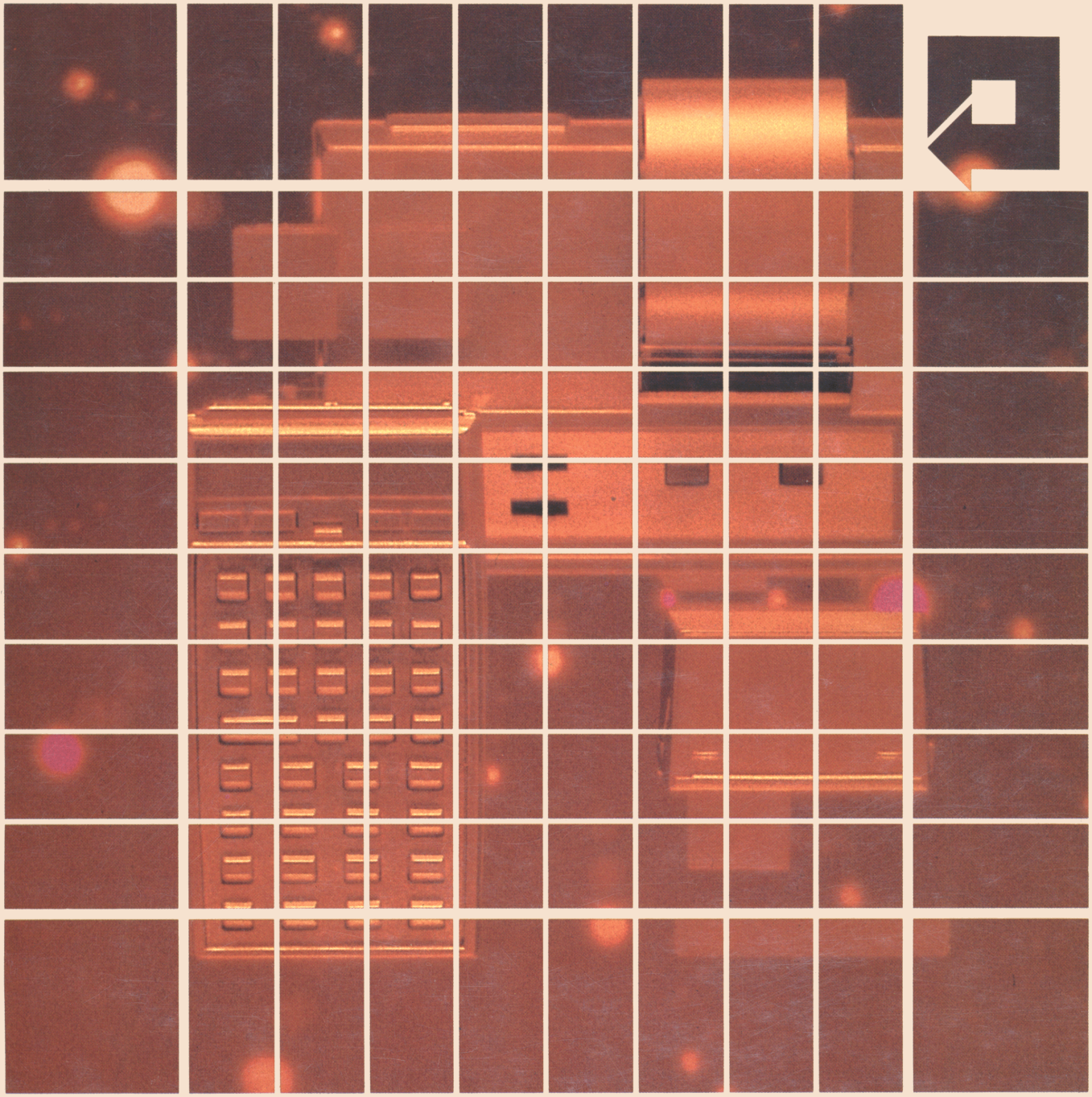
HEWLETT-PACKARD

HP 82184A

Plotter Module

OWNER'S MANUAL

For Use With the HP-41



NOTICE

Hewlett-Packard Company makes no express or implied warranty with regard to the program material offered or the merchantability or the fitness of the program material for any particular purpose. The program material is made available solely on an ''as is'' basis, and the entire risk as to its quality and performance is with the user. Should the program material prove defective, the user (and not Hewlett-Packard Company nor any other party) shall bear the entire cost of all necessary correction and all incidental or consequential damages. Hewlett-Packard Company shall not be liable for any incidental or consequential damages in connection with or arising out of the furnishing, use, or performance of the program material.

HP 82184A
Plotter Module

Owner's Manual

For Use With the HP-41

September 1982

82184-90001

Printed in U.S.A.

Preface

This manual was written with the assumption that you have a working knowledge of the HP-41 Handheld Computer, the HP 82160A HP-IL Module, a compatible plotter, and—for those using the plotter module's bar code functions—the HP 82153A Optical Wand. For information regarding the operation of any of these units, please refer to the appropriate owner's manual.

To use the HP 82184A Plotter Module, you need to know some basic operating information, plus the operation of the module's Utility Plotting Program and/or individual plotting functions. The two main parts of this manual cover these topics as follows:

- Part I (sections 1 and 2) describes module installation, HP-41 memory and system configuration, and how to use this manual; then provides three plotting examples and a detailed description of the Utility Plotting Program.
- Part II (sections 3 through 8) describes the plotter module's individual plotting and bar code functions, as well as programs you can use to expedite bar code generation.

Appendix A describes plotter module-related error messages. Appendix B provides care, warranty, and service information. Appendix C provides annotated program listings including three programs that enhance the use of the plotter module's Utility Plotting Program, and flowcharts. Appendix D provides bar code for the programs described in section 7, as well as for several other programs listed in the manual. Appendix E provides reference information relative to HP-41 programming and the Hewlett-Packard Graphics Language, a comparison of plotter module and Hewlett-Packard Series 80 graphics functions. Appendix F provides a series of charts and tables that are useful aids to designing direct execution and paper keyboard bar code.

A function index marked by the blue-edged pages near the end of the manual enables you to rapidly locate the primary description of each plotter module function and each of the routines in the Utility Plotting Program. Following the function index is a subject index that is useful for locating information describing a wide range of plotter module topics. An additional reference, the *HP 82184A Plotter Module Quick Reference Card*, is also provided with your module as a "memory jogger." This pocket-sized document can help you to recall operating details that you have read, but may have forgotten.

The plot illustrations used in this manual were generated using an HP 7470A plotter (option 003*).

---

*Option 003 on the HP 7470A Plotter allows the plotter to operate in the Hewlett-Packard Interface Loop (HP-IL).

Contents

Introduction 7

Part I: Using the Plotter Module

Section 1: Getting Started 10

Contents—Installing and Removing the Module—Memory and Equipment Configuration—Using This Manual—Plotting a Function—Plotting a Line Graph and a Bar Chart—Returning I/O Buffer Registers to Available Memory—Where To Read Next?

Section 2: The Utility Plotting Program 19

Contents—Introduction—Program Overview—User Keyboard, Key Assignments, and Keyboard Overlay—The NEWPLOT Routine—The REPLOT Routine—Plotting Examples—The PLINIT Routine—The PLTUXY Routine—The PLANOT Routine—Increasing Your Plotting Skills

Part II: Plotter Module Functions

Section 3: The Plotting Area and Scale 64

Contents—Introduction—Initializing and Clearing the I/O Buffer—Specifying Graphic Limits—Setting the User Scale—Revising the Plot Bounds—Page Advance—Saving Steps

Section 4: Plotting 78

Contents—Framing the Plotting Area—Moving the Pen and Drawing Lines—Using Plot-Option Functions—Rotating the Plot Axes—Other Pen Control Functions

Section 5: Labels and Axes 92

Contents—Introduction—Using Labels—Using Axes

Section 6: Digitizing 104

Contents—Introduction—Selecting a Point—Identifying a Point—Digitizing Plot Bounds

Section 7: Bar Code 108

Contents—Introduction—Terminology—Recommended Pen and Paper—Controlling Bar and Space Proportions—Using a Program to Generate Bar Code—Bar Code Subroutines—Basic Bar Code Functions—Printing HP-41 Bar Code on the HP 82162A Thermal Printer—Utility Bar Code Functions—Four-Bit Mirror Checksum—Four-Bit Summation Checksum—BC and Null Bytes—Eight-Bit Summation Checksum—Plotting Alternative (Non HP-41) Bar Code

Section 8: Plotting Conditions and Input/Output Buffer 148

Contents—Introduction—The 26-Register I/O Buffer—Default Plotting Conditions

Appendix A: Error Messages 154

Appendix B: Care, Warranty, and Service Information 156

Appendix C: Program Documentation 160

Appendix D: Bar Code 195

Appendix E: Reference Information 210

Appendix F: Bar Code Specification Charts 213

Function Index 221


HEWLETT · PACKARD
41CV
PLOTTER
APR
JUN
Month
AUG
OCT
ROW
ROW 2:
ROW 3:
ROW 4:
ROW 5:
ROW 6:
ROW 7:
1200
600
0
0
2

Introduction

The HP 82184A Plotter Module enables an HP-41 Handheld Computer to control Hewlett-Packard plotting devices that operate in the Hewlett-Packard Interface Loop (HP-IL) and are compatible with the Hewlett-Packard Graphics Language (HP-GL). The plotter module allows you to control such plotting devices through the HP 82160A HP-IL Module, either directly from the HP-41's keyboard or in a running program.

The module contains 52 plotter functions (including 10 bar code functions) and the Utility Plotting Program. In addition, a line graph program and a bar chart program are included in the manual to help you get started and to provide applications programs you can modify to suit your needs.

The bar code plotting functions included in the module enable you to use your HP-41/plotter system to plot HP-41 bar code or one of three other types of bar code, and to print HP-41 bar code using the HP 82162A Thermal Printer. The interactive PLOTBC program described early in section 7 allows you to easily generate HP-41 data or program bar code on a plotter. This program is recommended for novices and for anyone who needs only to generate a series of program or data bar code rows on a page. You can use the bar code subroutines and the bar code functions (described in section 7) for designing your own bar code programs or for controlling bar code generation from the keyboard. The utility bar code functions at the end of the section enable technically advanced users to generate paper keyboard, direct execution, or non HP-41 bar code. It is not necessary to have an HP 82153A Optical Wand available when you use the plotter module to generate bar code. However, using a wand helps you to quickly verify the contents and readability of your bar code.

Part I
Using the Plotter Module

Section 1

Getting Started

Contents

Installing and Removing the Module 10
Memory and Equipment Configuration 10
Memory Requirements and the I/O Buffer 10
The Hewlett-Packard Interface Loop 11
Using This Manual 11
Function Descriptions 12
Displays 12
Setting Up Your System for Examples 12
How to Use the Rest of Section 1 12
Plotting a Function 12
Plotting a Line Graph and a Bar Chart 15
Configuring HP-41 Memory 15
Line Graph Example 15
Bar Chart Example 16
Returning I/O Buffer Registers to Available Memory 18
Where to Read Next? 18

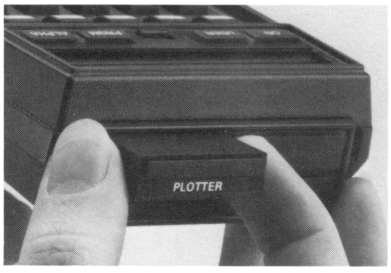
Installing and Removing the Module

Install the plotter module as described below, and use an HP 82160A HP-IL Module to connect your HP-41 to an appropriate plotter. (The interface loop can also contain other devices without affecting plotter operation.)

CAUTION

Be sure the HP-41 is turned off before inserting or removing the plotter module. If this is not done, the HP-41 may be damaged or its operation may be disrupted.

The HP 82184A Plotter Module plugs into any of the HP-41's ports. (If any HP 82106A Memory Modules are also plugged in, the plotter module must be in a higher-numbered port than the memory modules.) Push in the module until it snaps into place. When you remove the module, remember to place a port cap over the unused port.



Memory and Equipment Configuration

Memory Requirements and the I/O Buffer

The I/O Buffer. The module uses a block of 26 HP-41 registers to form a plotting data storage unit (termed an *input/output*—or *I/O—buffer*) for storing parameters used by the module to control the plotter. (The PINIT function described on page 68 creates this buffer. Most of the functions described in part II either use data stored in the buffer or change that data.)

Minimum Memory Requirements. Since the plotter module automatically controls the I/O buffer, you need be concerned *only* that there are enough memory registers available to create the buffer when you first execute [PINIT]. There is no other action you need to take concerning the I/O buffer. Of course, when determining how many registers you need available for a particular application, you need consider not only the 26 registers needed for the I/O buffer, but also how many registers you need for data storage and program memory.

Ensuring that Enough Unused Registers Are Available. To easily verify how many unused memory registers are available, press [GTO] [.] [.] to pack memory, then set the HP-41 to program mode and check the number of registers indicated by the displayed **00 REG *nn*** message. If the number shown by ***nn*** is less than the number of registers you need, create the necessary additional registers by executing [SIZE] or clearing one or more programs from memory (or, if you are using the model HP-41*C*, by installing the HP 82170A Quad Memory Module or one or more HP 82106A Memory Modules).

I/O Buffer Permanence. Once the I/O buffer has been created, it remains in memory until you either execute the [PCLBUF] (*clear plotter buffer*) function (described on page 69) or remove the plotter module from the HP-41, then turn on the HP-41. When you perform either of these operations, the 26 registers used by the I/O buffer are restored to available memory.

What to Expect When Executing Examples. Each of the examples in this part begin with a series of keystrokes that help ensure that your HP-41 contains enough properly configured memory to perform the example. If you are relatively inexperienced at manipulating the HP-41's memory, these aids will help you learn how to anticipate memory requirements for your plotting applications.

The examples in part II do not include keystrokes for configuring memory. When you are ready to begin executing the examples in part II, if you find that you need assistance to properly configure memory, reread the preceding four paragraphs. If necessary, refer also to the description of the [SIZE] function in your HP-41 owner's manual.

