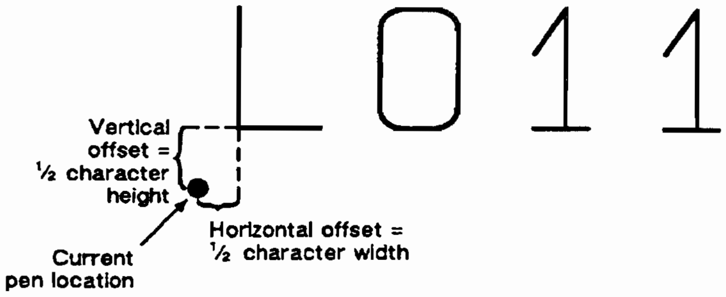
- **Position** — The position numbers are graphically illustrated below. Each dot represents the current pen location.



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.

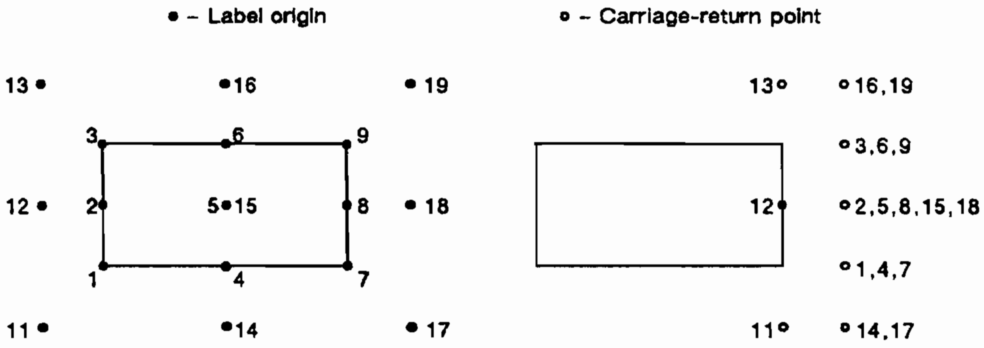


The amount of offset is equal to one-half the character's width and one-half of the character's height as specified by the most recent SI or SR instruction. The offset is shown in the following illustration.

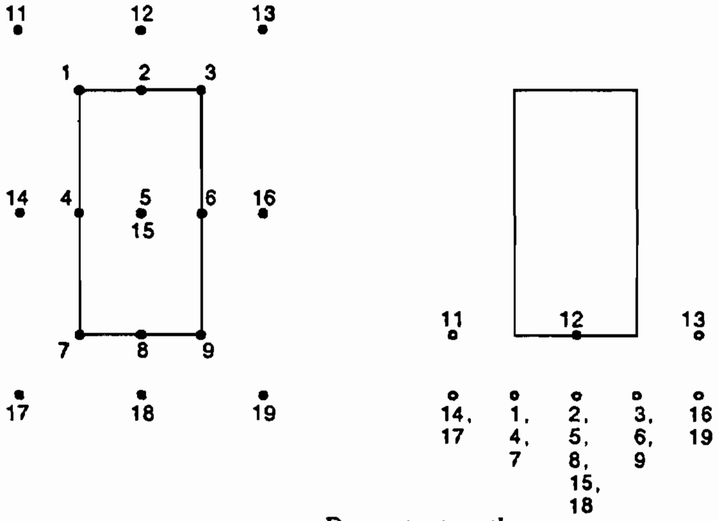

LO11
Vertical offset = ½ character height
Current pen location
Horizontal offset = ½ character width

Thereafter, LO updates the carriage-return point to the current location. 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 its previous location prior to the next label instruction, you can send a carriage return, CHR$(13), after the label but before the label terminator.

The define variable text path (DV) instruction rotates the label origins set by LO. The following illustration shows the label origins and normal carriage-return points for right and down text paths. The position of these points is mirrored for left and up text paths. The numbers in the illustration refer to the label origin position numbers illustrated on the preceding page.



*Right text path*



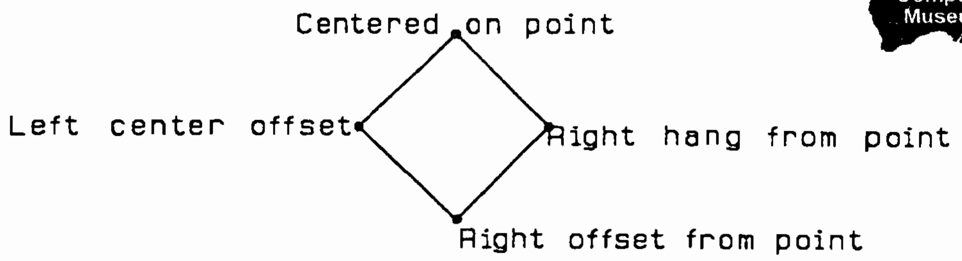
*Down text path*

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, "INPS5ØØØ,7ØØØSP1SI.17,.26PAØ,5ØØ"
30  PRINT #1, "PD-5ØØ,Ø,Ø,-5ØØ,5ØØ,Ø,Ø,5ØØ"
40  PRINT #1, "CI1Ø,LO4LBCentered on point"+CHR$(3)
50  PRINT #1, "PU-5ØØ,ØCI1Ø,LO18"
60  PRINT #1, "LBLeft center offset"+CHR$(3)
70  PRINT #1, "PUØ,-5ØØCI1Ø,LO13"
80  PRINT #1, "LBRight offset from point"+CHR$(3)
90  PRINT #1, "PA5ØØ,Ø;CI1Ø,LO3"
100 PRINT #1, "LBRight hang from point"+CHR$(3)
110 PRINT #1, "PG;"
120 END
```



**RELATED INSTRUCTIONS:**

DV, Define Variable Text Path
LB, Label

## SI, Absolute Character Size

**USE:** Specifies the size of labeling characters in centimetres. 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 | −32 767.9999 to 32 767.9999 | dependent* |
| height | clamped real | −32 767.9999 to 32 767.9999 | 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 set are not affected by changes in P1 and P2.

- **No Parameters** — Defaults the width of the absolute character size to the current pitch and the height to one half of the current font height set by the AD or SD instructions.

- **Width** — Specifies the width of the nominal character in centimetres. A negative width parameter mirrors labels in the right-to-left direction.

  The nominal character width is the en space. The en space (a typographical measurement) is equal to one half of the width of the font. SI adjusts the width of the en space to equal the specified width.

  **NOTE:** Changing character size also changes the width of line used to draw labels according to the stroke wiegbt attribute of AD or SD. ■

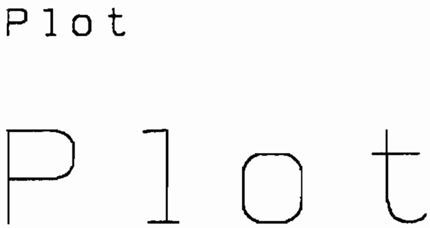
- **Height** — Specifies the height of the nominal character in centimetres. A negative height parameter mirrors labels in the top-to-bottom direction.

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.

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.

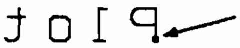
**EXAMPLE:** The following draws the word 'plot' in two sizes: the default and 1 cm wide by 1.5 cm high.

```
10  'Insert configuration statement here
20  PRINT #1, "INPS5000,7000SP1PA2500,3000"
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)
```

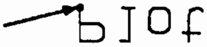


Current pen location

A negative height parameter mirrors labels in the top-to-bottom direction.

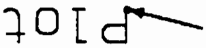
```
"SI.3,-.45;LBPlot"+CHR$(3)
```

Current pen location



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)
```



Current pen location

**RELATED INSTRUCTIONS:**

AD, Alternate Font Definition
SD, Standard Font Definition
SR, Relative Character Size
DI, Absolute Direction
DR, Relative Direction

## SL, Character Slant

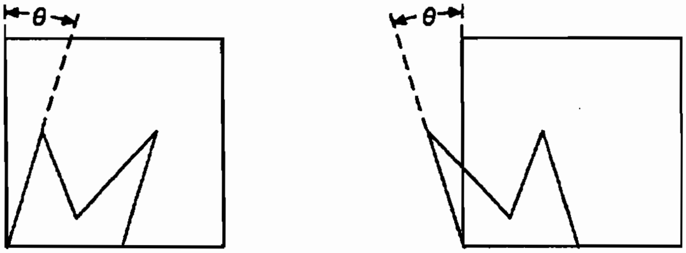
**USE:** Specifies the slant at which labels are drawn. Use SL to create slanted text for emphasis, or to reestablish upright labeling after an SL instruction with parameters has been in effect.

**SYNTAX:** SL*tangent of angle (;)*
or
SL*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| tangent of angle | clamped real | −32 767.9999 to 32 767.9999 | 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 only 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.

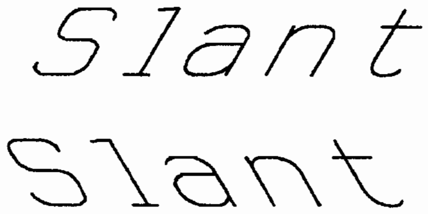
You can specify the actual tangent value, or you can use the TAN function available on most computers. A table of tangent values for selected angles follows.

| 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.

```
10 'Insert configuration statement here
20 PRINT #1, "INPS5000,7000SP1SI.7,1PA1000,1000"
30 PI=3.141593
40 A=TAN(20*(PI/180))
50 PRINT #1, "SL"A"LBSlant"+CHR$(3)
60 PRINT #1, "PA1000,300SL-.36LBSlant"+CHR$(3)
70 PRINT #1, "PG;"
80 END
```



**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. ■

**RELATED INSTRUCTIONS:**

DI, Absolute Direction
DR, Relative Direction
LB, Label

## 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)* | — |

*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 (e.g., (*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 centered at each subsequent X,Y coordinate. The symbol is drawn in addition to the usual function of each plotting instruction.

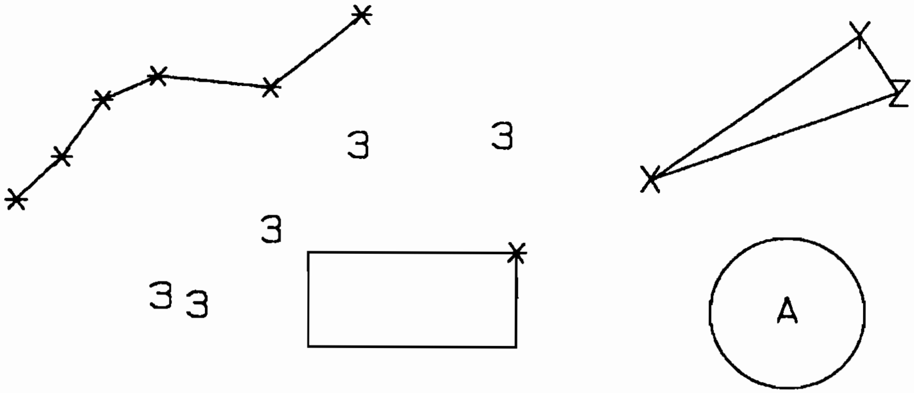
  The character is drawn in the character set selected at the time the SM instruction is executed. The character does not change even if you subsequently select a new character set. If you have defined downloadable characters in set -1, you can use these in symbol mode plotting, refer to the downloadable character (DL) instruction. Also, the size (SI and SR), slant (SL), and direction (DI and DR) instructions affect how the character is drawn. 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, "INPS5000,7000SP1SC0,1,0,1,2"
30  PRINT #1, "SM*PA0,1000PD200,1230,400,1560"
40  PRINT #1, "PD700,1670,1300,1600,1800,2000PU"
50  PRINT #1, "SM3PA700,500,900,450,1300,850"
60  PRINT #1, "PA1750,1300,2500,1350PUSM"
70  PRINT #1, "PA3300,1100PDSMYPA4400,1890SMZ"
80  PRINT #1, "PA4600,1590SMXPA3300,1100PU"
90  PRINT #1, "SMAPA4000,400CI400"
100 PRINT #1, "SM*PA2600,700EA1500,200"
110 PRINT #1, "PG;"
120 END
```



**RELATED INSTRUCTIONS:**

PA, Plot Absolute
PD, Pen Down
PE, Polyline Encoded
PR, Plot Relative
PU, Pen Up

## 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 | −32 767.9999<br>to 32 767.9999 | 0.75% of P2X–P1X |
| height | clamped real | −32 767.9999<br>to 32 767.9999 | 1.5% or P2Y–P1Y |

**REMARKS:** While SR is in effect, with or without parameters, the size characters of the currently selected font are affected by changes in P1 and P2.

- **No Parameters** — Defaults the relative character width to 0.75% of the distance (P2X - P1X) and the height to 1.5% of the 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 line used to draw labels according to the stroke wieght attribute of AD or SD. ■

- **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 from the specified parameters as follows:

width = width/100 × (P2X - P1X)

height = height/100 × (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:

width = 2/100 × ( 6956 – (–6956) ) = 278.24 plotter units or 0.695 cm

height = 3.5/100 × (4388 – ( –4388) ) = 307.16 plotter units or 0.768 cm

If you changed P1 and P2 settings 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

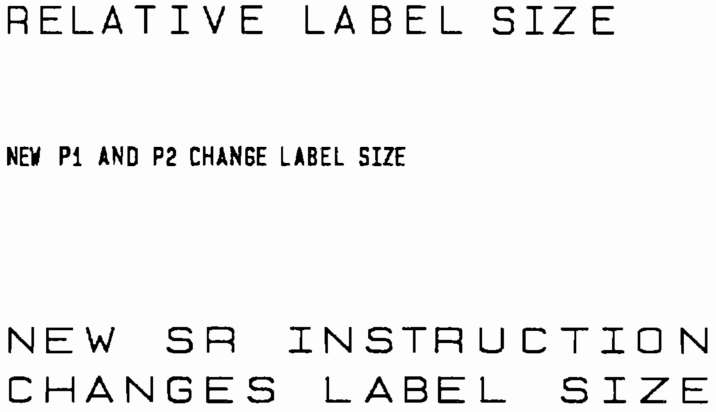
Note that in most languages the width of a letter is typically less than the height. If you set your characters to have a different 'aspect ratio', they may look odd to your readers.

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. ■

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.

**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, "INPS7ØØØ,8ØØØSP1IP-6956,-4388,6956,4388"
3Ø  PRINT #1, "SRPAØ,27ØØLBRELATIVE LABEL SIZE"+CHR$(3)
4Ø  '
5Ø  PRINT #1, "IPØ,Ø,55ØØ,55ØØPAØ,2ØØØ"
6Ø  PRINT #1, "LBNEW P1 AND P2 CHANGE LABEL SIZE"+CHR$(3)
7Ø  '
8Ø  PRINT #1, "PAØ,1ØØØSR2.5,2.5"
9Ø  PRINT #1, "LBNEW SR INSTRUCTION"+CHR$(3)+"CP"
1ØØ PRINT #1, "LBCHANGES LABEL SIZE"+CHR$(3)
11Ø PRINT #1, "PG;"
12Ø END
```



RELATIVE LABEL SIZE

NEW P1 AND P2 CHANGE LABEL SIZE

NEW SR INSTRUCTION
CHANGES LABEL SIZE

**RELATED INSTRUCTIONS:**

SI, Absolute Character Size
DI, Absolute Direction
DR, Relative Direction

## 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:** The plotter interprets the parameters as follows.

- **No Parameters** — Defaults the labeling mode to normal. Equivalent to (*TD0*).

- Mode — Selects the normal or transparent mode for labeling.

  **0** **Normal.** Characters with an associated functionality perform their function and do not print. Refer to the table of control characters in Appendix B.

  1 **Transparent.** All characters print and perform no other function (except the currently defined label terminator, which prints and terminates the label.) The plotter prints a space for nonprinting or undefined characters.

**RELATED INSTRUCTIONS:**

DT, Define Label Terminator
LB, Label
SD, Standard Font Definition

8

# Using Character Sets

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

- Plot with variable- and fixed-space font character sets.
- Use downloadable characters.
- Select standard and alternate character sets.

The following instructions are described in this chapter.

AD, Alternate Font Definition
DL, Download Character
SA, Select Alternate Font
SS, Select Standard Font
SD, Standard Font Definition

## Working with Character Sets

The plotter has 22 character sets and three fonts. (Refer to Appendix A for information on using the Kanji character set, available only in emulation mode.) A font is simply a style of lettering. Your plotter has three fonts you can use when drawing from 21 of its character sets. The differences between these fonts are described as follows.

- **Fixed vector font** — Each character occupies an equal horizontal space (like a typewriter) and are always drawn using a fixed number of vectors.

- **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.

- **Fixed arc font** — Characters all occupy an equal horizontal space (like a typewriter) and are drawn using arcs for smoother contours.

- **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 a way to avoid confusion between lines and figures as described in the following table. The set also contains symbols used in drafting, such as 1 or n . Refer to Appendix B to see the entire set. The HP Drafting font is a fixed-space vector font.

| | |
|---|---|
| -B- | Bottom wider than top. |
| -6- | Large body, stem curved and open. |
| -8- | Lower part larger than upper,<br>full and round to avoid blur. |
| -9- | Large body, stem curved but open. |

The following illustration shows a portion of the default character set (Roman8) in the following fonts: fixed vector, variable arc, and fixed arc. The difference in the total space required to plot the sets emphasizes the difference between fixed-space fonts and variable space fonts. Also, note the variation in the smoothness of characters between the arc and vector fonts. Refer to the alternate font definition (AD) and standard font definition (SD) instructions for a complete list of the your plotter's character sets. The character sets are shown in Appendix B (except Kanji, shown in Appendix A).

```
! "#$%&' ()*+, -./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ [\]^_`
abcdefghijklmnopqrstuvwxyz {|}~
```

*Fixed Vector Font*

```
!"#$%&'()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ [\]^_`
abcdefghijklmnopqrstuvwxyz{|}~
```

*Variable Arc Font*

```
! "#$%&' ()*+, -./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ [\]^_`
abcdefghijklmnopqrstuvwxyz {|}~
```

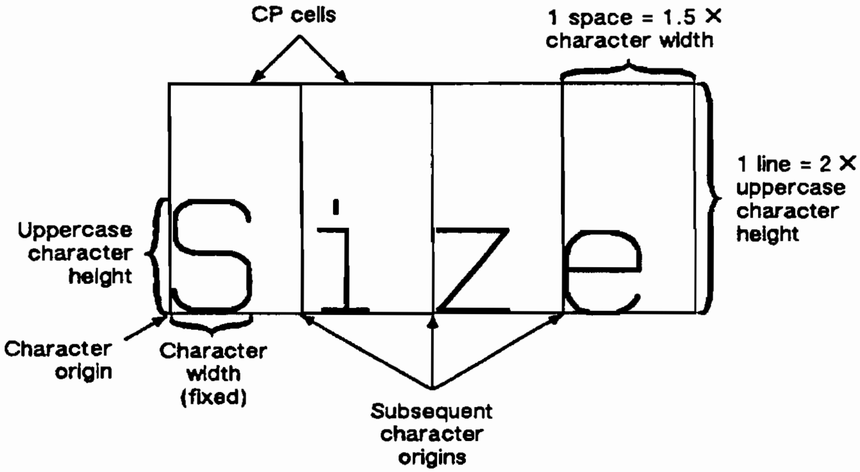
*Fixed Arc Font*

All sets except HP Special Symbols, HP Roman Extensions, and HP Katakana draw the same upper- and lowercase letters and numbers. With these exceptions, the sets differ only in the additional characters that are needed in certain languages.

## Plotting with Variable-Space Fonts

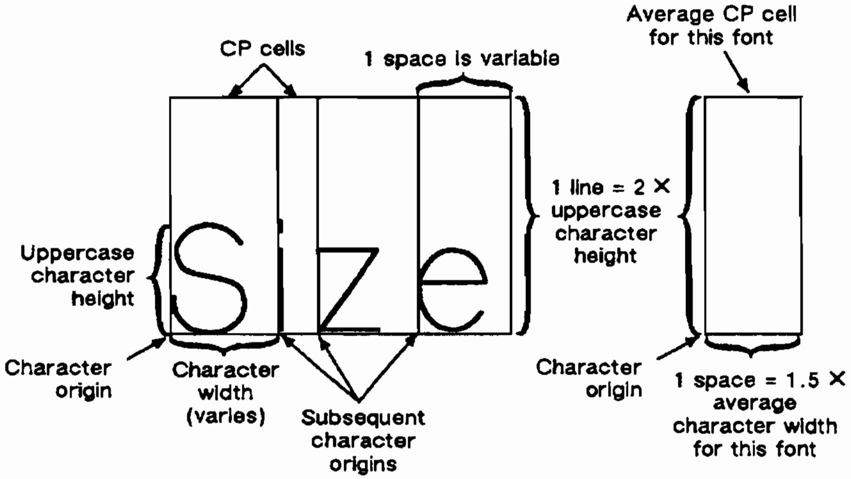
The variable-space fonts, by definition, use different amounts of horizontal space for each letter. This variance produces some differences in the definition of the character plot cell and in the way some of the labeling instructions work with variable-space fonts. These differences are described in this section.

As mentioned in Chapter 7, the character plot cell is defined as being 2 times the height of the uppercase A and 1½ times the width. Refer to the following illustration.


CP cells
1 space = 1.5 X character width
1 line = 2 X uppercase character height
Uppercase character height
Size
Character origin
Character width (fixed)
Subsequent character origins

*Fixed-Space Font*

With variable-space fonts, the character plot cell remains 2 times the height of an uppercase A, but the 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, but a blank space will always be 1½ times the width of an average character. Refer to the following illustration.



Average CP cell for this font
CP cells
1 space is variable
1 line = 2 × uppercase character height
Uppercase character height
Size
Character origin
Character width (varies)
Subsequent character origins
Character origin
1 space = 1.5 × average character width for this font

*Variable-Space Font*

When plotting in variable-space fonts, the CP instruction (*CPspaces, lines*) and the ES instruction (*ESspaces, lines*) use the average character plot cell in computing lines and spaces. Both the SI and SR instructions (*SIwidth, height*) and (*SRwidth, height*) 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 tends to make them look as though they were fixed-space. Refer to the following illustration.

ES CAUSES
THIS SPACING.

ES-.1,-.25 CAUSES
THIS SPACING.

ES.2,.25 CAUSES
THIS SPACING.

### 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 that are stored in a buffer for repeated use. The character plot cell used by the DL instruction is always a fixed-space character cell, regardless of the font currently selected.

## Designating and Selecting Fonts

If you always intend to label with the default fixed-space font, you do not need to use the SD or AD instructions for designating standard and alternate fonts. 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. Refer to the descriptions for each of these instructions at the end of this chapter.

### 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 set 0 (the default) as your standard set, 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 either using 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.

- Use the equivalent ANSI ASCII English character on your keyboard in the label string. (Refer to the *Character Sets and ASCII Codes* table in Appendix B.) For example, to draw the character '½' in set 5, you can use the 'x' from an English keyboard.

  ½

- 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).

  ½

- Use the SS and SA instructions to shift fonts to use a character from another set.

```
10  ´Insert configuration statement here
20  PRINT #1, "INPS5000,7000SP1"
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 compa"+CHR$(124)+"ia?"+CHR$(3)
70  PRINT #1, "PG;"
80  END
```

```
USASCII, fixed vector font

    SPANISH SET, variable arc font

            ¿su compañia?
```

- 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.

```
10  'Insert configuration statement here
20  PRINT #1, "INPS5000,8000SP1"
30  PRINT #1, "PA300,1000SD1,115,3,3,4,30AD1,5"
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
```

```
3½-5 ÅR   3½-5 ÅR
```

## Control Characters

The plotter recognizes only the following control characters.

- Backspace, decimal code 8.

- Horizontal tab, decimal code 9.

- Line feed, decimal code 10.

- Carriage return, decimal code 13.

- Shift-out, decimal code 14.

- Shift-in, decimal code 15.

## 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.

**SYNTAX:** AD*kind,value...(,kind,value;)*
or
AD*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| kind | clamped integer | 1 to 7 | — |
| value | clamped real | kind dependent* | kind dependent* |

* Refer to the table following the parameter descriptions.

**REMARKS:**

- **No Parameters** — Defaults the alternate font attributes.
- **Kind** — Specifies the attribute for which you are setting a value: character set, font spacing, pitch, height, stroke weight, or typeface.
- **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 | ISO No. |
|---|---|---|---|---|
| Character set | 1 | 277 (0) | Roman8 (default) | — |
| | | 21 | ANSI USASCII | 006 |
| | | 14 | ECMA 94 Latin 1 | — |
| | | 6 | French v1 | 025 |
| | | 38 | French v2 | 069 |
| | | 39 | German | 021 |
| | | 563 | HP Drafting | — |
| | | 531 | HP-GL/2 Download | — |
| | | 267 | HP Kana8 | — |
| | | 43 | HP Katakana | — |
| | | 5 | HP Roman Extensions | — |
| | | 595 | HP Special Symbols | — |
| | | 85 | Intnl. Ref. Version | 002 |
| | | 9 | Italian | 015 |
| | | 11 | JIS ASCII | 014 |
| | | 4 | Norwegian v1 | 060 |
| | | 36 | Norwegian v2 | 061 |
| | | 147 | Portuguese | 016 |
| | | 115 | Swedish | 010 |
| | | 19 | Swedish Names | 011 |
| | | 83 | Spanish | 017 |
| | | 37 | United Kingdom | 004 |
| Font spacing | 2 | 0 | fixed spacing (default) | |
| | | 1 | variable spacing | |
| Pitch | 3 | 0 to 32 767.9999 | characters per inch<br>default: 5.942 | |
| Height | 4 | 0 to 32 767.9999 | font point size<br>default: 21.3 | |
| Stroke Weight | 6 | -7 | very light | |
| | | -3 | light | |
| | | 0 | normal (default) | |
| | | 3 | bold | |
| | | 7 | very bold | |
| Typeface | 7 | 48 | fixed vector (default) | |
| | | 49 | drafting | |
| | | 50 | fixed arc | |

*The kind parameter 5 is ignored.

## Character Sets

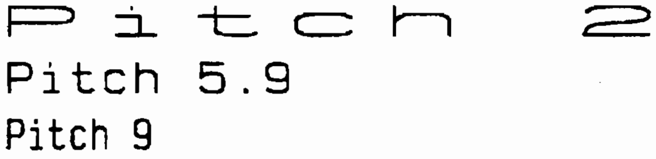
The character set attribute defines the character set to be used as the alternate set. Refer to Appendix B to see each of the plotter's 22 character sets.

## Font Spacing

The font spacing attribute defines whether the spacing is fixed (all characters occupying an equal horizontal space) or variable (each character occupying a space proportional to its size). Refer to *Using Fonts* in the beginning of this chapter.

## Pitch and Height

The pitch attribute defines the number of characters per inch. The following illustration shows text in three different pitches.

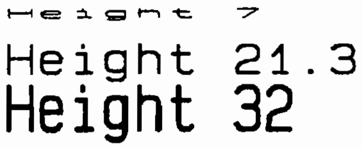


Pitch 2

Pitch 5.9

Pitch 9

The height attribute defines the font point size, i.e., the height of the character plot cell (discussed in Chapter 7). The following illustration shows text in three different heights.



Height 7

Height 21.3

Height 32

The pitch and height attributes apply only when (*SI*) is in effect. Otherwise, the SI or SR width and height parameters determine the character size.

## Stroke Weight

The stroke weight attribute defines the line width used in labels. The following illustration shows labels in the five different stroke weights.

SD6, -7 very light

SD6, -3 light

SD6, 0 normal

**SD6, 3 bold**

**SD6, 7 very bold**

### Typeface

The typeface attribute selects the font, which defines the smoothness of the lettering. Characters drawn with arcs have a smoother, more rounded appearance than characters drawn with vectors. The drafting font is for use with the HP Drafting character set. Refer to *Using Fonts* in the beginning of this chapter.

The following examples shown how to select each of the plotter's three fonts for an unspecified character set.

| | |
|---|---|
| Fixed vector | (*AD2,0,7,4,8,1,character set*) |
| Variable arc | (*AD2,1,1,character set*) |
| Fixed arc | (*AD2,0,7,50,1,character set*) |

**RELATED INSTRUCTIONS:**

LB, Label
SA, Select Alternate Font
SD, Standard Font Dinition
SI, Absolute Character Size
SR, Relative Character Size
SS, Select Standard Font

## 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 designating (SD or AD) and selecting (SS or SA) the HP-GL/2 Download character set (character set 531) use the DL instruction to create characters line by line, one character per command. Once defined with DL, characters can be used in the LB instruction. 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 (freeing the buffer space for polygons).

- **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).

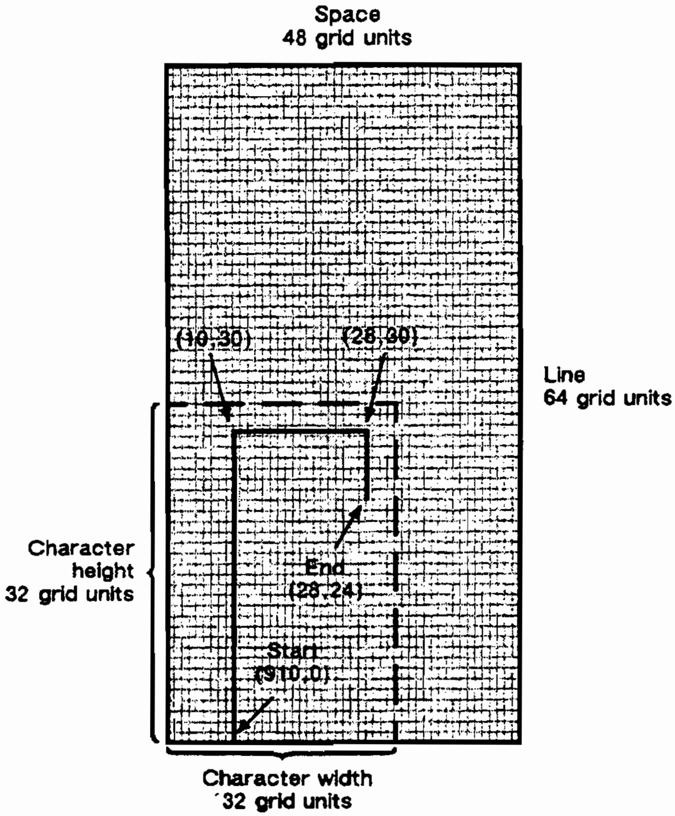
The number of parameters following the character number parameter is not restricted. Your plotter can download 94 characters, with at least 20 points in each character, and still have enough memory available for a 1000 point polygon.

## Defining a Downloadable Character

Define the character in absolute units on a 32 × 32 unit grid in a 48 × 64 unit cell. The origin (0,0) is in the lower-left corner. 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.

1. 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 two overhead bytes per defined character. The points in a DL character average 1½ bytes each.
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.

The diagram below shows the grid for a gamma symbol and is followed by the program that draws it.



**EXAMPLE:**

```
10  ´Insert configuration statement here
20 PRINT #1, "INPS5ØØØ,7ØØØSP1AD1,531,3,3,4,2Ø"
30 PRINT #1, "DL65,1Ø,Ø,1Ø,3Ø,28,3Ø,28,24"
4Ø PRINT #1, "PA3ØØ,3ØØ"
5Ø PRINT #1, "LB The symbol for gamma "+CHR$(3)
6Ø PRINT #1, "LBis "+CHR$(14)+"A"+CHR$(3)
7Ø PRINT #1, "PG;"
8Ø END
```

The symbol for gamma is Γ

**RELATED INSTRUCTIONS:**

AD, Alternate Font Definition
LB, Label
SA, Select Alternate Character Set
SD, Standard Font Definition
SS, Select Standard Character Set

**SPECIAL ERRORS:**

| Condition | Error | Plotter Response |
|---|---|---|
| buffer overflow | 7 | ignores instruction |

## 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 uses character set 277 (or 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:**

AD, Alternate Font Definition
LB, Label
SD, Standard Font Definition
SS, Select Standard Font

## SD, Standard Font Definition

**USE:** Defines the standard character set and its attributes: font spacing, pitch, height, stroke weight, and typeface.

**SYNTAX:** SD*kind,value... (,kind,value;)*
or
SD*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| kind | clamped integer | 1 to 7 | — |
| value | clamped real | kind dependent* | kind dependent* |

* Refer to the table following the parameter descriptions.

**REMARKS:**

- **No Parameters** — Defaults the standard font attributes.
- **Kind** — Specifies the attribute for which you are setting a value: character set, font spacing, pitch, height, stroke weight, or typeface.
- **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 | ISO No. |
|---|---|---|---|---|
| Character set | 1 | 277 (0) | Roman8 (default) | — |
| | | 21 | ANSI USASCII | 006 |
| | | 14 | ECMA 94 Latin 1 | — |
| | | 6 | French v1 | 025 |
| | | 38 | French v2 | 069 |
| | | 39 | German | 021 |
| | | 563 | HP Drafting | — |
| | | 531 | HP-GL/2 Download | — |
| | | 267 | HP Kana8 | — |
| | | 43 | HP Katakana | — |
| | | 5 | HP Roman Extensions | — |
| | | 595 | HP Special Symbols | — |
| | | 85 | Intnl. Ref. Version | 002 |
| | | 9 | Italian | 015 |
| | | 11 | JIS ASCII | 014 |
| | | 4 | Norwegian v1 | 060 |
| | | 36 | Norwegian v2 | 061 |
| | | 147 | Portuguese | 016 |
| | | 115 | Swedish | 010 |
| | | 19 | Swedish Names | 011 |
| | | 83 | Spanish | 017 |
| | | 37 | United Kingdom | 004 |
| Font spacing | 2 | 0 | fixed spacing (default) | |
| | | 1 | variable spacing | |
| Pitch | 3 | 0 to 32 767.9999 | characters per inch | |
| | | | default: 5.942 | |
| Height | 4 | 0 to 32 767.9999 | font point size | |
| | | | default: 21.3 | |
| Stroke Weight | 6 | -7 | very light | |
| | | -3 | light | |
| | | 0 | normal (default) | |
| | | 3 | bold | |
| | | 7 | very bold | |
| Typeface | 7 | 48 | fixed vector (default) | |
| | | 49 | drafting | |
| | | 50 | fixed arc | |

*The kind parameters 5 and 6 are ignored.

## Character Sets

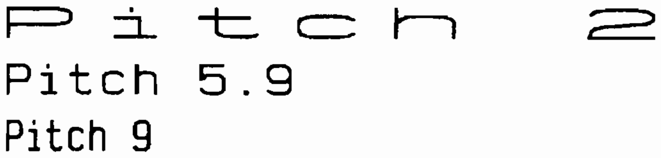
The character set attribute defines the standard character set. Refer to Appendix B to see each of the plotter's 22 character sets.

## Font Spacing

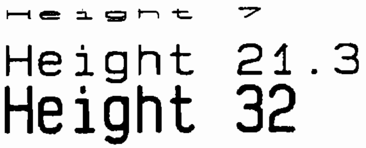
The font spacing attribute defines whether the spacing is fixed (all characters occupying an equal horizontal space) or variable (each character occupying a space proportional to its size). Refer to *Using Fonts* in the beginning of this chapter.

## Pitch and Height

The pitch attribute defines the number of characters per inch. The following illustration shows text in three different pitches.



The height attribute defines the font point size, i.e., the height of the character plot cell (discussed in Chapter 7). The following illustration shows text in three different heights.



The pitch and height attributes apply only when (*SI*) is in effect. Otherwise, the SI or SR width and height parameters determine the character size.

## Stroke Weight

The stroke weight attribute defines the line width used in labels. The following illustration shows labels in the five different stroke weights.

SD6, -7 very light

SD6, -3 light

SD6, 0 normal

SD6, 3 bold

**SD6, 7 very bold**

## Typeface

The typeface attribute selects the font, which defines the smoothness of the lettering. Characters drawn with arcs have a smoother, more rounded appearance than characters drawn with vectors. The drafting font is for use with the HP Drafting character set. Refer to *Using Fonts* in the beginning of this chapter.

The following examples shown how to select each of the plotter's three fonts for an unspecified character set.

| | |
|---|---|
| Fixed vector | (*AD2,0,7,4,8,1,character set*) |
| Variable arc | (*AD2,1,1,character set*) |
| Fixed arc | (*AD2,0,7,50,1,character set*) |

**RELATED INSTRUCTIONS:**

AD, Alternate Font Definition
LB, Label
SA, Select Alternate Font
SI, Absolute Character Size
SR, Relative Character Size
SS, Select Standard Font

## 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 character set 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 uses character set 277 (or 0). Character font 277 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:**

AD, Alternate Font Definition
LB, Label
SA, Select Alternate Character Set
SD, Standard Font Definition

9

# Changing Picture Area and Orientation

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

- Use windows (soft-clip limits).
- Enlarge and reduce plots.
- Draw equal-sized and mirror-imaged plots.
- Rotate plots.

The following instructions are described in this chapter.

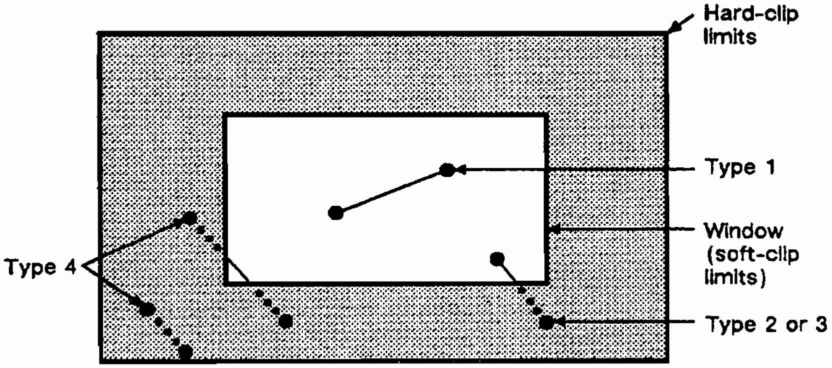
IW, Input Window
OH, Output Hard-Clip Limits
OP, Output P1 and P2
RO, Rotate Coordinate System

## 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.

| | Last Point | | New Point |
|---|---|---|---|
| 1. | Inside window area | to | inside window area |
| 2. | Inside window area | to | outside window area |
| 3. | Outside window area | to | inside window area |
| 4. | Outside window area | to | outside window area |



The IW instruction enables you to 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 *Soft-Clip Limits* in Chapter 2 and the IW instruction description later in this chapter.

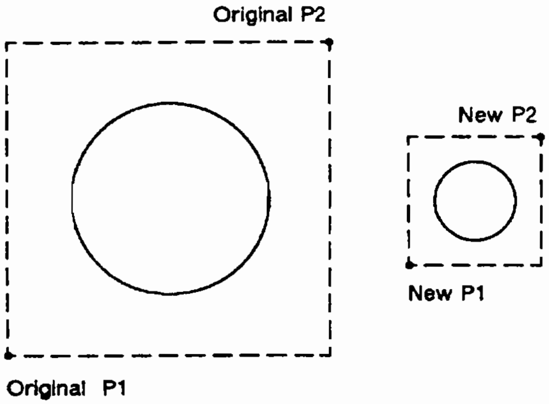
## Enlarging or Reducing a Picture

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 be able to plot the picture on any 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, respectfully, 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 drawing a circle 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 is smaller but is proportionally identical.

```
1Ø  'Insert configuration statement here
2Ø  PRINT #1, "INPS5ØØØ,7ØØØIPØ,Ø,2ØØØ,2ØØØ"
3Ø  PRINT #1, "SCØ,1Ø,Ø,1ØSP1"
4Ø  PRINT #1, "PA5,5;CI3"
5Ø  PRINT #1, "IP25ØØ,5ØØ,35ØØ,15ØØ"
6Ø  PRINT #1, "PA5,5CI3PG;"
7Ø  END
```

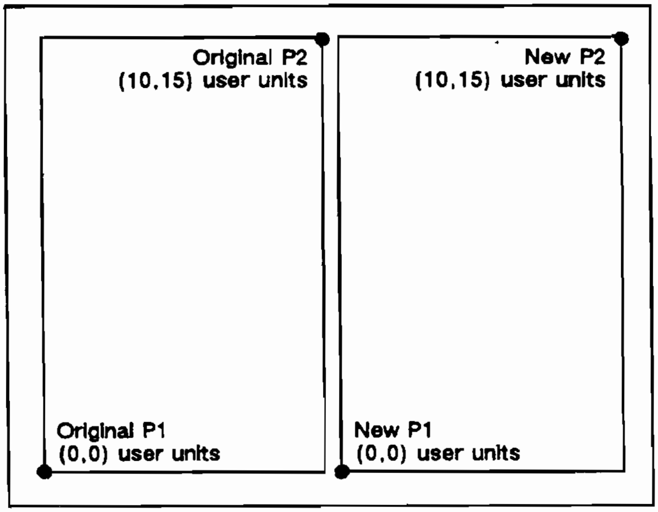


## Drawing Equal-Sized Pictures on One Page

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.

For example, the following locates P1 and P2 on the left side of the paper and scales the area. After drawing a boundary using the scaling parameters, only the P1 location is moved to the right side of the paper; P2 automatically tracks P1 so the plotting area retains the same dimensions as the first. The plotted rectangle around the second area shows P2 in its new location.

```
1Ø 'Insert configuration statement here
2Ø PRINT #1, "INPS8ØØØ,11ØØØIP5ØØ,5ØØ,545Ø,75ØØ"
3Ø PRINT #1, "SCØ,1Ø,Ø,15"
4Ø PRINT #1, "SP1PAØ,ØPD1Ø,Ø,1Ø,15,Ø,15,Ø,ØPU"
5Ø PRINT #1, "IP555Ø,5ØØ"
6Ø PRINT #1, "PAØ,ØPD1Ø,Ø,1Ø,15,Ø,15,Ø,Ø"
7Ø PRINT #1, "PG;"
8Ø END
```



**NOTE:** These P1/P2 frames are not windows or graphics limits; the pen can plot anywhere within the hard-clip limits. The new P1 and P2 retain their scaled values. This allows you to 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. ■

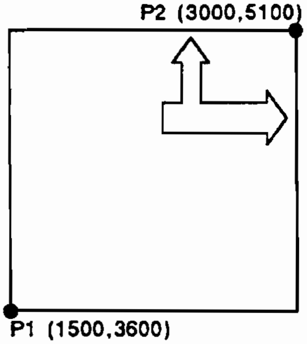
## Creating Mirror Images

For most plots, you will probably set P1 and P2 so that P1 is in the lower-left corner and P2 is in the upper-right corner of the scaling area. However, you can change the relationship of P1 and P2 to produce mirror imaging.

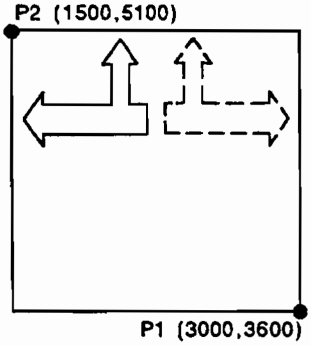
You can mirror image any *scaled* plot by changing the relative locations of P1 and P2. You can mirror image labels using the absolute direction and relative direction (DI and DR) instructions or the relative character size (SR) instruction. The DI, DR, and SR instructions are discussed in Chapter 7, *Labeling Your Plots*.

The following program uses a subroutine to draw the same picture (an arrow) four times. Because the program changes the relative locations of P1 and P2, the direction of the arrow is different in each of the four drawings. The program sets P1 and P2, draws the plot, then returns to reset P1 and P2 (using the IP instruction). This continues until all four possible mirror images are plotted. (The original plot is shown in each picture so you can compare the orientation of the mirror image.)

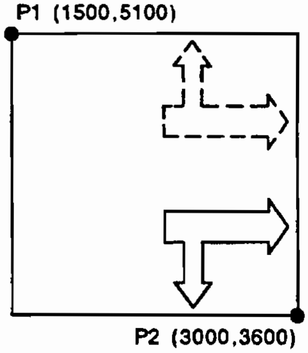
```
10  'Insert configuration statement here
20  PRINT #1, "INPS5000,7000SP1IP1000,0,2500,1600"
30  PRINT #1, "SC-15,15,-10,10"
40  GOSUB 130
50  PRINT #1, "IP2500,0,1000,1600"
60  GOSUB 130
70  PRINT #1, "IP1000,1600,2500,0"
80  GOSUB 130
90  PRINT #1, "IP2500,1600,1000,0"
100 GOSUB 130
110 PRINT #1, "PG;"
120 END
130 'draws the plot
140 PRINT #1, "PA1,2PD1,4,3,4,3,7,2,7,4,9,6,7,5,7"
150 PRINT #1, "PD5,4,12,4,12,5,14,3,12,1PU"
160 PRINT #1, "PD12,2,1,2PU"
170 RETURN
```



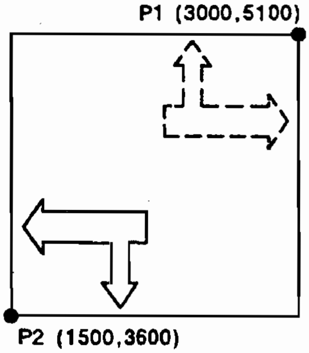
*Normal*
*(Lines 20-40)*



*Reversed*
*(Lines 50-60)*



*Upside Down*
*(Lines 70-80)*

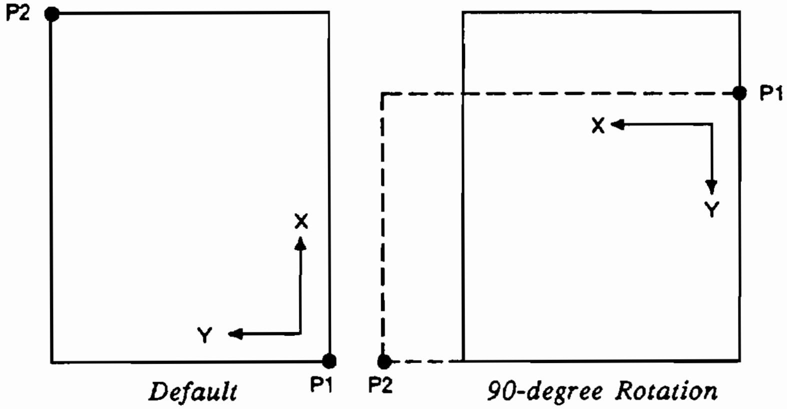


*Upside Down and Reversed*
*(Lines 90-100)*

## 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 and rotated orientation of the axes and locations of P1 and P2.



*Default* *90-degree Rotation*

Note that P2 is now off the page. This occurs because the X,Y coordinates of P1 and P2 do not change. You can use the IP instruction after the RO instruction to set the locations of P1 and P2 within the hard-clip limits. If you reset your coordinate system to its default orientation, remember to reset P1 and P2 (using the IP instruction again).

## Using the Output Instructions in This Chapter

**NOTE:** Do not use output instructions if the plotter will be used on a network which doesn't accomodate two-way communication, or if your program will be sent on a Centronics interface. ■

When changing a plot's size or orientation, you often need to know the hard-clip limits and the locations of P1 and P2. You can get this information from the plotter by using the output hard-clip limits (OH) and output P1 and P2 (OP) instructions described later in this chapter.

Output instructions are a special type of graphics instructions. They perform no plotting. They ask for information from the plotter to the computer. Output instructions can be used only in HP-IB and RS-232-C configurations. Read *Notes for Obtaining Plotter Output* in Chapter 10 before using the output instructions in this chapter.

## 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.

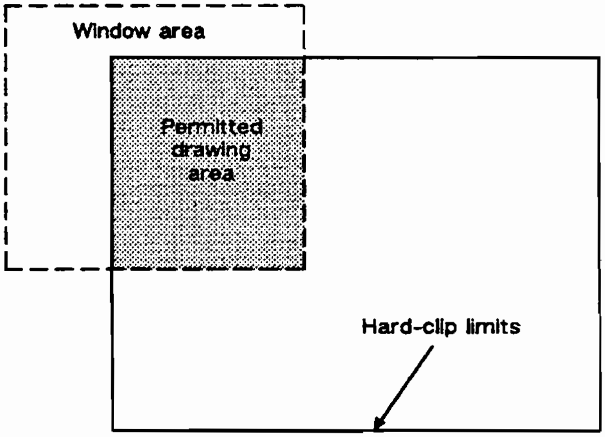
**SYNTAX:** IW *XLL,YLL,XUR,YUR*(;)
or
IW(;)

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates | current units | −67 108 863 to 67 108 863 | hard-clip limits |

**REMARKS:** The plotter interprets the parameters as follows.

- **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 and upper-right corners. Coordinates are interpreted in current units: as user units when scaling is on; as plotter units when scaling is off.

When you turn the plotter on, the window is automatically set to the hard-clip limits. You can define a window that extends beyond the hard-clip limits; however, the plotter can not 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.



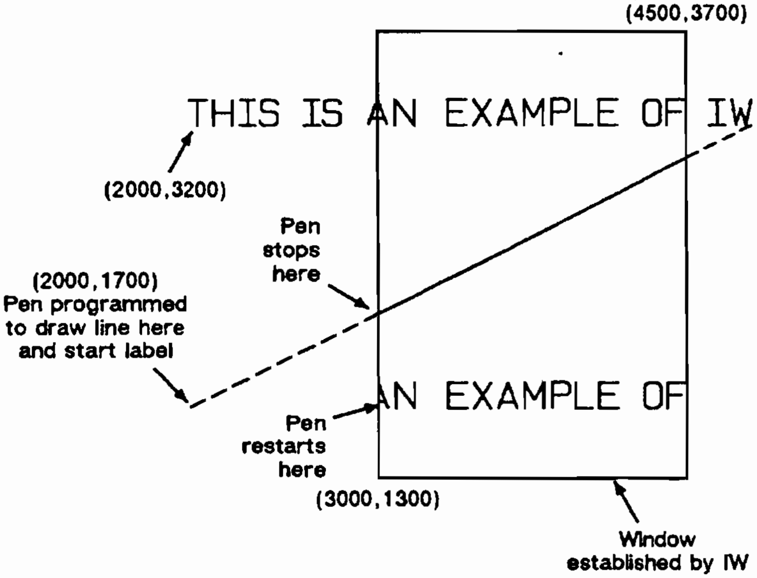
If the window falls entirely outside of the hard-clip limits, no plot will be drawn. This can happen when you define a window that is normally within the hard-clip limits and a subsequent rotate coordinate system (RO) instruction moves the window outside of the hard-clip limits.

When you define a window in user units, the size of window changes with any subsequent changes in P1 or P2. However, sending a subsequent SC instruction binds the window to its equivalent plotter units.

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

**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 clipped after the window has been established but not before.

```
10  'Insert configuration statement here
20  PRINT #1, "INPS8000,10000SP1"
30  PRINT #1, "SI.2,.35PA2000,3200"
40  PRINT #1, "LBTHIS IS AN EXAMPLE OF IW"+CHR$(3)
50  PRINT #1, "IW3000,1300,4500,3700"
60  PRINT #1, "PD2000,1700"
70  PRINT #1, "LBTHIS IS AN EXAMPLE OF IW"+CHR$(3)
80  PRINT #1, "PU3000,1300PD4500,1300,4500,3700"
90  PRINT #1, "PD3000,3700,3000,1300PU"
100 PRINT #1, "PG;"
110 END
```



**RELATED INSTRUCTIONS:**

| | |
|---|---|
| IP, | Input P1 and P2 |
| IR, | Input Relative P1 and P2 |
| SC, | Scale |

## 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 |
|---|---|---|---|
| — | XLL,YLL,XUR,YUR | integer | hard-clip limits |

**REMARKS:** The coordinates are always expressed in plotter units and represent the lower-left and upper-right corners of the hard-clip limits. After sending the OH instruction, have your program immediately read the plotter's output response.

| Plotter | Hard-Clip Limits | |
|---|---|---|
| D/A1-size (Model 240D) | X: 0 to 35 376 | Y: 0 to 24 000 |
| E/A0-size (Model 240E) | X: 0 to 47568 | Y: 0 to 35 840 |

**RELATED INSTRUCTIONS:** PS, Plot Size

## 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 |
|---|---|---|---|
| — | P1X,P1Y,P2X,P2Y | integer | -67 108 863 to 67 108 863* |

*P2 tracks P1 and may be outside this 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 in Chapter 10).

**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, "INOP;"
30 INPUT #1, A,B,Y,Z
40 PRINT A,B,Y,Z
50 END
```

**RELATED INSTRUCTIONS:**

IP, Input P1 and P2
IR, Input Relative P1 and P2
OS, Output Status
PS, Plot Size

## RO, Rotate Coordinate System

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

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

| Parameter | Format | Range | Default |
|---|---|---|---|
| angle | clamped integer | 0, 90, 180, or 270 degrees | 0 degrees |

**REMARKS:** The plotter interprets the parameters as follows:

- No Parameter — Defaults the orientation of the coordinate system to 0 degrees (horizontal). Equivalent to (*RO0*).

- **Angle** — Specifies the degree of rotation.

| | |
|---|---|
| **0** | Sets the orientation to horizontal. |
| **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. |

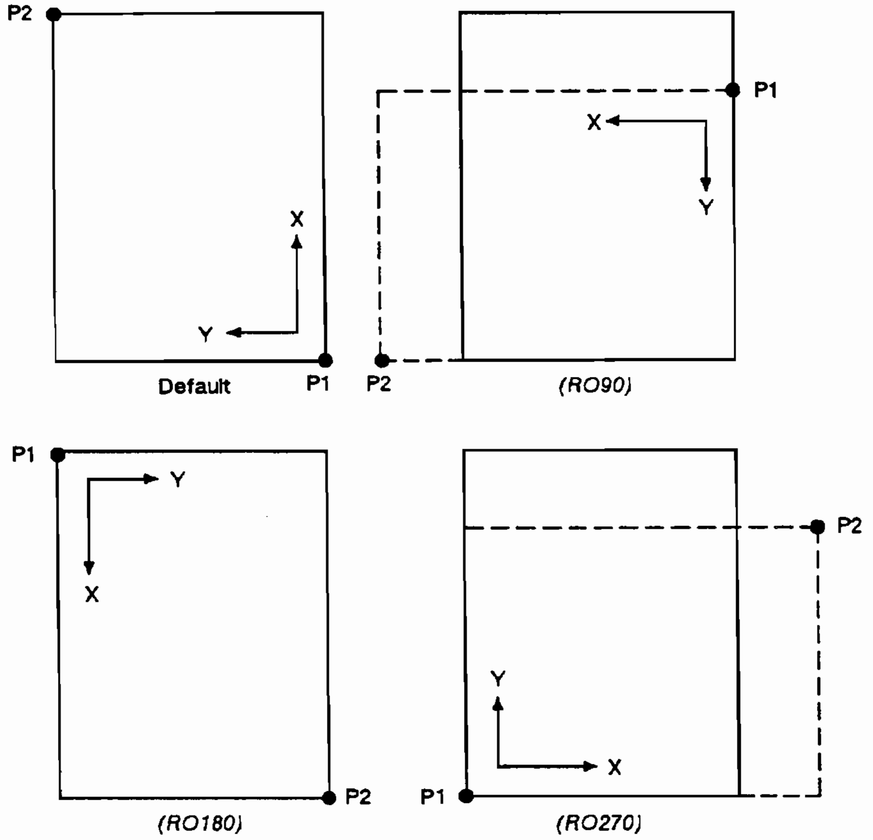
Scaling points P1 and P2 rotate with the coordinate system. However, they maintain the same X,Y coordinate values as before the rotation. This means that P1 and P2 can be located outside of the hard-clip limits. Follow (*RO90*) or (*RO270*) with (*IP*) or (*IR*) to relocate points P1 and P2 within the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

The soft-clip limits are also rotated if IW has been set. If a portion of a window is rotated outside the hard-clip limits, it is clipped. Use (*IW*) to reset the window to the hard-clip limits.

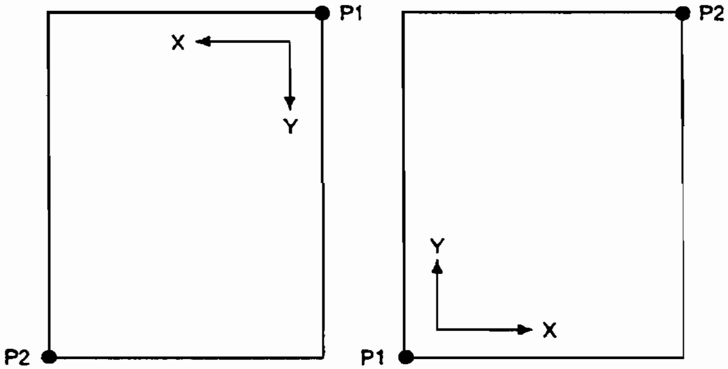
Note that the pen location does not change when you rotate the coordinate system. Instead, the plotter updates the pen's X,Y coordinate location to reflect the new orientation.

The RO instruction remains in effect until the rotation 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.



(RO90IP) (RO2701P)

Refer to Example 4 in Appendix C for additional illustrations of plot rotation.

**RELATED INSTRUCTIONS:**

IP, Input P1 and P2
IR, Input Relative P1 and P2
IW, Input Window

# 10

# Obtaining Information from the Plotter

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

- Use output instructions in HP-IB and RS-232-C configurations.

- Use output instructions to obtain error, plotter, and status byte information.

The following instructions are described in this chapter.

OE, Output Error
OI, Output Identification
OS, Output Status

## Notes on 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 computer 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 computer 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 only supports data transmission in one direction. 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 instruction (OE) until the circle is completely filled.

If you are sending device-control instructions along with HP-GL/2 output instructions, note that you can get an overwrite error if the outputs from both are simultaneous. Generally, you can prevent this by reading the output response for each output instruction or device-control instruction immediately, before sending any other instructions.

The plotter signals the end of its output response with 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 Chapter 12 in this manual and Chapter 6 in your User's Guide for more information regarding HP-IB addressing. ■

## For RS-232-C Users

The plotter outputs information according to the handshake protocol established by the ESC.P, ESC.M, ESC.N, and ESC.@ instructions. Use these instructions to specify turnaround delays and intercharacter delays as necessary to prepare your computer to receive the output response. Your computer documentation should specify whether or not such delays are required.

The plotter signals the end of its output response with an output response terminator, noted in the 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 none of the output instructions use parameters.

2. Read the plotter's output response immediately using a 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) simultaneously connected to your computer, it may be necessary to have the plotter output its identification so that you know it is on-line. 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 are the only errors that set the error bit of the status byte and response to the output status (OS) instruction. (Status byte information follows this section.)

### 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.

*Output Responses Types*

| Instruction | Parameters Returned* | Type and Range |
|---|---|---|
| OE | Error number | Integer, 1 digit |
| OH | XLL, YLL<br>XUR, YUR | Integer, ≤ 6 digits (plotter units) |
| OI | Model number | 6-character string |
| OP | P1X, P1Y<br>P2X, P2Y | Integer, ≤ 8 digits |
| OS | Status | Integer, ≤ 3 digits |

*In addition to these paramaeters, the output terminator [TERM] is always sent at the end of output, and commas are sent to separate parameters.

## OE, Output Error

**USE:** Outputs a number corresponding to the type of HP-GL/2 error (if any) received by the plotter after the most recent IN instruction, front-panel reset, or OE instruction. 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 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. |
| 2 | Wrong number of parameters. |
| 3 | Out-of-range parameter, or invalid character. |
| 4 and 5 | Not used. |
| 6 | Position overflow. |
| 7 | Buffer overflow/out of memory. |

Executing an OE instruction clears bit position 5 of the status byte (if previously set).

**EXAMPLE:** In the following program, line 20 contains two errors.

```
10   'Insert configuration statement here
20   PRINT #1, "INSP1PA1000,1000,20EDOE;"
30   INPUT #1, A
40   PRINT A
50   END
```

```
20   PRINT #1, "INSP1PA1000,1000,20EDOE;"
```

first error (wrong number of parameters)

second error (instruction not recognized)

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.

**RELATED INSTRUCTIONS:** ESC.E, Output Extended Error

## 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 on-line, 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 | C1600A or C1601A | character string | — |

**REMARKS:** The plotter outputs its order number and letter as a six-character string. For your reference, the following table lists the different means of identifying your plotter.

| Plotter Size | Family | Model Number | Order Number |
|---|---|---|---|
| D/A1 | 7600 Series | Model 240D | C1600A |
| E/A0 | 7600 Series | Model 240E | C1601A |

**EXAMPLE:** The following program outputs the plotter identification.

```
10 'Insert configuration statement here
20 PRINT #1, 'INOI;"
30 INPUT #1, A$
40 PRINT A$
50 END
```

**RELATED**
**INSTRUCTIONS:** ESC.A, Output Identification

## 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 No. |
|---|---|---|
| 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.

```
10 'Insert configuration statement here
20 PRINT #1, "INOS;"
30 INPUT #1, S
40 PRINT S
50 END
```

**RELATED INSTRUCTIONS:**

IN, Initialize
IP, Input P1 and P2
OE, Output Error
OP, Output P1 and P2
PD, Pen Down

# 11

# Using Device-Control Instructions

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

- Send device-control instructions.
- Work with device-control syntax and omit optional parameters.
- Use device-control instructions to establish default conditions; identify the plotter; monitor buffer sizes; and verify errors or operating conditions.

The following instructions are described in this chapter.

| | |
|---|---|
| ESC.A, | Output Identification |
| ESC.B, | Output Buffer Space |
| ESC.E, | Output Extended Error |
| ESC.J, | Abort Device Control |
| ESC.K, | Abort Graphics |
| ESC.L, | Output Buffer Size When Empty |
| ESC.O, | Output Extended Status |
| ESC.R, | Reset |
| ESC.S, | Output Configurable Memory Size |
| ESC.@, | Set Plotter Configuration |

## Working with Device-Control Instructions

Device-control instructions serve two basic purposes: to provide information about certain internal plotter conditions, and to control data transfer between the computer and plotter. Device-control instructions differ from graphics instructions in the following ways.

- Device-control instructions are not represented by mnemonics; they consist of the single ASCII character **ESC** (decimal code 27) followed by a period ( . ) and a character that represents the instruction's unique function.

- Device-control instructions follow specific syntax conventions that are different from the graphics instructions.

- Device-control instructions don't enter the plotter's buffer but are processed immediately. (Graphics instructions enter the buffer and are processed in the order they are received.)

### Sending Device-Control Instructions

The principles for sending device-control instructions to the plotter are the same as for graphics instructions; send them as a literal string in an output statement such as PRINT or OUTPUT.

Send the **ESC** character the same way you send other ASCII characters; either by using a character string function such as CHR$, or by producing the character directly from the keyboard. Send the period and the final character as a literal string. You may need to link the CHR$ function with the literal string using one of the following symbols: +, &, or ;. Use the symbol required by your system or language.

The examples in this chapter and Chapter 12 use the CHR$ function with the concatenation (linking) symbol +. The following shows the ESC.L instruction.