The Hewlett-Packard Interface Loop

You control a plotter by using your HP-41, the plotter module, and an HP 82160A HP-IL Module. If you use more than one plotter in the loop, plotter module functions will always address the first plotter in the loop unless you use the HP-IL [SELECT] function to specify another plotter. For further information about HP-IL operation, refer to the *HP 82160A HP-IL Module Owner's Manual.*

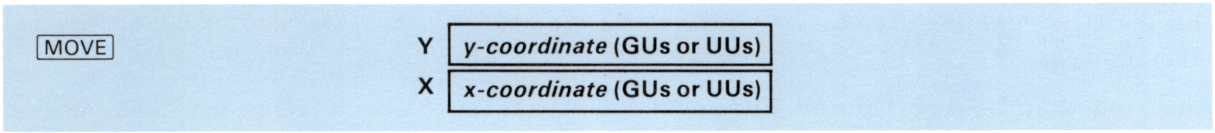
> **Note:** When using an HP 7470A or similar plotter, you should avoid executing plotter module functions while the plotter's VIEW function is active or while the CHART HOLD/CHART LOAD lever is in the "up" position. Executing some plotter module functions while the plotter's VIEW function is active results in a **TRANSMIT ERR** message. Execution of non-pen functions may be delayed by VIEW. Executing a pen movement function while the CHART HOLD/CHART LOAD lever is in the "up" position causes the plotter to ignore the function. Also, while VIEW is active or while the CHART lever is up, some HP-41 functions may take longer than normal to execute.

Using This Manual

For simplicity, plotter module functions (and any other functions not on the standard HP-41 keyboard) are represented by single, colored keys—such as [MOVE]. You can execute such functions in two ways: By using [XEQ] [ALPHA] ***name*** [ALPHA], or by assigning the function to a key using [ASN] and pressing that key in User mode. (Refer to the owner's manual for your HP-41.)

Function Descriptions

The description of each function (in part II) is preceded by a summary of information required by that function. This provides a quick, visual description of how to execute the function. For example:



This indicates that to move the plotter pen to a new point (defined by the current scale—which is specified in either graphic units or user units) you must place the point's *y*-axis coordinate in the Y-register and *x*-axis coordinate in the X-register before you execute MOVE —from the keyboard or in a program.

Displays

Unless otherwise indicated, all examples assume that you are beginning with the display cleared and set to FIX 4 display mode (that is, 0.0000).

If at any time your HP-41 displays an error message, refer to appendix A, Error Messages, for an explanation of its cause. For certain conditions the error message may not be displayed until after a short delay.

Setting Up Your System for Examples

Before you begin each example, ensure that your HP-41/plotter system is ready for operation. Unless otherwise indicated, this includes a fresh sheet of paper in your plotter.

How to Use the Rest of Section 1

At various times you may need to quickly generate a standard plot. The Utility Plotting Program enables you to do so with a minimum of preparation. To see how easy this is, step through the example that follows the next heading. Or, if you are familiar with plotters and wish to begin using the plotter immediately with your own programs and keystroke routines, you may want to skip the rest of part I for now and begin reading in part II. However, if you are new to plotters, expect to use your plotter module mainly with programs written by others, or want to see further demonstrations of the plotter module's capabilities, work through the last two examples in this part. Later, if you want to begin developing your own plotting programs, you will find in part II the information you need.

The three examples that follow are intended to demonstrate the plotter module's capabilities rather than to instruct you in plotter module operation. Therefore, in each of these examples, most of the text pertains to the general progression of the example instead of to the purpose and operating details of individual functions. Instructional material is provided later, in section 2 and in part II. So, if you are ready, plug in your plotter module, connect the HP-41 to your plotter, and begin plotting!

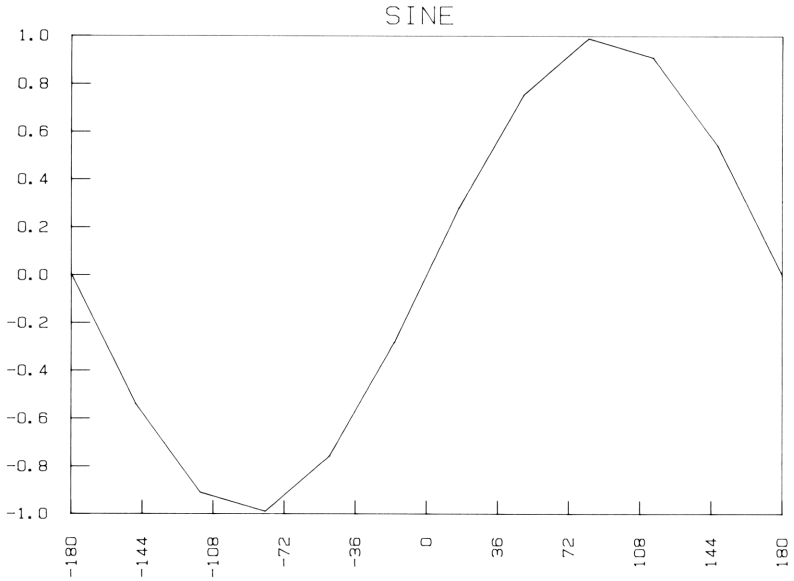
Plotting a Function

The Utility Plotting Program in your plotter module contains five plotting routines* that use your inputs to automatically generate a complete plot—that is, a plot that includes:

- A graphic representation of a math function or series of points you specify.
- A frame defining the plotting area.
- Scale annotations on the *x*- and *y*-axes.

*A *routine* in the HP-41 is a sequence of program instructions that you can execute either independently of other program instructions or as a subroutine. A *program* in the HP-41 consists of all program instructions between the top of program memory and the first **END** instruction, or between any **END** instruction and the next **END** instruction.

Utility Plotting Program Example: Generate the following plot of the sine function between −180° and 180°.



Your first step is to ensure that there are enough data registers available for use by the plotting program (R_{00} through R_{11}). While doing so you can also determine if there are enough unused memory registers for the short program used in the example (2 registers), and for the I/O buffer mentioned on page 10 (26 registers). Then load into memory a program that, given an *x*-coordinate (in this case, any angle between −180° and 180°), returns to the X-register a *y*-coordinate (in this case, the sine of the angle originally placed in the X-register). This program will be used by the plotter module's Utility Plotting Program to generate the *y*-coordinate of each point to be plotted. Before you execute the following keystrokes, turn your plotter off, then on, to ensure that it is set to its default graphic limits.

Keystrokes	Display	
SIZE 012		Ensures that R_{00} through R_{11} are available. (Does not affect the display remaining from any of your previous HP-41 operations.)
DEG		Ensures that the HP-41 is in Degrees mode.
■ GTO · ·		Packs program memory.
PRGM	00 REG *nn*	Switches HP-41 into Program mode. The number represented by ***nn*** must be 28 or greater.*

*This assumes that the I/O buffer has not been created by some other plotting operation you may have executed before turning to this example. If in doubt as to whether or not the I/O buffer currently exists, switch your HP-41 out of Program mode and execute PCLBUF. If the buffer exists, this function clears it and returns the registers used by the buffer to unused memory. If the buffer does not exist, **PL:PLS PINIT** is displayed. In either case, switch the calculator back into Program mode. If you still have fewer than 28 registers available, refer to Ensuring that Enough Unused Registers Are Available (page 11), before proceeding with the above example.

Keystrokes	Display	
■ LBL	01 LBL__	
ALPHA SINE ALPHA	01 LBL"SINE	Enters subroutine label.
SIN	02 SIN	Enters sine function.
PRGM	0.0000	Switches HP-41 out of program mode.

Now you are ready to begin executing the Utility Plotting Program. The HP-41 will prompt you for several parameters that control the basic plotting operation. The first parameter you will enter identifies the name of the subroutine you keyed in to calculate the *y*-coordinates (SINE). The next two parameters specify the lower and upper limits of the *x*-axis.

Keystrokes	Display	
NEWPLOT	NAME=?	Prompts you to key in the name of the subroutine that calculates the *y*-coordinate of the plot.
USER		Deactivates User keyboard activated by NEWPLOT. (The User keyboard is not needed in this example. Its general use is described on page 21 in section 2.)
SINE R/S	XMIN=−1.000?	Enters function name. HP-41 prompts you to select the minimum *x*-axis value. (The −1.000 in the display is the default *x*-axis minimum that is set whenever you execute NEWPLOT without specifying a minimum.)
180 CHS R/S	XMAX=1.000?	Enters minimum *x*-axis value. HP-41 prompts you to select the maximum *x*-axis value. (The displayed 1.000 is the default *x*-axis maximum.)
180 R/S	XINC=−11.000?	Enters maximum *x*-axis value. HP-41 prompts you to select the *x*-increment. (Ignore the minus sign for now.)

The preceding display prompts you to specify the number of equal increments (that is, the intervals between *x*-coordinates). The displayed default parameter is 11 increments.

Keystrokes	Display	
R/S	YMIN=−1.000?	Enters the displayed default number of increments. HP-41 prompts you to select the minimum *y*-axis value.

The next two parameters you enter specify the top and bottom boundaries of the plot.

Keystrokes	Display	
R/S	YMAX=1.000?	Enters default *y*-minimum shown in the preceding display. HP-41 prompts you for *y*-maximum value.
R/S	PLOT?	Enters default *y*-maximum. The HP-41 sounds a tone and prompts you to indicate whether or not to generate a plot.

You have now specified the source of the *y*-coordinates for your plot, the number of equally-spaced *x*-increments, and the scale and boundaries of your plot. The system is now ready to generate a plot.

Keystrokes	Display	
R/S		Plots the sine function with appropriate chart annotation.
	PLOT?	Prompts you to indicate whether or not to generate another plot.
■ FIX 4 ←	0.0000	Sets display mode to FIX 4 and clears display.

NEWPLOT and the other routines in the Utility Plotting Program enable you to easily perform complete plots using a variety of data options. Section 2, The Utility Plotting Program, describes these options and other operating details you will need to know to make full use of the program.

Plotting a Line Graph and a Bar Chart

The preceding example introduced you to your plotter module's utility plotting program. However, you may also have a need to perform unique plotting operations that are best undertaken using other, more specialized plotting programs that you create. Such programs use plotter module functions (described in part II of this manual), HP-41 functions, and, wnere desired, functions belonging to HP-41 extensions and/or peripherals. If you plan either to write plotting programs yourself or to use programs written by others, the following two examples will help you visualize some of the ways the plotter module can be adapted to your specific needs.

Configuring HP-41 Memory

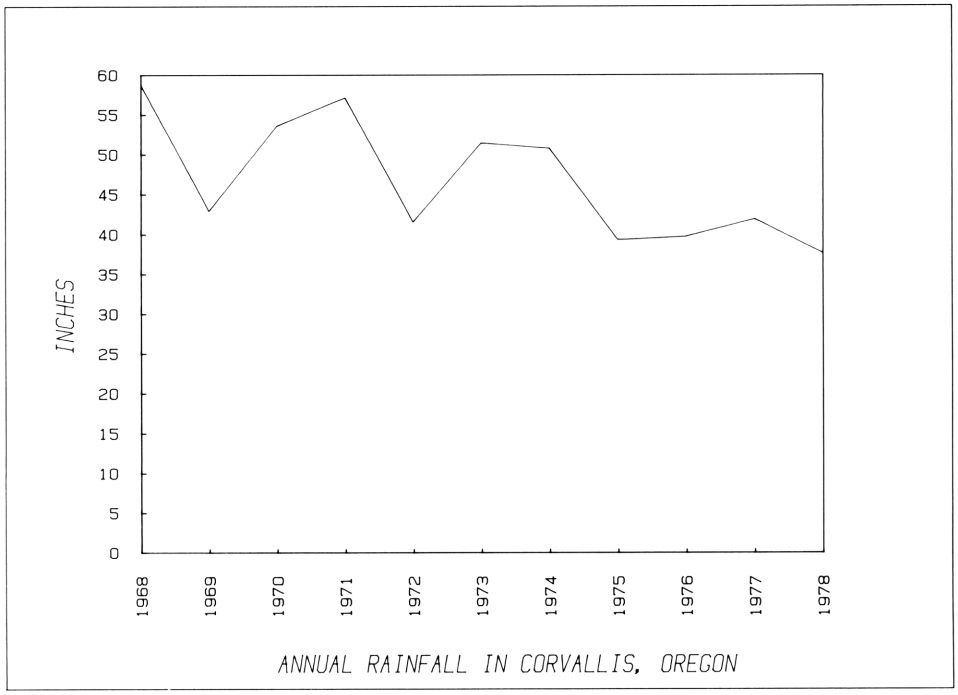
The next example requires that you have a minimum of 80 registers available in your HP-41.* That is, 31 memory registers to hold the program, 26 unused registers (for the I/O buffer mentioned on page 10), and 23 data storage registers. The next keystroke series ensures this configuration.

Keystrokes	Display	
■ GTO · ·	0.0000	Packs program memory.
SIZE 023	0.0000	Ensures that R_{00} through R_{22} are available.
PCLBUF	0.0000	If the I/O buffer remains from an earlier plotting operation, this function restores to unused memory the registers used by the buffer. If no buffer exists, the message PL:PLS PINIT is displayed.
PRGM	00 REG *nn*	Switches the HP-41 to Program mode. The number of registers indicated by ***nn*** must be 57 or more. If not, refer to Ensuring that Enough Unused Registers Are Available, page 11.
PRGM	0.0000	Removes HP-41 from Program mode.

Line Graph Example

Suppose you wanted to plot the average annual rainfall in Corvallis, Oregon, for the 11-year period from 1968 through 1978. You could write and load into program memory an independent program that plots years on the *x*-axis and inches on the *y*-axis. (That is, a program that does not use the plotter module's built-in Utility Plotting Program.) Press ■ GTO · · and load the RAIN program from either the bar code on page 203 or the program listing on page 160. Then execute RAIN and key in the rainfall for each year. When you press the keys shown on the next page, the plotter generates a graph like the following:

*If you are using a model HP-41*C*, you will need one HP 82106A Memory Module or the HP 82170A Quad Memory Module.



Before you execute the following keystrokes, turn your plotter off, then on, to ensure that it is set to its default graphic limits.

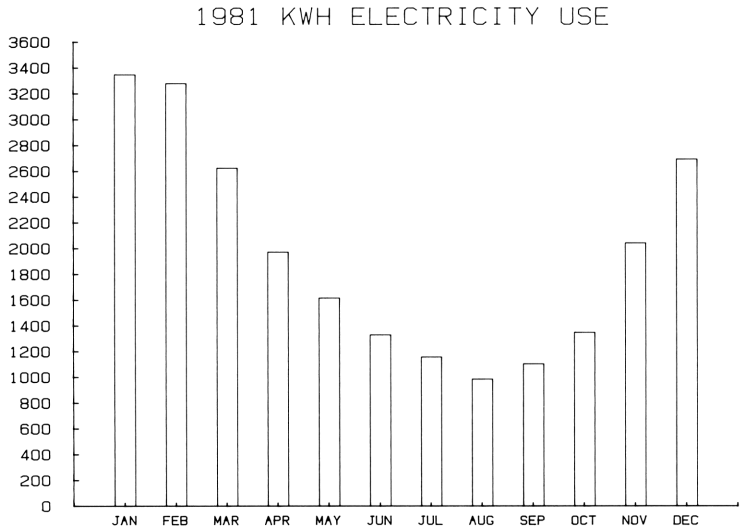
Keystrokes	Display	
[XEQ] [ALPHA] RAIN [ALPHA]	XEQ RAIN_ 1968=?	Executes RAIN. HP-41 prompts for the 1968 rainfall.
58.73 [R/S]	1969=?	Enters rainfall for each displayed year, then prompts for the next year.
42.92 [R/S]	1970=?	
53.60 [R/S]	1971=?	
57.15 [R/S]	1972=?	
41.56 [R/S]	1973=?	
51.44 [R/S]	1974=?	
50.76 [R/S]	1975=?	
39.33 [R/S]	1976=?	
39.73 [R/S]	1977=?	
41.91 [R/S]	1978=?	
37.64 [R/S]		Enters rainfall for final year, then plots graph.
	0.0000	Plot completed.

Bar Chart Example

This example requires that you have a minimum of 78 registers available in your HP-41. If you have not altered memory since executing the preceding example, just use the following keystrokes to reconfigure memory for the next program. Otherwise, execute the keystrokes under Configuring HP-41 Memory on page 15 before you proceed with the following keystrokes.

Keystrokes	Display	
[CLP] [ALPHA] RAIN [ALPHA]	CLP RAIN_ 0.0000	Clears RAIN program from memory to allow enough space for next program.
[SIZE] 013	0.0000	Reconfigures memory to 13 data registers (R_{00} through R_{12}).

Suppose you wanted a bar chart showing your monthly household electrical consumption in kilowatt-hours (kwh) for 1981. You could write and load into program memory an independent program that plots months on the x-axis and kilowatt-hours on the y-axis. To illustrate, enter the KWH program from either the bar code on page 196 or the annotated listing on page 161. Then execute KWH and key in the electrical consumption in kilowatt-hours for each month. When you press the keys shown at the bottom of this page, the plotter generates the following chart.



Keystrokes	Display	
XEQ ALPHA KWH ALPHA	XEQ KWH_ JAN	Executes KWH and prompts you to key in January's electrical consumption (in kwh).
3346 R/S	FEB	Enters electricity consumption for each displayed month, then prompts for the next month's consumption.
3278 R/S	MAR	
2625 R/S	APR	
1973 R/S	MAY	
1616 R/S	JUN	
1330 R/S	JUL	
1158 R/S	AUG	
986 R/S	SEP	
1105 R/S	OCT	
1350 R/S	NOV	
2043 R/S	DEC	
2694 R/S		Enters kwh for December, then plots the bar chart.
	0.0000	Plot completed.

This concludes the plotting examples in section 1. To summarize, the first example (page 13) uses the plotter module's built-in Utility Plotting Program. The remaining two examples (pages 15 and 16) use specialized programs of the type that you can create and load into the HP-41's program memory. Each of these examples automatically created the 26-register I/O buffer when needed. If you just completed the preceding example, this buffer remains in HP-41 memory.

Returning I/O Buffer Registers to Available Memory

When you complete a plotting session, you may want to clear the I/O buffer from memory so that the 26 registers used by the buffer will be available for other HP-41 operations. (The two procedures you can use to do so are described under I/O Buffer Permanence on page 11.) If the buffer remains in your HP-41 from the preceding example (or from any other plotting operation), the following keystrokes clear the buffer.

Keystrokes	Display	
[GTO] [.] [.]	0.0000	Packs program memory.
[PRGM]	00 REG *nn*	Switches HP-41 to Program mode and displays number of unused registers (***nn***).
[PRGM]	0.0000	Removes HP-41 from Program mode.
[PCLBUF] [PRGM]	00 REG *nn*	Clears I/O buffer from memory, then returns HP-41 to Program mode. Number of unused registers (*nn*) has increased by 26 because I/O buffer has been cleared.
[PRGM]	*0.0000*	Removes HP-41 from Program memory.

Where To Read Next?

The example on page 13 introduces you to the plotter module's built-in Utility Plotting Program. If you wish to learn the details of how to use this program, turn to section 2.

The RAIN and KWH examples demonstrate the kinds of results that can be achieved with user-designed programs. If you wish to learn about the individual plotter module functions that are provided for use in such programs, turn to part II, which begins on page 63.

Section 2

The Utility Plotting Program

Contents

Introduction 19
Program Overview 19
User Keyboard, Key Assignments, and Keyboard Overlay 21
The NEWPLOT Routine 22
The Plotting Data Base 22
How to Use NEWPLOT 22
The NAME and XINC Parameters 24
The XMIN, XMAX, YMIN, and YMAX Parameters 24
Automatic Transfer to REPLOT 24
The REPLOT Routine 25
When to Use REPLOT 25
How to Use REPLOT 25
The XAXAT and YAXAT Parameters 27
Parameter Elements 28
The ANNOT Parameter 28
The PLTPRM Parameter 29
Plotting Examples 31
The PLINIT Routine 38
When to Execute PLINIT Manually 38
Controlling the Physical Size of the Plotting Area 38
The PLTUXY Routine 39
General Plotting Options 39
Determining Parameter Elements 39
Plotting a Function 40
Prompting for Coordinates: The X? and Y? Subroutines 41
Plotting From Buffers 48
The Autoscale Option 55
Filling a Buffer 59
The PLANOT Routine 61
When to Use PLANOT 61
PLANOT Operation 61
Increasing Your Plotting Skills 61

Introduction

The HP 82184A Plotter Module's Utility Plotting Program generates complete plots—that is, framed, labeled plots of functions or data. This program enables you to use the plotter module in many applications *without* having to first learn how to use the plotter module's individual functions. If you are generally unfamiliar with plotter operation or want to learn how to quickly generate some of the more common types of plots, reading through this section may be the best way for you to proceed. When you are ready to learn how to use the plotter module's individual functions in your own programs or in step-by-step plotter control operations from the HP-41 keyboard, turn to part II, which begins on page 63.

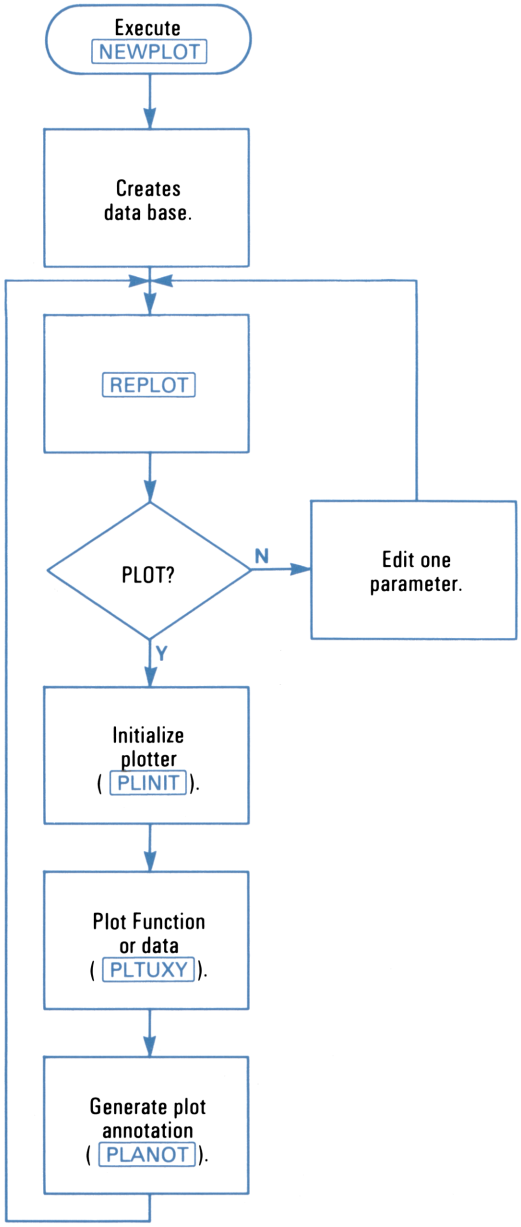
Program Overview

The Utility Plotting Program in your plotter module contains five major routines:

1. NEWPLOT (*new plot*) enables you to prepare for plotting by prompting you for several parameters, automatically assigning several other default parameters, and transferring execution to the REPLOT routine. NEWPLOT uses data registers R_{00} through R_{11} as a *plotting data base* that contains the plotting parameters.

2. REPLOT (*review/plot*) prompts you with **PLOT?**, to perform either of the following operations:
   - Automatically generate a complete plot by sequentially executing the PLINIT, PLTUXY, and PLANOT routines, then return to the **PLOT?** prompt. (This choice is demonstrated in the example of plotting a function on page 13.)
   - Review and, if desired, edit the contents of any register in the plotting data base (R_{00} through R_{11})—or any other data storage register—then return to the **PLOT?** prompt.
3. PLINIT (*plotter initialize*) initializes your plotter according to the parameters entered in the plotting data base during execution of NEWPLOT and/or REPLOT.
4. PLTUXY (*plot user x, y*) plots a function or data specified by parameters you have entered in the plotting data base.
5. PLANOT (*plot annotation*) draws a frame around the plotting area, then—guided by parameters in the plotting data base—labels the axes, and prints a plot title.

As shown in the following flowchart, executing NEWPLOT automatically executes the other routines, in order, to generate a complete plot.



To best understand how to use the Utility Plotting Program you should learn how the routines are used to set up and generate a plot, and learn how to select plotting parameters that give you the results you want. This section can help you make a good start. However, because the Utility Plotting Program offers numerous options for plot generation, the best way for you to truly master the program is to work through this section, then use it as a reference while you experiment with various combinations of plotting parameters. This procedure can help you to develop an insight into what is needed for any set of plotting requirements and to understand how the parameters interact to meet those requirements.

The preceding flowchart and the example on page 13 introduce you to the use of the five plotting routines in an automatic series. However, you can also execute any of these routines individually to perform only certain parts of the overall plotting procedure. This feature is useful, for example, when you want to plot two or more functions on a single page. In such cases it is often necessary to initialize the plotter and plot the annotation only once. The information following the next heading describes how to quickly execute the individual routines in order to perform such operations. The choice as to when to use any single plotting routine depends on your plotting applications. You will be better able to make such choices after you read through the detailed descriptions of the plotting parameters and routines provided later in this section. Appendix C, Program Documentation, includes an individual flowchart for each of these routines.

User Keyboard, Key Assignments, and Keyboard Overlay

Executing [NEWPLOT] activates the User keyboard.* Whenever the HP-41 is set to any program and the User keyboard is active, if you press a top-row key to which you have not already assigned a program label or HP-41 function, the HP-41 begins searching in the current program for the (default) local label corresponding to that key. (Local labels are described in your HP-41 owner's manual.) You can access [NEWPLOT], [REPLOT], [PLINIT], [PLTUXY], and [PLANOT] using local labels, as shown in the illustration below. For this reason, when the User keyboard is active, you can use the top row keys to execute each of these routines so long as the HP-41 is currently set to any program line in the Utility Plotting Program. (Since all five of the plotting routines are within the same program, setting the HP-41 to any of them automatically includes all five of them in the "current program."†)

Utility Plotting Program User Keyboard

Plotting Routine:	[NEWPLOT]	[REPLOT]	[PLINIT]	[PLTUXY]	[PLANOT]
Corresponding Local Label:	([LBL] A)	([LBL] B)	([LBL] C)	([LBL] D)	([LBL] E)
Corresponding Top Row Key:	[Σ+]	[1/x]	[√x]	[LOG]	[LN]

The preceding key assignments, plus ten other assignments designed for use with the **PLOT?** prompt (in [REPLOT]) for editing data base parameters, are printed on the keyboard overlay shipped with your plotter module.

*Line 03 of the [NEWPLOT] program sets flag 27, which is the HP-41's User keyboard flag. (Refer to the listing of [NEWPLOT] on page 162 and to the section describing flags in your HP-41 owner's manual.)

†In the HP-41, all program lines between two consecutive **END** instructions constitute a "program." Thus, a program may contain several smaller, functionally separate "routines." The five routines that form the Utility Plotting Program are within the same program in your plotter module. When you set the HP-41 to any program line, the "current program" consists of all program lines between the preceeding and following **END** instructions.

The NEWPLOT and REPLOT routines introduced on page 19 enable you to set and edit plotting parameters. The PLINIT, PLTUXY, and PLANOT routines use the plotting parameters to generate a plot. Let's examine NEWPLOT, REPLOT, and the parameters they control; then, after working through a pair of examples illustrating these topics, we'll examine the remaining three routines in the Utility Plotting Program.

The remainder of this section is divided into five units that describe the Utility Plotting Program's five plotting routines. The various parameters are described with the routines by which they are entered (NEWPLOT or REPLOT). The use of these parameters is described with the routines which they control. Thus, to learn, for example, the different kinds of XINC parameters you can use, turn to the NEWPLOT routine, which is the next topic in this section. To learn how XINC controls the plotting of points, turn to the PLTUXY routine, which describes how PLTUXY acquires coordinates and plots points.

The NEWPLOT Routine

NEWPLOT is designed for use whenever you want to establish a new set of parameters in the plotting data base. It requires that you enter at least one parameter and enables you to enter up to six parameters.