```
PRINT #1, CHR$(27)+".L"
```

## Syntax for Device-Control Instructions

A complete device-control instruction includes the three character sequence consisting of **ESC** and . followed by the character or uppercase letter indicating its function (e.g., @, A, or M). Some instructions also include parameters and a terminator according to the following syntax conventions.

**ESC** The single ASCII character, escape (decimal code 27) is shown in bold, condensed print.

( ) Parameters in parentheses are optional.

; Separates parameters. A semicolon without a parameter sets the parameter to its default value.

: Terminates any instruction that uses parameters, whether or not the parameters are used. Instructions that do not have parameters need no terminator.

[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: RS-232-C, a carriage return (**CR**); and HP-IB, carriage return and line feed (**CR LF**)

### Omitting Optional Parameters

Unlike the graphics instructions, you can set a parameter of a device control instruction to its default value by entering only the semicolon or colon. Thus, you *can* send an instruction such as (**ESC**.M50;;10;:). The omitted parameters are set to their default values.

# Using Device-Control Instructions

All device-control instructions with the word 'output' in their title cause the plotter to send information (called an output response) back to your computer. Immediately after sending a device-control instruction, have your program read the output response into a variable. Refer to *Notes on Obtaining Plotter Output* in Chapter 10.

## Establishing Default Conditions

You should already be acquainted with the following methods for establishing known plotter conditions.

- Turning the power switch off then on.

- Pressing **RESET** on the front control panel.

- Sending the DF or IN instructions to your plotter.

The first two methods require your presence at the plotter. The third method is not immediate; the DF and IN instructions are processed in the input buffer with other graphics instructions in the order they are received. The following three device-control instructions can establish specific default conditions and/or immediately clear the buffers of any HP-GL/2 instructions.

- ESC.J, Abort Device-Control, aborts any device-control instruction partially decoded.

- ESC.K, Abort Graphics, aborts any partially sent HP-GL/2 graphics instruction and clears the input buffer of all remaining graphics instructions (but does not clear the plotter's disc).

- ESC.R, Reset, resets certain input conditions to power-up default states, including clearing the buffers. (In an RS-232-C interface, this instruction sets the handshake protocol to default conditions.)

Note that you can also use the ESC.M, Output Mode, instruction (in an RS-232-C interface) to change the output terminator from its default value (carriage return). This instruction, however, is primarily used in establishing certain handshaking conditions and is discussed in detail in Chapter 12.

## Identifying the Plotter

When using the plotter in a remote location, you may need to confirm that the plotter is connected to your computer, and that it is turned on. Use the ESC.A instruction to immediately output the plotter's model number and firmware revision level.

## Monitoring the Buffer Sizes

You can use the following device-control instructions to monitor the sizes of the plotter's buffers.

- ESC.B, Output Buffer Space, outputs the number of bytes currently available in the logical input buffer.

- ESC.L, Output Buffer Size When Empty, outputs the size of the logical input buffer when empty.

- ESC.S, Output Configurable Memory Size, outputs the current size of allocated memory.

## Verifying Errors or Operating Conditions

The following device-control instructions are useful in debugging operations.

- ESC.E, Output Extended Error, outputs any input error related to device-control instructions, and is used in RS-232-C block input error checking.

- ESC.O, Output Extended Status, outputs information about the plotter's current operating status.

# ESC.A, Output Identification

**USE:** Outputs the plotter's model number and firmware revision level.

**SYNTAX:** ESC.A

| Parameter | Response | Format | Range |
|---|---|---|---|
| none | order number | character string | C1600A or C1601A |
| | firmware revision level | integer | 1 to 32 767 |

**RELATED INSTRUCTIONS:** OI, Output Identification

## ESC.B, Output Buffer Space

**USE:** Outputs the plotter's currently available logical input buffer space.

**SYNTAX:** ESC.B

| Parameter | Response | Format | Range |
|---|---|---|---|
| none | available logical input buffer space | integer | 0 to 4096 bytes |

**REMARKS:** The plotter outputs the number of unused bytes in the logical input buffer. If your program continually interrogates the plotter until the response indicates a specific amount of available space, use a pause routine to allow the plotter to execute other instructions; this increases the amount of available space.

*For HP-IB Users Only:* The plotter outputs only four responses when using a parallel interface. These responses are interpreted as follows.

| | |
|---|---|
| **0** | Buffer full. |
| **1** | Buffer at least half full. |
| **2049** | Buffer less than half full. |
| **4096** | Buffer empty. |

**RELATED INSTRUCTIONS:** ESC.L, Output Buffer Size When Empty

## ESC.E, Output Extended Error

**USE:** Outputs the error number for any input error related to device-control instructions. Use OE when debugging a program to determine which programming errors have occurred (if any). *For RS-232-C users,* use with ESC.@ for block input error checking.

**SYNTAX:** ESC.E

| Parameter | Response | Format | Range |
|---|---|---|---|
| none | error number | integer | 0, 10 to 18 |

**REMARKS:** The plotter's response is one of the following error numbers. *For RS-232-C users,* all numbers listed are valid. *For HP-IB users,* the valid error numbers are 0 and 11 through 14.

| Error # | Description |
|---|---|
| 0 | No input error has occured. |
| 10 | (*RS-232-C only*) New output was requested before previous ouput was finished being transmitted. The new output will be ignored (thus causing the error). |
| 11 | Invalid character received after first two character (ESC.) in a device-control instruction. |
| 12 | Invalid character received while parsing a device-control instruction. The parameter containing the invalid character and all following parameters are defaulted. |
| 13 | One or more parameters are out of range. |
| 14 | Too many parameters received. Additional parameters beyond the proper number are ignored; parsing of the instruction ends when a colon (normal termination) or the next ESC character (abnormal termiation) is received. |
| 15 | (*RS-232-C only*) A parity error has been detected. |
| 16 | (*RS-232-C only*) The physical input buffer has overflowed. As a result, one or more characters have been lost; an HP-GL/2 error will probably occur. |
| 18 | Input error of indeterminate cause. |

*RS-232-C Users Only:* In addition to checking input errors, you can use the ESC.E instruction to start and end a data block for block input checking. If block input error checking has been activated (bit 4 of the second parameter of the ESC.@ instruction is set to 1), the ESC.E instruction will terminate a block.

If there are no input errors (response to ESC.E is 0), the plotter executes the data normally. If the response to ESC.E indicates an input error, the plotter discards the entire data block. You can then retransmit the entire block of data and prevent errors in the plot.

**RELATED INSTRUCTIONS:** OE, Output Error
ESC.@, Set Plotter Configuration

## ESC.J, Abort Device-Control

**USE:** Aborts any device-control instruction that may be partially decoded or executed. Use ESC.J in an initialization sequence when you first access the plotter.

**SYNTAX:** ESC.J

**REMARKS:** Unspecified parameters of partial instructions are defaulted. All pending or partially transmitted output requests, from either HP-GL/2 or device-control instructions, are immediately terminated, including handshake outputs. Intermediate output operations, such as turnaround delay and echo suppression, are aborted, and buffer input is enabled. Only the specified execution of an output operation is aborted. The handshake and output mode parameters remain as specified.

**RELATED INSTRUCTIONS:** ESC.K, Abort Graphics

## ESC.K, Abort Graphics

**USE:** Aborts any partially decoded HP-GL/2 instruction and discards remaining instructions in the input buffer. Use ESC.K as part of an initialization sequence when starting a new program or to terminate plotting of HP-GL/2 instructions in the buffer. The portion on the plot already stored on disc is not affected.

**SYNTAX:** ESC.K

**REMARKS:** This instruction allows the instruction or vector being executed to finish, aborts any partially decoded instruction, and clears the input buffer of all remaining graphic instructions. The plotter finishes storing its current vector on the disc, but all remaining vectors are discarded.

**RELATED INSTRUCTIONS:** ESC.J, Abort Device Control

## ESC.L, Output Buffer Size When Empty

**USE:** Outputs the size (in bytes) of the logical input buffer.

**SYNTAX:** ESC.L

| Parameter | Response | Format | Range |
|---|---|---|---|
| none | available logical input buffer space | integer | 2 to 4096 bytes* |

**For HP-IB users*, the input buffer is always 4096 bytes.

**REMARKS:** The plotter outputs an ASCII integer equal to the number of bytes allocated to the logical input buffer. The response is not transmitted by the plotter until the buffer is empty.

**NOTE:** If your plotter is in block mode, using ESC.L could disrupt communication. If you use ESC.L while in block mode, send it immediately after reading the ESC.E response. ■

**RELATED INSTRUCTIONS:** ESC.B, Output Buffer Space

## ESC.O, Output Extended Status

**USE:** Outputs the plotter's extended status. Use this instruction to obtain information about the current operating status of the plotter.

**SYNTAX:** ESC.O

| Parameter | Response | Format | Range |
|---|---|---|---|
| none | operating status | integer | 0 to 63 |

**REMARKS:** When used with an RS-232-C interface, the instruction is subject to any turnaround or intercharacter delays specified by ESC.M and ESC.N.

The operating status is the decimal equivalent of a 16-bit extended status word. The status word bits are defined in the following table. Bits 6–15 are always zero.

| Bit Position | Logic State | Decimal Value | Meaning |
|---|---|---|---|
| 0 | 0 | 0 | Not used. |
| | 1 | 1 | Roll media loaded. |
| 1 | 0 | 0 | ‘Clean’ media loaded or at power-up. |
| | 1 | 2 | Current page is not clean (PA or other drawing command executed). Set after executing PD or any instruction with an automatic pen down. |
| 2 | 0 | 0 | No media advance (no PG instruction executed) since last ESC.O instruction. ESC.O resets the bit to 0. |
| | 1 | 4 | Plotting done. |
| 3 | 0 | 0 | Input buffer not empty. |
| | 1 | 8 | Input buffer empty and ready for data. |
| 4 | 0 | 0 | Control-panel **ON-LINE** light on. |
| 5 | | | Processing HP-GL/2 instructions. |
| 4 | 1 | 16 | Control-panel **ON-LINE** light off. |
| 5 | 0 | 0 | Media loaded but graphics suspended. |
| 4<br>5 | 0<br>1 | 0<br>32 | Out of paper detected. Media not loaded, graphics suspended. |

**RELATED INSTRUCTION:** OS, Output Status

## ESC.R, Reset

**USE:** Resets certain input conditions to power-up default states. Use ESC.R to establish known conditions when starting a new plot.

**SYNTAX:** ESC.R

**REMARKS:** The ESC.R instruction aborts any device-control instruction currently in use, aborts any partially parsed HP-GL/2 instruction, resets the parser, clears all buffers.

ESC.R is equivalent to sending the following three instructions:

ESC.J

ESC.K

ESC.P: (without parameters)

(Refer to the detailed descriptions of ESC.J and ESC.K in this chapter. Refer to Chapter 12 for a detailed description of the ESC.P instruction.)

After sending ESC.R, send the ESC.L instruction to ensure that all conditions have been reset before any subsequent instructions are parsed. In this application, the ESC.L response is not important. However, to avoid potential errors, read the output response before sending more data. Refer to the ESC.L instruction for additional information.

**RELATED INSTRUCTIONS:**

| | |
|---|---|
| ESC.J, | Abort Device Control |
| ESC.K | Abort Graphics |
| ESC.P, | Set Handshake Mode |

## ESC.S, Output Configurable Memory Size

USE: Outputs the total memory size of user-definable RAM, or the memory space available in one of its two configurable buffers: the polygon buffer and downloadable character buffer. Use ESC.S to determine how much memory is currently allocated to each buffer or to confirm the allocation performed by ESC.R.

SYNTAX: ESC.S *n*:

| Parameter | Response | Format | Range |
|---|---|---|---|
| n | (table follows) | integer | 0 to 16 900 |

RESPONSE:

| Parameter | Response | Format | Range |
|---|---|---|---|
| none | allocated memory size | integer | 0 to 16 900 |

REMARKS: The following table shows each parameter value with its corresponding memory designation.

| Parameter Value | Memory Specification |
|---|---|
| 0 | total configuration memory |
| 1 | not used; 0 is output |
| 2 | polygon buffer |
| 3 | downloadable character buffer |
| ≥4 | 0 is output |

RELATED INSTRUCTION:

ESC.R Reset
ESC.@ Set Plotter Configuration

## ESC.@, Set Plotter Configuration

**USE:** *For RS-232-C users*, sets an effective logical input buffer size and controls hardwire handshake, communications protocol, and block input error checking.

**SYNTAX:** ESC.@ *(logical input buffer size);(input conditions)*:

| Parameter | Format | Range | Default |
|---|---|---|---|
| logical input buffer size | integer | 0 to 4096 bytes | 4096 |
| input conditions | integer | 0 to 31 bytes | 0 |

**REMARKS:** The physical input buffer is the actual portion of the graphics memory where storage and parsing of HP-GL/2 instructions takes place. The logical input buffer can be thought of as the operational subset of the physical input buffer; it is not a separate buffer. It is the size of the logical input buffer that limits plotter input functions such as the quantity of HP-GL/2 instructions waiting to be parsed, and threshold levels in certain handshaking methods.

*For HP-IB users only*, this instruction is ignored. The logical buffer is always 4096.

The logical input buffer *cannot* be larger than the physical input buffer. The physical input buffer is always 4096 bytes. At power-on, the logical input buffer is equal to the physical input buffer.

Use ESC.@ to change the logical input buffer size. The plotter interprets the parameters as follows.

- **Logical input buffer size** — Sets the size of the logical input buffer. If you do not specify this parameter, the plotter sets the logical input buffer to the same size as the physical input buffer.

- **Input conditions** — Specifies an integer equivalent value that controls the states of bits 0 through 4. When using an RS-232-C interface, these bits control hardwire handshake and block input error checking.

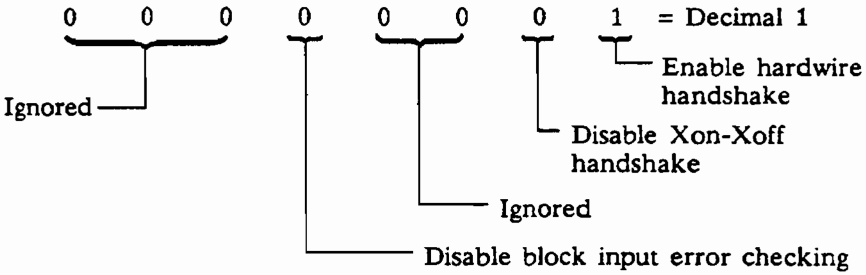
  At power-on, bits 0 and 1 are set according to the position of the **XON-XOFF/HARDWIRE** switch; bits 2 through 4 are set to zero. When hardwire handshake is enabled, the plotter uses the Data Terminal Ready (DTR) line in the same manner that buffer threshold indicators are used in the Xon-Xoff handshake. DTR is set high at power-on and is not set low until available buffer space is less than the data block size specified by either ESC.H or ESC.I (default is 80 bytes). DTR remains low until the buffer space is greater than twice the data block size available. The condition occurring first causes DTR to be set high.

The following table shows the logic states of the significant bits 0-4.

| Bit Number | Logic State | Description |
|---|---|---|
| 0 | 0 | Disable hardwire handshake mode (set and hold pin 20 of RS-232-C port connector high) |
| | 1 | Enable hardwire handshake mode. |
| 1 | 0 | Enable Xon-Xoff handshake mode. |
| | 1 | Disable Xon-Xoff handshake mode. |
| 2 | 0 | Ignored. |
| 3 | 0 | Ignored. |
| 4 | 0 | Disable block input error checking. |
| | 1 | Enable block input checking. |

**EXAMPLE:**

```
CHR$(27)+".@5000;1:"
```


0 0 0 0 0 0 0 1 = Decimal 1
Ignored
Enable hardwire handshake
Disable Xon-Xoff handshake
Ignored
Disable block input error checking

**RELATED INSTRUCTIONS:** ESC.E, Output Extended Error
ESC.L, Output Buffer Size When Empty

# 12

# Interfacing and Handshaking

This chapter provides information on the following interfaces. Read the section that applies to the interface you are using.

- Centronics parallel interfacing.
- HP-IB (IEEE-488) parallel interfacing, including normal and secondary command support modes.
- RS-232-C/CCITT V.24 serial interfacing and handshaking.
- Using device-control instructions to establish handshakes.
- Data transmission modes.

The following instructions are described at the end of the chapter.

ESC.H, Set Handshake Mode 1
ESC.I, Set Handshake Mode 2
ESC.M, Set Output Mode
ESC.N, Set Extended Output and Handshake Mode
ESC.P, Set Handshake Mode

## Centronics Information

The Centronics interface is recommended over RS-232-C for faster throughput with your plotter. The Centronics interface transfers data and commands between the computer and plotter on 15 signal lines. Eight data input lines are reserved for the transfer of data and other messages in a byte-serial, bit parallel manner. Data and message transfer is asynchronous, coordinated by three handshake lines. The remaining four lines are for management of bus activity.

Addressing is not necessary with this interface since it is not a shared bus. The following table lists and describes the Centronics connector pin assignments. (The second number in the Pen No. column refers to a ground wire that is twisted with the signal wire.)

| Pen No.* | Signal | Source | Description |
|---|---|---|---|
| 1, 19 | STROBE | Computer | A low pulse of at least 0.5 µsec causes plotter to read 1 byte of data. |
| 2, 20 | DATA1 | Computer | Data bit 0 |
| 3, 21 | DATA2 | Computer | Data bit 1 |
| 4, 22 | DATA3 | Computer | Data bit 2 |
| 5, 23 | DATA4 | Computer | Data bit 3 |
| 6, 24 | DATA5 | Computer | Data bit 4 |
| 7, 25 | DATA6 | Computer | Data bit 5 |
| 8, 26 | DATA7 | Computer | Data bit 6 |
| 9, 27 | DATA8 | Computer | Data bit 7 |
| 10, 28 | ACKNLG | Plotter | Plotter sends a low pulse indicating it has accepted a byte of data and is ready for more data. |
| 11, 29 | BUSY | Plotter | A high logic level indicates plotter can't receive data due to data entry, full buffer, or error status. |
| 12 | Out of Paper | Plotter | A high logic level indicates the plotter is out of paper. |
| 13 | SLCT | Plotter | Always high. Indicates the plotter is selected. |
| 16 | SIG GND | | |
| 17 | CHS GND | | |
| 31, 30 | INIT | Computer | A low pulse of at least 0.5 µsec resets the plotter to power-up conditions. |
| 32 | ERROR | Plotter | A low level indicates a self-test failure, out of media, plotter off-line, or any other hardware error. |
| 33 | SIG GND | | |

*Pins 14,15,18,34,35,and 36 are not used.

When using Centronics, the plotter responds to the following three device-control instructions, described in Chapter 11. (The plotter recognizes but ignores 14 additional device-control instructions, which are used in HP-IB and/or RS-232-C interfaces.)

ESC.J
ESC.K
ESC.R

The Centronics interface only supports data transmission in one direction. All output instructions are ignored when using this interface.

## HP-IB (IEEE-488) Information

The Hewlett-Packard Interface Bus (HP-IB) provides for compatibility between all devices adhering to the ANSI/IEEE-488 (1978) standard. HP-IB is the most common interface between Hewlett-Packard computers, peripherals and instruments. HP-IB is recommended over RS-232-C for faster throughput with your plotter. Refer to the *User's Guide* for instructions on setting the plotter's switches for the address code and for implementation of normal or secondary command support.

### Interface Modes: Normal or Secondary Command Support

Two command modes are available on the plotter: the standard (normal) HP-IB mode and secondary command support (SCS). To change command mode, turn off the plotter and select either HP-IB or SCS using the appropriate interface switch. Turn on the plotter; the command mode you selected is now in effect.

The following table shows what functions are implemented according to the mode selected.

| Interface Function Name | Normal Implementation | SCS Implementation |
|---|---|---|
| Source Handshake | SH1 | SH1 |
| Acceptor Handshake | AH1 | AH1 |
| Talker | T6 | — |
| Listener | L3 | — |
| Extended Talker | — | TE6 |
| Extended Listener | — | LE4 |
| Service Request | SR1 | — |
| Parallel Poll | PP2 or PP0* | PP2 |
| Device Clear | DC1 | DC1 |
| No Remote Local | RL0 | RL0 |
| No Device Trigger | DT0 | DT0 |
| No Controller | C0 | C0 |

*PP0 if listen only mode; PP1 if address > 8; PP2 if address < 8.

When using the HP-IB or SCS, the plotter responds to the following 10 device-control instructions (which can also be used in an RS-232-C interface.)

| | | |
|---|---|---|
| ESC.A | ESC.J | ESC.O |
| ESC.B | ESC.K | ESC.R |
| ESC.E | ESC.L | ESC.S |

All of the above instructions are described in Chapter 11. The plotter recognizes five additional device-control instructions which are valid *only* for RS-232-C interfacing and handshaking (described at the end of this chapter).

## Normal HP-IB Protocol

This is the standard implementation of HP-IB that establishes mechanical, electrical, timing, and data compatibility between devices. It allows high-speed communication between many peripherals on one computer port. A device on the HP-IB may function in the following ways.

- As a 'listener' that receives data sent over the bus.

- As a 'talker' that transmits data to other devices on the bus.

- As a 'controller' (computer) that regulates interaction of the devices on the HP-IB system.

The plotter functions primarily as a 'listener,' receiving data sent over the bus.

## Controlling Addressing Sequences

Using HP-IB addresses, the controller can identify and individually access various devices on the interface. Select the HP-IB address using the plotter's interface switches (refer to the User's Guide).

An addressing sequence is made up of three major parts, with the ATN (Attention) line true.

*<Unlisten Command> <Talk Address> <Listen Addresses>*

The purpose of these parts is as follows.

- **Unlisten command** — A universal bus command; its character is ? (ASCII decimal code 63). It unaddresses all listeners. After transmitting the unlisten command, no active listeners remain on the bus.
- **Talk address** — Indicates the device that is to talk, or send data. A new talk address automatically unaddresses the previous talker.
- **Listen addresses** — Indicate one or more devices that are to listen, or receive data. A listen address adds the designated device as listener along with other addressed listeners.

This addressing sequence directs who talks to whom. You can implement the commands (unlisten, talk, listen) by putting data on the bus and setting the ATN control line true. The unlisten command (?) plays a vital role in this sequence. It is important that a device receive only the data that is intended for it.

When a new talk address is transmitted in the addressing sequence, the previous talker is unaddressed. Therefore, only the new talker can send data on the bus and you don't need to use an untalk command in the same manner as the unlisten command.

For example, to tell a computer at address 21 to talk and a plotter at address 05 to listen, the controller (usually the computer) sets the proper control line true and sends the following sequence.

**? U %**

where: ? — tells all devices on the bus to unlisten,
U — designates the device at address 21 as the talker,
% — designates the device at address 05 as the listener.

To have the plotter talk and the computer listen, you would set the control lines and set the following, with ATN true.

```
? E 5
```

## Listen-Only Mode

In listen-only mode, the plotter plots all data transmitted over the bus without being addressed by a computer.

To activate listen-only mode, turn on the **LISTEN ONLY** switch as shown in the *User's Guide*.

**NOTE:** Listen-only mode is disabled if Secondary Command Support is selected for the plotter. ■

## Serial and Parallel Polling

Serial and Parallel Polling are processes used by the controller to determine if any of the devices (such as plotter and printers) on the HP-IB require service. Like most HP-IB devices, your plotter can request service.

### Serial Polling

Refer to your computer's documentation to determine whether or not your system has serial poll capability, and for the necessary (enable/disable) commands.

In a serial poll, the computer polls the devices on the bus in sequence, one at a time. The computer requests the status of one specific device by addressing only that device and asking it to respond. The plotter responds to this request by sending a status byte. The definition of each bit in the status byte is shown in the following table.

*Status Byte Bit Definition*

| Bit | Bit Value When Set | Condition Indicated When Bit Set to 1 |
|---|---|---|
| 0 | 1 | Pen is down. |
| 1 | 2 | P1 or P2 changed; cleared by OP. |
| 2 | 4 | (Reserved) |
| 3 | 8 | Initialized; cleared by OS. |
| 4 | 16 | Ready for data; always set to 1. |
| 5 | 32 | HP-GL/2 error occured; cleared by OE. |
| 6 | 64 | (Reserved) |
| 7 | 128 | Not used; always set to 0. |

Use an interrupt routine to halt your program when the computer receives a service request. Then, send a serial poll to determine which device on the line needs service. Sending a serial poll clears bit 6 of the status byte.

NOTE: The plotter cannot respond to a serial poll when it is in either listen-only mode or secondary command support mode. ■

### Parallel Polling

Refer to your computer's documentation to determine if your system has parallel polling capability and for the necessary (enable/disable) commands.

Parallel polling provides a fast way for the computer to determine which, if any, of the devices on the bus needs service. The computer performs a parallel poll by asserting the EOI (End or Identify) and ATN (Attention) lines simultaneously. If it requires service, the plotter responds by asserting its assigned DI/O line.

*When enabled in normal HP-IB mode*, the plotter responds to a parallel poll when it is out of media. Loading media terminates the plotter's response.

*When enabled in SCS mode*, the plotter responds to a parallel poll under the following conditions: ready for data, ready to transmit, media error, and power up. Eliminating the cause (loading media, for example) terminates the response.

NOTE: In listen-only mode, the plotter cannot respond to a parallel poll. ■

### Bus Commands

Devices on the HP-IB receive special instructions in the form of commands. To send a command over the bus, the controller asserts the ATN line. Once the ATN line is asserted, the devices on the bus understand that what follows is a command, not data.

#### Listen Address (LAD) X01AAAAA

AAAAA represents the HP-IB address of the device for which the command is intended. This command causes the plotter to become a listener.

#### Device Clear (DCL) X0010100

If the plotter is parsing an instruction, DCL aborts it; if the plotter is executing an instruction (processing and storing it internally), it is completed. DCL does not affect instructions already stored on the plotter's disc.

#### Selected Device Clear (SDC) X0000100

This command clears only those devices on the bus that are selected to listen. Except for this difference, SDC has the same effect as DCL.

#### Interface Clear (IFC)

The controller uses the IFC line to override all bus operations and return the bus to a known inactive state. All pending output on the bus is cleared.

## Secondary Command Support

Your system must support secondary commands for the plotter to function in this mode. Secondary command support extends the capabilities of normal HP-IB protocol. Implementation of secondary commands avoids HP-IB lock-up in a multi-user environment.

When this protocol is selected, listen-only mode and the service request are automatically disabled. The plotter can be identified with addresses 0–7.

Secondary commands have a two-byte structure to carry their more detailed instructions. The plotter functions as Extended Talker or Extended Listener. The first byte is always a primary talk or listen command, which sets up the second byte as a more specific talk or listen instruction.

The plotter supports four secondary commands, two talk and two listen.

## Secondary Talk Commands

The secondary talk commands are Talk Data, Device Specified Jump, and I/O Status.

### Talk Data (DATA) X1100000

This command instructs the plotter to send data in response to an output command.

### Device Specified Jump (DSJ) X1110000

When the plotter receives the primary talk command followed by a secondary DSJ command, it will respond with one byte of data with the EOI line asserted. The value of the byte will be decimal zero, one, or two.

| Decimal Value of Response Byte | Plotter Status |
|---|---|
| 0 | Ready to receive |
| 1 | Ready to send |
| 2 | Status change (power cycle, out of paper, paper motion failure) |

### I/O Status X1101110

The I/O status is a means by which the controller checks the current plotter status. When the plotter receives the primary talk command followed by the secondary I/O status command, it will respond with one byte of data with the EOI line asserted. Bits 2 through 5 of the status byte are undefined. If the plotter shuts down, bit 6 of the status byte is cleared.

| Bit | Bit Value When Set | Condition Indicated When Bit Set to 1 |
|---|---|---|
| 0 | 1 | Power cycle |
| 1 | 2 | Out of paper |
| 2-5 | — | Undefined |
| 6 | 64 | Ready for data |
| 7 | 128 | On-line |

### Unrecognized Secondary Talk Commands

When the plotter receives a secondary talk command that it does not recognize, it will respond by sending a null byte (00000000) with the EOI line asserted.

## Secondary Listen Commands

The secondary listen commands are Device Clear and Data.

### Device Clear X1110000

This command performs the same function as a DCL. Device Clear is followed by a required parity byte, used by other devices, which the plotter ignores.

### Data X1100000

This command tells the plotter that bytes following the command contain data to be plotted. For maximum throughput, the plotter should receive data bursts of 1000 bytes (or less) in length. A burst length that exceeds 1000 bytes may be processed at a slightly reduced rate.

### Unrecognized Secondary Listen Commands

When the plotter receives a secondary listen command that is does not recognize, it will still acknowledge the command as valid. It then reads but ignores all incoming data bytes until it receives an Unlisten command or a data byte with the EOI asserted.

### Identify Command Sequence

This command is used by the controller to identify devices on the bus and determine their characteristics. When the plotter receives a primary untalk command followed by a secondary identify command specifying the plotter's address in the lowest three bits, it will respond with two data bytes. The first byte is a general device classification which tells the controller that the plotter is a plotter/terminal device. The second byte tells the controller that the plotter is an HP 7600A. The response bytes are shown below.

| | | |
|---|---|---|
| Byte 1 (printer/terminal) | 00100000 | (20 hex) |
| Byte 2 (HP 7600A designator) | 00010011 | (13 hex) |

## RS-232-C Interfacing and Handshaking

Interfacing establishes communication by matching a set of conditions between the computer and the plotter. Your system's requirements dictate the interface conditions you must set on your plotter. To establish compatible interface conditions, your computer and plotter must agree on the following.

- **Number of data bits** — The plotter uses standard 7-bit ASCII code; it is not compatible with 6-bit or 12-bit ASCII devices. (If data from the computer is not in this format, you need a protocol converter.)

- **Parity** — ASCII characters are coded in seven bits, with an eighth bit optionally used for parity or data. Refer to the User's Guide to set the parity the same parity as your computer.

- **Baud rate** — The rate at which transmitted data is sent (equal to bits per second). You must set the plotter's baud rate to match the baud rate of the computer; otherwise, the plotter will not be able to understand the data. Refer to the User's Guide to set the baud rate to the same as the computer.

- **Number of stop bits** — On the plotter, baud rates 75 and 110 use two stop bits while baud rates of 300 and faster use one stop bit.

If your computer is not listed in the User's Guide, check your computer system documentation to determine the values for the above information.

## Choosing a Handshake

Once you establish the interface conditions, you must select a handshake method. The plotter can implement any of the following four handshakes. (Refer to *Connecting Your Plotter to a Computer* in your *User's Guide* for information on setting the plotter's configuration switches for your interface.)

- Xon-Xoff (factory default).

- Hardwire.

- Enquire/acknowledge (ENQ/ACK).

- Software checking.

**NOTE:** Using the Xon–Xoff handshake is strongly recommended. ■

Handshaking establishes the manner in which your plotter and computer transmit data once the interface is established. A handshake method is required to prevent data from being lost or misinterpreted.

For information on which handshake to use, consult your system's documentation or the installation manual for your computer and/or graphics software package. Most graphics software packages designed for use with RS-232-C plotters contain the instructions necessary to set up a handshake. You may need to provide your software with parameters suitable for your system. Instructions for setting the parameters can be found in the software installation guide.

The following paragraphs describe each of the handshake methods listed above. Use these descriptions along with your computer documentation to determine the handshake you should use. You must know your computer's requirements to have your computer and plotter communicate efficiently.

### Xon-Xoff Handshake

The Xon-Xoff handshake is the plotter's factory default handshake and is strongly recommended. The Xon-Xoff handshake is controlled by the plotter. You can use this handshake *only* if your computer supports an Xon-Xoff protocol. When using the Xon-Xoff handshake, the plotter transmits a control character to the computer when the plotter's I/O buffer is almost full, and another character when the buffer is ready to receive more data.

### Hardwire Handshake

The hardwire handshake uses the plotter's Data Terminal Ready (DTR) control line (pin 20) to control handshaking. You can use this handshake if your computer can monitor pin 20.

**NOTE:** Using a hardwire handshake in a PC environment is likely to cause device timeouts. Since the DTR line remains low while the plot is rasterized and executed, the PC is likely to time-out while waiting for the DTR to go high. ■

The hardwire handshake requires that the plotter be connected directly to the computer; there can be no intermediary devices (such as modems).

### Enquire/Acknowledge Handshake

The enquire/acknowledge (ENQ/ACK) handshake is controlled by the computer. The computer sends an enquire character that prompts the plotter to respond when there is enough room in the buffer for more data. When sufficient input buffer space is available, the plotter sends an acknowledgment string that signals the computer to send data. The enquire/acknowledge handshake is not as efficient as the Xon-Xoff handshake.

### Software Checking Handshake

Probably the least efficient of the handshakes, you can implement this method on almost any computer system. You *must* use it, however, if your computer system can not implement any of the other handshaking methods. This handshake is managed by the programmer. To use this handshake, you must include device-control instructions in your program to repeatedly check the availability of input buffer space.

## Handshaking Through Device-Control Instructions

You can establish each handshake programmatically using device-control instructions. This enables you to switch from one handshake to another for different applications.

The following list of device-control instructions directly affect your handshake and handshake capabilities. Use these instructions in combination with the device-control instructions in Chapter 11 to enhance the communication between your computer and plotter.

| | |
|---|---|
| ESC.H, | Set Handshake Mode 1 |
| ESC.I, | Set Handshake Mode 2 |
| ESC.M, | Set Output Mode |
| ESC.N, | Set Extended Output and Handshake |
| ESC.P, | Define Handshake |

### Xon-Xoff Handshake

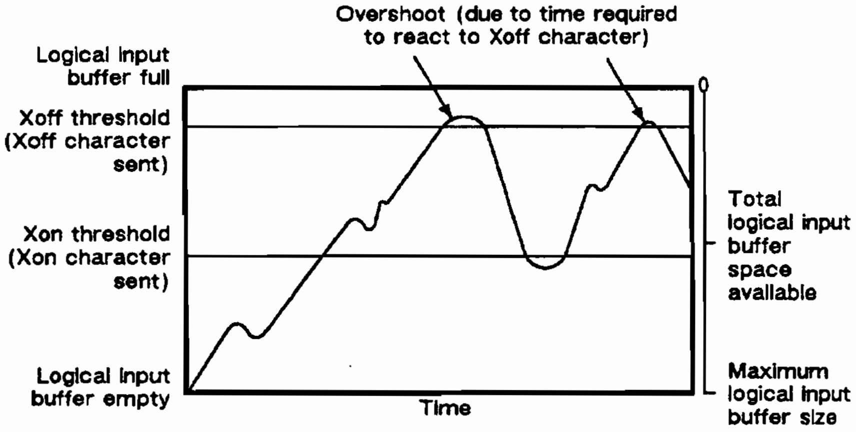
Xon-Xoff is the factory default handshake. Using an Xon-Xoff handshake is strongly recommended. Refer to your computer system's documentation to determine whether or not you can use the Xon-Xoff handshake. Your plotter's **XON-XOFF/HARDWIRE** switch must be set correctly, refer to *Connecting Your Plotter to a Computer* in your *User's Guide*.

When using the Xon-Xoff handshake method, the plotter controls the data exchange sequence by signaling the computer when it has sufficient room in its input buffer for data and when to stop sending data. The plotter uses buffer threshold indicators (an Xon trigger level and an Xoff trigger leve) to prevent buffer overflow, as follows.

1. IF data enters the plotter's buffer faster than it can be processed (unlikely even at 19 200 baud), and the buffer starts to fill. When the data in the input buffer reaches the Xoff threshold level, the plotter sends the Xoff trigger character to the computer.

2. The plotter's buffer empties as data is processed. When the Xon threshold level is reached, the plotter sends the Xon trigger character to the computer, restarting the flow of data.

This process is repeated until all data has been sent. The following is a graphic representation of this process.



**NOTE:** There is a delay between the time the signal is sent from the plotter and the time the computer stops sending data. Allow extra buffer room to avoid losing data. ■

Use either of the following device-control instructions to initiate a Xon-Xoff handshake in your program.

ESC.P
ESC.I and ESC.N

### Using ESC.P To Initiate an Xon-Xoff Handshake

Using ESC.P with a parameter of 1 (ESC.P1:) sets several Xon-Xoff parameters for you. It is the simplest way to initiate an Xon-Xoff handshake with a device-control instruction. Be aware that these parameter values may not be the best for your computer. If your computer requires other values, consider the following instructions.

- ESC.I to change the Xoff threshold level and Xon character.
- ESC.N to change the intercharacter delay and Xoff character.
- ESC.M to change the turnaround delay, output trigger character, echo terminate character, and output terminator.

To override parameter values set by the ESC.P instruction, place the appropriate instruction *after* the ESC.P instruction.

### Using ESC.I and ESC.N To Initiate an Xon-Xoff Handshake

To initiate an Xon-Xoff handshake using the ESC.I instruction, you must omit its second parameter (enquiry character) or set it to zero. Also, send the ESC.N instruction to establish an intercharacter delay and the Xoff trigger character. The following shows an example using the ESC.N and ESC.I instructions. The ASCII decimal code 27 sends the ESC character.

```
CHR$(27)+".I10;;17:"
CHR$(27)+".N0;19:"
```

The parameter values defined by the above example are as follows.

| | | |
|---|---|---|
| ESC.I | | |
| Xoff threshold level | = | 10 bytes |
| Enquiry character | = | omitted (indicates Xon-Xoff) |
| Xon character | = | 17 (ASCII DC1) |
| ESC.N | | |
| Intercharacter delay | = | 0 (no delay) |
| Xoff character | = | 19 (ASCII DC3) |

These are commonly used Xon-Xoff character values. Check your system documentation to determine your system's requirements.

## Hardwire Handshake

The hardwire handshake takes place in the hardware rather than the software. To use hardwire handshake, you must connect the plotter directly to the computer with the appropriate cable (refer to the cable schematics in Appendix A of the *User's Guide*); there can be no intermediate hardware (such as modems) between the plotter and computer. You can turn hardwire handshake on or off using the plotter's interface switches. Software can also turn it on or off using the ESC.@ instruction.

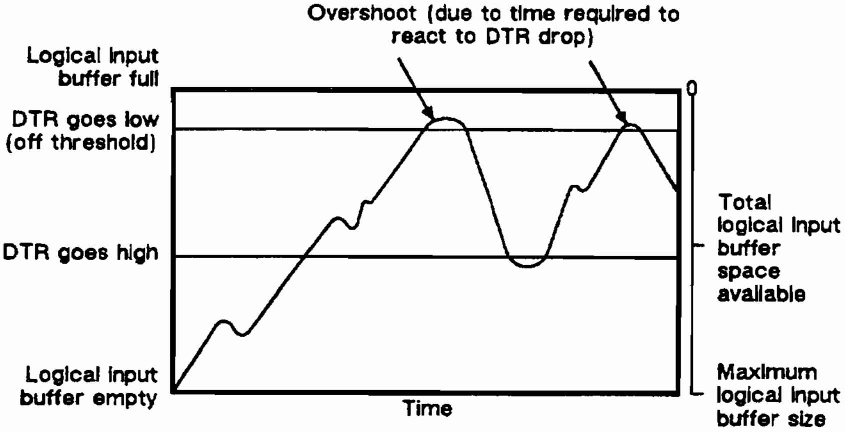
In a hardwire handshake, the computer monitors one of the interface lines from the plotter. The hardwire handshake works as follows.

1. When the plotter has room for data in its input buffer, it signals the computer to send data by setting the DTR line high (turning it on). The DTR line is pin 20 on the plotter's RS-232-C connector. The DTR line is high at power-on.

   The plotter raises the DTR line when the input buffer space is twice the data block size plus 80 bytes (default data block size is 80 bytes).

2. The plotter has a threshold level that determines when the input buffer is in danger of overflowing and losing data. When the threshold level (the data block size specified by the ESC.H or ESC.I instructions) is reached, the plotter sets the DTR line low (off).

The computer continually checks the status of the line: if the line is high, it sends data; if the line is low, the computer waits until the line is high again before sending more data. This prevents the computer from over-filling the plotter's input buffer.



NOTE: Using a hardwire handshake in a PC environment is likely to cause device timeouts. Since the DTR line remains low while the plot is rasterized and executed, the PC is likely to time-out while waiting for the DTR to go high. ■

Use either of the following device-control instructions to initiate a hardwire handshake in your program.

```
ESC.P
ESC.@
```

### Using ESC.P To Initiate a Hardwire Handshake

The ESC.P device-control instruction with a parameter of 3 (ESC.P3:) sets several parameters for you. It is the simplest method to initiate a hardwire handshake with a device-control instruction. Be aware that these parameter values may not be the best for your computer system. If your system requires other values, consider the following device-control instructions.

- ESC.M to change the output terminator.
- ESC.I to change the 'off' threshold level.

To override parameter values set by the ESC.P instruction, place the appropriate instruction *after* ESC.P.

### Using ESC.@ To Initiate a Hardwire Handshake

If a hardwire handshake has been disabled by a previous program, use ESC.@ to enable it. As with the ESC.P instruction, you can add the ESC.I instruction to set the threshold level. Since the other parameters of ESC.I are unnecessary for a hardwire handshake, they need not be included. The following shows an example using the ESC.@ and ESC.I instructions. ASCII decimal code 27 sends the ESC character. The first parameter of the ESC.@ instruction is set to its default value by using just the semicolon.