The Plotting Data Base

NEWPLOT uses R_{00} through R_{11} in your HP-41 as a *plotting data base*. Each time you execute NEWPLOT it replaces any values currently in the data base with up to 12 default plotting parameters, depending upon how many parameters you enter in response to NEWPLOT prompts.

> **Note:** The plotting data base is not protected from access by the HP-41 STO function. Thus, you should be careful to not accidentally alter the contents of the data base if you temporarily halt a plotting session to perform other HP-41 operations involving STO.

How to use NEWPLOT

NEWPLOT operation proceeds as described in the following four steps:

1. Activates the User keyboard.
2. Prompts you to specify the limits of the axes and to name the source of the x- and y-coordinates of each point to be plotted. Possible source names include:
   - An Alpha string that matches the label of a subroutine (where the subroutine generates an x- or y-coordinate).
   - A number specifying a block of HP-41 data registers (termed a *plotting buffer*) containing the coordinates for a series of points you wish to plot. The number is termed a *buffer pointer*. (Plotting buffers are described in detail under Plotting From Buffers on page 48.)
   - A subroutine that prompts you to key in a y-coordinate.
3. Automatically sets an additional six plotting parameters to default values.
4. Transfers execution to the REPLOT program, where you can generate a complete plot or inspect the contents of any data registers, including those in the plotting data base.

The example on page 13 uses NEWPLOT to automatically generate a complete plot. The following table illustrates each NEWPLOT prompt and the types of parameters you can enter in response to these prompts. The table also illustrates the **PLOT?** prompt, which is displayed after execution transfers from NEWPLOT to REPLOT. Following the table are detailed descriptions of the parameters you can enter in response to NEWPLOT prompts.

[NEWPLOT] Prompts and Options for Response

Step	Instructions	Keystrokes	Resulting Display
1	Execute the [NEWPLOT] routine. **Note:** When [NEWPLOT] prompts you with **NAME=?**, your HP-41 is automatically set to the Alpha keyboard.	[NEWPLOT]	**NAME=?**
2	Enter an Alpha string corresponding to the label of a subroutine that either returns *y* for a given *x* or prompts you to key in *y*; *or* enter a buffer pointer containing the beginning (***bbb***) and ending (***eee***) register numbers and buffer type (***t***). (For a description of buffer pointers, refer to How to Access a Buffer, page 49.)	*label* [R/S] *or* [ALPHA] ***bbb.eeet*** [R/S]	**XMIN=-1.000?**
3	Specify −1 for the *x*-axis minimum; *or* enter the *x*-axis minimum.	[R/S] *or* *xmin* [R/S]	**XMAX=1.000?**
4	Specify 1 for the *x*-axis maximum; *or* enter the *x*-axis maximum.	[R/S] *or* *xmax* [R/S]	**XINC=-11.000?**
5	Specify 11 equal intervals between *x*-coordinates (that is, 12 equally-spaced points); *or* enter the desired number of equal *x*-intervals (−*n*) *or* enter the interval (*int*) you want between equally-spaced *x*-coordinates. *or* enter an Alpha string corresponding to the label of a subroutine that prompts you for *x* (or computes *x*).	[R/S] *or* *n* [CHS] [R/S] *or* *int* [R/S] *or* [ALPHA] *label* [R/S]	**YMIN=-1.000?**
6	Specify −1 for the *y*-minimum; *or* key in the *y*-minimum.	[R/S] *or* *ymin* [R/S]	**YMAX=1.000?**
7	Specify 1 for the *y*-maximum; *or* key in the *y*-maximum.	[R/S] *or* *ymax* [R/S]	**PLOT?**
8	To automatically generate a complete plot, press [R/S].*	[R/S]	**PLOT?**

* After you execute step 7, [NEWPLOT] automatically sets six additional default parameters to default values and then transfers execution to the [REPLOT] routine, which is indicated by the **PLOT?** prompt.

The NAME and XINC Parameters

The NAME and XINC parameters are used by PLTUXY to obtain the x- and y-coordinates for the points in a plot. XINC determines the x-coordinates;* NAME determines the y-coordinates. (PLTUXY iterates once for each point in a plot.) Thus, your choice of parameters to use for NAME and XINC depends upon the source of the x- and y-coordinates of the points you want to plot.

The NAME Parameter. PLTUXY uses an Alpha-string NAME parameter to identify and execute a subroutine that determines the y-coordinate corresponding to each x-coordinate used in a plot. (That is, for each x-coordinate generated using XINC, NAME provides a corresponding y-coordinate.) As indicated in step 2 of the preceding table, the NAME parameter can be an Alpha string that matches the label of a subroutine that either calculates y or simply prompts you to key in y. (There is a subroutine built into your plotter module for this purpose. For further information, refer to Prompting For Coordinates: The X? and Y? Subroutines, page 41.) Thus, when you want the HP-41 to *calculate* y, write and enter in program memory a subroutine that, when given an x-coordinate in the x-register, calculates the corresponding y-coordinate and leaves it in the X-register.

A buffer pointer entered in response to the **NAME?** prompt is used to determine either y-coordinates only or both the x- and y-coordinates.

The XINC Parameter. Whenever NAME is used only to specify the source of y-coordinates in a plot, XINC specifies the x-coordinates. When you use a *numeric* XINC parameter—that is, a value that specifies either the equal interval between x-coordinates (int) or the number of equal intervals ($-n$)—the XMIN and XMAX parameters become the leftmost and rightmost x-coordinates in the plot. As indicated in step 5 of the preceding table, you can also use for XINC an Alpha string corresponding to a subroutine label. The subroutine can be designed either to prompt you to key in an x-parameter† or to calculate an x-parameter. When you execute NEWPLOT, if you do not specify an XINC parameter, XINC is set to the −11.000 default value (that is, $-n$, or 11 equal intervals) indicated at step 5 in the preceding table.

> **Note:** When PLTUXY is executed, if XINC was specified as $-n$—the number of x-axis intervals—PLTUXY always changes this value to int—the equal interval between x-coordinates.‡ Also, when the NAME parameter is a buffer pointer that determines *both* the x- and y-coordinates, the XINC parameter is ignored.

The XMIN, XMAX, YMIN, and YMAX Parameters

These define the limits of the axes and control plotting scale. When you execute NEWPLOT, they are set to the default values indicated in the preceding table unless you key in your own parameters.

Automatic Transfer to REPLOT

The NAME, XINC, XMIN, XMAX, YMIN, and YMAX parameters control the basic configuration of all plots. The remaining six parameters (to be described later in this section), which NEWPLOT automatically sets to default values, either control optional plotting features or provide internal control or storage for various operations performed by PLTUXY. At the completion of NEWPLOT, program execution automatically transfers to the REPLOT program and prompts you with **PLOT?**. If you need to edit any of the parameters that were either specified by you during execution of NEWPLOT or were automatically set by NEWPLOT to default values, you can do so using REPLOT.

---

*Unless both x and y are obtained from a buffer identified by the **NAME?** parameter.

†Refer to the first footnote on page 41.

‡This conversion consists of replacing $-n$ with a value representing the quotient of the expression $|(\text{XMAX} - \text{XMIN}) / (-n)|$.

The REPLOT Routine

REPLOT provides a convenient method for you to review or edit the contents of the plotting data base in R_{00} through R_{11} (as well as the contents of any other data storage register and to automatically generate a complete plot. The editing feature is especially useful when you want to edit any of the default parameters you are not prompted for during NEWPLOT execution.* Four of these parameters are used to control plotting options that are described later in this section. (The remaining two parameters are automatically maintained for internal use by the Utility Plotting Program *and should not normally be changed in any way by users.*)

When to Use REPLOT

Whenever you want to edit an existing plotting data base or begin automatic generation of a complete plot, if the **PLOT?** prompt is not already displayed, execute REPLOT to display this prompt.

How to Use REPLOT

The PLOT? Prompt. Whenever REPLOT is executed, the HP-41 displays **PLOT?** At this point you can do any one of the following:

- Automatically generate a complete plot. To do so, press R/S. (When plotting is completed, the HP-41 returns to REPLOT and prompts you again with **PLOT?**.)
- Examine and, if you wish, edit the contents of any register in the plotting data base (R_{00} through R_{11}) or any other data storage register, then return to the **PLOT?** prompt. This operation is described under the Register Editing Procedure, below.
- Switch from automatic plotting control to individual plotting control. (That is, instead of pressing R/S, which causes REPLOT to execute PLINIT, PLTUXY, and PLANOT in an automatic sequence, you can manually execute one or more of these routines from the keyboard.) Under manual control of PLINIT, PLTUXY, and PLANOT, the HP-41 does not return to the **PLOT?** prompt in REPLOT unless the User keyboard is active and you are executing these routines by using the keys to which they are assigned. (If the HP-41 halts without displaying the **PLOT?** prompt, you can return to this prompt by executing REPLOT.)
- Execute a program that is stored in the HP-41's program memory and automatically return to the **PLOT?** prompt. To do so, press ALPHA to activate the Alpha keyboard, key in the program's global Alpha label, and press R/S.

The flowchart on page 20 illustrates the first two of the preceding options. The third option involves selective execution of PLINIT, PLTUXY, and PLANOT. (If you wish to selectively edit PLINIT, PLTUXY, and/or PLANOT, you should first refer to the description of the PLINIT routine on page 38.)

The Register Editing Procedure. To use this feature you must know which register in the plotting data base contains the parameter you want to edit. The following chart lists data base registers with their corresponding parameter names and REPLOT prompts. Several of these parameters are user-accessible when NEWPLOT is executed and are described in the table on page 23 and in the paragraphs following that table. The descriptions of parameters that are user-accessible only when REPLOT is executed begin following the tables on the next two pages.

*Refer to item 3 on page 22, and to the lower portion of the table on page 26.

The Data Base

Access	Storage	Parameter	REPLOT Prompt	Refer to Page
NEWPLOT or REPLOT	R_{08}	Alpha String or Buffer Pointer	NAME=*label* or *n*?	24
	R_{00}	*X*-Minimum	XMIN=*n*?	24
	R_{01}	*X*-Maximum	XMAX=*n*?	24
	R_{05}	*X*-Increment	XINC=*n*?	24
	R_{04}	*Y*-Minimum	YMIN=*n*?	24
	R_{07}	*Y*-Maximum	YMAX=*n*?	24
REPLOT Only	R_{02}	Plot Parameter	PLTPRM=*n* or *label*?	29
	R_{03}	Annotation Control	ANNOT=*n*?	28
	R_{06}	*X*-Axis Intercept	XAXAT=*n*?	24
	R_{09}	*Y*-Axis Intercept	YAXAT=*n*?	24
	R_{10}	Current *X*	R10=*n*?	—
	R_{11}	Point Counter	R11=*n*?	—

A REPLOT prompt for a data storage register numbered higher than R_{11} is similar to those illustrated for R_{10} and R_{11}.

The plotter module keyboard overlay quickly shows you which digit key to press (while **PLOT?** is displayed) to access the plotting data base parameters (in R_{00} through R_{09}). (The parameters in R_{10} and R_{11} are variables that are automatically maintained by NEWPLOT and PLTUXY and are usually not of interest to users.)

When the HP-41 displays **PLOT?**, and you key in a register number and press R/S:

- If the specified register contains a plotting parameter (from R_{00} through R_{09} in the plotting data base), the HP-41 displays the parameter name and current value.
- If the specified register is numbered higher than R_{09}, the HP-41 displays the register number and the data contained in that register.

If, after the HP-41 displays the contents of a register, you press R/S again—*without pressing any other key*—the HP-41 returns to the **PLOT?** prompt without changing the contents of the register you specified. (This feature allows you to inspect the data in a register without changing that data.) But if you instead key in numeric or Alpha data, then press R/S, the data you keyed in replaces the current contents of that register. The following chart illustrates the register editing operation.

Register Editing Procedure

Step	Instructions	Keystrokes	Resulting Display
1	Execute REPLOT automatically or from the keyboard.	-As Appropriate-	PLOT?
2	Select the register (*R*) containing the plotting parameter or other value you wish to review or edit, then do one of the following:	*R* R/S *then*	Parameter=*n*? or *label*? or R*nn*=*n*?

Step	Instructions	Keystrokes	Resulting Display
2 (continued)	• Retain the displayed value in the indicated register; *or* • Store a new number in the indicated register; *or* • Store a new Alpha string (*s*)—up to six characters—in the indicated register. To select another register, repeat step 2. Otherwise, go to step 3.	R/S *or* *n* R/S *or* ALPHA *s* R/S	**PLOT?**
3	Automatically generate a complete plot (execution returns to REPLOT after plot completed); *or* Execute PLINIT, PLTUXY, or PLANOT. (If you execute any of these routines by pressing XEQ ALPHA *name* ALPHA, the HP-41 does not automatically return to the **PLOT?** prompt. Instead, when execution terminates, the HP-41 displays whatever number was left in the X-register by the routine you executed.)	R/S *or* PLINIT *or* PLTUXY *or* PLANOT	**PLOT?** **PLOT?** or ***variable***

Note: During register editing, if you accidentally key in an Alpha string for any parameter when a numeric parameter was required (or vice versa), you must terminate the entering of the erroneous data by pressing R/S. To correct the error, use the register editing procedure to reedit the parameter. *Simply clearing the erroneous data with the ← key, then pressing ALPHA to clear (or set) the Alpha keyboard does not correct the error.* This is because REPLOT uses the numeric and Alpha entry flags (flags 22 and 23—refer to the section describing flags in your HP-41 owner's manual) to determine whether your entry was numeric or Alpha. When one of these two flags is set by a keyboard entry, it cannot be cleared by clearing the display with the ← key.

As you can see from the Data Base chart on page 26, all parameters in the plotting data base can be accessed by REPLOT. To review or edit any of these parameters, use the procedure illustrated in the preceding register editing chart. The NAME, XMIN, XMAX, XINC, YMIN, and YMAX parameters are initially accessed by NEWPLOT and are described under The NEWPLOT Routine. The remaining parameters are accessed only by REPLOT and are described in the following text.

The XAXAT and YAXAT Parameters

The x-intercept parameter (XAXAT) is used by PLANOT to specify where the x-axis intercepts the y-axis. Executing NEWPLOT sets XAXAT to the same value that you specify for the y-minimum.