```
CHR$(27)+".@;1:"
CHR$(27)+".I1Ø:"
```

The following defines the parameter values indicated by the above example.

ESC.@
Input Buffer size = 4 96 bytes (default)
Hardwire handshake = hardwire handshake (enabled)

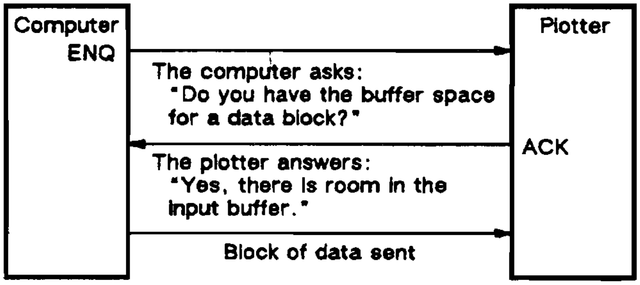
ESC.I
'Off' Threshold level = 10 bytes

## Enquire/Acknowledge Handshake

This handshake method derives its name from the two ASCII characters, ENQ and ACK, used on some systems as the enquiry character and acknowledgment string. The enquire/acknowledge handshake prevents the computer from sending the plotter more data than its buffer can accommodate. The computer sends the enquiry character to the plotter, asking if there is enough room in its input buffer for a block of data. The computer waits until it receives an acknowledgment string from the plotter signaling that there is sufficient room in the input buffer for a block of data. Only then does the computer send a block of data. In this way, the computer does not send the plotter more data than its buffer can accommodate.

When a proper enquire/acknowledge handshake is not established, the plotter responds with a 'dummy' acknowledge to any enquire character received. Refer to the ESC.H and ESC.I instruction descriptions at the end of this chapter.

The diagram below illustrates how an enquire/acknowledge handshake works.


Computer
ENQ
Plotter
The computer asks:
"Do you have the buffer space for a data block?"
ACK
The plotter answers:
"Yes, there is room in the input buffer."
Block of data sent

Use any of the following device-control instructions to initiate an enquire/acknowledge handshake in your program.

ESC.P
ESC.I
ESC.H

The method you use depends on your computer system's requirements. If your computer documentation lists the enquire/acknowledge handshake, you should consider either ESC.P or ESC.I. You can use the ESC.H instruction in systems where the output trigger character, echo terminator, and output terminator must be used with the enquire and acknowledge exchange in addition to being used with plotter output responses.

### Using ESC.P to Initiate Enquire/Acknowledge Handshake

Using ESC.P with a parameter of 2 (ESC.P2:) sets several enquire/acknowledge parameters for you. Be aware that these parameter values may not be the best for your system. If your computer requires other values, consider the following instructions.

- ESC.I to change the block size, enquiry character, and acknowledgment string.
- ESC.M to change the turnaround delay, output trigger, echo terminate, and output terminate characters.
- ESC.N to change the intercharacter delay and immediate response string.

To override the parameter values set by the ESC.P instruction, place the appropriate instruction *after* ESC.P.

### Using ESC.I to Initiate an Enquire/Acknowledge Handshake

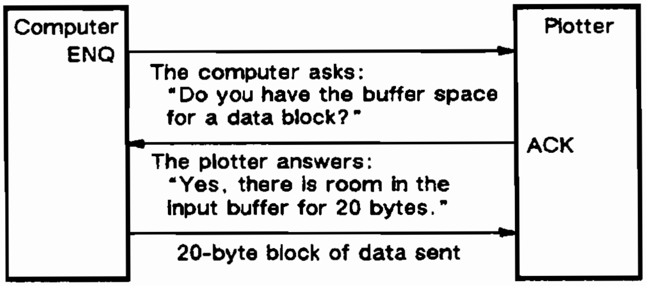
The ESC.I instruction also lets you establish an enquire/acknowledge handshake. If your computer does not *require* that you use the output trigger character, echo terminator, or output initiator with the handshake characters, use ESC.I to initiate the enquire/acknowledge handshake.

```
CHR$(27)+".I2Ø;5;6:"
```

The following lists the parameter values this instruction specifies.

| | | |
|---|---|---|
| Block size | = | 20 byte |
| Enquiry character | = | 5 (ASCII ENQ) |
| Acknowledgment string | = | 6 (ASCII ACK) |

The following illustration shows the data exchange this handshake implements.


Computer
ENQ
Plotter
The computer asks:
"Do you have the buffer space for a data block?"
ACK
The plotter answers:
"Yes, there is room in the input buffer for 20 bytes."
20-byte block of data sent

Note that you can also use the ESC.M instruction to specify the turnaround delay and the ESC.N instruction to change the intercharacter delay and immediate response string.

The following escape sequences are in effect when the plotter's default Xon-Xoff handshake is used.

```
ESC.I;;17:
ESC.N;19:
ESC.H
ESC.@;Ø:
```

Upon receipt of any escape sequence that sets up handshaking, the plotter overrides the above Xon-Xoff setting and retruns to a no-handshake state before implementing the programmed handshake. The prevents setting up ENQ/ACK using ESC.I with an unwanted immediate response string residual in the second parameter of (ESC.N;19:) (otherwise interpreted as the Xoff trigger character for Xon-Xoff).

### Using ESC.H to Initiate an Enquire/Acknowledge Handshake

If your computer cannot implement a true enquire/acknowledge handshake, you can use the ESC.H to initiate a form of a software checking enquire/acknowledge handshake that can be implemented by your software and used on all computers. This method, however, is time consuming and requires that you specify ESC.M and ESC.N parameters. The following shows an example of this generic enquire/acknowledge handshake. The ASCII decimal code 27 sends the ESC character.

```
CHR$(27)+".M;17;Ø;13;:"
CHR$(27)+".N;21:"
CHR$(27)+".H2Ø;5;6:"
```

The following lists the parameter values these instructions specify.

ESC.M

| | | |
|---|---|---|
| Turnaround delay | = | none |
| Output trigger character | = | 17 (ASCII DC1) |
| Echo terminate character | = | none |
| Output terminator | = | 13 (ASCII CR) |
| Output initiator | = | none |

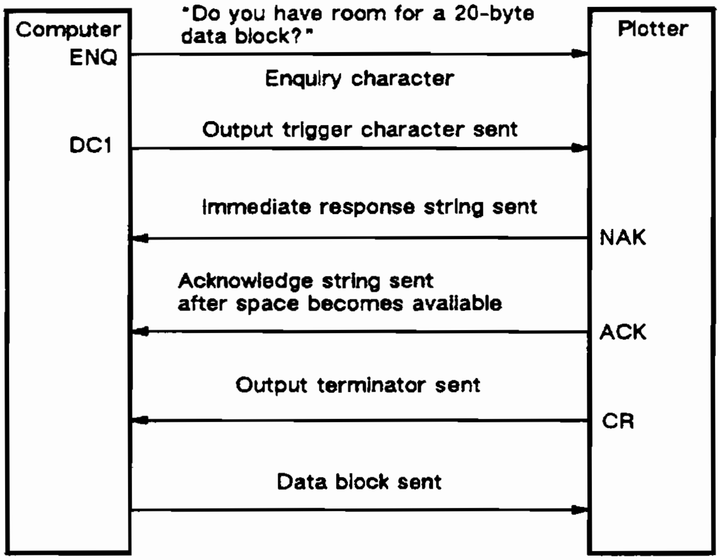
ESC.N

| | | |
|---|---|---|
| Intercharacter delay | = | none |
| Immediate response string | = | 21 (ASCII NAK) |

ESC.H

| | | |
|---|---|---|
| Data block size | = | 20 bytes |
| Enquiry character | = | 5 (ASCII ENQ) |
| Acknowledgment string | = | 6 (ASCII ACK) |

The following illustrates the data exchange this handshake would implement.


Computer
ENQ
"Do you have room for a 20-byte data block?"
Plotter
Enquiry character
DC1
Output trigger character sent
Immediate response string sent
NAK
Acknowledge string sent after space becomes available
ACK
Output terminator sent
CR
Data block sent

The following table summarizes the maximum specifications (instructions and parameters) needed to establish an enquire/acknowledge handshake using the ESC.I or ESC.H instructions. In particular, it shows which parameters of the ESC.M instruction are used with ENQ/ACK characters depending on the device-control instruction that initiates the handshake. All specified ESC.M parameters are used with plotter output responses regardless of the device-control instruction establishing the handshake.

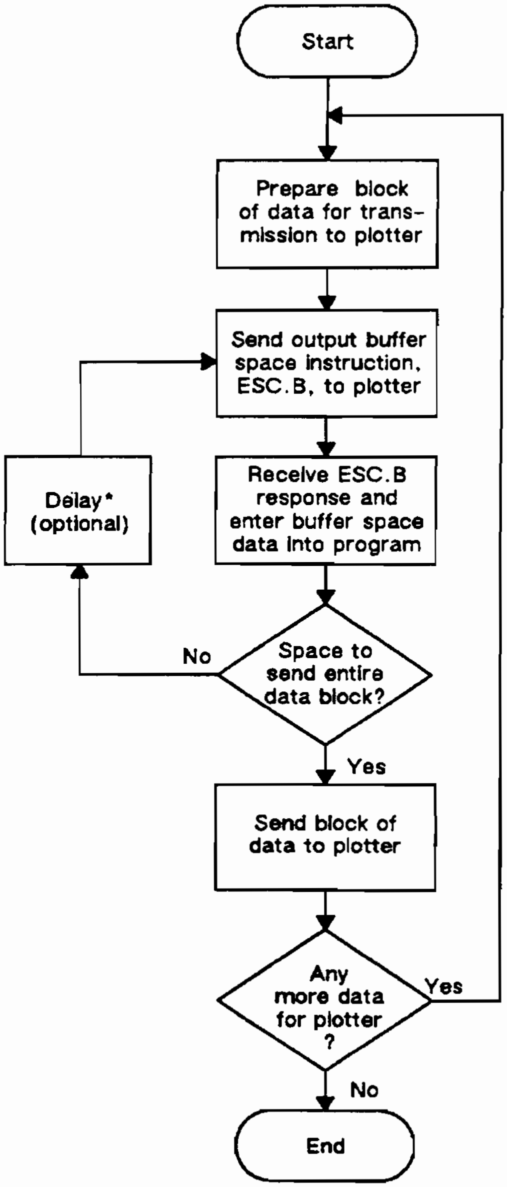
*Enquire/Acknowledge Instructions and Parameters Used in Handshake Response*

| | |
|---|---|
| ESC.H<br>block size<br>enquiry character<br>acknowledgment string | ESC.I<br>block size<br>enquiry character<br>acknowledgment string |
| ESC.M<br>turnaround delay<br>output trigger character<br>echo terminate character<br>output terminator<br>output initiator | ESC.M<br>turnaround delay |
| ESC.N<br>intercharacter delay<br>immediate response string | ESC.N<br>intercharacter delay<br>immediate response string |

## Software Checking Handshake

When your computer system cannot use any of the previously described handshakes, you must incorporate some form of software checking handshake to prevent data from being lost when transferred between the computer and plotter. The ESC.B, output buffer space, instruction is an effective way to monitor the plotter's input buffer. After you send the ESC.B instruction, the plotter outputs the number of empty bytes in the input buffer. Send the block of data when there is sufficient room in the plotter's input buffer. Repeat this process until all data has been sent.

The following chart illustrates how a typical software checking handshake works within a program.


Start
Prepare block of data for transmission to plotter
Send output buffer space instruction, ESC.B, to plotter
Receive ESC.B response and enter buffer space data into program
Delay* (optional)
Space to send entire data block?
No
Yes
Send block of data to plotter
Any more data for plotter ?
Yes
No
End

* Insert computer instructions that will temporarily halt the program, such as a WAIT statement or a FOR...NEXT loop.

*Software Checking Handshake*

This method can use large amounts of computer and plotter input processing time and is inefficient in any environment. It is especially slow in time-shared environments. The following techniques can help reduce the inquiries about the availability of input buffer space.

1. Count the number of bytes to send to the plotter, then fill the input buffer before you send the initial inquiry about available input buffer space.

2. After filling the input buffer, or receiving a negative reply concerning available buffer space, wait a short time before sending another buffer space inquiry.

Use the following instructions to match the requirements of your computer system. Check your computer's documentation to determine which of the following parameters are required.

| Device-Control Instruction | Parameter |
|---|---|
| ESC.M | Turnaround delay<br>Output trigger character<br>Echo terminate character<br>Output terminator<br>Output initiator |
| ESC.N | Intercharacter delay |

### Using ESC.B to Initiate a Software Checking Handshake

You can use the ESC.B, Output Buffer Space, instruction to set up a software checking handshake. (Refer to Chapter 11 for a complete description of this device-control instruction.) In addition to the ESC.B instruction, the following example also uses the ESC.M and ESC.N instructions to establish a turnaround delay and an intercharacter delay. Since all other parameters are omitted, they assume their default values. Use the ESC.B instruction to send a block of data *after* ESC.M and ESC.N. The ASCII decimal code 27 sends the ESC character.

```
CHR$(27)+".M250:"
CHR$(27)+".N50:"
CHR$(27)+".B:"
```

The parameter values for the above example are as follows.

ESC.M
Turnaround delay = 250 milliseconds
Output terminator = ASCII code 13 (CR; default value)

ESC.N
Intercharacter delay = 50 milliseconds

## Data Transmission Modes

When you are using the plotter in an RS-232-C environment, you can use two modes of data transmission: normal mode and block mode. Normal mode is the default mode. Use the ESC.@ instruction to switch from one mode to the other. Refer to Chapter 11 for a complete description of the ESC.@ instruction.

### Normal Mode

The plotter is in normal mode when you turn it on. In normal mode, all HP-GL/2 instructions are stored in an input buffer where they are parsed in the order in which they are received. Device-control instructions do not enter the buffer, but are executed immediately.

### Block Mode

Block mode lets you monitor the transmission of a block of data for any errors that may reach the plotter. This allows you to retransmit the block of data again so that the plotter receives it correctly, thus preventing errors in the plot.

After sending the ESC.@ instruction to turn on block mode, you define a block by sending some data followed by the ESC.E instruction. The ESC.E instruction serves as a block separator by terminating the first block of data and signaling the beginning of the subsequent block (if any). Refer to the ESC.E instruction in Chapter 11 for more information.

In block mode, all HP-GL/2 instructions are stored in the input buffer until you send an ESC.E instruction to determine whether or not errors occurred; device-control and handshake instructions are not stored. You should always send the ESC.E instruction before you turn off block mode (using the ESC.@ instruction). Device-control instructions do not enter the buffer, but are executed immediately.

Block mode has no effect on the type of handshake used or on the handshaking parameters defined. For example, if the number of characters in the block exceeds the logical buffer size, your handshake should prevent a buffer overflow error (error 16) from occurring.

**NOTE:** Do not use the ESC.L instruction in block mode as it may disrupt communication. If you must use the ESC.L instruction in block mode, send it immediately after an ESC.E instruction. Send the ESC.E instruction, read the response, send the ESC.L instruction, read the response, and then send the additional HP-GL instructions. ■

The following shows an example of a block mode transmission process.

*Block Input Error Checking*

| Computer | Plotter | Comments |
|---|---|---|
| ESC.@;16: → | | Enable block input checking. |
| Data block A → | | Send a block of data. [Assume a character gets garbled (e.g., bad parity).] |
| ESC.E → | | Any input errors? |
| | ← 15[TERM] | Parity error. At this point, the plotter discards the block because an error occurred. |
| Data block A → | | Retransit the block. |
| ESC.E → | | Any input errors? |
| | ← 0[TERM] | No errors.<br>Plotter executes block. |
| Data block B → | | Send a block of data. [Assume a handshake character gets lost, and buffer overflows.] |
| ESC.E → | | Any input errors? |
| | ← 16[TERM] | Buffer overflow.<br>Plotter discards block because an error occurred. |
| Data block B → | | Retransmit the block. |
| ESC.E → | | Any input errors? |
| | ← 0[TERM] | No errors.<br>Plotter executes block. |

## ESC.H, Set Handshake Mode 1 (Software Enq/Ack)

**USE:** Configures the plotter for enquire/acknowledge handshake when the computer requires the parameters of ESC.M and ESC.N to be used during the handshaking sequence. Use ESC.H in systems where the output trigger character, echo terminator, and output terminator must be used with the enquire and acknowledge exchange in addition to being used with plotter output responses.

**SYNTAX:** ESC.H (data block size);(enquiry character);(acknowledgment string):
or
ESC.H:

| Parameter | Format | Range | Default |
|---|---|---|---|
| data block size | integer | 0 to 4096 bytes | 80 bytes |
| enquiry character | ASCII value | 0 to 26, 28 to 31* | 0 (no character) |
| acknowledge string | ASCII value | 0 to 126 | 0 (no character) |

*Practical range; printable characters (ASCII codes 32 to 126) should be avoided as they are required to send the HP-GL instructions.

**REMARKS:** When you send an ESC.H instruction without parameters (ESC.H:), the enquire/acknowledge handshake is disabled, data block size is 80 bytes, and there is no enquiry character or acknowledgment string. If however, the computer is configured to send an ENQ character anytime it is ready to send data to the plotter, the plotter automatically responds with ACK when it receives ENQ. This 'dummy' handshake is not dependent on available buffer space and does not protect against buffer overflow.

The plotter interprets the parameters as follows.

- **No Parameters** — Establishes default dummy enquire/acknowledge handshake.

- **Data Block Size** — Specifies the maximum size of each block of data the plotter will receive from the computer. This must be less than the logical input buffer size.

- **Enquiry Character** — Prompts the plotter to acknowledge when there is room in the input buffer for a block of data. Any value other than zero enables the enquire/acknowledge handshake. Decimal code 5 (ENQ) is generally used as the enquiry character.

- **Acknowledgment String** — Signals the computer that the plotter has space available in the input buffer for a block of data. This parameter can be a string of up to 10 ASCII character codes, each subsequent character code separated from the previous one by a semicolon. Zero is not transmitted and terminates the string. Decimal code 6 (ACK) is generally used as the acknowledgment string. Avoid using characters that have a special meaning to the computer.

**RELATED INSTRUCTIONS:**

| | |
|---|---|
| ESC.I, | Set Handshake Mode 2 |
| ESC.M | Set Output Mode |
| ESC.N, | Set Extended Output and Handshake Mode |

## ESC.I, Set Handshake Mode 2 (Operating System)

**USE:** Configures the plotter for either the Xon-Xoff or enquire/acknowledge handshakes when the computer does not expect the parameters of the ESC.M and ESC.N instructions to be used during the handshaking sequence. This is often true when the handshake protocol is part of the computer's operating system.

**SYNTAX for Xon-Xoff:** ESC.I *(Xoff threshold level);(omitted);(Xon trigger character(s))*:

| Parameter | Format | Range | Default |
|---|---|---|---|
| Xoff threshold level | integer | 0 to 4093 | 80 bytes |
| omitted | integer | 0* | — |
| Xon character(s) | ASCII value | 0 to 126, 1 to 10 decimal codes | 0 (no character) |

*You can designate the omitted parameter by entering a 0 or by putting the semicolon without a parameter.

**SYNTAX for Enquire/Acknowledge:** ESC.I*(data block size);(enquiry character);(acknowledgment string)*:

| Parameter | Format | Range | Default |
|---|---|---|---|
| data block size | integer | 0 to 4093 bytes | 80 bytes |
| enquiry character | ASCII value | 0 to 26, 28 to 126 decimal code* | 0 (no character) |
| acknowledgement string | ASCII value | 0 to 126, 1 to 10 decimal codes | 0 (no character) |

*Practical range; printable characters (ASCII codes 32 to 126) should be avoided as they are required to send the HP-GL/2 instructions.

**REMARKS:** When you send an ESC.I instruction without parameters (ESC.I:), neither the Xon-Xoff nor enquire/acknowledge handshake is enabled, the data block size is 80 bytes, and there is no enquiry character or acknowledgment string. However, if the computer is configured to send an ENQ character anytime it is ready to send data to the plotter, the plotter automatically responds with ACK when it receives ENQ. This 'dummy' handshake is not dependent on available buffer space and does not protect against buffer overflow.

For an Xon-Xoff handshake, the plotter interprets the parameters as follows.

- **Xoff Threshold Level** — Specifies the number of available bytes in the input buffer when the Xoff character is to be sent. Xon is sent when the 80 bytes over the threshold are available (or when the buffer is empty if 80 bytes over the threshold is greater than the buffer size).

- **Omitted Parameter** — Sets an Xon-Xoff handshake when you omit this parameter by entering only the semicolon, or the value zero followed by the semicolon. To enable the Xon-Xoff handshake, you must specify the next parameter.

- **Xon Character** — Specifies the character or characters the plotter sends the computer when there is sufficient space available in the input buffer. Although the DC1 character (decimal code 17) is often used as the Xon character, you can specify up to ten character codes separated by semicolons.

For an enquire/acknowledge handshake, interpret the parameters as follows.

- **Data Block Size** — Specifies the maximum size of each block of data the plotter will receive from the computer.

- **Enquiry Character** — Prompts the plotter to acknowledge when there is room in the input buffer for a block of data. Any value other than zero enables the enquire/acknowledge handshake. Decimal code 5 (ENQ) is generally used as the enquiry character.

- **Acknowledgment String** — Signals the computer that the plotter has space available in the input buffer for a block of data. This parameter can be a string of up to ten ASCII character codes, each subsequent character separated from the previous one by a semicolon. Zero is not transmitted and terminates the string. Decimal code 6 (ACK) is generally used as the acknowledgment string. Avoid using characters that have a special meaning to the computer.

**EXAMPLE: For the Xon-Xoff Handshake**

```
CHR$(27)+".I1Ø;;17:"
```

The ASCII decimal code 27 sends the ESC character. The parameters set the Xoff threshold level to 10 (the Xoff character is sent when 10 bytes remain in the plotter's input buffer) and the Xon trigger character to DC1. Note that the second parameter is omitted (as required by this handshake) by entering the semicolon only. Set the Xoff trigger character using the ESC.N instruction.

**For the Enquire/Acknowledge Handshake**

```
CHR$(27)+".I;5;6:"
```

The ASCII decimal code 27 sends the ESC character. The parameters set the block size to its default value of 80 bytes, the ASCII character ENQ as the enquiry character, and the single ASCII character ACK as the acknowledgment string. No output initiator will precede it, even if one is defined, and no output terminator will follow it.

For more information concerning these handshakes, refer to *Xon-Xoff Handshake* and *Enquire/Acknowledge Handshake* earlier in this chapter.

**RELATED INSTRUCTIONS:** ESC.H, Set Handshake Mode 1
ESC.N, Set Extended Output and Handshake Mode

## ESC.M, Set Output Mode

**USE:** Establishes parameters for the plotter's communication format. Use ESC.M to establish a turnaround delay, an output trigger character, an echo terminator, and an output initiator character. Also use it to change the output terminator from its default value, ASCII decimal code 13 (carriage return). *This instruction is valid only when using an RS-232-C interface.*

**SYNTAX:** ESC.M *(turnaround delay);(output trigger);(echo terminator);(output terminator);(output initiator)*:

| Parameter | Format | Range | Default |
|---|---|---|---|
| turnaround delay* | integer | 0 to 32 767 | 0 |
| output trigger | ASCII value | 0 to 4, 6 to 26, 28 to 126 decimal code | 0 (no character) |
| echo terminator | ASCII value | 0 to 4, 6 to 26, 28 to 126 decimal code | 0 (no character) |
| output terminator | ASCII value | 0 to 4, 6 to 26, 28 to 126,<br>1 or 2 decimal codes | 13 (carriage return) |
| output initiator | ASCII value | 0 to 126 decimal code | 0 (no character) |

*If the delay is odd, the plotter adds 1 to make it even.

**REMARKS:** The plotter interprets the parameters as follows.

- **Turnaround Delay** — Prevents the plotter from sending data until the computer is ready to receive and process it. (An intercharacter delay set by ESC.N increments the turnaround delay.)

- **Output Trigger Character** — Is the last character output by the computer when making a request of the plotter. This signals the plotter to respond to the request. Most computers use the LF character (decimal code 10) for output trigger.

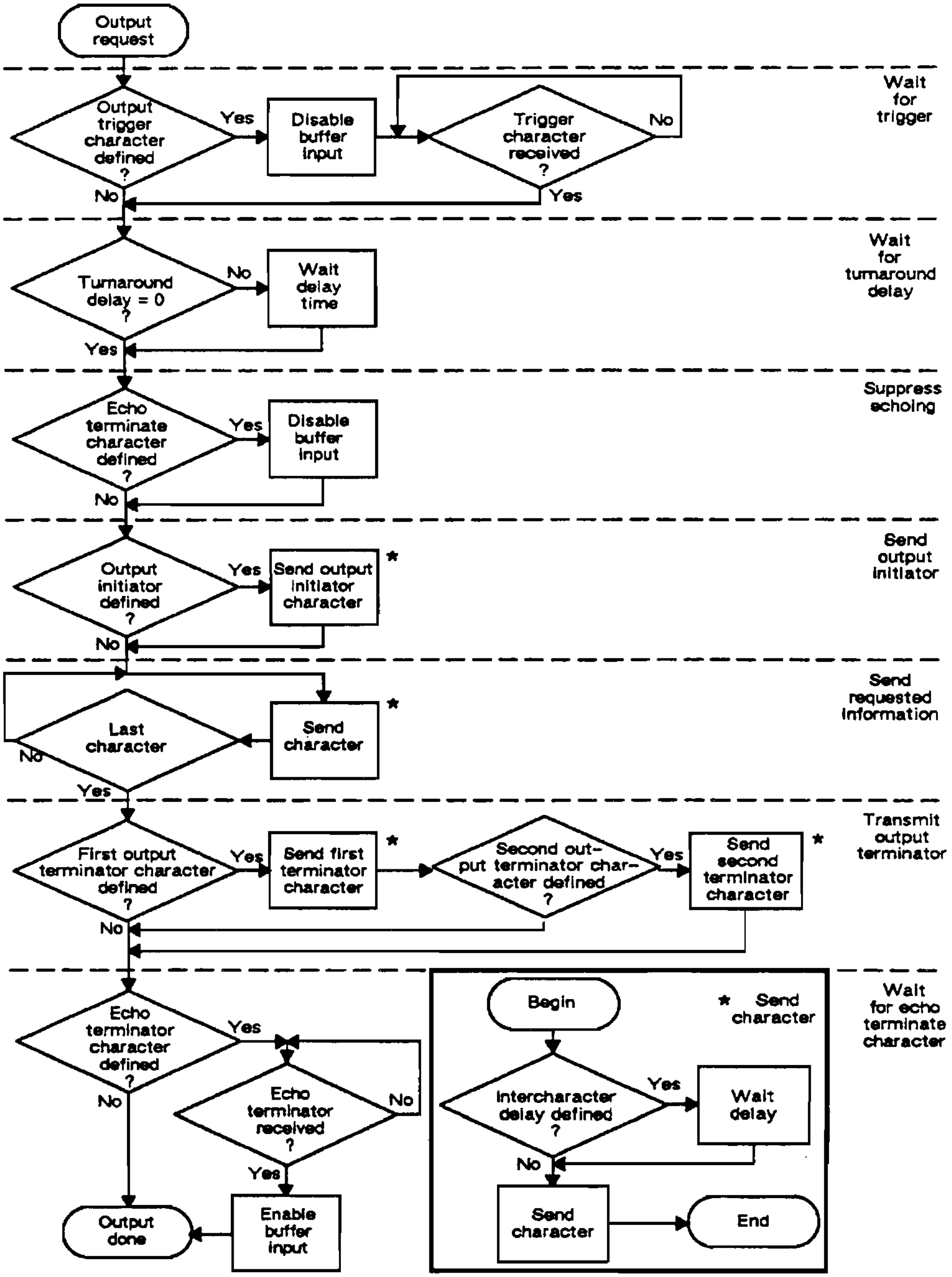
- **Echo Terminator** — Closes the plotter's input buffer while the computer echoes the plotter's responses. The input buffer remains closed until the echo response terminator is received. Refer to your computer documentation to determine if your computer echoes data. If your computer does not echo data, specify this parameter as zero or omit it.

The LF character (decimal code 10) is often used for the echo terminate character. If the computer echoes the plotter's response without a terminating character, then use the plotter's output terminator (next parameter) as the echo terminator.

- **Output Terminator** — Indicates the end of plotter response to the computer. This can be a one- or two-character terminator. If the next parameter (output initiator) is to be specified, the output terminator must consist of two characters, or the second character must be set to zero or omitted (by entering a semicolon). Most computers use the CR character (decimal code 13).

- **Output Initiator** — Indicates the beginning of a plotter response to the computer. To specify an output initiator, the output terminator must consist of two characters (see previous parameter description). Most computers do not use an output initiator. However, many of those that do use **STX** for the output initiator.

The flowchart on the next page illustrates a plotter output request.


Output request
Wait for trigger
Output trigger character defined ?
Yes
No
Disable buffer input
Trigger character received ?
No
Yes
Wait for turnaround delay
Turnaround delay = 0 ?
No
Yes
Wait delay time
Suppress echoing
Echo terminate character defined ?
Yes
No
Disable buffer input
Send output initiator
Output initiator defined ?
Yes
No
Send output initiator character
*
Send requested information
Last character
No
Yes
Send character
*
Transmit output terminator
First output terminator character defined ?
Yes
No
Send first terminator character
*
Second output terminator character defined ?
Yes
Send second terminator character
*
Wait for echo terminate character
Echo terminator character defined ?
Yes
No
Echo terminator received ?
No
Yes
Enable buffer input
Output done
Begin
* Send character
Intercharacter delay defined ?
Yes
No
Wait delay
Send character
End

## ESC.N, Set Extended Output and Handshake Mode

**USE:** Establishes parameters for the plotter's communication format. Use ESC.N to specify an intercharacter delay in all handshake modes and either the immediate response string for the enquire/acknowledge handshake or the Xoff trigger character(s) for the Xon-Xoff handshake.

**SYNTAX:** ESC.N *(intercharacter delay);(handshake dependent parameter)*:

| Parameter | Format | Range | Default |
|---|---|---|---|
| intercharacter delay | integer | 0 to 32 767 | 0 |
| handshake dependent parameter<br>for Xon-Xoff:<br>Xoff trigger character(s) | ASCII value | 0 to 126, up to 10 decimal codes | 0 (no character) |
| for Enquire/Acknowledge:<br>immediate response string | ASCII value | 0 to 126, up to 10 decimal codes | 0 (no character) |

*If the delay is odd, the plotter adds 1 to make it even.

**REMARKS:** The plotter interprets the parameters as follows.

- **Intercharacter Delay** — Specifies the length of transmission delay between each character output by the plotter. This allows extra time for computers with limited input port buffering capability to process data. The intercharacter delay (minus the character transmission time) is added to the turnaround delay (if one has been specified using the ESC.M instruction) before the plotter sends the first character, and is also inserted before each subsequent character in a string being sent to the computer.

- **Handshake Dependent Parameter** — Depends on the method implemented on your plotter. It is valid in either an Xon-Xoff or enquire/acknowledge handshake environment and is interpreted as follows.

  - **Xoff Character** — (Xon-Xoff handshake environment only) signals the computer to temporarily stop sending data while the plotter processes what it has already received. The DC3 character (decimal code 19) is often used for the Xoff trigger character.

- **Immediate Response String** — (Enquire/Acknowledge handshake environment only) indicates to the computer that the plotter is acknowledging receipt of an enquiry character and is checking for available buffer space. The NAK character (decimal code 21) is often used for the immediate response string.

In either handshake environment, the handshake dependent parameter can list up to ten ASCII characters, each separated from another by a semicolon.

**RELATED INSTRUCTIONS:** ESC.I, Set Handshake Mode 2
ESC.M, Set Output Mode

## ESC.P, Set Handshake Mode

**USE:** Sets one of three standard handshakes, or no handshake.

**SYNTAX:** ESC.P (*handshake*):

| Parameter | Format | Range | Default |
|---|---|---|---|
| handshake | integer | 0 (none)<br>1 (Xon-Xoff)<br>2 (ENQ/ACK)<br>3 (hardwire) | 0 |

**REMARKS:** You can use this instruction to select among the three standard handshakes. (Use ESC.@, H, I, M, and N to enhance, or select nonstandard handshakes.) The following table shows the parameters and the handshakes they implement.

| Parameter Value and Handshake Medthod | | | | |
|---|---|---|---|---|
| 0<br>None | 1<br>Xon-<br>Xoff* | 2<br>ENQ/<br>ACK | 3<br>Hardwire | Predefined Handshake Parameters Established |
| 0 | 50 | 0 | 0 | Turnaround delay |
| 0 | 0 | 17 | 0 | Output trigger character |
| 0 | 10 | 10 | 0 | Echo terminate character |
| 13 | 13 | 13 | 13 | Output terminator |
| 80 | N/A | 80 | N/A | Block (record) size |
| 4096 | 4096 | 4096 | 4096 | Logical input buffer size |
| 0 | 10 | 0 | 0 | Intercharacter delay |
| 0 | N/A | 0 | 0 | Immediate response string |
| N/A | 17 | N/A | N/A | Xon character |
| N/A | 19 | N/A | N/A | Xoff character |
| 0 | 0 | 5 | 0 | Equire character |
| 6** | N/A | 6 | 6** | Acknowledgement string |
| N/A | 80 | N/A | 80 | Threshold level |

*This handshake method is established as though you had used the ESC.I instruction.

**This is a 'dummy' acknowledge and does not prevent buffer overflow.

**RELATED INSTRUCTIONS:**

ESC.H, Set Handshake Mode 1
ESC.I, Set Handshake Mode 2
ESC.M, Set Output Mode
ESC.N, Set Extended Output and Handshake Mode
ESC.@, Set Plotter Configuration

A

# Using Emulate Mode

In emulate mode, the plotter can successfully run on software written for an HP 7586B pen plotter. This appendix discusses differences in program instructions between HP-GL/2 and HP 7586B emulate mode in the following areas.

- Coordinate systems, including plot size and P1/P2 locations.
- Pen selection.
- Instructions not available, NOPed, or with functional differences in emulate mode.
- Kanji character set accessability.

The differences between using your plotter in emulate mode and using an HP 7586B pen plotter are discussed in the product note supplied with the plotter, *Using HP 7586B Emulation Mode*. (The information is repeated in Chapter 8 of the *User's Guide.)*

## Coordinate Systems

In HP-GL/2 mode, the origin of the coordinate system is in the lower-left corner of the hard-clip limits. In emulate, the origin of the coordinate system is in the center of the hard-clip limits.

### Plot Size

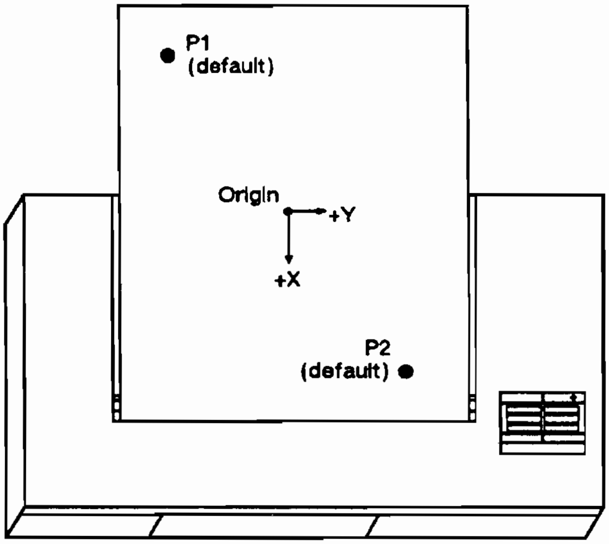
The plot size is the area within the hard-clip limits. The size of the hard-clip limits are the same for both HP-GL/2 and emulate modes (since the electrostatic plotter uses a fixed media width.) Note that the plotting area on your plotter in emulate mode is larger than the plotting area of an HP 7586B pen plotter.

*Hard-Clip Limits in Emulate Mode*

| | X-Axis Range | Y-Axis Range |
|---|---|---|
| Model 240D | ±17 688 | ±12 000 |
| Model 240E | ±23 784 | ±17 920 |

## P1 and P2 Locations

The following illustrations compare the default locations of the scaling points P1 and P2 in emulate mode.



Because the plotting area is larger in emulate mode than on an HP 7586B pen plotter, distance between the scaling points P1 and P2 is larger. (P1 and P2 are located 600 plotter units (15 mm) within the hard-clip limits in emulate mode.) You can expect the output of scaled plots to be larger on your plotter in emulate mode than on an HP 7586B pen plotter.

## Pen Selection

In emulate mode, your plotter operates as an eight-pen plotter by using eight 'logical' pens of different line widths to represent eight physical pens. The following table lists the line widths of the eight logical pens.

| Select Pen Instructions | Line Widths (mm) | Pixel Widths |
|---|---|---|
| SP1; | 0.375 | 6 |
| SP2; | 0.6875 | 11 |
| SP3; | 0.1875 | 3 |
| SP4; | 0.25 | 4 |
| SP5; | 0.3125 | 5 |
| SP6; | 0.50 | 8 |
| SP7; | 0.625 | 10 |
| SP8; | 1.0 | 16 |

## HP-GL Instruction Comparison

In HP-GL/2 mode, your plotter uses HP-GL/2 instructions. In emulate mode, your plotter uses HP-GL instructions. HP-GL instructions do not use clamped integer and clamped real ranges.

The syntax recommended for use with HP-GL instructions differs slightly from HP-GL/2. In emulate mode, you can insert a space or comma between the letters of an instruction mnemonic, and terminate instructions with a line feed when using HP-IB. In HP-GL/2 mode, you can not use either of these syntax variations.

The following sections list the differences between the instruction sets of HP-GL and HP-GL/2. Certain instructions available in one set are not available in the other.

## Instructions Not Available in Emulate Mode

The following instructions are HP-GL/2 instructions and can be used only in HP-GL/2 mode.

| | |
|---|---|
| AC, | Anchor Corner |
| AD, | Alternate Font Definition |
| AT, | Absolute Arc Three Point |
| CF, | Character Fill* |
| IR, | Input P1 and P2 Relative |
| LA, | Line Attributes |
| PE, | Polyline Encoded |
| RF, | Raster Fill Definition |
| RT, | Relative Arc Three Point |
| SD, | Standard Font Definition |
| TD, | Transparent Data |
| UL, | User-Defined Line Type |
| WU, | Pen Width Unit Selection |

## Instructions NOPed in Emulate Mode

Although the following instructions are used with the HP 7586B pen plotter, they are 'no operation' (NOP) instructions in emulate mode. The plotter ignores these instructions and does not generate an error.

| | |
|---|---|
| AP, | Automatic Pen Operations |
| AS, | Acceleration Select |
| DC, | Digitize Clear |
| DP, | Digitize Point |
| EC, | Enable Cut Line |
| FS, | Force Select |
| GP, | Group Pen |
| PT, | Pen Thickness |
| VS, | Velocity Select |

The plotter also ignores these instructions and *does* generate an error (these instructions are used by plotters other than the HP 7586B): BF, CV, GC, IC, KY, OB, OG, OK, VA, VN, and WD.

*The plotter recognizes but ignores CF in HP-GL/2 mode.

## Instructions With Functional Differences in Emulate Mode

The following list briefly describes instructions whose mnemonics are the same that can be used in both modes but whose functionality differs between modes.

**CP, Character Plot** — An automatic pen up is included in HP-GL/2.

**DT, Define Terminator*** — A parameter is added in HP-GL/2 which can disable printing of the label terminator.

**DV, Define Variable Text Path (Direction Vertical, HP-GL)** — Four text directions and two directions of line feeds are added in HP-GL/2.

**EA and ER, Edge Rectangle Absolute and Relative** — The current line type is used in HP-GL/2; a solid line is used in HP-GL.

**FT, Fill Type** — Shading and raster fill are added in HP-GL/2.

**LB, Label** — VT is ignored and HT performs a horizontal tab in HP-GL/2.

**LT, Line Type*** — Up to eight line type patterns are available in HP-GL/2; up to six are available in HP-GL.

**RO, Rotate*** — 180- and 270-degree rotation is added in HP-GL/2.

**SA and SS, Select Alternate and Standard Character Sets** — HP 8-bit mode is the default in HP-GL/2; HP 7-bit compatibility mode is the default in HP-GL.

**SC, Scale*** — Isotropic point-factor scaling is added in HP-GL/2.

**SR, Relative Character Size** — The default sizes are larger in HP-GL/2 (width is 0.75% and height is 1.5%).

*The extended capabilities of the instructins are also available in emulate mode although they are not available on the HP 7586B.

## Instructions Used In Emulate Mode Only

The following instructions are HP-GL instructions and can be used only in emulate mode. The functionality of these instructions is described in detail in the *Hewlett-Packard 7580B, 7585B, and 7586B Drafting Plotters Interfacing and Programming Manual.*

| | |
|---|---|
| AF, | Advance Full Page |
| AH, | Advance Half Page |
| BL, | Buffer Label |
| CA, | Designate Alternate Character Set |
| CC, | Character Chord Angle |
| CM, | Character Selection Mode |
| CS, | Designate Standard Character Set |
| DS, | Designate Character Set into Slot |
| IM, | Input Mask |
| IV, | Invoke Character Slot |
| NR, | Not Ready |
| OA, | Output Actual Pen Status |
| OC, | Output Commanded Pen Status |
| OD, | Output Digitized Point and Pen Status* |
| OF, | Output Factors* |
| OL, | Output Label Length |
| OO, | Output Options |
| OT, | Output Carousel Type* |
| OW, | Output Window |
| PB, | Print Buffered Label |
| SG, | Select Pen Group** |
| TL, | Tick Length |
| UC, | User-Defined Character |
| UF, | User-Defined Fill Type |
| XT, | X-Tick |
| YT, | Y-Tick |

*OD returns a 'dummy' response of 0,0,0; OF returns a 'dummy' response of 40,40; OT returns 'dummy' repsonse of -1,255.

**SG is treated the same as SP.

## Accessing the Kanji Character Set

The Kanji character set can only be accessed in emulate mode. Designate either Kanji set (100 or 101) for use as the standard or alternate character set just as you do other character sets in emulate mode, using the designate standard character set (*CS set;*) or designate alternate character set (*CA set;*) instructions. Of the two sets, only set 101 contains printing characters. For this reason, you will probably only want to use set 101 as either the standard or alternate character set.

Access and use the nonprinting control characters in set 101 just as you do when labeling with other character sets. However, because of the quantity of printing characters in set 101, you must supply two pieces of information (two bytes) to access and draw any printing character. To draw printing characters, both bytes must be within the ASCII decimal value range 33 to 126. The first type accesses the character group; the second byte accesses the printing character within the group. The following table lists the valid first byte values and the character group they access. Since only some of the character groups use all of the available 94 ASCII codes (33–126), the table also lists the valid second-byte ranges for the printing characters in that group.

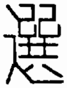
| First Byte Value | Group | Second Byte Value Range |
|---|---|---|
| 33<br>34 | General Graphics | 33–126<br>33–46 |
| 35 | Numbers | 48–57 |
| 35 | Latin alphabet | 65–90, 97–122 |
| 36 | Hiragana | 33–115 |
| 37 | Katakana | 33–118 |
| 38 | Greek alphabet | 33–56, 65–88 |
| 39 | Cyrillic alphabet | 33–65, 81–113 |
| 48–78<br>79 | Kanji | 33–126<br>33–83 |

For example, the first printing character in the general graphics character group would be specified by its group character byte, then by the first byte of the valid printing character range (33, 33). The first printing character of the Cyrillic alphabet is (39, 33). The last printing character of the Kanji group is (79, 83).

If you specify two bytes within the 33-126 range for which there is no printing character, the plotter draws a substitution character. The substitution character is four dots set in a square configuration. All of the printing characters for set 101 are shown at the end of this appendix.

You can specify each byte using the BASIC CHR$ function, or by using the corresponding keyboard character for that byte value. For example, when character set 101 is selected, the following examples access the same Kanji character.

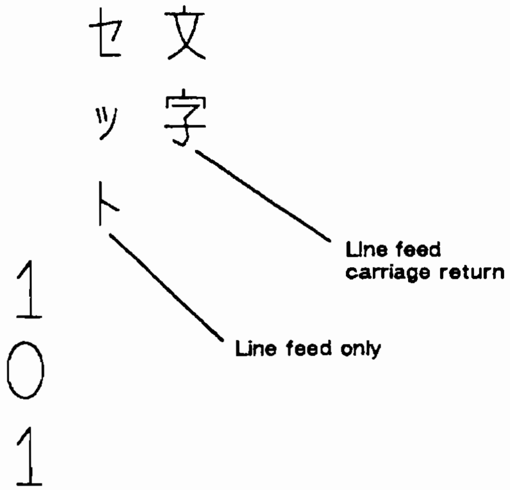
```
PRINT #1, "LB";CHR$(65);CHR$(42);CHR$(3)