The y-intercept parameter (YAXAT) is used by PLANOT to specify where the y-axis intercepts the x-axis. Executing NEWPLOT sets YAXAT to the same value that you specify for the x-minimum.

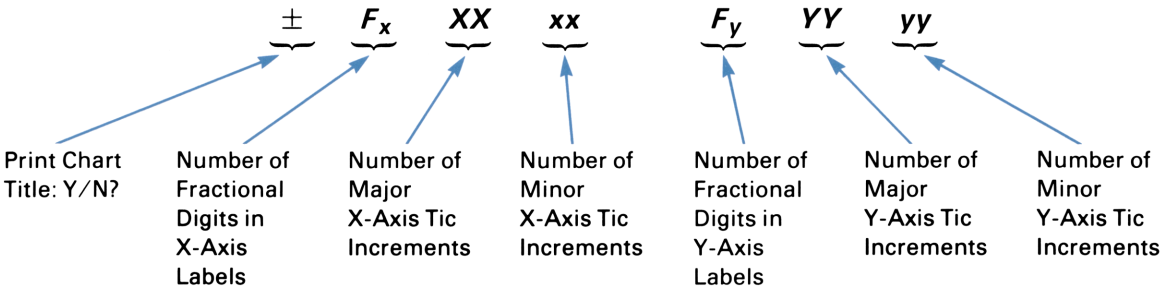
The plot generated by the introductory example on page 13 shows the results of the default x- and y-intercept parameters. The example on pages 34 through 36 uses REPLOT to edit these parameters.

Parameter Elements

Some plotting data base parameters contain only one data item or *element*. For example, the XMIN parameter contains one element—the minimum or leftmost limit of the *x*-axis. Other data base parameters contain two or more elements. The following parameter, ANNOT, contains seven elements.

The ANNOT Parameter

The annotation parameter (ANNOT) contains seven elements used by PLANOT to control various aspects of plot labeling. These elements are interpreted in the following format:



The Default ANNOT Parameter. NEWPLOT sets the annotation parameter to (0)1000.01000, which results in the following plot annotation when you execute PLANOT:

- If the NAME parameter (R_{08}) is Alpha data, it is printed as the plot title.
- Ten major tic increments with labels, are plotted on each axis.
- Tic labels are formatted automatically in FIX or SCI notation.
- No minor tics are plotted on either axis.

Editing the ANNOT Parameter. To change one or more elements in the annotation parameter, you must enter a complete, new parameter in R_{03}. To do so, determine all of the elements you need in the parameter to generate the desired annotation, then use the register editing procedure to access R_{03} and to enter the new parameter. The following chart describes each element of the annotation parameter.

ANNOT Parameter Elements

Parameter	Purpose
+	If the NAME parameter (R_{08}) contains an Alpha string, then NAME causes PLANOT to print that string as the chart title. If the NAME parameter is numeric data, no title is printed.
−	Suppresses chart title that would otherwise be printed if NAME contains an Alpha string.
F_x	Specifies FIX display setting for digits in *x*-axis labels. If $F_x = 0$, PLANOT computes an appropriate FIX or SCI setting.
XX	Specifies number of major *x*-axis tic increments.
xx	Specifies number of minor tic increments between major *x*-axis tics. (Minor tics are not labeled.) If XX = 0, xx specifies the number of tic increments on the *x*-axis.
F_y	Controls number of digits in *y*-axis labels in the same way as F_x does for the *x*-axis labels.
YY	Specifies number of major *y*-axis tic increments.
yy	Specifies *y*-axis minor tic increments in the same way that *xx* specifies *x*-axis tic increments.

To determine the major x-axis tic element to use for a given application, divide the x-axis length by the (equal) interval you want between the major (labeled) tics. Use the same procedure for the major y-axis tics and the minor tics used in both axes.

If you specify zero for either the F_x or the F_y parameter, the PLANOT program determines the display setting as follows:

- Where the *complete* tic label (all digits) can be printed using a FIX display setting of 0 through 5, the appropriate FIX display setting is automatically determined.
- Where i is the increment between major tics:

  If

  $$i \geqslant 10{,}000$$

  or

  $$i < 0.0001$$

  the major tic labels on the applicable axis are printed in the SCI 4 format.

  If

  $$0.0001 \leqslant i < 10{,}000$$

  and there are more than five fractional digits in the increment, the labels are likewise printed in the SCI 4 format.

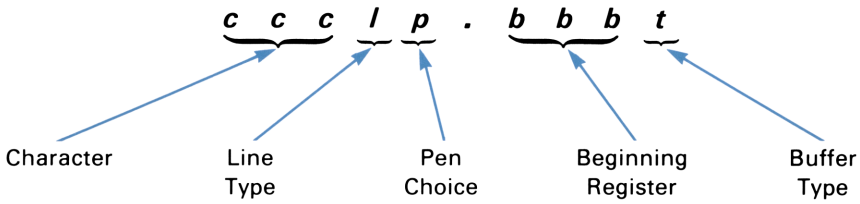
The plot illustration in the introductory example on page 13 demonstrates the result of the default annotation parameter. The entry of non-default annotation parameters are shown on pages 35, 44, and 46.

The PLTPRM Parameter

As indicated in the introduction to this section, the PLTUXY program is the part of the plotting package that actually plots the function or series of points you specify. PLTUXY uses the PLTPRM parameter to control one or more of the following options:

- Line type.
- Choice of pen.
- Choice of character to draw at each point.
- Buffer filling.
- Automatic scaling (*autoscale*).
- Executing a user-defined program at each plotting point.

In its *numeric* form, PLTPRM contains five elements. The format for these elements is:



In its *Alpha* form, the PLTPRM parameter contains only one element, an Alpha string representing any global* label in your HP-41.

*A *global* label is any Alpha label except local labels A through J and a through e.

Default PLTPRM Parameters. Executing NEWPLOT defaults PLTPRM to line type 1 (a solid line) and pen number 1. The remaining elements are set to null values.

The following paragraphs describe the purpose and format of each PLTPRM parameter. However, some aspects of plotting under optional PLTPRM controls require an understanding of PLTUXY operating details. For this reason, the integration of PLTPRM options into plotting operations is provided later, under The PLTUXY Routine, page 39.

The Character, Line Type, Pen, and Buffer-Filling Options. When you use numeric PLTPRM parameters, the integer portion of PLTPRM specifies the character (if any) to plot at each point, the line type, and the pen choice.

ccc is the ASCII decimal code* of the single character that can be printed at each point in a plot. Executing NEWPLOT defaults ***ccc*** to zero. The character code table on page 212 lists the characters you can select and their equivalent ***ccc*** code.

l specifies the line type used to connect points in your plot. (Refer to the description of the LTYPE (*line type*) function on page 89.)

p specifies the number of the pen stall containing the pen with which you wish to plot. (For pen stall information, refer to your plotter owner's manual.)

bbb specifies the beginning data storage register in a block of registers you wish to use for storing the coordinates of a series of points (a buffer-filling operation, which is described under Filling a Buffer, page 59).

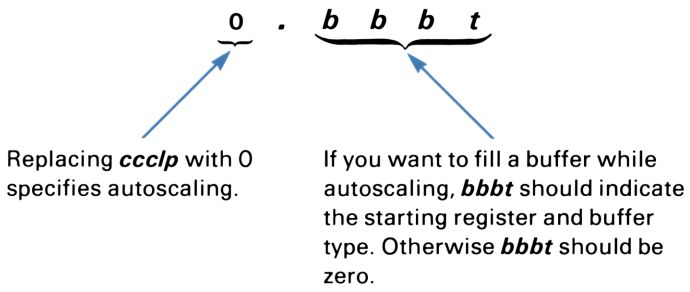
> **Note:** The beginning register must be numbered 12 or higher. Using a lower-numbered register destroys information in the plotting data base.

t specifies the type of plotting buffer you want. Type 0 and type 2 buffers contain both the x- and y-coordinates of each point to be plotted. (A type 1 buffer contains only the y-coordinates of the points you want to plot.) For the purposes of this element, types 0 and 2 are identical.

When ***bbbt*** = 0, no buffer filling takes place.

The Autoscale Parameter. Autoscaling is designed for use either when you cannot estimate the maximum and minimum values to be derived by XINC and/or NAME accurately enough to specify the axes limits (XMIN, XMAX, YMIN, and YMAX) or when you want to fill a buffer without simultaneously plotting it.

To specify autoscaling, use register editing to set the integer portion of PLTPRM (R_{02}) to 0. That is:



When you execute an autoscaling operation, XINC and/or NAME generate x- and y-coordinates in the same way as for a plotting operation. (That is, when you specify autoscaling, PLTUXY generates the x- and y-coordinates, but does not plot the corresponding points.) The limits of both axes (XMIN, XMAX, YMIN,

*Characters whose decimal codes are 32 through 127 are standard printable characters as defined by the American Standard Code for Information Interchange.

and YMAX) are reset to match the respective minimum and maximum coordinates that PLTUXY generates. Also, PLTPRM is changed from 0 to (000)11.000 (the null character code, line type 1, pen 1, and no buffer-filling), which is the same as the default PLTPRM that is set when you execute NEWPLOT. Because autoscaling automatically resets PLTPRM to its default value, you can autoscale a set of parameters, then immediately generate a plot of those parameters without having to manually change any plotting parameters. Autoscaling and its relationship to PLTUXY is described in further detail under The Autoscale Option on page 55.

Executing a Subroutine at Each Plotting Point. When you replace the numeric ***cccIp.bbbt*** PLTPRM parameter with an Alpha string representing a global label, the points generated by XINC and NAME when you execute PLTUXY are not plotted. Instead, the subroutine named by the Alpha label in PLTPRM is executed. This option enables you to plot special shapes or diagrams at the desired points in a plot.

When you write a subroutine to be used by PLTPRM as described in the preceding paragraph, *the first instruction after the initial subroutine label should be the plotter module's* MOVE *function.* This is because PLTUXY executes the subroutine *after* placing the *x*- and *y*-coordinates of the next plotting point into the HP-41's X- and Y-registers.

In the example on page 34, this option is used to plot an octagon around each of several points in a plot.

Plotting Examples

The preceding discussion of PLTPRM parameters completes your introduction to NEWPLOT, REPLOT, and control of the Utility Plotting Program. The following examples illustrate the basics of how to use the program and parameters. Once you have worked through these examples you may want to begin using the Utility Plotting Program in your applications. However, if you wish to understand further details of how PLINIT, PLTUXY, and PLANOT use the parameters in the data base, or if you plan to use the autoscaling and buffer features, you should also read through the remaining material in this section.

> **Note:** If you inadvertantly enter the wrong data for any of the parameters in the following examples, complete the data entry as shown by the printed keystrokes, then correct the erroneous data using the register editing procedure introduced on page 25.

Keystrokes	Display	
PCLBUF		If the I/O buffer remains from an earlier plotting operation, restores to unused memory the registers used by the buffer. Otherwise displays **PL:PLS PINIT**.
SIZE 013		Reallocates memory to 13 data storage registers (R_{00} through R_{12}).
■ GTO · ·		Packs program memory.
PRGM	**00 REG *nn***	Displays available memory. If the SINE subroutine from the example on page 13 is still in program memory, the value represented by ***nn*** should be at least 26. Otherwise, it should be at least 28.
PRGM		Removes HP-41 from program mode.

Memory Requirements. Use the following keystrokes to set up the plotting data base (R_{00} through R_{11}) and to ensure that R_{12} is available (for use in the last of these examples).

Example of NEWPLOT Operation. Use NEWPLOT to enter a series of plotting parameters and generate a complete plot of a sine function (in Degrees mode). Then use REPLOT to increase the number of points in the plot by changing the XINC parameter, and replot the function.

If the SINE subroutine used in the example on page 13 is not in your HP-41's memory, use the following keystrokes to load it into memory.

Keystrokes	Display	
PRGM		
■ LBL ALPHA	01 LBL _	
SINE ALPHA	01 LBL"SINE	Enters subroutine label.
SIN	02 SIN	Calculates sine of x-coordinate (generated by XINC and placed in X by PLTUXY).
PRGM		Removes HP-41 from Program mode.

Use NEWPLOT to set up the plotting data base as follows:

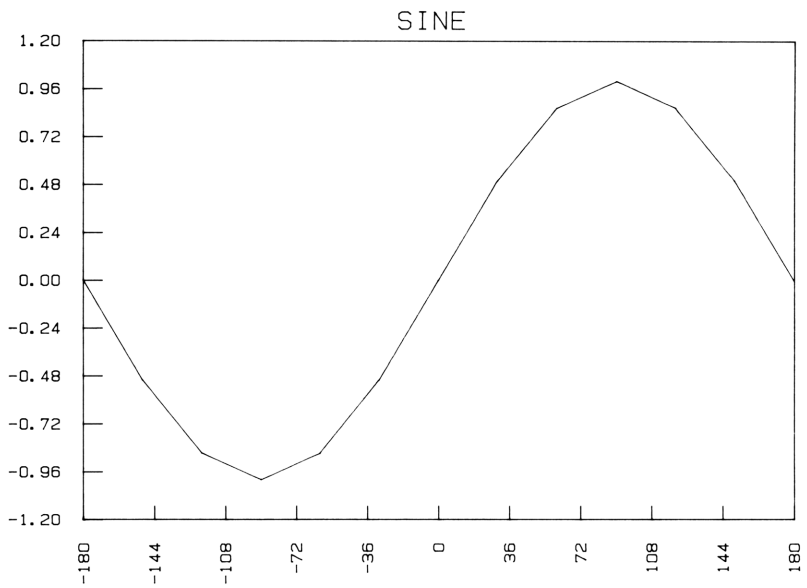
1. Set NAME to use the SINE program for determining your plot's y-coordinates.
2. Set the x-axis scale to a minimum of –180 and a maximum of 180.
3. Set the x-interval to 30. This produces the x-coordinates of the plot. The first x-coordinate is –180. The interval between succeeding x-coordinates will be 30.
4. Set the y-minimum to –1.2 and the y-maximum to 1.2.

Keystrokes	Display	
DEG		Ensures that the HP-41 is set to Degrees mode.
NEWPLOT	NAME= ?	Executes NEWPLOT program. Prompts you for parameter that names source of y-coordinate.
SINE R/S	XMIN=–1.000?	Enters Alpha string for NAME parameter. HP-41 prompts you for x-axis minimum.
180 CHS R/S	XMAX=1.000?	Enters –180 for XMIN. HP-41 prompts you for x-axis maximum.
180 R/S	XINC=–11.000?	Enters 180 for XMAX. HP-41 prompts you for x-increment.
30 R/S	YMIN=–1.000?	Enters XINC of 30. HP-41 prompts you for y-axis minimum.
1.2 CHS R/S	YMAX=1.000?	Enters YMIN of –1.2. HP-41 prompts you for y-axis maximum.
1.2 R/S	PLOT?	Enters 1.2 for YMAX. HP-41 prompts you to either generate a plot or edit the data base.

You are now ready to plot the sine function. Pressing R/S at this point causes the HP-41 to automatically generate a complete plot. (That is, to sequentially execute PLINIT, PLTUXY, and PLANOT.)

Keystrokes	Display	
R/S		Generates complete plot of sine function.
	PLOT?	Prompts you to either generate a plot or edit the data base.

The following illustration shows the resulting plot. (Leave your HP-41 turned on to preserve the PLOT? prompt for the next part of this example.)

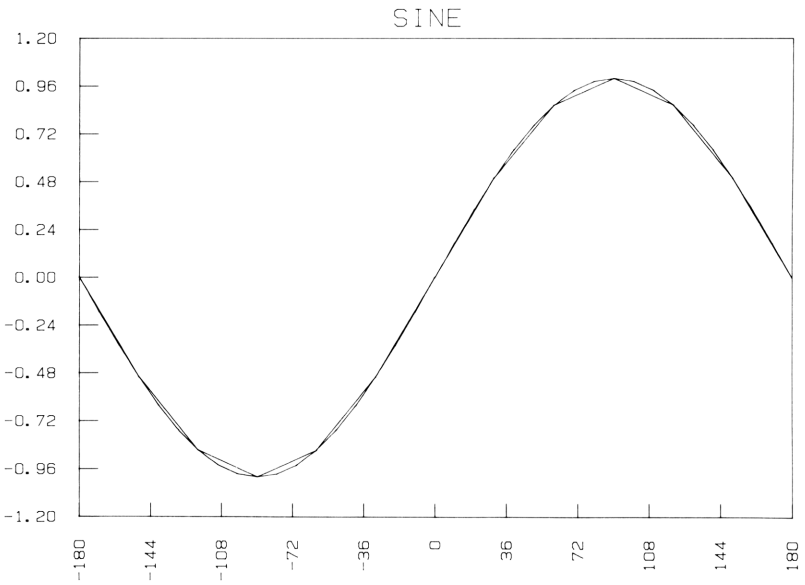


Now let's plot a more accurate representation of the sine curve by increasing the number of plotting points. To do so, decrease the interval between x-axis coordinates by using the register editing procedure to change the XINC parameter (R_{05}) from 30 to 10. (Each time XINC generates an x-coordinate, NAME generates a y-coordinate.) Since the plotter is already initialized and the annotation has already been plotted (by the automatic execution of [PLINIT] and [PLANOT]), it is unnecessary to reexecute the entire plotting procedure. Instead, just replot the function by executing [PLTUXY] alone. (If your HP-41's User keyboard is active (**USER** annunciator displayed) and you have not assigned any function or program label to the [LOG] key, pressing [LOG] executes [PLTUXY].)

Keystrokes	Display	
	PLOT?	Display remaining from preceding operation.
5 [R/S]	XINC=30.0000?	Displays the current XINC parameter.
10 [R/S]	PLOT?	Replaces the current XINC parameter with 10. HP-41 prompts you to either generate a plot or edit the data base.
[PLTUXY]		Executes [PLTUXY] to replot the sine function. ([PLINIT] and [PLANOT] are not executed.)
	0.000	Plot completed.*
	-or-	
	PLOT?	

*Using the top row key convention described under User Keyboard, Key Assignments, and Keyboard Overlay on page 21 results in the **PLOT?** Prompt. If you do not use the top row key convention, the HP-41 displays the last number placed in the X-register before termination of the routine instead of displaying **PLOT?**.

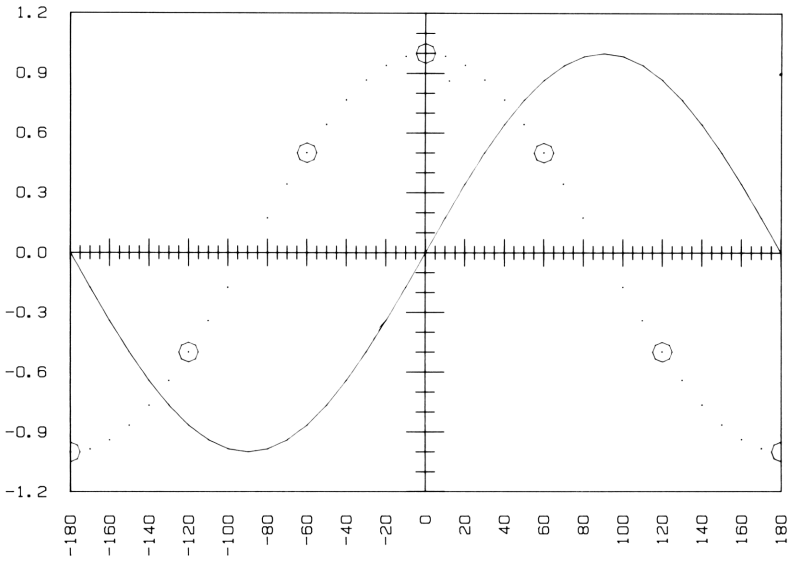
As shown in the following illustration, the sine function plot now includes a more accurately plotted curve.



Retain the plotting data base and the SINE subroutine in your HP-41's memory for use in the next example, and leave your plotter turned on to preserve the current plotter settings.

Example of Two Function Plots on the Same Page. Place a new sheet of paper in your plotter and enter in program memory a subroutine that calculates the cosine of a number. Then:

1. Adjust the plot annotation and move the positions of the axes by changing the ANNOT (R_{03}), XAXAT (R_{06}), and YAXAT (R_{09}) parameters.
2. Plot the annotation for a new chart.
3. Plot the sine function.
4. Change the line type and pen elements of the PLTPRM (R_{02}) parameter so that the plotter will use pen 2 and will plot a dotted line instead of a solid line. Change the NAME parameter (R_{08}) from SINE to COS.
5. Plot the cosine function.
6. To highlight plotting points, enter in program memory a subroutine that plots octagons around the plotting points.
7. Change the PLTPRM parameter so that it specifies the subroutine in step 6, then execute PLTUXY to plot the octagons.



(This example assumes that the data base and plotter settings created in the preceding example remain in your HP-41 and plotter, and that you have not assigned any functions or labels to any of the top row keys.)

Keystrokes	Display	
■ GTO · ·		Packs program memory.
PRGM	00 REG *nn*	Sets HP-41 to Program mode. If the value represented by ***nn*** is not at least 01, refer to Ensuring that Enough Unused Registers Are Available, page 11.
■ LBL ALPHA	01 LBL _	Program to calculate the cosine of x.
COS ALPHA	01 LBL"COS	
COS	02 COS	
PRGM		

Change the annotation parameter from the default value set by NEWPLOT to a new value that specifies 18 major x-axis tic intervals, with 4 minor tic intervals in each major interval; and 8 major y-axis tic intervals with 3 minor tic intervals in each major interval.

Keystrokes	Display	
REPLOT	PLOT?	HP-41 prompts you for next operation.
3 R/S	ANNOT=1000.01 =1000.01000?	Accesses plot annotation parameter. (Display scrolls.)
1804.00803 CHS	-1804.00803 _	Enters new ANNOT parameter.
R/S	PLOT?	HP-41 prompts you for next operation.

Now change the axes intercepts to zeroes.

Keystrokes	Display	
6 R/S	XAXAT=-1.200	Accesses x-axis intercept.
0 R/S	PLOT?	Enters 0 for x-axis intercept.
9 R/S	YAXAT=-180.00 XAT=-180.000?	Accesses y-axis intercept. (Display scrolls.)
0 R/S	PLOT?	Enters 0 for y-axis intercept.

You are now ready to generate the plot annotation. This operation uses the parameters that you edited in the preceding keystrokes.

Keystrokes	Display	
PLANOT		Generates the plot annotation.
	0.0000	Plot completed.*
	-or-	
	PLOT?	

For the next step, use PLTUXY to generate a plot of the sine function (as specified by the SINE Alpha string entered earlier for the NAME parameter). Then switch the line type and pen number by changing the PLTPRM parameter, change the Alpha string in the NAME parameter to represent the COS subroutine, and plot the cosine function.

Keystrokes	Display	
PLTUXY		Plots sine function.
	0.0000	Plot completed.*
	-or-	
	PLOT?	
REPLOT	PLOT?	Executes REPLOT. (This step is unnecessary if plot is already displayed as a result of the preceding instruction.)
2 R/S	PLTPRM=11.000 TPRM=11.0000?	Accesses PLTPRM parameter. Displayed parameter elements—***cccIp*** = 00011—specify line type 1 and pen 1. (Display scrolls.)
32 R/S	PLOT?	Enters new PLTPRM parameter that specifies line type 3 and pen 2.
8 R/S	NAME= SINE?	Accesses NAME parameter.
COS R/S	PLOT?	Enters new NAME parameter that specifies the COS subroutine.
PLTUXY		Plots cosine function.
	0.000	Plot completed.*
	-or-	
	PLOT?	

Now enter a subroutine labeled OCTA that plots octagons around plotting points, then edit the PLTPRM parameter so that it contains an Alpha string representing the label of this subroutine. Thus, when you reexecute PLTUXY, it executes the OCTA subroutine at each point instead of plotting a point.

Keystrokes	Display	
■ GTO · ·		Packs program memory.
PRGM	00 REG *nn*	Sets the HP-41 to Program mode. (If the number represented by ***nn*** is less than 06 refer to Ensuring that Enough Unused Registers Are Available, page 11.)

*Refer to the footnote on page 33.

Keystrokes	Display	
■ LBL ALPHA	01 LBL _	Enters in program memory a subroutine that plots an octagon at a point whose coordinates are given in the X- and Y-registers.
OCTA ALPHA	01 LBL"OCTA	
MOVE	02 MOVE	
1	03 1 _	
PEN	04 PEN	
LTYPE	05 LTYPE	
.40545	06 .40545 _	
STO 12	07 STO 12	
■ LBL 00	08 LBL 00	
0	09 0 _	
ENTER↑	10 ENTER↑	
5 CHS	11 -5 _	
RPLOT	12 RPLOT	
RCL 12	13 RCL 12	
INT	14 INT	
PDIR	15 PDIR	
■ ISG 12	16 ISG 12	
■ GTO 00	17 GTO 00	
PRGM	0.0000	Switches HP-41 out of Program mode.
REPLOT	PLOT?	Prompts you for next plotting operation.
2 R/S	PLTPRM=32.000	Accesses the PLTPRM parameter.
	TPRM=32.0000?	
ALPHA OCTA R/S	PLOT?	Enters Alpha string representing OCTA subroutine label. HP-41 prompts you for next plotting operation.

Earlier you set the XINC parameter so that a point would be plotted every 10 units on the x-axis. If you now execute PLTUXY, because of the change you made to the PLTPRM parameter, octagons will be plotted at 10-unit intervals on the cosine curve, which is an unnecessarily small interval for the purposes of this example. To reduce the frequency of octagon plots, reduce the number of plotting points. To do so, increase the interval between x-coordinates by changing XINC from 10 to 60.

Keystrokes	Display	
5 R/S	XINC=10.000?	Accesses XINC parameter.
60 R/S	PLOT?	Edits XINC. HP-41 prompts you for next operation.

You have now edited the PLTPRM and XINC parameters to plot an octagon at intervals of 60 units on the x-axis. If you execute PLTUXY, it will use XINC and NAME to generate each point's x- and y-coordinates (just as it did in previous examples). However, now that the PLTPRM parameter specifies a subroutine label instead of a line type and pen number, PLTUXY executes that subroutine at each plotting point instead of drawing a line to the point.* (When an Alpha PLTPRM parameter is used, plotting by PLTUXY is done using pen 1 and line type 1.†)

*The main purpose of this example is to demonstrate parameter editing. If the use of these parameters is unclear to you at this point, you can learn more about how they affect plotting later, when you read the material under The PLTUXY Routine on page 39 and the PLTUXY flowchart on page 176.

†You can override this default by specifying another line type and pen number in the program named by PLTPRM. Refer to the PEN and LTYPE functions described under Other Pen Control Functions on page 88.

Keystrokes	Display	
PLTUXY		Plots octagons around plotting points.
	0.000	Plot completed.*
	-or-	
	PLOT?	

The PLINIT Routine

After NEWPLOT and REPLOT are executed, but before plotting can begin, the plotting area, scale, line type, label, and other parameters must be established in the plotter module's I/O buffer. The PLINIT routine automatically performs this operation for you. That is, PLINIT executes PINIT, then executes additional plotter module functions that use either the parameters included in the PLINIT program or the scale parameters (XMIN, XMAX, YMIN, and YMAX) in the plotting data base (R_{00} through R_{11}) created by NEWPLOT. (Execution of PLINIT leaves the plotter module's label origin setting at LORG 5. Refer to changing the Label's Location, page 94.)

As described under Program Overview on page 19, each time you use REPLOT to generate a complete plot (by pressing R/S while the **PLOT?** prompt is displayed), PLINIT is automatically executed (followed by PLTUXY and PLANOT). Thus, the only time that you need be concerned with PLINIT execution is when you want to generate a plot by executing PLTUXY and (if needed) PLANOT manually from the keyboard instead of automatically under REPLOT control.

When to Execute PLINIT Manually

Once the I/O buffer has been initialized by executing PLINIT, it remains unchanged until you reexecute PLINIT.† Because the only plotting data base parameters (R_{00} through R_{11}) used by PLINIT are the four scale parameters, it is necessary to manually execute PLINIT only when you are not automatically generating a complete plot. That is, you should execute PLINIT after you have used NEWPLOT to create or recreate the plotting data base, or after you have used REPLOT to change a scale parameter.