PRINT #1, "LBA*";CHR$(3)
```



When labeling with Kanji characters, use the define variable text path (DV) instruction to label the characters vertically. In vertical mode, a carriage return moves the pen up to the carriage return point instead of moving left to the carriage return point. A line feed moves the pen to the left instead of down. Thus labeling in vertical mode is from top to bottom, right to left.

The following illustrates carriage returns and line feeds in vertical mode. Note that the listing uses both methods to access Kanji characters.

```
PRINT #1, "CS0;CA101;SA;DV1;"
PRINT #1, "LBJ8;Z"+CHR$(13)+CHR$(10)+CHR$(3)
PRINT #1, "LB";CHR$(37)+CHR$(59);CHR$(37)+CHR$(67);
               CHR$(37)+CHR$(72);CHR$(10)+CHR$(3)
PRINT #1, "SS;LB101"+CHR$(3)
```



In the first column (at the far right), the first character is accessed using the characters 'J8'; the second character is accessed by the characters ',z'. The characters in the second column are accessed using ASCII decimal values.

## Using Kanji Characters in Symbol Mode

The symbol mode (SM) instruction has been modified for use with Kanji. Usually the SM instruction lets you specify only a single character for use in symbol mode. When you are using character sets 100 or 101, you must specify two characters to properly define your character. Symbol mode allows you to define a single printing character to use as a symbol in geometric drawings or graphs. When you are not using either character set, you can specify only one character byte for the symbol. If you specify more than one byte or character, only the first character byte is used.

In symbol mode, both of the character bytes you specify must be within the printing character range. However, note that printing character (33, 33) is a space, which terminates symbol mode. If either the first or second character is a printing character but the other character is a nonprinting control character (has a value of 32 or less), then symbol mode is terminated. If the second character is the control character, the plotter also generates an "out of range" error.

Note that you can not specify one character and then use a semicolon to terminate the SM instruction. Symbol mode uses the semicolon as the second byte that defines your symbol character. However, (*SM;;;*) is valid for Kanji.

## Terminating Kanji Labels

The define label terminator (DT) instruction has not been modified for character set 101. There is no two-byte form of the DT instruction. The default label terminator is the ETX (end of text) character (ASCII decimal code 3). Only ASCII codes 1–9, 11–26, 28–31 or the space character (ASCII decimal code 32) can be used as label terminators. You can not use the LF character (ASCII decimal code 10) or the ESC character (ASCII character code 27). Nor can you use the semicolon (ASCII decimal code 59). If the DT instruction defines a printing character as a label terminator, that character terminates labels for all character sets but sets 100 and 101; the designated terminator for sets 100 and 101 is not altered.

## The Kanji Character Set

The following pages show the characters in set 101. Note that the value of the first byte (which specifies a particular group of characters) is listed vertically along the side of the page. The second byte (which specifies a particular character in the group) is listed horizontally across the top of the page. The illustration below shows how the characters are arranged on the following pages. The number in the box is the page number; the numbers along the top and the left side show the values listed on that page.

| | 33–44 | 45–56 | 57–68 | 69–80 | 81–92 | 93–104 | 105–116 | 117–126 |
|---|---|---|---|---|---|---|---|---|
| 33<br>39<br>48<br>53 | D-7 | D-8 | D-9 | D-10 | D-11 | D-12 | D-13 | D-14 |
| 54<br>66 | D-15 | D-16 | D-17 | D-18 | D-19 | D-20 | D-21 | D-22 |
| 67<br>79 | D-23 | D-24 | D-25 | D-26 | D-27 | D-28 | D-29 | D-30 |

*Kanji Character Set*

| Decimal Value | | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 |
| FIRST BYTE | 33 | | 、 | 。 | ， | ． | ・ | ： | ； | ？ | ！ | ゛ | ゜ |
| | 34 | ◆ | □ | ■ | △ | ▲ | ▽ | ▼ | ※ | 〒 | → | ← | ↑ |
| | 35 | . .<br>. . | . .<br>. . | . .<br>. . | . .<br>. . | . .<br>. . | . .<br>. . | . .<br>. . | . .<br>. . | . .<br>. . | . .<br>. . | . .<br>. . | . .<br>. . |
| | 36 | ぁ | あ | ぃ | い | ぅ | う | ぇ | え | ぉ | お | か | が |
| | 37 | ァ | ア | ィ | イ | ゥ | ウ | ェ | エ | ォ | オ | カ | ガ |
| | 38 | Α | Β | Γ | Δ | Ε | Ζ | Η | Θ | Ι | Κ | Λ | Μ |
| | 39 | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К |
| | 48 | 亜 | 唖 | 娃 | 阿 | 哀 | 愛 | 挨 | 姶 | 逢 | 葵 | 茜 | 穐 |
| | 49 | 院 | 陰 | 隠 | 韻 | 吋 | 右 | 宇 | 烏 | 羽 | 迂 | 雨 | 卯 |
| | 50 | 押 | 旺 | 横 | 欧 | 殴 | 王 | 翁 | 襖 | 鴬 | 鴎 | 黄 | 岡 |
| | 51 | 魁 | 晦 | 械 | 海 | 灰 | 界 | 皆 | 絵 | 芥 | 蟹 | 開 | 階 |
| | 52 | 粥 | 刈 | 苅 | 瓦 | 乾 | 侃 | 冠 | 寒 | 刊 | 勘 | 勧 | 巻 |
| | 53 | 機 | 帰 | 毅 | 気 | 汽 | 畿 | 祈 | 季 | 稀 | 紀 | 徽 | 規 |

*(Table continues)*

*Kanji Character Set*

| Decimal Value | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIRST BYTE | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
| 33 | ´ | ｀ | ¨ | ＾ | ￣ | ＿ | ヽ | ヾ | ゝ | ゞ | 〃 | 仝 |
| 34 | ↓ | 〓 | | | | | | | | | | |
| 35 | | | | ０ | １ | ２ | ３ | ４ | ５ | ６ | ７ | ８ |
| 36 | き | ぎ | く | ぐ | け | げ | こ | ご | さ | ざ | し | じ |
| 37 | | ギ | ク | グ | ケ | ゲ | コ | ゴ | サ | ザ | シ | ジ |
| 38 | Ν | Ξ | Ο | Π | Ρ | Σ | Τ | Υ | Φ | Χ | Ψ | Ω |
| 39 | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц |
| 48 | 悪 | 握 | 渥 | 旭 | 葦 | 芦 | 鯵 | 梓 | 圧 | 斡 | 扱 | 宛 |
| 49 | 鵜 | 窺 | 丑 | 碓 | 臼 | 渦 | 嘘 | 唄 | 欝 | 蔚 | 鰻 | 姥 |
| 50 | 沖 | 荻 | 億 | 屋 | 憶 | 臆 | 桶 | 牡 | 乙 | 俺 | 卸 | 恩 |
| 51 | 貝 | 凱 | 劾 | 外 | 咳 | 害 | 崖 | 慨 | 概 | 涯 | 碍 | 蓋 |
| 52 | 喚 | 堪 | 姦 | 完 | 官 | 寛 | 干 | 幹 | 患 | 感 | 慣 | 憾 |
| 53 | 記 | 貴 | 起 | 軌 | 輝 | 飢 | 騎 | 鬼 | 亀 | 偽 | 儀 | 妓 |

*(Table continues)*

*Kanji Character Set*

| Decimal Value | | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 |
| FIRST BYTE | 33 | 々 | 〆 | 〇 | ー | ― | ‐ | ／ | ＼ | 〜 | ‖ | ｜ | … |
| | 34 | | ∈ | ∋ | ⊆ | ⊇ | ⊂ | ⊃ | ∪ | ∩ | | | |
| | 35 | ９ | | | | | | | | Ａ | Ｂ | Ｃ | Ｄ |
| | 36 | す | ず | せ | ぜ | そ | ぞ | た | だ | ち | ぢ | っ | つ |
| | 37 | ス | ズ | セ | ゼ | ソ | ゾ | タ | ダ | チ | ヂ | ッ | ツ |
| | 38 | | | | | | | | | α | β | γ | δ |
| | 39 | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | | | |
| | 48 | 姐 | 虻 | 飴 | 絢 | 綾 | 鮎 | 或 | 粟 | 袷 | 安 | 庵 | 按 |
| | 49 | 厩 | 浦 | 瓜 | 閏 | 噂 | 云 | 運 | 雲 | 荏 | 餌 | 叡 | 営 |
| | 50 | 温 | 穏 | 音 | 下 | 化 | 仮 | 何 | 伽 | 価 | 佳 | 加 | 可 |
| | 51 | 街 | 該 | 鎧 | 骸 | 浬 | 馨 | 蛙 | 垣 | 柿 | 蛎 | 鈎 | 劃 |
| | 52 | 換 | 敢 | 柑 | 桓 | 棺 | 款 | 歓 | 汗 | 漢 | 澗 | 潅 | 環 |
| | 53 | 宜 | 戯 | 技 | 擬 | 欺 | 犠 | 疑 | 祇 | 義 | 蟻 | 誼 | 議 |

*(Table continues)*

*Kanji Character Set*

| | Decimal Value | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| FIRST BYTE | 33 | ‥ | ‘ | ’ | “ | ” | （ | ） | 〔 | 〕 | ［ | ］ | ｛ |
| | 34 | | | | | | ∧ | ∨ | ￢ | ⇒ | ⇔ | ∀ | ∃ |
| | 35 | Ｅ | Ｆ | Ｇ | Ｈ | Ｉ | Ｊ | Ｋ | Ｌ | Ｍ | Ｎ | Ｏ | Ｐ |
| | 36 | づ | て | で | と | ど | な | に | ぬ | ね | の | は | ば |
| | 37 | ヅ | テ | デ | ト | ド | ナ | ニ | ヌ | ネ | ノ | ハ | バ |
| | 38 | ε | ζ | η | θ | ι | κ | λ | μ | ν | ξ | ο | π |
| | 39 | | | | | | | | | | | | |
| | 48 | 暗 | 案 | 闇 | 鞍 | 杏 | 以 | 伊 | 位 | 依 | 偉 | 囲 | 夷 |
| | 49 | 嬰 | 影 | 映 | 曳 | 栄 | 永 | 泳 | 洩 | 瑛 | 盈 | 穎 | 頴 |
| | 50 | 嘉 | 夏 | 嫁 | 家 | 寡 | 科 | 暇 | 果 | 架 | 歌 | 河 | 火 |
| | 51 | 嚇 | 各 | 廓 | 拡 | 撹 | 格 | 核 | 殻 | 獲 | 確 | 穫 | 覚 |
| | 52 | 甘 | 監 | 看 | 竿 | 管 | 簡 | 緩 | 缶 | 翰 | 肝 | 艦 | 莞 |
| | 53 | 掬 | 菊 | 鞠 | 吉 | 吃 | 喫 | 桔 | 橘 | 詰 | 砧 | 杵 | 黍 |

*(Table continues)*

*Kanji Character Set*

| Decimal Value | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIRST BYTE | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 |
| 33 | ｝ | 〈 | 〉 | 《 | 》 | 「 | 」 | 『 | 』 | 【 | 】 | ＋ |
| 34 | . . | . . | . . | . . | . . | . . | . . | . . | . . | . . | . . | ∠ |
| 35 | Ｑ | Ｒ | Ｓ | Ｔ | Ｕ | Ｖ | Ｗ | Ｘ | Ｙ | Ｚ | . . | . . |
| 36 | ぱ | ひ | び | ぴ | ふ | ぶ | ぷ | へ | べ | ぺ | ほ | ぼ |
| 37 | パ | ヒ | ビ | ピ | フ | ブ | プ | ヘ | ベ | ペ | ホ | ボ |
| 38 | ρ | σ | τ | υ | φ | χ | ψ | ω | . . | . . | . . | . . |
| 39 | а | б | в | г | д | е | ё | ж | з | и | й | к |
| 48 | 委 | 威 | 尉 | 惟 | 意 | 慰 | 易 | 椅 | 為 | 畏 | 異 | 移 |
| 49 | 英 | 衛 | 詠 | 鋭 | 液 | 疫 | 益 | 駅 | 悦 | 謁 | 越 | 閲 |
| 50 | 珂 | 禍 | 禾 | 稼 | 箇 | 花 | 苛 | 茄 | 荷 | 華 | 菓 | 蝦 |
| 51 | 角 | 赫 | 較 | 郭 | 閣 | 隔 | 革 | 学 | 岳 | 楽 | 額 | 顎 |
| 52 | 観 | 諌 | 貫 | 還 | 鑑 | 間 | 閑 | 関 | 陥 | 韓 | 館 | 舘 |
| 53 | 却 | 客 | 脚 | 虐 | 逆 | 丘 | 久 | 仇 | 休 | 及 | 吸 | 宮 |

*(Table continues)*

*Kanji Character Set*

| | Decimal Value | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 |
| FIRST BYTE | 33 | − | ± | × | ÷ | ＝ | ≠ | ＜ | ＞ | ≦ | ≧ | ∞ | ∴ |
| | 34 | ⊥ | ⌒ | ∂ | ∇ | ≡ | ≒ | ≪ | ≫ | √ | ∽ | ∝ | ∵ |
| | 35 | · · | · · | · · | · · | ａ | ｂ | ｃ | ｄ | ｅ | ｆ | ｇ | ｈ |
| | 36 | ぽ | ま | み | む | め | も | ゃ | や | ゅ | ゆ | ょ | よ |
| | 37 | ポ | マ | ミ | ム | メ | モ | ャ | ヤ | ュ | ユ | ョ | ヨ |
| | 38 | · · | · · | · · | · · | · · | · · | · · | · · | · · | · · | · · | · · |
| | 39 | л | м | н | о | п | р | с | т | у | ф | х | ц |
| | 48 | 維 | 緯 | 胃 | 萎 | 衣 | 謂 | 違 | 遺 | 医 | 井 | 亥 | 域 |
| | 49 | 榎 | 厭 | 円 | 園 | 堰 | 奄 | 宴 | 延 | 怨 | 掩 | 援 | 沿 |
| | 50 | 課 | 嘩 | 貨 | 迦 | 過 | 霞 | 蚊 | 俄 | 峨 | 我 | 牙 | 画 |
| | 51 | 掛 | 笠 | 樫 | 橿 | 梶 | 鰍 | 潟 | 割 | 喝 | 恰 | 括 | 活 |
| | 52 | 丸 | 含 | 岸 | 巌 | 玩 | 癌 | 眼 | 岩 | 翫 | 贋 | 雁 | 頑 |
| | 53 | 弓 | 急 | 救 | 朽 | 求 | 汲 | 泣 | 灸 | 球 | 究 | 窮 | 笈 |

*(Table continues)*

*Kanji Character Set*

| | Decimal Value | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 |
| FIRST BYTE | 33 | ♂ | ♀ | ° | ′ | ″ | ℃ | ￥ | ＄ | ￠ | ￡ | ％ | ＃ |
| | 34 | ∫ | ∬ | | | | | | | | Å | ‰ | ♯ |
| | 35 | ｉ | ｊ | ｋ | ｌ | ｍ | ｎ | ｏ | ｐ | ｑ | ｒ | ｓ | ｔ |
| | 36 | ら | り | る | れ | ろ | ゎ | わ | ゐ | ゑ | を | ん | |
| | 37 | ラ | リ | ル | レ | ロ | ヮ | ワ | ヰ | ヱ | ヲ | ン | ヴ |
| | 38 | | | | | | | | | | | | |
| | 39 | ч | ш | щ | ъ | ы | ь | э | ю | я | | | |
| | 48 | 育 | 郁 | 磯 | 一 | 壱 | 溢 | 逸 | 稲 | 茨 | 芋 | 鰯 | 允 |
| | 49 | 演 | 炎 | 焔 | 煙 | 燕 | 猿 | 縁 | 艶 | 苑 | 薗 | 遠 | 鉛 |
| | 50 | 臥 | 芽 | 蛾 | 賀 | 雅 | 餓 | 駕 | 介 | 会 | 解 | 回 | 塊 |
| | 51 | 渇 | 滑 | 葛 | 褐 | 轄 | 且 | 鰹 | 叶 | 椛 | 樺 | 鞄 | 株 |
| | 52 | 顔 | 願 | 企 | 伎 | 危 | 喜 | 器 | 基 | 奇 | 嬉 | 寄 | 岐 |
| | 53 | 級 | 糾 | 給 | 旧 | 牛 | 去 | 居 | 巨 | 拒 | 拠 | 挙 | 渠 |

*(Table continues)*

*Kanji Character Set*

| | Decimal Value | SECOND BYTE | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 |
| FIRST BYTE | 33 | ＆ | ＊ | ＠ | § | ☆ | ★ | ○ | ● | ◎ | ◇ |
| | 34 | ♭ | ♪ | † | ‡ | ¶ | | | | | ◯ |
| | 35 | ｕ | ｖ | ｗ | ｘ | ｙ | ｚ | | | | |
| | 36 | | | | | | | | | | |
| | 37 | ヵ | ヶ | | | | | | | | |
| | 38 | | | | | | | | | | |
| | 39 | | | | | | | | | | |
| | 48 | 印 | 咽 | 員 | 因 | 姻 | 引 | 飲 | 淫 | 胤 | 蔭 |
| | 49 | 鴛 | 塩 | 於 | 汚 | 甥 | 凹 | 央 | 奥 | 往 | 応 |
| | 50 | 壊 | 廻 | 快 | 怪 | 悔 | 恢 | 懐 | 戒 | 拐 | 改 |
| | 51 | 兜 | 竃 | 蒲 | 釜 | 鎌 | 噛 | 鴨 | 栢 | 茅 | 萱 |
| | 52 | 希 | 幾 | 忌 | 揮 | 机 | 旗 | 既 | 期 | 棋 | 棄 |
| | 53 | 虚 | 許 | 距 | 鋸 | 漁 | 禦 | 魚 | 亨 | 享 | 京 |

*(Table continues)*

*Kanji Character Set*

| Decimal Value | SECOND BYTE 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 54 | 供 | 侠 | 僑 | 兇 | 競 | 共 | 凶 | 協 | 匡 | 卿 | 叫 | 喬 |
| 55 | 掘 | 窟 | 沓 | 靴 | 轡 | 窪 | 熊 | 隈 | 粂 | 栗 | 繰 | 桑 |
| 56 | 検 | 権 | 牽 | 犬 | 献 | 研 | 硯 | 絹 | 県 | 肩 | 見 | 謙 |
| 57 | 后 | 喉 | 坑 | 垢 | 好 | 孔 | 孝 | 宏 | 工 | 巧 | 巷 | 幸 |
| 58 | 此 | 頃 | 今 | 困 | 坤 | 墾 | 婚 | 恨 | 懇 | 昏 | 昆 | 根 |
| F I R S T 59 | 察 | 拶 | 撮 | 擦 | 札 | 殺 | 薩 | 雑 | 皐 | 鯖 | 捌 | 錆 |
| 60 | 次 | 滋 | 治 | 爾 | 璽 | 痔 | 磁 | 示 | 而 | 耳 | 自 | 蒔 |
| B Y T E 61 | 宗 | 就 | 州 | 修 | 愁 | 拾 | 洲 | 秀 | 秋 | 終 | 繍 | 習 |
| 62 | 勝 | 匠 | 升 | 召 | 哨 | 商 | 唱 | 嘗 | 奨 | 妾 | 娼 | 宵 |
| 63 | 拭 | 植 | 殖 | 燭 | 織 | 職 | 色 | 触 | 食 | 蝕 | 辱 | 尻 |
| 64 | 登 | 摺 | 寸 | 世 | 瀬 | 畝 | 是 | 凄 | 制 | 勢 | 姓 | 征 |
| 65 | 繊 | 羨 | 腺 | 舛 | 船 | 薦 | 詮 | 賎 | 践 | 選 | 遷 | 銭 |
| 66 | 臓 | 蔵 | 贈 | 造 | 促 | 側 | 則 | 即 | 息 | 捉 | 束 | 測 |

*(Table continues)*

*Kanji Character Set*

| Decimal Value | | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
| FIRST BYTE | 54 | 境 | 峡 | 強 | 彊 | 怯 | 恐 | 恭 | 挟 | 教 | 橋 | 況 | 狂 |
| | 55 | 鍬 | 勲 | 君 | 薫 | 訓 | 群 | 軍 | 郡 | 卦 | 袈 | 祁 | 係 |
| | 56 | 賢 | 軒 | 遣 | 鍵 | 険 | 顕 | 験 | 鹸 | 元 | 原 | 厳 | 幻 |
| | 57 | 広 | 庚 | 康 | 弘 | 恒 | 慌 | 抗 | 拘 | 控 | 攻 | 昂 | 晃 |
| | 58 | 梱 | 混 | 痕 | 紺 | 艮 | 魂 | 些 | 佐 | 叉 | 唆 | 嵯 | 左 |
| | 59 | 鮫 | 皿 | 晒 | 三 | 傘 | 参 | 山 | 惨 | 撒 | 散 | 桟 | 燦 |
| | 60 | 辞 | 汐 | 鹿 | 式 | 識 | 鴫 | 竺 | 軸 | 宍 | 雫 | 七 | 叱 |
| | 61 | 臭 | 舟 | 蒐 | 衆 | 襲 | 讐 | 蹴 | 輯 | 週 | 酋 | 酬 | 集 |
| | 62 | 将 | 小 | 少 | 尚 | 庄 | 床 | 廠 | 彰 | 承 | 抄 | 招 | 掌 |
| | 63 | 伸 | 信 | 侵 | 唇 | 娠 | 寝 | 審 | 心 | 慎 | 振 | 新 | 晋 |
| | 64 | 性 | 成 | 政 | 整 | 星 | 晴 | 棲 | 栖 | 正 | 清 | 牲 | 生 |
| | 65 | 銑 | 閃 | 鮮 | 前 | 善 | 漸 | 然 | 全 | 禅 | 繕 | 膳 | 糎 |
| | 66 | 足 | 速 | 俗 | 属 | 賊 | 族 | 続 | 卒 | 袖 | 其 | 揃 | 存 |

*(Table continues)*

### *Kanji Character Set*

| Decimal Value | | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 |
| FIRST BYTE | 54 | 狭 | 矯 | 胸 | 脅 | 興 | 蕎 | 郷 | 鏡 | 響 | 饗 | 驚 | 仰 |
| | 55 | 傾 | 刑 | 兄 | 啓 | 圭 | 珪 | 型 | 契 | 形 | 径 | 恵 | 慶 |
| | 56 | 弦 | 減 | 源 | 玄 | 現 | 絃 | 舷 | 言 | 諺 | 限 | 乎 | 個 |
| | 57 | 更 | 杭 | 校 | 梗 | 構 | 江 | 洪 | 浩 | 港 | 溝 | 甲 | 皇 |
| | 58 | 差 | 査 | 沙 | 瑳 | 砂 | 詐 | 鎖 | 裟 | 坐 | 座 | 挫 | 債 |
| | 59 | 珊 | 産 | 算 | 纂 | 蚕 | 讃 | 賛 | 酸 | 餐 | 斬 | 暫 | 残 |
| | 60 | 執 | 失 | 嫉 | 室 | 悉 | 湿 | 漆 | 疾 | 質 | 実 | 蔀 | 篠 |
| | 61 | 醜 | 什 | 住 | 充 | 十 | 従 | 戎 | 柔 | 汁 | 渋 | 獣 | 縦 |
| | 62 | 捷 | 昇 | 昌 | 昭 | 晶 | 松 | 梢 | 樟 | 樵 | 沼 | 消 | 渉 |
| | 63 | 森 | 榛 | 浸 | 深 | 申 | 疹 | 真 | 神 | 秦 | 紳 | 臣 | 芯 |
| | 64 | 盛 | 精 | 聖 | 声 | 製 | 西 | 誠 | 誓 | 請 | 逝 | 醒 | 青 |
| | 65 | 噌 | 塑 | 岨 | 措 | 曾 | 曽 | 楚 | 狙 | 疏 | 疎 | 礎 | 祖 |
| | 66 | 孫 | 尊 | 損 | 村 | 遜 | 他 | 多 | 太 | 汰 | 詑 | 唾 | 堕 |

*(Table continues)*

## *Kanji Character Set*

| Decimal Value | | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| FIRST BYTE | 54 | 凝 | 尭 | 暁 | 業 | 局 | 曲 | 極 | 玉 | 桐 | 粁 | 僅 | 勤 |
| | 55 | 慧 | 憩 | 掲 | 携 | 敬 | 景 | 桂 | 渓 | 畦 | 稽 | 系 | 経 |
| | 56 | 古 | 呼 | 固 | 姑 | 孤 | 己 | 庫 | 弧 | 戸 | 故 | 枯 | 湖 |
| | 57 | 硬 | 稿 | 糠 | 紅 | 紘 | 絞 | 綱 | 耕 | 考 | 肯 | 肱 | 腔 |
| | 58 | 催 | 再 | 最 | 哉 | 塞 | 妻 | 宰 | 彩 | 才 | 採 | 栽 | 歳 |
| | 59 | 仕 | 仔 | 伺 | 使 | 刺 | 司 | 史 | 嗣 | 四 | 士 | 始 | 姉 |
| | 60 | 偲 | 柴 | 芝 | 屡 | 蕊 | 縞 | 舎 | 写 | 射 | 捨 | 赦 | 斜 |
| | 61 | 重 | 銃 | 叔 | 夙 | 宿 | 淑 | 祝 | 縮 | 粛 | 塾 | 熟 | 出 |
| | 62 | 湘 | 焼 | 焦 | 照 | 症 | 省 | 硝 | 礁 | 祥 | 称 | 章 | 笑 |
| | 63 | 薪 | 親 | 診 | 身 | 辛 | 進 | 針 | 震 | 人 | 仁 | 刃 | 塵 |
| | 64 | 静 | 斉 | 税 | 脆 | 隻 | 席 | 惜 | 戚 | 斥 | 昔 | 析 | 石 |
| | 65 | 租 | 粗 | 素 | 組 | 蘇 | 訴 | 阻 | 遡 | 鼠 | 僧 | 創 | 双 |
| | 66 | 妥 | 惰 | 打 | 柁 | 舵 | 楕 | 陀 | 駄 | 騨 | 体 | 堆 | 対 |

*(Table continues)*

*Kanji Character Set*

| Decimal Value | | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 |
| FIRST BYTE | 54 | 均 | 巾 | 錦 | 斤 | 欣 | 欽 | 琴 | 禁 | 禽 | 筋 | 緊 | 芹 |
| | 55 | 継 | 繋 | 罫 | 茎 | 荊 | 蛍 | 計 | 詣 | 警 | 軽 | 頚 | 鶏 |
| | 56 | 狐 | 糊 | 袴 | 股 | 胡 | 菰 | 虎 | 誇 | 跨 | 鈷 | 雇 | 顧 |
| | 57 | 膏 | 航 | 荒 | 行 | 衡 | 講 | 貢 | 購 | 郊 | 酵 | 鉱 | 砿 |
| | 58 | 済 | 災 | 采 | 犀 | 砕 | 砦 | 祭 | 斎 | 細 | 菜 | 裁 | 載 |
| | 59 | 姿 | 子 | 屍 | 市 | 師 | 志 | 思 | 指 | 支 | 孜 | 斯 | 施 |
| | 60 | 煮 | 社 | 紗 | 者 | 謝 | 車 | 遮 | 蛇 | 邪 | 借 | 勺 | 尺 |
| | 61 | 術 | 述 | 俊 | 峻 | 春 | 瞬 | 竣 | 舜 | 駿 | 准 | 循 | 旬 |
| | 62 | 粧 | 紹 | 肖 | 菖 | 蒋 | 蕉 | 衝 | 裳 | 訟 | 証 | 詔 | 詳 |
| | 63 | 壬 | 尋 | 甚 | 尽 | 腎 | 訊 | 迅 | 陣 | 靭 | 笥 | 諏 | 須 |
| | 64 | 積 | 籍 | 績 | 脊 | 責 | 赤 | 跡 | 蹟 | 碩 | 切 | 拙 | 接 |
| | 65 | 叢 | 倉 | 喪 | 壮 | 奏 | 爽 | 宋 | 層 | 匝 | 惣 | 想 | 捜 |
| | 66 | 耐 | 岱 | 帯 | 待 | 怠 | 態 | 戴 | 替 | 泰 | 滞 | 胎 | 腿 |

*(Table continues)*

## Character Sets

The plotter has 22 character sets. The following table lists these character sets and their corresponding AD or SD value and ISO number.

*Plotter Character Sets*

| Description | Value | ISO No. |
|---|---|---|
| Roman8 (default) | 277 (0) | — |
| ANSI USASCII | 21 | 006 |
| ECMA 94 Latin 1 | 14 | — |
| French v1 | 6 | 025 |
| French v2 | 38 | 069 |
| German | 39 | 021 |
| HP Drafting | 563 | — |
| HP-GL/2 Download | 531 | — |
| HP Kana8 | 267 | — |
| HP Katakana | 43 | — |
| HP Roman Extensions | 5 | — |
| HP Special Symbols | 595 | — |
| Intnl. Ref. Version | 85 | 002 |
| Italian | 9 | 015 |
| JIS ASCII | 11 | 014 |
| Norwegian v1 | 4 | 060 |
| Norwegian v2 | 36 | 061 |
| Portuguese | 147 | 016 |
| Swedish | 115 | 010 |
| Swedish Names | 19 | 011 |
| Spanish | 83 | 017 |
| United Kingdom | 37 | 004 |

The following pages show the plotter's character sets in fixed vector font. (The Kanji character set is shown in Appendix A.) All printing ASCII characters and their decimal codes (33–126) are listed for each character set.

| Decimal Value | | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 |
| FIRST BYTE | 54 | 狗 | 玖 | 矩 | 苦 | 躯 | 駆 | 駈 | 駒 | 具 | 愚 | 虞 | 喰 |
| | 55 | 潔 | 穴 | 結 | 血 | 訣 | 月 | 件 | 倹 | 倦 | 健 | 兼 | 券 |
| | 56 | 檎 | 瑚 | 碁 | 語 | 誤 | 護 | 醐 | 乞 | 鯉 | 交 | 佼 | 侯 |
| | 57 | 拷 | 濠 | 豪 | 轟 | 麹 | 克 | 刻 | 告 | 国 | 穀 | 酷 | 鵠 |
| | 58 | 咲 | 崎 | 埼 | 碕 | 鷺 | 作 | 削 | 咋 | 搾 | 昨 | 朔 | 柵 |
| | 59 | 脂 | 至 | 視 | 詞 | 詩 | 試 | 誌 | 諮 | 資 | 賜 | 雌 | 飼 |
| | 60 | 守 | 手 | 朱 | 殊 | 狩 | 珠 | 種 | 腫 | 趣 | 酒 | 首 | 儒 |
| | 61 | 初 | 所 | 暑 | 曙 | 渚 | 庶 | 緒 | 署 | 書 | 薯 | 藷 | 諸 |
| | 62 | 冗 | 剰 | 城 | 場 | 壌 | 嬢 | 常 | 情 | 擾 | 条 | 杖 | 浄 |
| | 63 | 翠 | 衰 | 遂 | 酔 | 錐 | 錘 | 随 | 瑞 | 髄 | 崇 | 嵩 | 数 |
| | 64 | 千 | 占 | 宣 | 専 | 尖 | 川 | 戦 | 扇 | 撰 | 栓 | 栴 | 泉 |
| | 65 | 痩 | 相 | 窓 | 糟 | 総 | 綜 | 聡 | 草 | 荘 | 葬 | 蒼 | 藻 |
| | 66 | 醍 | 題 | 鷹 | 滝 | 瀧 | 卓 | 啄 | 宅 | 托 | 択 | 拓 | 沢 |

*(Table continues)*

## ASCII Control Codes

The plotter's reactions in label mode to nonprinting ASCII control codes (decimal codes 0–32 and +128 codes) are shown in the following table. These reactions are true regardless of the character set currently used for labeling.

*Reaction to Nonprinting Control Characters*

| Decimal Code | ASCII Character | All Sets |
|---|---|---|
| 0 | NULL | No Operation (NOP) |
| 1 | SOH | NOP |
| 2 | STX | NOP |
| 3 | ETX | Terminate Label Instruction |
| 4 | ETO | NOP |
| 5* | ENQ | NOP |
| 6 | ACK | NOP |
| 7 | BEL | NOP |
| 8 | BS | Backspace |
| 9** | HT | Horizontal Tab |
| 10 | LF | Line Feed |
| 11 | VT | NOP |
| 12 | FF | NOP |
| 13 | CR | Carriage Return |
| 14 | SO | Shift-Out<br>(Select Alternate Character Set) |

*With an RS-232-C interface, unless 5 has been established as an enquiry character, the plotter will respond to an ENQ with an ACK.

**Using control character horizontal tab (decimal code 9) inside a label string moves the pen one-half character space back (equivalent to a *CP-0.5,0;*).

*(Table continues)*

## *Kanji Character Set*

| Decimal Value | | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 |
| FIRST BYTE | 67 | 叩 | 但 | 達 | 辰 | 奪 | 脱 | 巽 | 竪 | 辿 | 棚 | 谷 | 狸 |
| | 68 | 帖 | 帳 | 庁 | 弔 | 張 | 彫 | 徴 | 懲 | 挑 | 暢 | 朝 | 潮 |
| | 69 | 邸 | 鄭 | 釘 | 鼎 | 泥 | 摘 | 擢 | 敵 | 滴 | 的 | 笛 | 適 |
| | 70 | 董 | 蕩 | 藤 | 討 | 謄 | 豆 | 踏 | 逃 | 透 | 鐙 | 陶 | 頭 |
| | 71 | 如 | 尿 | 韮 | 任 | 妊 | 忍 | 認 | 濡 | 禰 | 祢 | 寧 | 葱 |
| | 72 | 函 | 箱 | 硲 | 箸 | 肇 | 筈 | 櫨 | 幡 | 肌 | 畑 | 畠 | 八 |
| | 73 | 鼻 | 柊 | 稗 | 匹 | 疋 | 髭 | 彦 | 膝 | 菱 | 肘 | 弼 | 必 |
| | 74 | 福 | 腹 | 複 | 覆 | 淵 | 弗 | 払 | 沸 | 仏 | 物 | 鮒 | 分 |
| | 75 | 法 | 泡 | 烹 | 砲 | 縫 | 胞 | 芳 | 萌 | 蓬 | 蜂 | 褒 | 訪 |
| | 76 | 曼 | 蔓 | 味 | 未 | 魅 | 巳 | 箕 | 岬 | 密 | 蜜 | 湊 | 蓑 |
| | 77 | 諭 | 輸 | 唯 | 佑 | 優 | 勇 | 友 | 宥 | 幽 | 悠 | 憂 | 揖 |
| | 78 | 痢 | 裏 | 裡 | 里 | 離 | 陸 | 律 | 率 | 立 | 葎 | 掠 | 略 |
| | 79 | 蓮 | 連 | 錬 | 呂 | 魯 | 櫓 | 炉 | 賂 | 路 | 露 | 労 | 婁 |

*(Table continues)*

*Kanji Character Set*

| | Decimal Value | SECOND BYTE | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 |
| FIRST BYTE | 67 | 瀦 | 猪 | 苧 | 著 | 貯 | 丁 | 兆 | 凋 | 喋 | 寵 |
| | 68 | 汀 | 碇 | 禎 | 程 | 締 | 艇 | 訂 | 諦 | 蹄 | 逓 |
| | 69 | 橙 | 当 | 痘 | 禱 | 等 | 答 | 筒 | 糖 | 統 | 到 |
| | 70 | 弐 | 迩 | 匂 | 賑 | 肉 | 虹 | 廿 | 日 | 乳 | 入 |
| | 71 | 舶 | 薄 | 迫 | 曝 | 漠 | 爆 | 縛 | 莫 | 駁 | 麦 |
| | 72 | 樋 | 簸 | 備 | 尾 | 微 | 枇 | 毘 | 琵 | 眉 | 美 |
| | 73 | 封 | 楓 | 風 | 葺 | 蕗 | 伏 | 副 | 復 | 幅 | 服 |
| | 74 | 宝 | 峰 | 峯 | 崩 | 庖 | 抱 | 捧 | 放 | 方 | 朋 |
| | 75 | 抹 | 末 | 沫 | 迄 | 侭 | 繭 | 麿 | 万 | 慢 | 満 |
| | 76 | 訳 | 躍 | 靖 | 柳 | 薮 | 鑓 | 愉 | 愈 | 油 | 癒 |
| | 77 | 藍 | 蘭 | 覧 | 利 | 吏 | 履 | 李 | 梨 | 理 | 璃 |
| | 78 | 烈 | 裂 | 廉 | 恋 | 憐 | 漣 | 煉 | 簾 | 練 | 聯 |
| | | . . | . . | . . | . . | . . | . . | . . | . . | . . | . . |
| | 79 | . . | . . | . . | . . | . . | . . | . . | . . | . . | . . |

*Kanji Character Set*

| Decimal Value | | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 |
| FIRST BYTE | 67 | 湛 | 炭 | 短 | 端 | 箪 | 綻 | 耽 | 胆 | 蛋 | 誕 | 鍛 | 団 |
| | 68 | 長 | 頂 | 鳥 | 勅 | 捗 | 直 | 朕 | 沈 | 珍 | 賃 | 鎮 | 陳 |
| | 69 | 店 | 添 | 纏 | 甜 | 貼 | 転 | 顛 | 点 | 伝 | 殿 | 澱 | 田 |
| | 70 | 胴 | 萄 | 道 | 銅 | 峠 | 鴇 | 匿 | 得 | 徳 | 瀆 | 特 | 督 |
| | 71 | 嚢 | 悩 | 濃 | 納 | 能 | 脳 | 膿 | 農 | 覗 | 蚤 | 巴 | 把 |
| | 72 | 塙 | 蛤 | 隼 | 伴 | 判 | 半 | 反 | 叛 | 帆 | 搬 | 斑 | 板 |
| | 73 | 氷 | 漂 | 瓢 | 票 | 表 | 評 | 豹 | 廟 | 描 | 病 | 秒 | 苗 |
| | 74 | 聞 | 丙 | 併 | 兵 | 塀 | 幣 | 平 | 弊 | 柄 | 並 | 蔽 | 閉 |
| | 75 | 帽 | 忘 | 忙 | 房 | 暴 | 望 | 某 | 棒 | 冒 | 紡 | 肪 | 膨 |
| | 76 | 鵡 | 椋 | 婿 | 娘 | 冥 | 名 | 命 | 明 | 盟 | 迷 | 銘 | 鳴 |
| | 77 | 郵 | 雄 | 融 | 夕 | 予 | 余 | 与 | 誉 | 輿 | 預 | 傭 | 幼 |