Controlling the Physical Size of the Plotting Area

It is not necessary to understand this topic in order to use PLINIT for general plotting. However, users needing to vary the size and position of plots they generate using the Utility Plotting Program should read the following information, as well as the referenced material in section II of this manual.

The actual size and position of the plotting area established by PLINIT depends upon the plotter you are using and the graphic limits to which your plotter is currently set.‡ Using the HP 7470A plotter with its default graphic limits, PLINIT establishes a plotting area of approximately 188 mm on the *x*-axis and 126 mm on the *y*-axis. To quickly set the default graphic limits, turn the plotter off, then on, before you begin using the Utility Plotting Program.

---

*Refer to the footnote on page 33.

†Unless you execute an individual plotter module function that alters any data held in the I/O buffer. Such functions are described in part II.

‡The *graphic limits* define the maximum limits of pen movement. PINIT, which is executed by the PLINIT routine, sets the graphic limits to those currently maintained by your plotter. All examples in this section assume the default graphic limits, which you can set by turning your plotter off, then on, before you execute any of the plotting applications programs. Changing the graphic limits changes the size and/or the location of subsequent plots. If you wish to experiment with various sized graphic limits, refer to Graphic Limits (page 64) and Specifying the Graphic Limits (page 69).

The PLTUXY Routine

When you automatically generate a complete plot, the REPLOT program executes PLINIT, PLTUXY, and PLANOT for you. However, when you want to generate a plot when you don't need to reinitialize the plotter module or draw the plot annotation, just execute PLTUXY from the keyboard.

The PLTUXY routine uses the XINC and/or NAME parameters to acquire *x*- and *y*-coordinates, and the PLTPRM parameter to determine what to do with the coordinates. Each iteration of PLTUXY generates one point. PLTUXY automatically performs as many iterations as are required to produce all points specified by XINC and/or NAME. (The PLTUXY flowchart on pages 176 through 178 illustrates this operation.) Before you execute PLTUXY, you should perform each of the following two operations at least once:

1. Use NEWPLOT and/or REPLOT to enter the parameters you want into the plotting data base.
2. Execute PLINIT. (It is not necessary to execute PLINIT more than once in any plotting session unless one or more of the parameters controlling scale (XMIN, XMAX, YMIN, or YMAX) are changed between executions of PLTUXY.)

General Plotting Options

The range of possible XINC, NAME, and PLTPRM parameters gives you several options for PLTUXY use. Generally, these options include:

- Plotting a function from *x*-coordinates determined by a value in the XINC parameter and *y*-coordinates determined by a subroutine specified in the NAME parameter. The examples on pages 31 through 38 demonstrate this option.
- Plotting points where you are prompted to key in one or both coordinates of each succeeding point while the plot is in progress.
- Plotting points where either the *y*-coordinate of each point or both coordinates of each point are taken from a buffer you loaded previously.
- Building a buffer with or without simultaneously plotting it.
- Where you are unsure of the scale to use, letting PLTUXY determine the scale (termed *autoscaling*).
- Highlighting points in a plot by plotting a special shape or design at each plotting point.

Determining Parameter Elements

To determine the elements to use for the XINC, NAME, and PLTPRM parameters:

1. Identify:
   - How you want PLTUXY to acquire the *x*-coordinates. (That is, do you want coordinates that occur at equal intervals, coordinates that you enter when prompted from the keyboard, or coordinates stored in a buffer?)
   - How you want PLTUXY to acquire the *y*-coordinates. (That is, do you want coordinates derived from a calculation that returns *y* when given *x*, or do you want the coordinates derived in one of the ways mentioned above for the *x*-coordinate?)
   - What you want done with the points defined by the *x*- and *y*-coordinates. (That is, do you want them plotted and/or stored in a buffer, or used for autoscaling?*)
2. Assign to XINC and NAME the parameter elements that address the sources of your coordinates, and to PLTPRM the elements that indicate the plotting operation you want.

---

*Generating coordinates and storing them in a buffer without simultaneously plotting them requires that you specify both the buffer and autoscaling operations. However, you can also choose to autoscale without simultaneously storing coordinates in a buffer.

The following two charts identify the types of XINC, NAME, and PLTPRM parameter elements you can select, and either the sources to which they refer or the plotting operations they initiate. The charts are intended to help you visualize how to control PLTUXY. For descriptions of the XINC, NAME, and PLTPRM parameters themselves, refer to The NAME and XINC Parameters on page 24 and The PLTPRM Parameter on page 29, or to the subject index that begins on page 223. While you are learning how to use the parameters in these charts, it may be helpful to refer to the PLTUXY flowchart on pages 176 through 178. The paths through PLTUXY as shown by this flowchart are controlled by the values you specify for the NAME and PLTPRM parameters. In this regard, it is helpful to divide the flowchart into two parts. The upper part illustrates how PLTUXY acquires the coordinates of each successive point. The lower part illustrates what PLTUXY does with each successive point.

Note: When PLTUXY is ready to perform the operation(s) specified by PLTPRM, the *x*-coordinate of the current point is in the X-register and the *y*-coordinate of the current point is in the Y-register.

Parameters That Acquire X- and Y-Coordinates

XINC Options	Number of equal *x*-coordinate intervals (*–n*). Interval between *x*-coordinates (*int*). Label of subroutine that prompts for (or calculates) *x*.
NAME Options	Label of subroutine that calculates *y* for a given *x*. Label of subroutine that prompts for *y*. Buffer pointer for buffer containing only *y*-coordinates (***iii.fff*1**). Buffer pointer for buffer containing both *x*- and *y*-coordinates (***iii.fff*0** or ***iii.fff*2**).*
* When a NAME parameter buffer pointer acquires both the *x*- and *y*-coordinates of each point, the XINC parameter—which at other times is used to acquire *x*-coordinates—is ignored.	

Parameters That Specify What to Do With the Coordinates Identified by XINC and/or NAME

Integer Part of Numeric PLTPRM Option	Plot the points. Use nonzero ***ccclp*** values to determine character number, line type, and pen number. (If you do not want a character plotted at each point, use 0 for ***ccc***.) Autoscale (***ccclp*** = 0).
Fractional Part of Numeric PLTPRM Option	Do not build a buffer (***bbbt*** = 0). Build a buffer (***bbbt*** ≠ 0).
Alpha PLTPRM Parameter Option	Plot a special shape at each plotting point by executing a subroutine at each point. To do so, use for the PLTPRM parameter the global Alpha label that names the subroutine.

Plotting a Function

To plot a function:

1. Write and store in program memory a subroutine that, given an *x*-value in the X-register, calculates and leaves in the X-register a corresponding *y*-value. The subroutine should have a global label.
2. Determine the XINC (R_{05}) parameter that will provide the desired *x*-coordinate for each point.

3. Use [NEWPLOT] or [REPLOT] to specify:
   - The program label described in 1, above, as the NAME parameter.
   - The XINC parameter and any other data base parameters that need to be entered or edited before you begin plotting.
4. Execute [PLTUXY] either automatically or manually. (If you execute [PLTUXY] manually, ensure that [PLINIT] has been executed once since the last execution of [NEWPLOT] or change in the XMIN, XMAX, YMIN, or YMAX parameters.)

When you execute [PLTUXY], it uses XINC to determine an x-value, places a copy of that value in the X-register, then executes the program specified by the label used as the NAME parameter. [PLTUXY] repeats this procedure for each successive point.

Prompting For Coordinates: The X? and Y? Subroutines

You can design a [PLTUXY] operation to prompt you to key in one or both coordinates of each point. The Utility Plotting Program contains two subroutines, labeled X? and Y?, that you can use for this purpose.

How to Use X? and Y?. If you want the HP-41 to prompt you to key in each successive x-coordinate used in a plot, use X? for the XINC parameter (R_{05}).* If you want the HP-41 to prompt you to key in each successive y-coordinate, use Y? for the NAME parameter.

How the X? and Y? Subroutines Operate. When you use X? as the XINC parameter, each time a new x-coordinate is required [PLTUXY] executes the X? subroutine, which prompts you to key in the desired x-coordinate. [PLTUXY] keeps track of the number of points that have already been plotted and includes in the X? subroutine's prompt a number in parentheses to indicate the sequence number of the current point. The first point in any series will be point 0. Thus, the first X? prompt will be **X⟨0⟩= ?**. The Y? subroutine operates in the same way.

Terminating Keyboard Entry of Coordinates. When you are using the X? and/or Y? subroutines to prompt you for point coordinates, and you want to terminate point acquisition, wait until the next X? or Y? prompt appears, then press [R/S] *without keying in a number*.† When you do so, execution of [PLTUXY] terminates. (If [PLTUXY] was executed automatically by [REPLOT], execution transfers to the [PLANOT] routine.)

Where only one coordinate of each point is to be entered from the keyboard, if the parameter controlling the other coordinate determines the number of points in the series, [PLTUXY] automatically terminates after processing the final point. For example:

- If NAME contains Y? and XINC contains a value specifying either the number of equal x-intervals or the interval between each x-coordinate, XINC determines the number of points that [PLTUXY] plots.

---

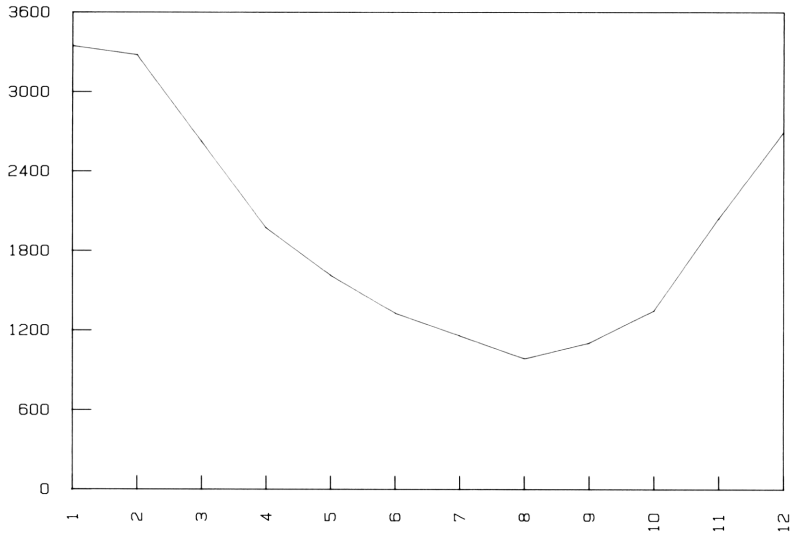
*If the XINC parameter already contains a numeric element, you will have to activate the Alpha keyboard by pressing [ALPHA] before you key in the label name. If the XINC parameter already contains an Alpha element, the Alpha keyboard is automatically activated when the XINC prompt is displayed. The same applies to the NAME parameter when you are using the register editing procedure. (However, the Alpha keyboard is always activated when the NAME prompt is displayed as a result of executing [NEWPLOT].)

†[PLTUXY] uses the HP-41's Numeric Input Flag (flag 22) to determine whether or not you key in a number in response to either the X? or Y? subroutine prompt. If you do not key in a number, flag 17 is set, which causes [PLTUXY] to stop generating coordinates and to halt or to transfer execution back to [REPLOT]. Thus, pressing [R/S] without a keyboard data entry terminates [PLTUXY] automatically. (If you design your own subroutine for XINC or NAME, you can use flag 17 for this same purpose.)

- If XINC uses the X? subroutine, and if NAME contains the label of another subroutine that returns y for a given x, neither parameter determines the number of points.

The first of the following two examples demonstrates [PLTUXY] operation when the Y? subroutine label is used for the NAME parameter and the XINC parameter determines the number of points plotted. The second example demonstrates [PLTUXY] operation where neither XINC or NAME determine the number of points plotted.

Example Using Y? In NAME. The bar chart example on page 16 in section 1 used a specialized program to plot kilowatt-hour electrical consumption over a 12-month period. The data supplied for that example indicates that energy consumption ranged from a low of 986 kilowatts per month to a high of 3,346 kilowatts per month. Suppose you wanted to use the same input data with the Utility Plotting Program to generate a simple line chart like the one shown below.



You can easily set up the plotter to generate the preceding chart by using [NEWPLOT] and [REPLOT] to initialize the data base. Before you begin to plot, ensure that there are 12 data storage registers available for the plotting data base and at least 26 unused memory registers available for the I/O buffer.

Keystrokes	Display	
[PCLBUF]		Clears the I/O buffer if it exists. If the buffer does not exist, **PL:PLS PINIT** is displayed.
[SIZE] 012		Ensures that R_{00} through R_{11} are available for the plotting data base.
[■] [GTO] [·] [·]		Packs program memory.
[PRGM]	**00 REG *nn***	Switches HP-41 to Program mode. The number of registers indicated by ***nn*** must be 26 or more. (If it is not, refer to Ensuring that Enough Unused Registers Are Available, page 11.)
[PRGM]		Removes HP-41 from Program mode.

Now use the Utility Plotting Program to plot a line chart of energy consumption.

Keystrokes	Display	
NEWPLOT	NAME=?	Prompts you to key in the name of the subroutine that calculates the *y*-coordinate of the plot.
Y? R/S	XMIN=-1.000?	Enters name of Y? subroutine included in Utility Plotting Program. HP-41 prompts you to select the minimum *x*-axis value.

Since the minimum *x*-axis value should represent the first month in the period, and the maximum *x*-value should represent the last, or 12th month in the period, use 1 for XMIN and 12 for XMAX.

Keystrokes	Display	
1 R/S	XMAX=1.000?	Enters minimum *x*-axis value. HP-41 prompts you for maximum *x*-axis value.
12 R/S	XINC=-11.000?	Enters maximum *x*-axis value. HP-41 prompts you for the *x*-increment.

The *x*-axis represents 12 months. Because we expect to plot one point for each month, use 1 as the increment between *x*-coordinates.

Keystrokes	Display	
1 R/S	YMIN=-1.000?	Enters *x*-increment. HP-41 prompts you for the *y*-axis minimum value.

Since the minimum *y*-axis value should represent the lower limit of the energy consumption plot and the maximum *y*-axis value should represent the upper limit of consumption, use 0 for YMIN and 3600 for YMAX.