| | 78 | 旅 | 虜 | 了 | 亮 | 僚 | 両 | 凌 | 寮 | 料 | 梁 | 涼 | 猟 |
| | 79 | 蠟 | 郎 | 六 | 麓 | 禄 | 肋 | 録 | 論 | 倭 | 和 | 話 | 歪 |

*(Table continues)*

*Kanji Character Set*

| | Decimal Value | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 |
| FIRST BYTE | 67 | 竹 | 筑 | 蓄 | 逐 | 秩 | 窒 | 茶 | 嫡 | 着 | 中 | 仲 | 宙 |
| | 68 | 紬 | 爪 | 吊 | 釣 | 鶴 | 亭 | 低 | 停 | 偵 | 剃 | 貞 | 呈 |
| | 69 | 倒 | 党 | 冬 | 凍 | 刀 | 唐 | 塔 | 塘 | 套 | 宕 | 島 | 嶋 |
| | 70 | 呑 | 曇 | 鈍 | 奈 | 那 | 内 | 乍 | 凪 | 薙 | 謎 | 灘 | 捺 |
| | 71 | 培 | 媒 | 梅 | 楳 | 煤 | 狽 | 買 | 売 | 賠 | 陪 | 這 | 蝿 |
| | 72 | 否 | 妃 | 庇 | 彼 | 悲 | 扉 | 批 | 披 | 斐 | 比 | 泌 | 疲 |
| | 73 | 怖 | 扶 | 敷 | 斧 | 普 | 浮 | 父 | 符 | 腐 | 膚 | 芙 | 譜 |
| | 74 | 保 | 舗 | 鋪 | 圃 | 捕 | 歩 | 甫 | 補 | 輔 | 穂 | 募 | 墓 |
| | 75 | 翻 | 凡 | 盆 | 摩 | 磨 | 魔 | 麻 | 埋 | 妹 | 昧 | 枚 | 毎 |
| | 76 | 杢 | 勿 | 餅 | 尤 | 戻 | 籾 | 貰 | 問 | 悶 | 紋 | 門 | 匁 |
| | 77 | 慾 | 抑 | 欲 | 沃 | 浴 | 翌 | 翼 | 淀 | 羅 | 螺 | 裸 | 来 |
| | 78 | 塁 | 涙 | 累 | 類 | 令 | 伶 | 例 | 冷 | 励 | 嶺 | 怜 | 玲 |
| | 79 | . . .<br>. . . | . . .<br>. . . | . . .<br>. . . | . . .<br>. . . | . . .<br>. . . | . . .<br>. . . | . . .<br>. . . | . . .<br>. . . | . . .<br>. . . | . . .<br>. . . | . . .<br>. . . | . . .<br>. . . |

*(Table continues)*

## *Kanji Character Set*

| Decimal Value |  | SECOND BYTE |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 |
| FIRST BYTE | 67 | 恥 | 智 | 池 | 痴 | 稚 | 置 | 致 | 蜘 | 遅 | 馳 | 築 | 畜 |
|  | 68 | 佃 | 漬 | 柘 | 辻 | 蔦 | 綴 | 鍔 | 椿 | 潰 | 坪 | 壷 | 嬬 |
|  | 69 | 菟 | 賭 | 途 | 都 | 鍍 | 砥 | 砺 | 努 | 度 | 土 | 奴 | 怒 |
|  | 70 | 苫 | 寅 | 酉 | 瀞 | 噸 | 屯 | 惇 | 敦 | 沌 | 豚 | 遁 | 頓 |
|  | 71 | 廃 | 拝 | 排 | 敗 | 杯 | 盃 | 牌 | 背 | 肺 | 輩 | 配 | 倍 |
|  | 72 | 煩 | 頒 | 飯 | 挽 | 晩 | 番 | 盤 | 磐 | 蕃 | 蛮 | 匪 | 卑 |
|  | 73 | 頻 | 敏 | 瓶 | 不 | 付 | 埠 | 夫 | 婦 | 富 | 冨 | 布 | 府 |
|  | 74 | 変 | 片 | 篇 | 編 | 辺 | 返 | 遍 | 便 | 勉 | 娩 | 弁 | 鞭 |
|  | 75 | 朴 | 牧 | 睦 | 穆 | 釦 | 勃 | 没 | 殆 | 堀 | 幌 | 奔 | 本 |
|  | 76 | 妄 | 孟 | 毛 | 猛 | 盲 | 網 | 耗 | 蒙 | 儲 | 木 | 黙 | 目 |
|  | 77 | 用 | 窯 | 羊 | 耀 | 葉 | 蓉 | 要 | 謡 | 踊 | 遥 | 陽 | 養 |
|  | 78 | 倫 | 厘 | 林 | 淋 | 燐 | 琳 | 臨 | 輪 | 隣 | 鱗 | 麟 | 瑠 |
|  | 79 | 湾 | 碗 | 腕 | . . | . . | . . | . . | . . | . . | . . | . . | . . |

*(Table continues)*

*Kanji Character Set*

| | Decimal Value | SECOND BYTE | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 |
| FIRST BYTE | 54 | 空 | 偶 | 寓 | 遇 | 隅 | 串 | 櫛 | 釧 | 屑 | 屈 |
| | 55 | 剣 | 喧 | 圏 | 堅 | 嫌 | 建 | 憲 | 懸 | 拳 | 捲 |
| | 56 | 候 | 倖 | 光 | 公 | 功 | 効 | 勾 | 厚 | 口 | 向 |
| | 57 | 黒 | 獄 | 漉 | 腰 | 甑 | 忽 | 惚 | 骨 | 狛 | 込 |
| | 58 | 窄 | 策 | 索 | 錯 | 桜 | 鮭 | 笹 | 匙 | 冊 | 刷 |
| | 59 | 歯 | 事 | 似 | 侍 | 児 | 字 | 寺 | 慈 | 持 | 時 |
| | 60 | 受 | 呪 | 寿 | 授 | 樹 | 綬 | 需 | 囚 | 収 | 周 |
| | 61 | 助 | 叙 | 女 | 序 | 徐 | 恕 | 鋤 | 除 | 傷 | 償 |
| | 62 | 状 | 畳 | 穣 | 蒸 | 譲 | 醸 | 錠 | 嘱 | 埴 | 飾 |
| | 63 | 枢 | 趨 | 雛 | 据 | 杉 | 椙 | 菅 | 頗 | 雀 | 裾 |
| | 64 | 浅 | 洗 | 染 | 潜 | 煎 | 煽 | 旋 | 穿 | 箭 | 線 |
| | 65 | 装 | 走 | 送 | 遭 | 鎗 | 霜 | 騒 | 像 | 増 | 憎 |
| | 66 | 濯 | 琢 | 託 | 鐸 | 濁 | 諾 | 茸 | 凧 | 蛸 | 只 |

*(Table continues)*

*Kanji Character Set*

| Decimal Value | | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 |
| | 67 | 忠 | 抽 | 昼 | 柱 | 注 | 虫 | 衷 | 註 | 酎 | 鋳 | 駐 | 樗 |
| | 68 | 堤 | 定 | 帝 | 底 | 庭 | 廷 | 弟 | 悌 | 抵 | 挺 | 提 | 梯 |
| | 69 | 悼 | 投 | 搭 | 東 | 桃 | 梼 | 棟 | 盗 | 淘 | 湯 | 涛 | 灯 |
| | 70 | 鍋 | 楢 | 馴 | 縄 | 畷 | 南 | 楠 | 軟 | 難 | 汝 | 二 | 尼 |
| | 71 | 秤 | 矧 | 萩 | 伯 | 剥 | 博 | 拍 | 柏 | 泊 | 白 | 箔 | 粕 |
| FIRST BYTE | 72 | 皮 | 碑 | 秘 | 緋 | 罷 | 肥 | 被 | 誹 | 費 | 避 | 非 | 飛 |
| | 73 | 負 | 賦 | 赴 | 阜 | 附 | 侮 | 撫 | 武 | 舞 | 葡 | 蕪 | 部 |
| | 74 | 慕 | 戊 | 暮 | 母 | 簿 | 菩 | 倣 | 俸 | 包 | 呆 | 報 | 奉 |
| | 75 | 哩 | 槙 | 幕 | 膜 | 枕 | 鮪 | 柾 | 鱒 | 桝 | 亦 | 俣 | 又 |
| | 76 | 也 | 冶 | 夜 | 爺 | 耶 | 野 | 弥 | 矢 | 厄 | 役 | 約 | 薬 |
| | 77 | 莱 | 頼 | 雷 | 洛 | 絡 | 落 | 酪 | 乱 | 卵 | 嵐 | 欄 | 濫 |
| | 78 | 礼 | 苓 | 鈴 | 隷 | 零 | 霊 | 麗 | 齢 | 暦 | 歴 | 列 | 劣 |
| | 79 | . . | . . | . . | . . | . . | . . | . . | . . | . . | . . | . . | . . |

*(Table continues)*

*Kanji Character Set*

| Decimal Value | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIRST BYTE | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 |
| 54 | 菌 | 衿 | 襟 | 謹 | 近 | 金 | 吟 | 銀 | 九 | 倶 | 句 | 区 |
| 55 | 芸 | 迎 | 鯨 | 劇 | 戟 | 撃 | 激 | 隙 | 桁 | 傑 | 欠 | 決 |
| 56 | 鼓 | 五 | 互 | 伍 | 午 | 呉 | 吾 | 娯 | 後 | 御 | 悟 | 梧 |
| 57 | 鋼 | 閤 | 降 | 項 | 香 | 高 | 鴻 | 剛 | 劫 | 号 | 合 | 壕 |
| 58 | 際 | 剤 | 在 | 材 | 罪 | 財 | 冴 | 坂 | 阪 | 堺 | 榊 | 肴 |
| 59 | 旨 | 枝 | 止 | 死 | 氏 | 獅 | 祉 | 私 | 糸 | 紙 | 紫 | 肢 |
| 60 | 杓 | 灼 | 爵 | 酌 | 釈 | 錫 | 若 | 寂 | 弱 | 惹 | 主 | 取 |
| 61 | 楯 | 殉 | 淳 | 準 | 潤 | 盾 | 純 | 巡 | 遵 | 醇 | 順 | 処 |
| 62 | 象 | 賞 | 醤 | 鉦 | 鍾 | 鐘 | 障 | 鞘 | 上 | 丈 | 丞 | 乗 |
| 63 | 酢 | 図 | 厨 | 逗 | 吹 | 垂 | 帥 | 推 | 水 | 炊 | 睡 | 粋 |
| 64 | 摂 | 折 | 設 | 窃 | 節 | 説 | 雪 | 絶 | 舌 | 蝉 | 仙 | 先 |
| 65 | 掃 | 挿 | 掻 | 操 | 早 | 曹 | 巣 | 槍 | 槽 | 漕 | 燥 | 争 |
| 66 | 苔 | 袋 | 貸 | 退 | 逮 | 隊 | 黛 | 鯛 | 代 | 台 | 大 | 第 |

*(Table continues)*

B

# Reference Material

This appendix provides information on the following topics.

- ASCII character codes.
- ASCII control codes.
- Character sets. (Information on the Kanji character set is provided in Appendix A.)

## ASCII Character Codes

One of the most commonly used computer codes is ASCII. The plotter uses 7-bit ASCII encoding with an optional eight bit for parity on RS1–232-C. ASCII is used by the plotter for I/O operations. The parity bit here is set to 0. Characters in ASCII are either nonprinting control characters or graphic printing characters.

Often, it is convenient to refer to ASCII characters using their decimal codes (the decimal equivalents of their binary codes). The following table shows three ASCII characters, their binary codes, and their decimal codes. (The graphic representation of the decimal code is dependent on the currently selected character set.

| | ASCII | ASCII |
|---|---|---|
| **Character** | **Binary Code** | **Decimal Code** |
| A | 0100 0001 | 65 |
| B | 0100 0010 | 66 |
| ? | 0011 1111 | 63 |

*Kanji Character Set*

| | Decimal Value | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
| FIRST BYTE | 67 | 鱈 | 樽 | 誰 | 丹 | 単 | 嘆 | 坦 | 担 | 探 | 旦 | 歎 | 淡 |
| | 68 | 牒 | 町 | 眺 | 聴 | 脹 | 腸 | 蝶 | 調 | 諜 | 超 | 跳 | 銚 |
| | 69 | 鏑 | 溺 | 哲 | 徹 | 撤 | 轍 | 迭 | 鉄 | 典 | 填 | 天 | 展 |
| | 70 | 騰 | 闘 | 働 | 動 | 同 | 堂 | 導 | 憧 | 撞 | 洞 | 瞳 | 童 |
| | 71 | 猫 | 熱 | 年 | 念 | 捻 | 撚 | 燃 | 粘 | 乃 | 廼 | 之 | 埜 |
| | 72 | 鉢 | 溌 | 発 | 醗 | 髪 | 伐 | 罰 | 抜 | 筏 | 閥 | 鳩 | 噺 |
| | 73 | 畢 | 筆 | 逼 | 桧 | 姫 | 媛 | 紐 | 百 | 謬 | 俵 | 彪 | 標 |
| | 74 | 吻 | 噴 | 墳 | 憤 | 扮 | 焚 | 奮 | 粉 | 糞 | 紛 | 雰 | 文 |
| | 75 | 豊 | 邦 | 鋒 | 飽 | 鳳 | 鵬 | 乏 | 亡 | 傍 | 剖 | 坊 | 妨 |
| | 76 | 稔 | 脈 | 妙 | 粍 | 民 | 眠 | 務 | 夢 | 無 | 牟 | 矛 | 霧 |
| | 77 | 有 | 柚 | 湧 | 涌 | 猶 | 猷 | 由 | 祐 | 裕 | 誘 | 遊 | 邑 |
| | 78 | 劉 | 流 | 溜 | 琉 | 留 | 硫 | 粒 | 隆 | 竜 | 龍 | 侶 | 慮 |
| | 79 | 郎 | 弄 | 朗 | 楼 | 榔 | 浪 | 漏 | 牢 | 狼 | 篭 | 老 | 聾 |

*(Table continues)*

*Reaction to Nonprinting Control Character (Continued)*

| Decimal Code | ASCII Character | All Sets |
|---|---|---|
| 15 | SI | Shift-In<br>(Select Standard Character Set) |
| 16 | DLE | NOP |
| 17 | DC1 | NOP |
| 18 | DC2 | NOP |
| 19 | DC3 | NOP |
| 20 | DC4 | NOP |
| 21 | NAK | NOP |
| 22 | SYN | NOP |
| 23 | ETB | NOP |
| 24 | CAN | NOP |
| 25 | EM | NOP |
| 26 | SUB | NOP |
| 27 | ESC | Begins device-control instruction |
| 28 | FS | NOP |
| 29 | GS | NOP |
| 30 | RS | NOP |
| 31 | US | NOP |
| 32 | SP | Space |

*Kanji Character Set*

| Decimal Value | | SECOND BYTE | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| FIRST BYTE | 67 | 壇 | 弾 | 断 | 暖 | 檀 | 段 | 男 | 談 | 値 | 知 | 地 | 弛 |
| | 68 | 津 | 墜 | 椎 | 槌 | 追 | 鎚 | 痛 | 通 | 塚 | 栂 | 掴 | 槻 |
| | 69 | 電 | 兎 | 吐 | 堵 | 塗 | 妬 | 屠 | 徒 | 斗 | 杜 | 渡 | 登 |
| | 70 | 禿 | 篤 | 毒 | 独 | 読 | 栃 | 橡 | 凸 | 突 | 椴 | 届 | 鳶 |
| | 71 | 播 | 覇 | 杷 | 波 | 派 | 琶 | 破 | 婆 | 罵 | 芭 | 馬 | 俳 |
| | 72 | 氾 | 汎 | 版 | 犯 | 班 | 畔 | 繁 | 般 | 藩 | 販 | 範 | 釆 |
| | 73 | 錨 | 鋲 | 蒜 | 蛭 | 鰭 | 品 | 彬 | 斌 | 浜 | 瀕 | 貧 | 賓 |
| | 74 | 陛 | 米 | 頁 | 僻 | 壁 | 癖 | 碧 | 別 | 瞥 | 蔑 | 箆 | 偏 |
| | 75 | 謀 | 貌 | 貿 | 鉾 | 防 | 吠 | 頬 | 北 | 僕 | 卜 | 墨 | 撲 |
| | 76 | 姪 | 牝 | 滅 | 免 | 棉 | 綿 | 緬 | 面 | 麺 | 摸 | 模 | 茂 |
| | 77 | 妖 | 容 | 庸 | 揚 | 揺 | 擁 | 曜 | 楊 | 様 | 洋 | 溶 | 熔 |
| | 78 | 療 | 瞭 | 稜 | 糧 | 良 | 諒 | 遼 | 量 | 陵 | 領 | 力 | 緑 |
| | 79 | 賄 | 脇 | 惑 | 枠 | 鷲 | 亙 | 亘 | 鰐 | 詫 | 藁 | 蕨 | 椀 |

*(Table continues)*

| Decimal Value | SET | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 4 | 5 | 6 | 9 | 11 | 14 | 19 | 21 | 36 | 37 | 38 | 39 | 43 | 83 | 85 | 115 | 147 | 267 | 563 | 595 |
| 33 | ! | ! | À | ! | ! | ! | ! | ! | ! | ! | ! | ! | ! | ｡ | ! | ! | ! | ! | ｡ | ! | ! |
| 34 | " | " | Â | " | " | " | " | " | " | " | " | " | " | ｢ | " | " | " | " | ｢ | " | " |
| 35 | # | # | È | £ | £ | # | # | # | # | § | £ | £ | # | ｣ | £ | # | # | # | ｣ | # | # |
| 36 | $ | $ | Ê | $ | $ | $ | $ | ¤ | $ | $ | $ | $ | $ | ､ | $ | ¤ | ¤ | $ | ､ | $ | $ |
| 37 | % | % | Ë | % | % | % | % | % | % | % | % | % | % | ･ | % | % | % | % | ･ | % | % |
| 38 | & | & | Î | & | & | & | & | & | & | & | & | & | & | ｦ | & | & | & | & | ｦ | & | & |
| 39 | ' | ' | Ï | ' | ' | ' | ' | ' | ' | ' | ' | ' | ' | ｧ | ' | ' | ' | ' | ｧ | ' | ' |
| 40 | ( | ( | ´ | ( | ( | ( | ( | ( | ( | ( | ( | ( | ( | ｨ | ( | ( | ( | ( | ｨ | ( | ( |
| 41 | ) | ) | ` | ) | ) | ) | ) | ) | ) | ) | ) | ) | ) | ｩ | ) | ) | ) | ) | ｩ | ) | ) |
| 42 | * | * | ^ | * | * | * | * | * | * | * | * | * | * | ｪ | * | * | * | * | ｪ | * | * |
| 43 | + | + | ¨ | + | + | + | + | + | + | + | + | + | + | ｫ | + | + | + | + | ｫ | + | + |
| 44 | , | , | ~ | , | , | , | , | , | , | , | , | , | , | ｬ | , | , | , | , | ｬ | , | , |
| 45 | - | - | Ù | - | - | - | - | - | - | - | - | - | - | ｭ | - | - | - | - | ｭ | - | - |
| 46 | . | . | Û | . | . | . | . | . | . | . | . | . | . | ｮ | . | . | . | . | ｮ | . | . |
| 47 | / | / | £ | / | / | / | / | / | / | / | / | / | / | ｯ | / | / | / | / | ｯ | / | / |
| 48 | 0 | 0 | ¯ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ｰ | 0 | 0 | 0 | 0 | ｰ | 0 | 0 |
| 49 | 1 | 1 | Ý | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | ｱ | 1 | 1 | 1 | 1 | ｱ | 1 | 1 |
| 50 | 2 | 2 | ý | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | ｲ | 2 | 2 | 2 | 2 | ｲ | 2 | 2 |
| 51 | 3 | 3 | ° | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | ｳ | 3 | 3 | 3 | 3 | ｳ | 3 | 3 |
| 52 | 4 | 4 | Ç | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | ｴ | 4 | 4 | 4 | 4 | ｴ | 4 | 4 |
| 53 | 5 | 5 | ç | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | ｵ | 5 | 5 | 5 | 5 | ｵ | 5 | 5 |
| 54 | 6 | 6 | Ñ | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | ｶ | 6 | 6 | 6 | 6 | ｶ | 6 | 6 |
| 55 | 7 | 7 | ñ | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | ｷ | 7 | 7 | 7 | 7 | ｷ | 7 | 7 |
| 56 | 8 | 8 | ¡ | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | ｸ | 8 | 8 | 8 | 8 | ｸ | 8 | 8 |
| 57 | 9 | 9 | ¿ | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | ｹ | 9 | 9 | 9 | 9 | ｹ | 9 | 9 |
| 58 | : | : | ¤ | : | : | : | : | : | : | : | : | : | : | ｺ | : | : | : | : | ｺ | : | : |
| 59 | ; | ; | £ | ; | ; | ; | ; | ; | ; | ; | ; | ; | ; | ｻ | ; | ; | ; | ; | ｻ | ; | ; |
| 60 | < | < | ¥ | < | < | < | < | < | < | < | < | < | < | ｼ | < | < | < | < | ｼ | < | < |
| 61 | = | = | § | = | = | = | = | = | = | = | = | = | = | ｽ | = | = | = | = | ｽ | = | = |
| 62 | > | > | ƒ | > | > | > | > | > | > | > | > | > | > | ｾ | > | > | > | > | ｾ | > | > |
| 63 | ? | ? | ¢ | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ｿ | ? | ? | ? | ? | ｿ | ? | ? |
| 64 | @ | @ | â | à | § | @ | @ | É | @ | @ | @ | à | § | ﾀ | § | @ | @ | § | ﾀ | @ | @ |

*(Table continues)*

| Decimal Value | SET | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 4 | 5 | 6 | 9 | 11 | 14 | 19 | 21 | 36 | 37 | 38 | 39 | 43 | 83 | 85 | 115 | 147 | 267 | 563 | 595 |
| 65 | A | A | ê | A | A | A | A | A | A | A | A | A | A | ﾁ . | A | A | A | A | ﾁ | A | ▫ |
| 66 | B | B | ô | B | B | B | B | B | B | B | B | B | B | ﾂ | B | B | B | B | ﾂ | B | ● |
| 67 | C | C | û | C | C | C | C | C | C | C | C | C | C | ﾃ | C | C | C | C | ﾃ | C | ▲ |
| 68 | D | D | á | D | D | D | D | D | D | D | D | D | D | ﾄ | D | D | D | D | ﾄ | D | + |
| 69 | E | E | é | E | E | E | E | E | E | E | E | E | E | ﾅ | E | E | E | E | ﾅ | E | x |
| 70 | F | F | ó | F | F | F | F | F | F | F | F | F | F | ﾆ | F | F | F | F | ﾆ | F | ◆ |
| 71 | G | G | ú | G | G | G | G | G | G | G | G | G | G | ﾇ | G | G | G | G | ﾇ | G | ♠ |
| 72 | H | H | à | H | H | H | H | H | H | H | H | H | H | ﾈ | H | H | H | H | ﾈ | H | x |
| 73 | I | I | è | I | I | I | I | I | I | I | I | I | I | ﾉ | I | I | I | I | ﾉ | I | z |
| 74 | J | J | ò | J | J | J | J | J | J | J | J | J | J | ﾊ | J | J | J | J | ﾊ | J | y |
| 75 | K | K | ù | K | K | K | K | K | K | K | K | K | K | ﾋ | K | K | K | K | ﾋ | K | x |
| 76 | L | L | ä | L | L | L | L | L | L | L | L | L | L | ﾌ | L | L | L | L | ﾌ | L | x |
| 77 | M | M | ë | M | M | M | M | M | M | M | M | M | M | ﾍ | M | M | M | M | ﾍ | M | x |
| 78 | N | N | ö | N | N | N | N | N | N | N | N | N | N | ﾎ | N | N | N | N | ﾎ | N | ¦ |
| 79 | O | O | ü | O | O | O | O | O | O | O | O | O | O | ﾏ | O | O | O | O | ﾏ | O | ● |
| 80 | P | P | Å | P | P | P | P | P | P | P | P | P | P | ﾐ | P | P | P | P | ﾐ | P | _ |
| 81 | Q | Q | î | Q | Q | Q | Q | Q | Q | Q | Q | Q | Q | ﾑ | Q | Q | Q | Q | ﾑ | Q | ' |
| 82 | R | R | Ø | R | R | R | R | R | R | R | R | R | R | ﾒ | R | R | R | R | ﾒ | R | R |
| 83 | S | S | Æ | S | S | S | S | S | S | S | S | S | S | ﾓ | S | S | S | S | ﾓ | S | S |
| 84 | T | T | å | T | T | T | T | T | T | T | T | T | T | ﾔ | T | T | T | T | ﾔ | T | T |
| 85 | U | U | í | U | U | U | U | U | U | U | U | U | U | ﾕ | U | U | U | U | ﾕ | U | U |
| 86 | V | V | ø | V | V | V | V | V | V | V | V | V | V | ﾖ | V | V | V | V | ﾖ | V | V |
| 87 | W | W | æ | W | W | W | W | W | W | W | W | W | W | ﾗ | W | W | W | W | ﾗ | W | W |
| 88 | X | X | Ä | X | X | X | X | X | X | X | X | X | X | ﾘ | X | X | X | X | ﾘ | X | X |
| 89 | Y | Y | ì | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | ﾙ | Y | Y | Y | Y | ﾙ | Y | Y |
| 90 | Z | Z | Ö | Z | Z | Z | Z | Z | Z | Z | Z | Z | Z | ﾚ | Z | Z | Z | Z | ﾚ | Z | Z |
| 91 | [ | Æ | Ü | ° | ° | [ | [ | Ä | [ | Æ | [ | ° | Ä | ﾛ | ¡ | [ | Ä | Ã | ﾛ | [ | [ |
| 92 | \ | Ø | É | ç | ç | ¥ | \ | Ö | \ | Ø | \ | ç | Ö | ﾜ | Ñ | \ | Ö | Ç | ﾜ | ● | \ |
| 93 | ] | Å | ï | § | é | ] | ] | Å | ] | Å | ] | § | Ü | ﾝ | ¿ | ] | Å | Õ | ﾝ | ] | ] |
| 94 | ^ | ^ | ß | ^ | ^ | ^ | ^ | Ü | ^ | ^ | ^ | ^ | ^ | ﾞ | ^ | ^ | ^ | ^ | ﾞ | ⊥ | ^ |
| 95 | _ | _ | Ô | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | ﾟ | _ | _ | _ | _ | ﾟ | _ | _ |
| 96 | ` | ` | Á | ` | ù | ` | ` | é | ` | ` | ` | µ | ` | | ` | ` | ` | ` | | ` | ` |

*(Table continues)*

| Decimal Value | SET<br>0 | 4 | 5 | 6 | 9 | 11 | 14 | 19 | 21 | 36 | 37 | 38 | 39 | 43 | 83 | 85 | 115 | 147 | 267 | 563 | 595 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 97 | a | a | Ã | a | a | a | a | a | a | a | a | a | a |  | a | a | a | a |  | a | ∩ |
| 98 | b | b | ã | b | b | b | b | b | b | b | b | b | b |  | b | b | b | b |  | b | ⊃ |
| 99 | c | c | Ð | c | c | c | c | c | c | c | c | c | c |  | c | c | c | c |  | c | ⊂ |
| 100 | d | d | ð | d | d | d | d | d | d | d | d | d | d |  | d | d | d | d |  | d | ∪ |
| 101 | e | e | Í | e | e | e | e | e | e | e | e | e | e |  | e | e | e | e |  | e | ‾ |
| 102 | f | f | Ì | f | f | f | f | f | f | f | f | f | f |  | f | f | f | f |  | f | ■ |
| 103 | g | g | Ó | g | g | g | g | g | g | g | g | g | g |  | g | g | g | g |  | g | ■ |
| 104 | h | h | Ò | h | h | h | h | h | h | h | h | h | h |  | h | h | h | h |  | h | ■ |
| 105 | i | i | Õ | i | i | i | i | i | i | i | i | i | i |  | i | i | i | i |  | i | ~ |
| 106 | j | j | õ | j | j | j | j | j | j | j | j | j | j |  | j | j | j | j |  | j | ≤ |
| 107 | k | k | Š | k | k | k | k | k | k | k | k | k | k |  | k | k | k | k |  | k | ≥ |
| 108 | l | l | š | l | l | l | l | l | l | l | l | l | l |  | l | l | l | l |  | l | ≠ |
| 109 | m | m | Ú | m | m | m | m | m | m | m | m | m | m |  | m | m | m | m |  | m | Δ |
| 110 | n | n | Ÿ | n | n | n | n | n | n | n | n | n | n |  | n | n | n | n |  | n | Π |
| 111 | o | o | ÿ | o | o | o | o | o | o | o | o | o | o |  | o | o | o | o |  | o | Σ |
| 112 | p | p | Þ | p | p | p | p | p | p | p | p | p | p |  | p | p | p | p |  | p | ± |
| 113 | q | q | þ | q | q | q | q | q | q | q | q | q | q |  | q | q | q | q |  | q | ∓ |
| 114 | r | r | · | r | r | r | r | r | r | r | r | r | r |  | r | r | r | r |  | r | → |
| 115 | s | s | µ | s | s | s | s | s | s | s | s | s | s |  | s | s | s | s |  | s | ↑ |
| 116 | t | t | ¶ | t | t | t | t | t | t | t | t | t | t |  | t | t | t | t |  | t | ← |
| 117 | u | u | ¾ | u | u | u | u | u | u | u | u | u | u |  | u | u | u | u |  | u | ↓ |
| 118 | v | v | – | v | v | v | v | v | v | v | v | v | v |  | v | v | v | v |  | v | ∫ |
| 119 | w | w | ¼ | w | w | w | w | w | w | w | w | w | w |  | w | w | w | w |  | w | ÷ |
| 120 | x | x | ½ | x | x | x | x | x | x | x | x | x | x |  | x | x | x | x |  | x | × |
| 121 | y | y | ª | y | y | y | y | y | y | y | y | y | y |  | y | y | y | y |  | y | ∇ |
| 122 | z | z | º | z | z | z | z | z | z | z | z | z | z |  | z | z | z | z |  | z | • |
| 123 | { | æ | « | é | à | { | { | ä | { | æ | { | é | ä |  | ° | { | ä | ã |  | µ | { |
| 124 | \| | ø | ■ | ù | ò | \| | \| | ö | \| | ø | \| | ù | ö |  | ñ | \| | ö | ç |  | • | \| |
| 125 | } | å | » | è | è | } | } | å | } | å | } | è | ü |  | ç | } | å | õ |  | ● | } |
| 126 | ~ | ‾ | ± | ¨ | ì | ‾ | ‾ | ü | ‾ | \| | ~ | ¨ | ß |  | ~ | ‾ | ‾ | ° |  | ~ | ‾ |

# Sample Plots and Program Listings

This appendix provides six example programs. The first four examples are designed to give you a better understanding of some of the fundamental HP-GL/2 concepts, such as scaling, rotation, and using windows. The last two are written in FORTRAN and Pascal as well as BASIC.

The explanatory text for each program listing describes the combination of HP-GL/2 concepts illustrated by the program. As you enter each program, concentrate on seeing how these concepts interact with the rest of the program to bring about the desired result.

Each sample plot follows a listing of its program. Also, the first four examples use the PS instruction to fit in a 17 × 22 in. area (C-size); the remainder fit in an 8½ × 11 in. area (A-size).

**NOTE:** In the listings, a single program line may be printed on two lines to fit on the page. You should enter such a line as though it were unbroken on the page; otherwise, you may receive unexpected results. ■

## Example 1: Scaling and P1/P2 Locations

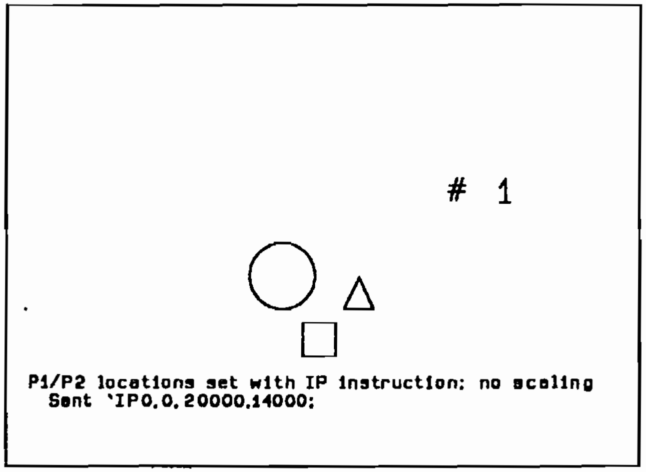
This program illustrates the effect of P1 and P2 on scaling and uses relative plotting to draw a series of figures at any location on the paper. The program is divided into four sections.

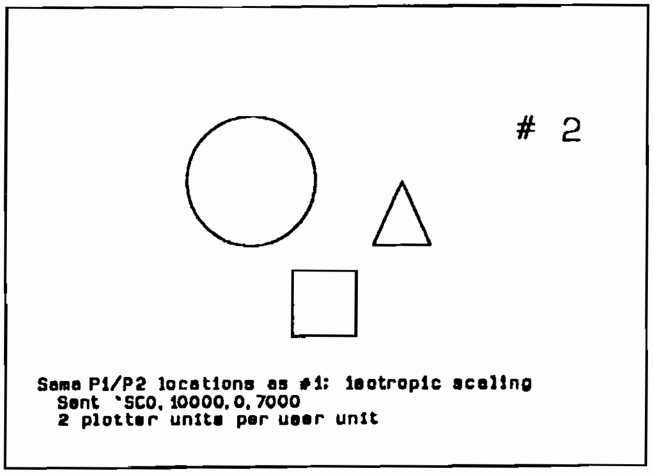
- Section 1 sets P1 and P2 to near-default locations and uses no scaling when drawing the figures.
- Section 2 adds isotropic scaling to the current P1/P2 settings.
- Section 3 turns off scaling and dramatically changes the locations of P1 and P2. (Note the similarity in size and shape to the figures in Section 1. When scaling is off, the locations of P1 and P2 have no affect on how a figure is drawn.)
- Section 4 isotropically scales these P1/P2 locations.

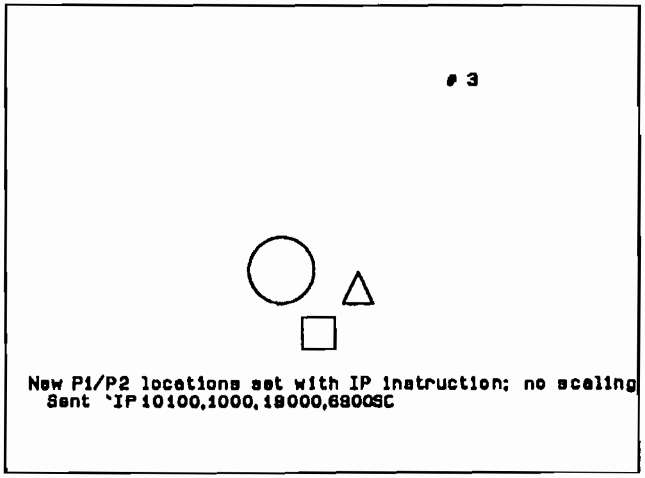
Each section is numbered and labeled on the plot. The numbers are labeled using size relative characters; that is, the size of the numbers is dependent on the relative locations of P1 and P2.

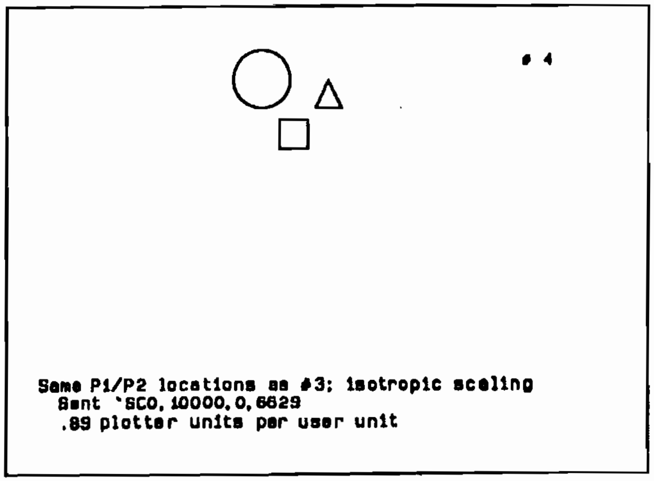
```
10  'Insert configuration statement here
15  '
25  ' Section 1 -- IP sets P1/P2; no scaling
30  PRINT #1, "INPSIP0,0,20000,14000"
35  '
45  'draws box around the quadrant
50  PRINT #1, "SP1PU0,14000"
60  PRINT #1, "PD9900,14000,9900,7100,0,7100,0,14000"
65  '
75  'draws figures
80   PRINT #1, "PU5000,10500"
90   GOSUB 600
105 '
115 'numbers quadrant
120 PRINT #1, "SR1.2,2.4PU7000,11000LB# 1"+CHR$(3)
130 PRINT #1, "SI"
135 '
145 'labels quadrant
150 PRINT #1, "PU0,7100"
160 PRINT #1, "CP3,4PR0,0PA"
170 PRINT #1, "LBP1/P2 locations set with IP instruction
                  no scaling"+CHR$(13)+CHR$(10)+CHR$(3)
180 PRINT #1, "LB  Sent `IP0,0,20000,14000' "+CHR$(3)
185 '
195 'Section 2 - Same P1/P2 locations; isotropic scaling
200 PRINT #1, "SC0,10000,0,7000"
205 '
210 PRINT #1, "SP2PU5050,3550PD5050,7000"
220 PRINT #1, "PD10000,7000,10000,3550,5050,3550"
225 '
230 PRINT #1, "PU9000,6000SR1.2,2.4LB# 2"+CHR$(3)
235 '
240 PRINT #1, "PA7525,6300"
250 GOSUB 600
255 '
260 PRINT #1, "PU5050,3550SICP3,4PR0,0PA"
270 PRINT #1, "LBSame P1/P2 locations as #1 isotropic
                  scaling"+CHR$(13)+CHR$(10)+CHR$(3)
280 PRINT #1, "LB  Sent `SC0,10000,0,7000' "+CHR$(13)
                  +CHR$(10)+CHR$(3)
290 PRINT #1, "LB  2 plotter units per user unit"+CHR$(3)
295 '
305 'Section 3 - Turn off scaling, move P1/P2
```

```
310 PRINT #1, "SR1.2,2.4IP10100,1000,19000,6900SC"
320 PRINT #1, "SP3PU0,0"
330 PRINT #1, "PD9900,0,9900,6900,0,6900,0,0"
335 '
340 PRINT #1, "PU5000,3500"
350 GOSUB 600
355 '
360 PRINT #1, "PU7000,5700LB# 3"+CHR$(3)
370 PRINT #1, "PU0,0SICP3,4PR0,0PA"
380 PRINT #1, "LBNew P1/P2 locations set with IP no
                 scaling"+CHR$(13)+CHR$(10)+CHR$(3)
390 PRINT #1, "LB  Sent `IP10100,1000,19000,6900SC'"
                 +CHR$(3)
395 '
400 'Section 4 - New P1/P2, isotropic scaling
410 PRINT #1, "IPSR1.2,2.4IP10100,1000,19000,6900"
420 PRINT #1, "SC0,10000,0,6629PU0,-1123SP4"
430 PRINT #1, "PD11123,-1123,11123,6629,0,6629,0,-1123"
435 '
440 PRINT #1, "PU5000,5979"
450 GOSUB 600
455 '
460 PRINT #1, "PU9000,5629"
470 PRINT #1, "LB# 4"+CHR$(3)
480 PRINT #1, "PU0,-1123SICP3,4PR0,0PA"
490 PRINT #1, "LBSame P1/P2 locations as #3 isotropic
                 scaling"+CHR$(13)+CHR$(10)+CHR$(3)
500 PRINT #1, "LB  Sent `SC0,10000,0,6629'"+CHR$(13)
                 +CHR$(10)+CHR$(3)
510 PRINT #1, "LB  .89 plotter units per user unit"+CHR$(3)
515 '
520 PRINT #1, "PG;"
530 END
535 '
600 'draws a circle, an equilateral triangle,
605 'and a square
620 PRINT #1, "PRPU-600,-600CI500PU1200,0"
630 PRINT #1, "PD250,-500,-500,0,250,500"
640 PRINT #1, "PU-350,-700"
650 PRINT #1, "PD-500,0,0,-500,500,0,0,500PU"
660 PRINT #1, "PA"
670 RETURN
```


# 1
P1/P2 locations set with IP instruction: no scaling
Sent 'IP0,0,20000,14000;


# 2
Same P1/P2 locations as #1: isotropic scaling
Sent 'SC0,10000,0,7000
2 plotter units per user unit

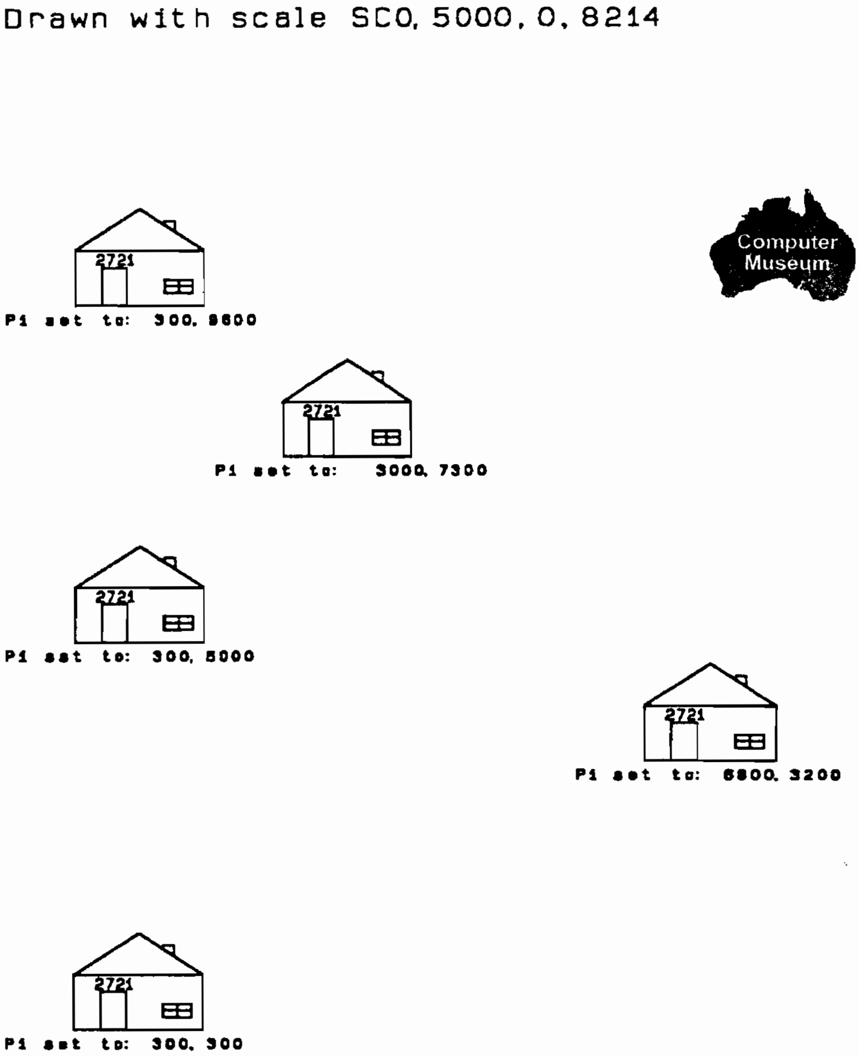

# 3
New P1/P2 locations set with IP instruction: no scaling
Sent 'IP10100,1000,19000,6800SC


# 4
Same P1/P2 locations as #3: isotropic scaling
Sent 'SC0,10000,0,6629
.89 plotter units per user unit

## Example 2: Moving a Scaled P1

This program illustrates that scaling moves with subsequent changes in the location of P1. This allows you to plot a figure using absolute coordinates anywhere on the page by changing the location of the scaled P1 point. The figure is drawn at the same scaled location each time it is drawn; only P1 moves. The scaling is isotropic.

```
10  'Insert configuration statement here
20  PRINT #1, "INPS17350,22450IP300,300,8300,17000"
30  PRINT #1, "SC0,5000,0,8214"
40  '
50  'label from scaling origin (0,0)
60  PRINT #1,"SP1PA0,0LO2LBOriginal P1: 300,300"+CHR$(3)
70  '
80  'draw figure
90  GOSUB 400
100 '
110 'change P1 only; P2 follows
120 PRINT #1,"IP300,5000"
130 PRINT #1,"PA0,0LBP1 set to: 300,5000"+CHR$(3)
150 GOSUB 400
160 '
170 'change P1, label, draw figure
180 PRINT #1,"IP300,9600"
190 PRINT #1,"PA0,0LBP1 set to: 300,9600"+CHR$(3)
200 GOSUB 400
210 '
220 'change P1, label, draw figure
230 PRINT #1,"IP3000,7300"
240 PRINT #1,"PA0,0LBP1 set to: 3000,7300"+CHR$(3)
250 GOSUB 400
260 '
270 'last P1 change
280 PRINT #1,"IP6800,3200"
290 PRINT #1,"PA0,0LBP1 set to: 6800,3200"+CHR$(3)
300 GOSUB 400
310 '
320 'return P1 to original values; label scaling
340 PRINT #1,"IP300,300;"
350 PRINT #1, "SCPU4400,19000LO5SI.45,.65CP0,-5"
360 PRINT #1,"LBDrawn with scale SC0,5000,0,8214"+CHR$(3)
370 PRINT #1,"PG;"
380 END
390 '
400 'draws a house using absolute coordinates
410 'main structure, roof, chimney, door,
420 'window frame, panes, address
430 PRINT #1,"PU500,500PDEA1500,100"
440 PRINT #1,"PD1000,800,1500,500"
450 PRINT #1,"PU1300,620PD1300,710,1200,710,1200,680"
460 PRINT #1,"PU700,100PD700,400,900,400,900,100"
470 PRINT #1,"PU1200,300PDEA1400,200"
480 PRINT #1,"PU1300,200PD1300,300PU1200,250"
490 PRINT #1, "PD1400,250PUSI.15,.25LO4"
500 PRINT #1, "PU800,420LB2721"+CHR$(3)
510 'restore program defaults
520 PRINT #1,"SILO2"
530 RETURN
```

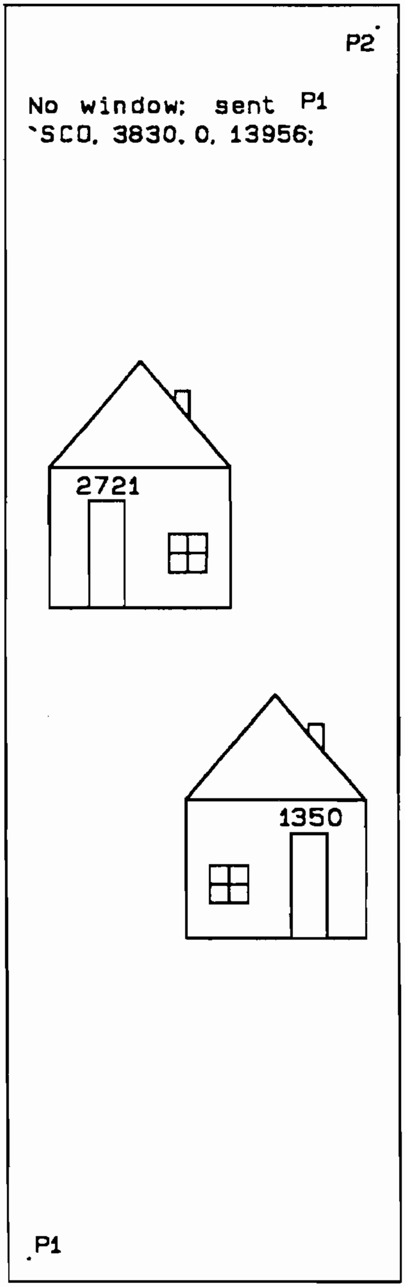

Drawn with scale SC0,5000,0,8214
2721
P1 set to: 300, 9600
2721
P1 set to: 3000, 7300
2721
P1 sat to: 300, 5000
2721
P1 set to: 6800, 3200
2721
P1 sat to: 300, 300

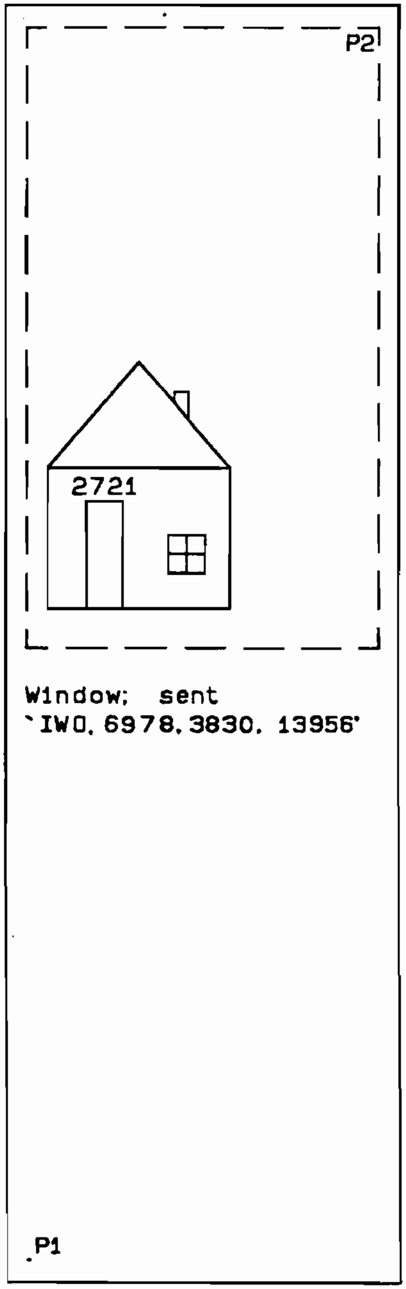
## Example 3: Using a Window

This program illustrates how a window restricts plotting to a specific area. The program first sets P1 and P2 to define an area on the left side of the paper, and then scales the area isotropically. After drawing the figures, the program moves only P1 to the right side of the paper (and P2 tracks P1). The program then establishes a window to restrict plotting to the top of the paper so that one of the figures is within the window area and the other is outside of the window. After drawing only what is within the window, the program removes the window and labels the area.

```
10  ´Insert configuration statement here
15 ´sets P1 and P2 to left side of paper
20  PRINT #1, "INPSIP250,250,3900,13650"
25 ´
35 ´frames P1/P2 area;labels
40  PRINT #1, "SP1PU0,0EA4150,13900"
50  PRINT #1, "PA250,250CI20LO11LBP1"+CHR$(3)
60  PRINT #1, "PA3900,13650CI20LO19LBP2"+CHR$(3)
70  PRINT #1, "PU250,13650CP0,-3LO1"
80 PRINT #1, "LBNo window sent"+CHR$(3)
90 PRINT #1, "CPLB`SC0,3830,0,13956´"+CHR$(3)
105 ´
115 ´scales area and draws figures
120 PRINT #1, "SC0,3830,0,13956"
130 PRINT #1, "PA1915,7350"
140 GOSUB 405
145 ´
205 ´resets P1 (P2 tracks), labels, frames area
210 PRINT #1, "IP4500,250SC"
220 PRINT #1, "PU4250,0EA8400,13900"
225 ´
230 PRINT #1, "PU4500,250CI20LO11LBP1"+CHR$(3)
240 PRINT #1, "PU8150,13650CI20LO19LBP2"+CHR$(3)
245 ´sets window and draws figures
250 PRINT #1, "SC0,3830,0,13956"
260 PRINT #1, "IW0,6978,3830,13956"
270 PRINT #1, "PU1915,7350"
280 GOSUB 405
285 ´
295 ´outlines window area
300 PRINT #1, "LT-2PU0,6978PD3830,6978,3830,13956"
310 PRINT #1, "PD0,13956,0,6978;PU;LT;IW;"
315 ´
325 ´indicate the window
330 PRINT #1, "SCPU4500,7000CP.5,-1CA5SA"
340 PRINT #1, "CP;LBWindow; sent "+CHR$(3)
```

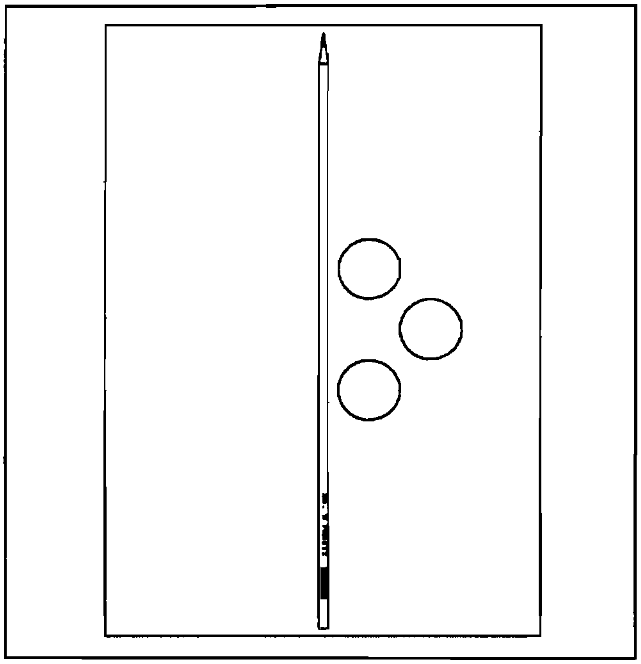
```
35Ø PRINT #1, "CP;LB`IWØ,6978,383Ø,13956;´"+CHR$(3)
36Ø PRINT #1, "PG;"
37Ø END
4Ø5 ´draws two houses
415 ´drawing the left house
42Ø PRINT #1, "SP2PRPU-17ØØ,17ØØER2ØØØ,-16ØØ"
43Ø PRINT #1,"PD1ØØØ,12ØØ,1ØØØ,-12ØØ"
44Ø PRINT #1,"PU-4ØØ,48ØPDØ,36Ø,-2ØØ,Ø,Ø,-12Ø"
45Ø PRINT #1,"PU-1ØØØ,-232ØPDØ,12ØØ,4ØØ,Ø,Ø,-12ØØ"
46Ø PRINT #1,"PU6ØØ,8ØØER4ØØ,-4ØØ"
47Ø PRINT #1,"PU2ØØ,-4ØØPDØ,4ØØPU-2ØØ,-2ØØPD4ØØ,ØPU"
48Ø ´label address and return to carriage return point
49Ø PRINT #1,"SI.15,.25LO4PU-12ØØ,68ØLB2721"
               +CHR$(13)+CHR$(3)
5Ø5 ´
515 ´drawing the bottom house
52Ø PRINT #1,"PU9ØØ,-35ØØPDER2ØØØ,-16ØØ"
53Ø PRINT #1,"PD1ØØØ,12ØØ,1ØØØ,-12ØØ"
54Ø PRINT #1,"PU-4ØØ,48ØPDØ,36Ø,-2ØØ,Ø,Ø,-12Ø"
55Ø PRINT #1,"PU2ØØ,-232ØPDØ,12ØØ,-4ØØ,Ø,Ø,-12ØØ"
56Ø PRINT #1,"PU-6ØØ,8ØØER-4ØØ,-4ØØ"
57Ø PRINT #1,"PU-2ØØ,ØPDØ,-4ØØPU-2ØØ,2ØØPD4ØØ,ØPU"
58Ø PRINT #1,"SI.15,.25LO4PU8ØØ,68ØLB135Ø"+CHR$(3)
585 ´
595 ´restore defaults
6ØØ PRINT #1,"SILO1PASP1"
61Ø RETURN
```


P2
No window; sent P1
`SC0, 3830, 0, 13956;
2721
1350
P1


P2
2721
Window; sent
`IW0, 6978, 3830, 13956'
P1

## Example 4: Rotating a Plot

This example illustrates the effect of rotaion and anisotropic scaling. First, use the following program to draw the plot shown below. The program uses absolute coordiantes to draw the pencil. then scales the area isotropically to draw the circles. Following the program are instructions for adding rotation and anisotropic scaling.

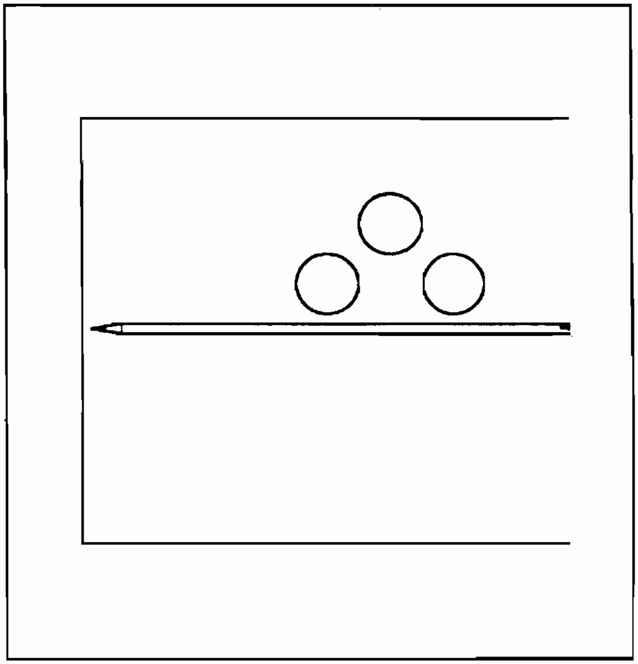


```
10  'Insert configuration statement here
20  PRINT #1,"IN"
30  PRINT #1,"PS22000,16500"
70  '
80  'draw the pencil
90  PRINT #1,"SP1PU1200,8000"
100 PRINT #1,"PD2200,8200,20800,8200,20800,7800,2200,7800,
              1200,8000"
110 PRINT #1,"PUPM0PD1600,8080,1600,7920,1200,8000PM2FT"
120 PRINT #1,"FPPU2200,8200PD2200,7800PU18800,8200"
130 PRINT #1,"PM0PD19800,8200,19800,7800,18800,7800,
              18800,8200PUPM2"
140 PRINT #1,"FT3,40,0FPFT"
150 PRINT #1,"SP2PU16300,8000LBNo. 2 Pencil"+CHR$(3)
```

```
160 '
170 'draw a frame around plot
180 PRINT #1,"SP3PU900,800"
190 PRINT #1,"EA21100,15200"
200 '
210 'isometric scale for unrotated P1/P2 values
220 PRINT #1,"SC0,9800,0,6900SP4"
230 '
240 'draw the circles
250 PRINT #1,"PU4900,5175CI500"
260 PRINT #1,"PU3900,4175CI500"
270 PRINT #1,"PU5900,4175CI500"
280 '
290 PRINT #1,"PG;"
300 END
```

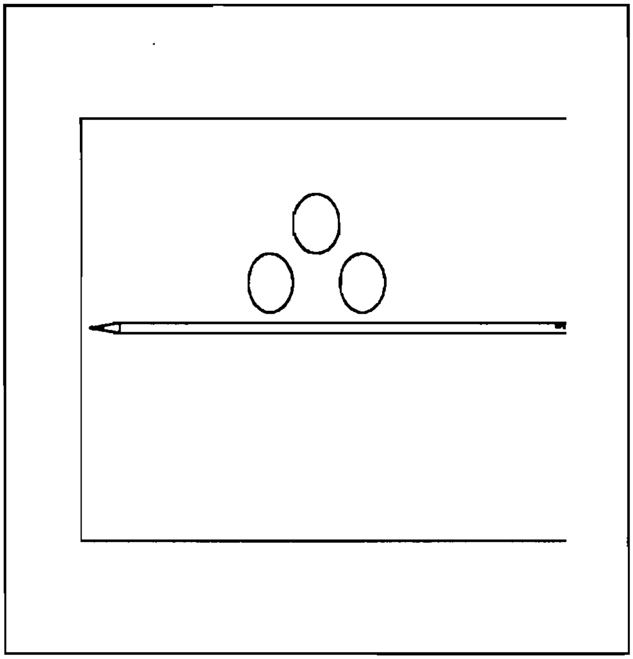
Add the following lines to rotate the plot. The pencil, drawn with absolute coordinates, is clipped when the plot is rotated.

```
40 'rotate plot and set P2 within rotated limits
50 PRINT #1,"RO90"
```



Add the following line to move P2 within the rotated limits. The resulting anisotropic scaling distorts the circles.

```
60 PRINT #1,"IP16500,0,0,16500"
```



## Example 5: FORTRAN and Pascal — AR Instruction

The following programs produce the example plot shown for the arc relative (AR) instruction. The GW BASIC program is shown first, followed by the FORTRAN and the Pascal programs. All three programs produce the same plot.

Use this example to help you get started with converting the program examples to FORTRAN or Pascal.

### GW BASIC

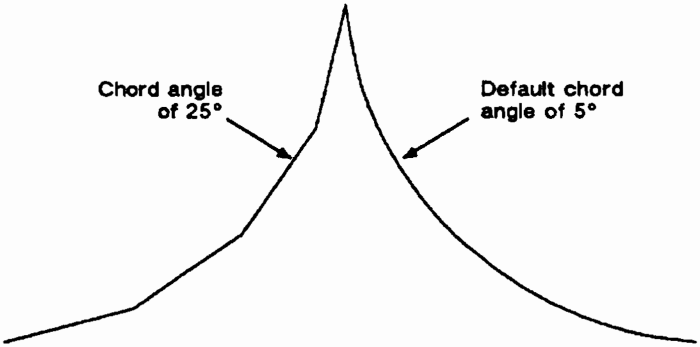
```
10 ´Insert configuration statement here
20 PRINT #1, "INSP1PS5000,7000PA1500,1500PD"
30 PRINT #1, "AR0,2000,80,25AR2000,0,80"
40 PRINT #1, "PG;"
50 END
```

### FORTRAN

```
      PROGRAM AR
      INTEGER ESCAPE
      ESCAPE = 27
      WRITE (6,10) ESCAPE,ESCAPE
10    FORMAT (1X,1R1,´.9´.P2:INSP1PS5000,7000PA1500,1500PD
               AR0,2000,80,25")
      WRITE (6,20) ESCAPE
20    FORMAT (1X,´´AR0,2000,80,25AR2000,0,80PG;´´,1R1,´´.Z´´)
      STOP
      END
```

## Pascal

```
PROGRAM AR (OUTPUT);
   CONST
      esc = CHR(27);

   PROCEDURE Arcs;
   BEGIN
      Writeln
(esc,´.(´,esc,´.P2:INSP1PS5ØØØ,7ØØØPA15ØØ,15ØØPD´);
      Writeln (´ARØ,2ØØØ,8Ø,25AR2ØØØ,Ø,8Ø´);
      Writeln (´PG;´,esc,´.z´);
   END;  (*Arcs*)

BEGIN
   Arcs
   Writeln
END (*AR*).
```



## Example 6: FORTRAN and Pascal — DI and LB Instruction

The following programs produce the example plot shown for the direction absolute (DI) instruction. The GW BASIC program is shown first, followed by the FORTRAN and the Pascal programs. All three programs produce the same plot.

Use these programs to help you convert program examples that use the label (LB) instruction to FORTRAN or Pascal.

## GW BASIC

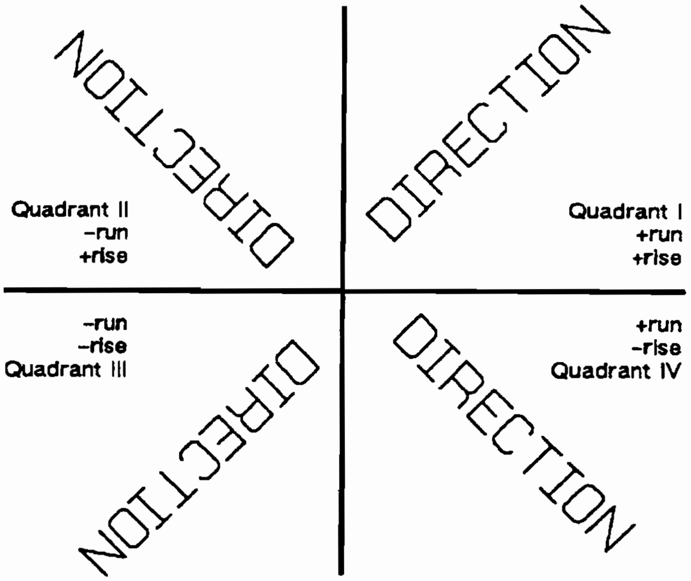
```
10   Insert configuration statement here
20  PRINT #1, "INPS5000,7000SP1PA3500,2500"
30  PRINT #1, "DI1,1LB  DIRECTION"+CHR$(13)+CHR$(3)
40  PRINT #1, "DI1,-1LB  DIRECTION"+CHR$(13)+CHR$(3)
50  PRINT #1, "DI-1,-1LB  DIRECTION"+CHR$(13)+CHR$(3)
60  PRINT #1, "DI-1,1LB  DIRECTION"+CHR$(13)+CHR$(3)
70  PRINT #1, "PG;"
80  END
```

## FORTRAN

```
         PROGRAM DI
         INTEGER ESCAPE,CR,ETX
         ESCAPE = 27
         CR = 13
         EXT = 3
         WRITE (6,10) ESCAPE,ESCAPE
10       FORMAT (1X,1R1,".(",1R1,".P2:INPS5000,7000SP1
                  PA3500,2500
         WRITE (6,20) CR,ETX
20       FORMAT (1X,"DI1,1LB  DIRECTION",1R1,1R1)
         WRITE (6,30) CR,ETX
30       FORMAT (1X,"DI1,-1LB  DIRECTION",1R1,1R1)
         WRITE (6,40) CR,ETX
40       FORMAT (1X,"DI-1,-1LB  DIRECTION",1R1,1R1)
         WRITE (6,50) CR,ETX
50       FORMAT (1X,"DI-1,1LB  DIRECTION",1R1,1R1)
         WRITE (6,60) ESCAPE
60       FORMAT (1X,"PG;",1R1,".Z")
         STOP
         END
```

## Pascal

```
PROGRAM DI (OUTPUT)
       CONST
                 esc = CHR(27);
                 cr = CHR(13);
                 etx = CHR(3);

       PROCEDURE LabelDirection;
       BEGIN
                 Writeln (esc,´.(´esc,´.P2:INPS5ØØØ,7ØØØ´);
                 Writeln (´SP1PA35ØØ,25ØØ´cr,etx);
                 Writeln (´DI1,1LB  DIRECTION´cr,etx);
                 Writeln (´DI1,-1LB  DIRECTION´cr,etx);
                 Writeln (´DI-1,-1LB  DIRECTION´cr,etx);
                 Writeln (´DI-1,1LB  DIRECTION´cr,etx);
       END ; (*LabelDirection*)

BEGIN
       LabelDirection;
       Writeln;
END (*DI*).
```



D

# Syntax Summary

This appendix lists the syntax and parameter ranges for HP-GL/2 and device-control instructions. For a complete description of each instruction, refer to the indicated page number.

## AA, Arc Absolute

Page 4-6

**SYNTAX:** AA*X center,Y center,sweep angle(,chord tolerance;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y center | current units | −67 108 863 to 67 108 863 | — |
| sweep angle | clamped real | −360 to 360 degrees | — |
| chord tolerance | | | |
| chord angle* | clamped real | 0.5 to 90 degrees | 5 degrees |
| chord deviation | current units | 0 to 67 108 863 | 5 degrees |

*Chord angle is the default interpretation of chord tolerance.

## AC, Anchor Corner

Page 5-6

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates | current units | −67 108 863 to 67 108 863 | — |

## AD, Alternate Font Definition

Page 8-9

SYNTAX: AD*kind,value...(,kind,value;)*
or
AD*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| kind | clamped integer | 1 to 7 | — |
| value | clamped real | kind dependent* | kind dependent* |

* Refer to the table following the parameter descriptions.

| Attribute | Kind* | Value | Description | ISO No. |
|---|---|---|---|---|
| Character set | 1 | 277 (0) | Roman8 (default) | — |
| | | 21 | ANSI USASCII | 006 |
| | | 14 | ECMA 94 Latin 1 | — |
| | | 6 | French v1 | 025 |
| | | 38 | French v2 | 069 |
| | | 39 | German | 021 |
| | | 563 | HP Drafting | — |
| | | 531 | HP-GL/2 Download | — |
| | | 267 | HP Kana8 | — |
| | | 43 | HP Katakana | — |
| | | 5 | HP Roman Extensions | — |
| | | 595 | HP Special Symbols | — |
| | | 85 | Intnl. Ref. Version | 002 |
| | | 9 | Italian | 015 |
| | | 11 | JIS ASCII | 014 |
| | | 4 | Norwegian v1 | 060 |
| | | 36 | Norwegian v2 | 061 |
| | | 147 | Portuguese | 016 |
| | | 115 | Swedish | 010 |
| | | 19 | Swedish Names | 011 |
| | | 83 | Spanish | 017 |
| | | 37 | United Kingdom | 004 |
| Font spacing | 2 | 0<br>1 | fixed spacing (default)<br>variable spacing | |
| Pitch | 3 | 0 to 32 767.9999 | characters per inch<br>default: 5.942 | |
| Height | 4 | 0 to 32 767.9999 | font point size<br>default: 21.3 | |
| Stroke Weight | 6 | −7<br>−3<br>0<br>3<br>7 | very light<br>light<br>normal (default)<br>bold<br>very bold | |
| Typeface | 7 | 48<br>49<br>50 | fixed vector (default)<br>drafting<br>fixed arc | |

*The kind parameter 5 is ignored.

## AR, Arc Relative

Page 4-8

**SYNTAX:** AR*X increment,Y increment,sweep angle(,chord tolerance;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y increments | current units | −67 108 863 to 67 108 863 | — |
| sweep angle | clamped real | −360 to 360 degrees | — |
| chord tolerance | | | |
| chord angle* | clamped real | 0.5 to 90 degrees | 5 degrees |
| chord deviation | current units | 0 to 67 108 863 | 5 degrees |

*Chord angle is the default interpretation of chord tolerance.

## AT, Absolute Arc Three Point

Page 4-10

**SYNTAX:** AT*X inter,Y inter,X end,Y end(,chord tolerance;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y increments | current units | −67 108 863 to 67 108 863 | — |
| sweep angle | clamped real | −360 to 360 degrees | — |
| chord tolerance | | | |
| chord angle* | clamped real | 0.5 to 90 degrees | 5 degrees |
| chord deviation | current units | 0 to 67 108 863 | 5 degrees |

*Chord angle is the default interpretation of chord tolerance.

## CI, Circle

Page 4-13

**SYNTAX:** CI*radius(;chord tolerance;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| radius | current units | −67 108 863 to 67 108 863 | — |
| chord tolerance | | | |
| chord angle* | clamped real | 0.5 to 90 degrees | 5 degrees |
| chord deviation | current units | 0 to 67 108 863 | 5 degrees |

*Chord angle is the default interpretation of chord tolerance.

## CP, Character Plot

Page 7-11

SYNTAX: CP*spaces,lines(;)*
or
CP*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| spaces | clamped real | −32 767.9999 to 32 767.9999 | — |
| lines | clamped real | −32 767.9999 to 32 767.9999 | — |

## CT, Chord Tolerance Mode

Page 4-16

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| n | integer | 0 or 1 | 0 |

## DF, Default Values

Page 2-18

SYNTAX: DF*(;)*

## DI, Direction Absolute

Page 7-14

SYNTAX: DI*run,rise(;)*
or
DI*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| run (or cos θ) | clamped real | −32 767.9999 to 32 767.9999 | 1 |
| rise (or sin θ) | clamped real | −32 767.9999 to 32 767.9999 | 0 |

## DL, Download Character

Page 8-13

**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 | — |

## DR, Relative Direction

Page 7-20

**SYNTAX:** DR*run,rise(;)*
or
DR*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| run | clamped real | –32 767.9999 to 32 767.9999 | 1% of P2X–P1X |
| rise | clamped real | –32 767.9999 to 32 767.9999 | 0% of P2Y–P1Y |

## DT, Define Label Terminator

Page 7-23

**SYNTAX:** DT*label term(,mode;)*
or
DT*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| label terminator | label | any character except **NULL**, **LF**, **ENQ**, and ; (decimal codes 0, 5, 27, and 59 respectively) | **ETX** (decimal code 3) |
| mode | clamped integer | 0 or 1 | 0 (printing) |

## DV, Define Variable Text Path

Page 7-25

**SYNTAX:** DVpath(,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) |

## EA, Edge Rectangle Absolute

Page 4-18

**SYNTAX:** EA*X,Y(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates | current units | −67 108 863 to 67 108 863 | — |

## EP, Edge Polygon

Page 6-7

**SYNTAX:** EP(;)

## ER, Edge Rectangle Relative

Page 4-20

**SYNTAX:** ER*X,Y(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y increments | current units | –67 108 863 to 67 108 863 | — |

## ES, Extra Space

Page 7-29

**SYNTAX:** ES*width(,height;)*
or
ES*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| width | clamped real | –32 767.9999 to 32 767.9999 | 0 |
| height | clamped real | –32 767.9999 to 32 767.9999 | 0 |

## EW, Edge Wedge

Page 4-22

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| radius | current units | –67 108 863 to 67 108 863 | — |
| start angle | clamped real | –360 to 360 degrees | — |
| sweep angle | clamped real | –360 to 360 degrees | — |
| chord tolerance<br>chord angle* | clamped real | 0.5 to 90 degrees | 5 degrees |
| chord deviation | current units | 0 to 67 108 863 | 5 degrees |

*Chord angle is the default interpretation of chord tolerance.

## FP, Fill Polygon

Page 6-8

SYNTAX: FP(;)

## FR, Frame Advance

Page 2-20

SYNTAX: FR(;)

## FT, Fill Type

Page 5-8

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| fill type | clamped integer | 1 to 4, 10, 11 | 1 |
| option1 and 2 | clamped real | type dependent* | type dependent* |

* Refer to the table following the parameter descriptions.

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

## IN, Initialize

Page 2-21

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| n | integer | 0 or 1 | no parameter |

## IP, Input P1 and P2

Page 2-22

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates | integer | -67 108 863<br>to 67 108 863 | hard-clip limits* |

*The hard-clip limits, in plotter units, of the D-size plotter are (0,0) to (35 376,24 000); the hard-clip limts of the E-size plotter are (0,0) to (47 568,35 840).

## IR, Input Relative P1 and P2

Page 2-24

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates | clamped real | -32 767.9999<br>to 32 767.9999 | 0,0,100,100% |

## IW, Input Window

Page 9-8

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates | current units | -67 108 863 to 67 108 863 | hard-clip limits |

## LA, Line Attributes

Page 5-11

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

| Parameter | Format | Range | Default |
|---|---|---|---|
| kind | clamped integer | 1 through 3 | 1 |
| value | clamped integer | kind dependent* | kind dependent* |

* Refer to the table following the parameter descriptions.

| 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 | 0 to 32 767.9999** | 5 (default, refer to description under *Miter Limit*) |

*Lines with a width of 0.80 mm or less always have rounded caps, regardless of the current attribute setting.

**Values less than 1.1 are set to 1.1, but do not cause an error.

## LB, Label

Page 7–31

**SYNTAX:** LB *c* ... *c*label terminator

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| c . . . c | label | any character | — |

## LO, Label Origin

Page 7-32

**SYNTAX:** LO*position(;)*
or
LO*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| position | clamped integer | 1 to 9 or 11 to 19 | 1 |

## LT, Line Type

Page 5-18

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| line type | clamped integer | –8 to 8<br>99 | solid line<br>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) |

## OE, Output Error

Page 10-5

**SYNTAX:** OE;

**NOTE:** You *must* use a terminator (;) with output instructions. ■

| Parameter | Response | Format | Range |
|---|---|---|---|
| none | error number | clamped integer | 0 to 7 |

## OH, Output Hard-Clip Limits

Page 9-11

**SYNTAX:** OH;

**NOTE:** You *must* use a terminator (;) with output instructions. ■

| Parameter | Response | Format | Range |
|---|---|---|---|
| — | XLL,YLL,XUR,YUR | integer | hard-clip limits |

## OI, Output Identification

Page 10-6

**SYNTAX:** OI;

**NOTE:** You *must* use a terminator (;) with output instructions. ■

| Parameter | Response | Format | Range |
|---|---|---|---|
| none | C1600A or C1601A | character string | — |

## OP, Output P1 and P2

Page 9-11

**SYNTAX:** OP;

**NOTE:** You *must* use a terminator (;) with output instructions. ■

| Parameter | Response | Format | Range |
|---|---|---|---|
| — | P1X,P1Y,P2X,P2Y | integer | -67 108 863 to 67 108 863* |

*P2 tracks P1 and may be outside this range.

## OS, Output Status

Page 10-7

**SYNTAX:** OS;

**NOTE:** You *must* use a terminator (;) with output instructions. ■

| Parameter | Response | Format | Range |
|---|---|---|---|
| none | status number | clamped integer | 0 to 255 |

## PA, Plot Absolute

Page 3-7

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates | current units | −67 108 863 to 67 108 863 | — |

## PD, Pen Down

Page 3-8

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates/ increments | current units | −67 108 863 to 67 108 863 | — |

## PE, Polyline Encoded

Page 3-10

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

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| flag | character | ':', '<', '>', '=', or '7' | — |
| value | character | flag dependent* | — |
| X,Y coordinates | character | −67 108 863 to 67 108 863 plotter units** | — |

*Refer to the table following the parameter description.

**PR and PE have extended ranges of $2^{31}$ to $2^{31}-1$ plotter units. 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.

## PG, Advance Page

Page 2-26

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

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| n | clamped integer | -32 767 to 32 767 | no parameter |

## PM, Polygon Mode

Page 6-10

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| polygon definition | clamped integer | 0, 1, and 2 | — |

## PR, Plot Relative

Page 3-16

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y increments | current units | -67 108 863 to 67 108 863* | — |

*PR and PE have extended ranges of $2^{31}$ to $2^{31}-1$ plotter units. 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.

## PS, Plot Size

Page 2-27

SYNTAX: PS*length(,width;)*
or
PS*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| length | integer | 400 to 609 920 plu | machine dependent** |
| width | integer | machine dependent* | paper width |

* The width range for'the D/A1-size plotter is 400 to 24 000 plu; the width range for the E/A0-size plotter is 400 to 35 840 plu.

** The default for the length parameter on the D/A1-size plotter is 35.4 in. (884 mm); the default length on the E/A0-size plotter is 47.8 in. (1189 mm).

## PU, Pen Up

Page 3-18

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

| Parameter | Format | Range | Default |
|---|---|---|---|
| X,Y coordinates/ increments | current units | -67 108 863 to 67 108 863 | — |

## PW, Pen Width

Page 2-31

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| width | clamped real | -32 767 to 32 767 | dependent* |
| pen | (ignore all but 0) | (ignore) | (ignore) |

*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.

## RA, Fill Rectangle Absolute

Page 5-22

SYNTAX: RA*X,Y(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y coordinates | current units | -67 108 863 to 67 108 863 | — |

## RF, Raster Fill Definition

Page 5-24

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

| Parameter | Format | Range | Default |
|---|---|---|---|
| index | clamped integer | 1 to 8 | 1 (solid) |
| width | clamped integer | 8, 16, 32, or 64 | — |
| height | clamped integer | 8, 16, 32, or 64 | — |
| pen number | integer | 0 to 67 108 863 | — |

## RO, Rotate Coordinate System

Page 9-12

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

| Parameter | Format | Range | Default |
|---|---|---|---|
| angle | clamped integer | 0, 90, 180, or 270 degrees | 0 degrees |

## RP, Replot

Page 2-33

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| n | clamped integer | 1 to 32 767 | 1 |

## RR, Fill Rectangle Relative

Page 5-26

**SYNTAX:** RR*X,Y(;)*

| Parameter | Format | Range | Default |
|---|---|---|---|
| X,Y increments | current units | -67 108 863 to 67 108 863 | — |

## RT, Relative Arc Three Point

Page 4-25

**SYNTAX:** RT*X incr inter,Y incr inter,X incr end,Y incr end(,chord tolerance;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| X,Y incr inter | current units | −67 108 863 to 67 108 863 | — |
| X,Y incr end | current units | −67 108 863 to 67 108 863 | — |
| chord tolerance | | | |
| chord angle* | clamped real | 0.5 to 90 degrees | 5 degrees |
| chord deviation | current units | 0 to 67 108 863 | 5 degrees |

*Chord angle is the default interpretation of chord tolerance.

## SA, Select Alternate Font

Page 8-16

**SYNTAX:** SA*(;)*

## SC, Scale

Page 2-34

**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 |
|---|---|---|---|
| $X_{min}$ | real | −67 108 863 to 67 108 863 | — |
| $X_{max}$ | real | −67 108 863 to 67 108 863 | — |
| $Y_{min}$ | real | −67 108 863 to 67 108 863 | — |
| $Y_{max}$ | real | −67 108 863 to 67 108 863 | — |
| type | clamped integer | 0 to 2 | 0 |
| left | clamped real | 0 to 100% | 50% |
| bottom | clamped real | 0 to 100% | 50% |
| $X_{factor}$ | clamped real | −32 767.9999<br>to 32 767.9999* | — |
| $Y_{factor}$ | clamped real | −32 767.9999<br>to 32 767.9999* | — |

*Excluding zero and values approaching zero

## SD, Standard Font Definition

Page 8-17

**SYNTAX:** SD*kind,value...(,kind,value;)*
or
SD*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| kind | clamped integer | 1 to 7 | — |
| value | clamped real | kind dependent* | kind dependent* |

* Refer to the table following the parameter descriptions.

| Attribute | Kind* | Value | Description | ISO No. |
|---|---|---|---|---|
| Character set | 1 | 277 (0) | Roman8 (default) | — |
| | | 21 | ANSI USASCII | 006 |
| | | 14 | ECMA 94 Latin 1 | — |
| | | 6 | French v1 | 025 |
| | | 38 | French v2 | 069 |
| | | 39 | German | 021 |
| | | 563 | HP Drafting | — |
| | | 531 | HP-GL/2 Download | — |
| | | 267 | HP Kana8 | — |
| | | 43 | HP Katakana | — |
| | | 5 | HP Roman Extensions | — |
| | | 595 | HP Special Symbols | — |
| | | 85 | Intnl. Ref. Version | 002 |
| | | 9 | Italian | 015 |
| | | 11 | JIS ASCII | 014 |
| | | 4 | Norwegian v1 | 060 |
| | | 36 | Norwegian v2 | 061 |
| | | 147 | Portuguese | 016 |
| | | 115 | Swedish | 010 |
| | | 19 | Swedish Names | 011 |
| | | 83 | Spanish | 017 |
| | | 37 | United Kingdom | 004 |
| Font spacing | 2 | 0 | fixed spacing (default) | |
| | | 1 | variable spacing | |
| Pitch | 3 | 0 to 32 767.9999 | characters per inch<br>default: 5.942 | |
| Height | 4 | 0 to 32 767.9999 | font point size<br>default: 21.3 | |
| Stroke Weight | 6 | -7 | very light | |
| | | -3 | light | |
| | | 0 | normal (default) | |
| | | 3 | bold | |
| | | 7 | very bold | |
| Typeface | 7 | 48 | fixed vector (default) | |
| | | 49 | drafting | |
| | | 50 | fixed arc | |

*The kind parameters 5 and 6 are ignored.

## SI, Absolute Character Size

Page 7-36

**SYNTAX:** SI*width, height(;)*
or
SI*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| width | clamped real | −32 767.9999 to 32 767.9999 | dependent* |
| height | clamped real | −32 767.9999 to 32 767.9999 | 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.

## SL, Character Slant

Page 7-38

**SYNTAX:** SL*tangent of angle(;)*
or
SL*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| tangent of angle | clamped real | −32 767.9999 to 32 767.9999 | 0 |

## SM, Symbol Mode

Page 7-40

**SYNTAX:** SM*character(;)*
or
SM*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| character | label | most printing characters (decimal codes 33–58, 60–126, 161, and 254)* | — |

*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 (e.g., (*SM*)).

## SP, Select Pen

Page 2-40

**SYNTAX:** *SPpen number(;)*
or
SP*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| pen number | integer | 0 to 67 108 863 | 0 |

## SR, Relative Character Size

Page 7-42

**SYNTAX:** SR*width height(;)*
or
SR*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| width | clamped real | –32 767.9999 to 32 767.9999 | 0.75% of P2X–P1X |
| height | clamped real | –32 767.9999 to 32 767.9999 | 1.5% or P2Y–P1Y |

## SS, Select Standard Font

Page 8-21

**SYNTAX:** *SS(;)*

## TD, Transparent Data

Page 7-45

**SYNTAX:** TD*mode(;)*
or
TD*(;)*

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| mode | clamped integer | 0 or 1 | 0 (normal) |

## UL, User-Defined Line Type

Page 5-29

**SYNTAX:** UL index *(,gap1,...gap20;)*
or
UL*(;)*

| Parameter | Format | Range | Default |
|---|---|---|---|
| index | clamped integer | \|1 through 8\| | — |
| gaps | clamped real | 0 to 32 767.9999 | default line types |

## WG, Fill Wedge

Page 5-31

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

| Parameter | Format | Functional Range | Default |
|---|---|---|---|
| radius | current units | −67 108 863 to 67 108 863 | — |
| start angle | clamped real | −360 to 360 degrees | — |
| sweep angle | clamped real | −360 to 360 degrees | — |
| chord tolerance |  |  |  |
| chord angle* | clamped real | 0.5 to 90 degrees | 5 degrees |
| chord deviation | current units | 0 to 67 108 863 | 5 degrees |

***Chord angle is the default interpretation of chord tolerance.**

## WU, Pen Width Unit Selection

Page 2-41

**SYNTAX:** WU*type(;)*
or
WU*(;)*

| Parameter | Format | Range | Default |
|---|---|---|---|
| type | clamped integer | 0 or 1 | 0 (metric) |

## ESC.A, Output Identification

Page 11-5

**SYNTAX:** ESC.A

| Parameter | Response | Format | Range |
|---|---|---|---|
| none | order number | character string | C1600A or C1601A |
| | firmware revision level | integer | 1 to 32 767 |

## ESC.B, Output Buffer Space

Page 11-6

**SYNTAX:** ESC.B

| Parameter | Response | Format | Range |
|---|---|---|---|
| none | available logical input buffer space | integer | 0 to 4096 bytes |

## ESC.E, Output Extended Error

Page 11-7

**SYNTAX:** ESC.E

| Parameter | Response | Format | Range |
|---|---|---|---|
| none | error number | integer | 0, 10 to 18 |

## ESC.H, Set Handshake Mode 1 (Software Enq/Ack)

Page 12-28

**SYNTAX:** ESC.H (data block size);(enquiry character);(acknowledgment string):
or
ESC.H:

| Parameter | Format | Range | Default |
|---|---|---|---|
| data block size | integer | 0 to 4096 bytes | 80 bytes |
| enquiry character | ASCII value | 0 to 26, 28 to 31* | 0 (no character) |
| acknowledge string | ASCII value | 0 to 126 | 0 (no character) |

*Practical range; printable characters (ASCII codes 32 to 126) should be avoided as they are required to send the HP-GL instructions.

## ESC.I, Set Handshake Mode 2 (Operating System)

Page 12-29

**SYNTAX for Xon-Xoff:** ESC.I *(Xoff threshold level);(omitted);(Xon trigger character(s))*:

| Parameter | Format | Range | Default |
|---|---|---|---|
| Xoff threshold level | integer | 0 to 4093 | 80 bytes |
| omitted | integer | 0* | — |
| Xon character(s) | ASCII value | 0 to 126, 1 to 10 decimal codes | 0 (no character) |

*You can designate the omitted parameter by entering a 0 or by putting the semicolon without a parameter.

**SYNTAX for Enquire/Acknowledge:** ESC.I*(data block size);(enquiry character);(acknowledgment string)*:

| Parameter | Format | Range | Default |
|---|---|---|---|
| data block size | integer | 0 to 4093 bytes | 80 bytes |
| enquiry character | ASCII value | 0 to 26, 28 to 126 decimal code* | 0 (no character) |
| acknowledgement string | ASCII value | 0 to 126, 1 to 10 decimal codes | 0 (no character) |

*Practical range; printable characters (ASCII codes 32 to 126) should be avoided as they are required to send the HP-GL/2 instructions.

## ESC.J, Abort Device-Control

Page 11-8

**SYNTAX:** ESC.J

## ESC.K, Abort Graphics

Page 11-9

**SYNTAX:** ESC.K

## ESC.L, Output Buffer Size When Empty

Page 11-9

**SYNTAX:** ESC.L

## ESC.M, Set Output Mode

Page 12-32

**SYNTAX:** ESC.M *(turnaround delay);(output trigger);(echo terminator);(output terminator);(output initiator)*:

| Parameter | Format | Range | Default |
|---|---|---|---|
| turnaround delay* | integer | 0 to 32 767 | 0 |
| output trigger | ASCII value | 0 to 4, 6 to 26, 28 to 126 decimal code | 0 (no character) |
| echo terminator | ASCII value | 0 to 4, 6 to 26, 28 to 126 decimal code | 0 (no character) |
| output terminator | ASCII value | 0 to 4, 6 to 26, 28 to 126,<br>1 or 2 decimal codes | 13 (carriage return) |
| output initiator | ASCII value | 0 to 126 decimal code | 0 (no character) |

*If the delay is odd, the plotter adds 1 to make it even.

## ESC.N, Set Extended Output and Handshake Mode

Page 12-35

**SYNTAX:** ESC.N *(intercharacter delay);(handshake dependent parameter)*:

| Parameter | Format | Range | Default |
|---|---|---|---|
| intercharacter delay | integer | 0 to 32 767 | 0 |
| handshake dependent parameter | | | |
| for Xon-Xoff: | | | |
| Xoff trigger character(s) | ASCII value | 0 to 126, up to 10 decimal codes | 0 (no character) |
| for Enquire/Acknowledge: | | | |
| immediate response string | ASCII value | 0 to 126, up to 10 decimal codes | 0 (no character) |

*If the delay is odd, the plotter adds 1 to make it even.

## ESC.O, Output Extended Status

Page 11-10

**SYNTAX:** ESC.O

## ESC.P, Set Handshake Mode

Page 12-36

**SYNTAX:** ESC.P (*handshake*):

| Parameter | Format | Range | Default |
|---|---|---|---|
| handshake | integer | 0 (none)<br>1 (Xon-Xoff)<br>2 (ENQ/ACK)<br>3 (hardwire) | 0 |

## ESC.R, Reset

Page 11-12

**SYNTAX:** ESC.R

## ESC.S, Output Configurable Memory Size

Page 11-13

**SYNTAX:** ESC.S *n*:

| Parameter | Response | Format | Range |
|---|---|---|---|
| n | (table follows) | integer | 0 to 16 900 |

## ESC.@, Set Plotter Configuration

Page 11-14

**SYNTAX:** ESC.@ *(logical input buffer size);(input conditions)*:

| Parameter | Format | Range | Default |
|---|---|---|---|
| logical input buffer size | integer | 0 to 4096 bytes | 4096 |
| input conditions | integer | 0 to 32 767 bytes | 4096 |

# Glossary

**Address**
The address specifies the plotter's location on the HP-IB (IEEE-488) interface cable (bus).

**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 or as an eighth data bit.

**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.

**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).

**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.

**Communication**
Data exchange between two or more devices.

**Configuration**

The way in which computer equipment and software is interconnected and set up to operate as a system. The plotter is configured using the switches on the front of the vector-to-raster converter. The software is configured to the plotter in the installation portion of the software.

**Configuration Switches**

The switches on the front of the vector-to-raster converter that configure the plotter for different software.

**Default**

A value or condition that is assumed if no other value or condition is specified.

**Device-Control Instruction**

The portion of the plotter's instruction set that controls internal plotter conditions such as buffer size, input/output conditions, RS-232-C interface conditions and handshakes.

**DPI**

Dots per inch, the plotter's resolution of raster images on the media. (This plotter has 406 dpi.)

**Driver**

Configuration data used by software to control input and output between the computer and a peripheral device (e.g., a plotter).

**Eavesdrop**

This plotter cannot be set up in an Eavesdrop configuration. Eavesdrop refers to a configuration in which the plotter is physically connected between a computer and some other device such as a modem, terminal, or another computer.

**Emulate**

To imitate. For example, this plotter can emulate an HP 7586B. For more information, refer to Appendix A.

**Escape Sequence**

A string of characters, beginning with the escape character (decimal 27 in ASCII). All HP-GL/2 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.

**Hewlett-Packard Graphics Language/2 (HP-GL/2)** The standardized, reduced graphics instruction set that is a subset of HP-GL.

**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.

**Interface** Anything (a cable, for example) used to join components of a computer system so they function in a compatible and coordinated fasion. Standards that allow systems to connect with each other (e.g., HP-IB, RS-232-C).

**Interface Cable** The data transmission cable used to connect a peripheral device to a computer. Most devices require an HP-IB, Centronics, or RS-232-C interface cable.

**I/O Error** Examples of I/O errors are mismatched interface conditions, such as baud rate and parity.

**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.

**P1** A scaling point the plotter uses that generally specifies the location of a plot's lower-left corner.

**P2** A scaling point the plotter uses that generally specifies the location of a plot'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.

**Peripheral (device)** A device separate from, but used with, a computer. For example, a disc drive,.printer, or plotter.

**Print Engine** The part of the plotter that prints the raster data on the media. The vector-to-raster converter is not part of the print engine.

**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.

**RS-232-C Interface** A serial interface standardized by the Electronic Industries Association Standard RS-232-C.

**Resolution** A measure of image sharpness. In electrostatic plotters, resolution is measured in dots per inch (dpi).

**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/2 scale (SC) instruction. These points, also known as P1 and P2, define opposite corners of a rectangular area.

**Serial Interface** A serial interface uses a single data line to transfer data bits sequentially between devices. RS-232-C is a serial interface.

**Standalone Configuration** When using the RS-232-C interface, a configuration where the plotter is connected to the computer via a separate (not a shared) cable.

**Stop Bit** In an RS-232-C configuration, one or two bits following a transmitted piece of information; used to notify the receiving device that the information is complete.

**Vector-to Raster Converter** (Also called the VRC.) A device that converts the vector data to raster data after all of the data has been sent to the plotter and stored on the internal disc.

# Subject Index