Keystrokes	Display	
0 R/S	YMAX=1.000?	Enters minimum *y*-axis value. HP-41 prompts you for maximum *y*-axis value.
3600 R/S	PLOT?	Enters maximum *y*-axis value. HP-41 prompts you to indicate whether or not to generate a plot.

If you generate the plot now, the default ANNOT (plot annotation) parameter (set in R_{03} of the plotting data base) by NEWPLOT will plot ten major tic intervals, with corresponding tic labels, on each axis. This same default parameter will print the Y? label in the NAME parameter as the plot label. Before generating the plot, use the parameter editing feature to change the ANNOT parameter to specify tics and labels that correspond to months on the *x*-axis and blocks of 600 kwh on the *y*-axis. (Specifying 11 *x*-axis tic intervals produces 12 labeled tics—1 through 12; specifying 6 *y*-axis tic intervals produces 7 labeled tics—0 through 3600.) Use the ANNOT sign convention to avoid printing a plot label (that is, use a negative parameter value).

Keystrokes	Display	
3 R/S	ANNOT=1000.01 T=1000.01000?	Uses the REPLOT register editing feature to access the ANNOT parameter in R_{03}. (Display scrolls.)
1100.006 CHS R/S	PLOT?	Enters a value that specifies 11 tic intervals on the x-axis and 6 tic intervals on the y-axis. Because the value is negative, no plot label will be printed. HP-41 prompts you to indicate whether or not to generate a plot.
R/S		Begins automatic generation of a complete plot.

Because of the numeric XINC parameter you entered earlier, PLTUXY computes the x-coordinate needed for each point. However, because you used the Y? subroutine label for the NAME parameter, each time PLTUXY requires a y-coordinate, the Y? subroutine, which prompts you for a y-coordinate, is executed.

Keystrokes	Display	
	Y⟨0⟩=?	HP-41 prompts you for first y-coordinate (which is the first month's electricity consumption).
3346 R/S	Y⟨1⟩=?	Enters the y-coordinate corresponding to each month's consumption, then prompts you for the next y-coordinate.
3278 R/S	Y⟨2⟩=?	
2625 R/S	Y⟨3⟩=?	
1973 R/S	Y⟨4⟩=?	
1616 R/S	Y⟨5⟩=?	
1330 R/S	Y⟨6⟩=?	
1158 R/S	Y⟨7⟩=?	
986 R/S	Y⟨8⟩=?	
1105 R/S	Y⟨9⟩=?	
1350 R/S	Y⟨10⟩=?	
2043 R/S	Y⟨11⟩=?	
2694 R/S		Enters 12th y-coordinate.

Because the XINC parameter you entered earlier allows 12 x-coordinates,* PLTUXY execution terminates, and no more points are plotted. Because PLTUXY was executed under automatic control, execution transfers to PLANOT and completes the plot.

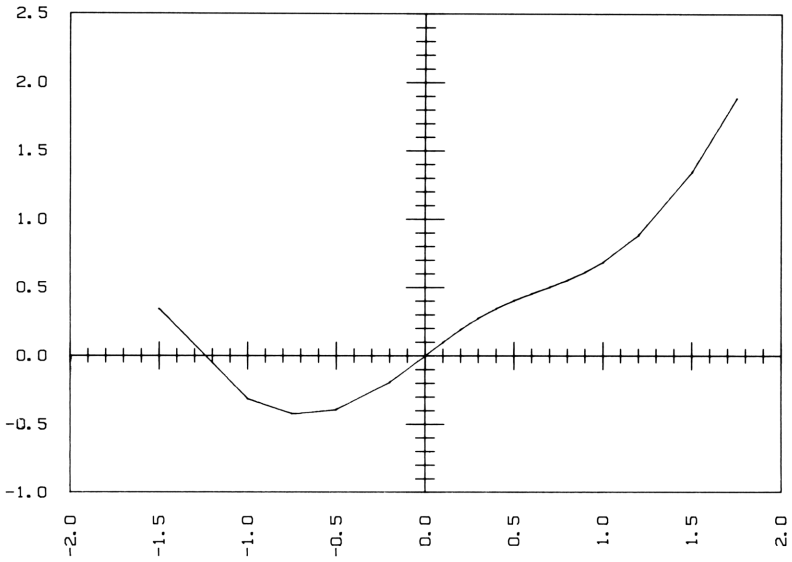
Keystrokes	Display	
None	PLOT?	Plot completed. HP-41 prompts you to indicate whether or not to generate another plot.

Example Using Keyboard-Entered X-Coordinates. Suppose that you wanted to generate a plot of the function

$$f(x) = x^2 - \ln(x^2 + e^{-x})$$

so that the portion of the function that lies between $x = 0$ and $x = 1$ is plotted with greater precision than the remainder of the function.

*The XMIN/XMAX range is 1 through 12. As XINC specifies 1 unit between x-coordinates, the result is 12 coordinates.



To generate this plot, load into program memory a subroutine that calculates $f(x)$. Use the subroutine's label for the NAME parameter. Use the label of the Utility Plotting Program's X? subroutine for the XINC parameter. This choice of parameters enables you to key in any x-coordinate you wish when prompted by X? for each successive point. The corresponding y-coordinate—$f(x)$—for each x-coordinate is calculated by the subroutine identified by NAME. Thus, while the plot is in the most interesting portion of the function, you can increase plotting precision by keying in x-coordinates that are closer together than the x-coordinates you enter for other portions of the plot. This method allows you to plot virtually as many points as you wish.

To begin, ensure that there is sufficient space in memory for this example (29 unused memory registers and 12 data storage registers).

Keystrokes	Display	
PCLBUF		Clears the I/O buffer if it exists. If the buffer does not exist, **PL:PLS PINIT** is displayed.
SIZE 012		Ensures that R_{00} through R_{11} are available for the plotting data base.
■ GTO · ·		Packs program memory.
PRGM	**00 REG *nn***	Switches HP-41 into Program mode. The number at the right of the display must be 29 or greater. (Refer to Ensuring that Enough Unused Registers Are Available, page 11.)

Now load into program memory a subroutine that generates $f(x)$ when x is given (in the X-register).

Keystrokes	Display	
■ LBL ALPHA	01 LBL _	
GRAPH ALPHA	01 LBL"GRAPH	Enters subroutine label.
ENTER↑	02 ENTER↑	Enters instructions for calculating the value of the expression $x^2 - \ln(x^2 + e^{-x})$.
ENTER↑	03 ENTER↑	
CHS	04 CHS	
■ e^x	05 E↑X	
$x \gtrless y$	06 X<>Y	
■ x^2	07 X↑2	
+	08 +	
LN	09 LN	
CHS	10 CHS	
$x \gtrless y$	11 X<>Y	
■ x^2	12 X↑2	
+	13 +	
PRGM		

Now execute NEWPLOT and specify an x-axis range of –2 through 2.

Keystrokes	Display	
NEWPLOT	NAME=?	Prompts you to key in the name of the subroutine that calculates the y-coordinate of the plot.
GRAPH R/S	XMIN=-1.000?	Enters subroutine name. HP-41 prompts you to key in the minimum x-axis value.
2 CHS R/S	XMAX=1.000?	Enters XMIN value. HP-41 prompts you to key in the maximum x-axis value.
2 R/S	XINC=-11.000?	Enters XMAX value. HP-41 prompts you to select the x-increment.

Because you want to control and vary the interval between x-coordinates, specify the Utility Plotting Program's X? subroutine. When executed by PLTUXY, X? prompts you to key in an x-value.

Keystrokes	Display	
ALPHA X? R/S	YMIN=-1.000?	Enters X? subroutine label for XINC parameter. HP-41 prompts you for minimum y-axis value.
R/S	YMAX=1.000?	Enters default YMIN value. HP-41 prompts you for maximum y-axis value.
2.5 R/S	PLOT?	Enters YMAX value. HP-41 prompts you to generate a plot or edit a parameter.

Generating a plot now would result in a plot annotation reflecting the default ANNOT parameters (refer to The ANNOT Parameter, page 28). Also, the axes would be drawn at the bottom and the left plotting limits. To enhance the plot's readability, specify that tics and labels be drawn at 0.5-unit intervals on both axes, and that the axes themselves be plotted at the 0 intercepts. Since the x-axis has a range of 4 units, specify 8 x-axis tics (4 / 0.5 = 8). Because the y-axis has a range of 3.5 units, specify 7 y-axis tics. Include five minor tic intervals between the major tics on both axes. The resulting ANNOT parameter is (00)805.00705.

Keystrokes	Display	
3 R/S	ANNOT=1000.01 T=1000.01000?	Uses register editing to access the ANNOT parameter. (Display scrolls.)

Keystrokes	Display	
805.00705 CHS R/S	PLOT?	Enters ANNOT parameter.
6 R/S	XAXAT=−1.000?	Accesses current (default) *y*-intercept parameter for *x*-axis.
0 R/S	PLOT?	Enters XAXAT parameter. HP-41 prompts you to generate a plot or edit a parameter.
9 R/S	YAXAT=−2.000?	Accesses current (default) *x*-axis intercept parameter for *y*-axis.
0 R/S	PLOT?	Enters YAXAT parameter. HP-41 prompts you to generate a plot or edit a parameter.

If you press R/S to automatically generate the plot, you will be prompted to key in *x*-coordinates without having a labeled plotting area to refer to while selecting the coordinates. To provide the plot annotation *before* you begin plotting points, first initialize the plotter by executing PLINIT, then executing PLANOT.

Keystrokes	Display	
PLINIT		Initializes plotter according to parameters currently in the plotting data base, then halts.*
	2.500	
	-or-	
	PLOT?	
PLANOT		Generates plot annotation, then halts.*
	0.0000	
	-or-	
	PLOT?	

Because the *x*-coordinate of each point to be plotted is a value you key in when prompted by X?, you have control over the interval between points. To plot the function, execute PLTUXY and respond to the prompts. You can terminate the operation whenever an X? prompt is displayed by pressing R/S *without* keying in any number. (In the following series of keystrokes, if you accidentally enter an erroneous value for an *x*-coordinate and press R/S, the resulting plotted point will be in error. However when the next prompt appears, you can recover by reentering the last correct coordinate and pressing R/S. This returns the pen to the last correct point. You can then resume entering coordinates. If you follow this procedure only the parameter number in the X⟨*n*⟩ = ? prompt will be in error.)

Keystrokes	Display	
PLTUXY	X⟨0⟩= ?	Begins plot generation. First iteration of PLTUXY prompts you for first *x*-coordinate.
1.5 CHS R/S	X⟨1⟩= ?	Enters first coordinate. PLTUXY plots first point, then begins second iteration and prompts you for next *x*-coordinate.
1 CHS R/S	X⟨2⟩= ?	Plots four more points at various *x*-intervals.
.75 CHS R/S	X⟨3⟩= ?	
.5 CHS R/S	X⟨4⟩= ?	
.2 CHS R/S	X⟨5⟩= ?	

*Refer to footnote on page 33.

Keystrokes	Display	
0 R/S	X⟨6⟩= ?	Plots eleven points that are separated by small, equal intervals to increase plotting precision.
.1 R/S	X⟨7⟩= ?	
.2 R/S	X⟨8⟩= ?	
.3 R/S	X⟨9⟩= ?	
.4 R/S	X⟨10⟩= ?	
.5 R/S	X⟨11⟩= ?	
.6 R/S	X⟨12⟩= ?	
.7 R/S	X⟨13⟩= ?	
.8 R/S	X⟨14⟩= ?	
.9 R/S	X⟨15⟩= ?	
1 R/S	X⟨16⟩= ?	
1.2 R/S	X⟨17⟩= ?	Plots three more points at various x-intervals.
1.5 R/S	X⟨18⟩= ?	
1.75 R/S	X⟨19⟩= ?	

Terminate the plot by pressing R/S without keying in any number in response to the prompt.

Keystrokes	Display	
R/S	0.000 *-or-* PLOT?	Terminates plot.*

Plotting From Buffers

For plotting purposes, a buffer is any block of consecutive HP-41 data storage registers where the initial register number is 12 or greater. A buffer can contain both the x- and y-coordinates of each point (termed an *x,y buffer*) or only the y-coordinates (termed a *y-only buffer*). If a buffer contains only y-coordinates, PLTUXY acquires the corresponding x-coordinates using the XINC parameter. As with all coordinate sources you can specify in the NAME parameter, you should think of a buffer specified in this parameter as a means by which PLTUXY *acquires* points. (Refer to the PLTUXY flowchart on pages 176 through 178.)

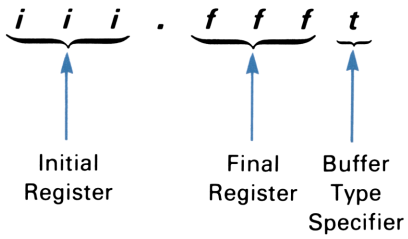
Setting Up a Buffer. To set up a buffer, first determine the number of coordinates you want the buffer to contain. Then ensure that enough storage registers exist to store the coordinates. (The first register in the block is termed $R_{initial}$, or R_i; the last register is termed R_{final}, or R_f.) An easy way to ensure that you have enough registers is by attempting to recall the current contents of R_f to the X-register. If R_f does not exist, then execute SIZE to create the necessary number of storage registers. The last step in setting up a buffer is to store the values of the coordinates into the desired registers. You can use one of three methods to do so:

- Key in each coordinate and store it using the STO key.
- Execute REPLOT and use the register editing feature to store coordinates.
- Execute PLTUXY with XINC and NAME parameters that acquire each successive coordinate pair, and a PLTPRM parameter that stores the points in a buffer.

Note: When Alpha data is stored in a buffer register, if you use a type 0 buffer pointer in the NAME parameter when you plot the buffer, the pen lifts when PLTUXY attempts to use the Alpha data as a coordinate for plotting a point. If PLTUXY subsequently acquires numeric data for both coordinates of a point, the pen moves to that point and drops. PLTUXY operates in this way because the plotter module's PLREGX function (described in part II) is used to plot type 0 buffers. However, PLREGX is not used to plot type 1 or type 2 buffers. Thus, an **ALPHA DATA** error message results if you attempt to plot a type 1 or type 2 buffer containing Alpha data.

*Refer to footnote on page 33.

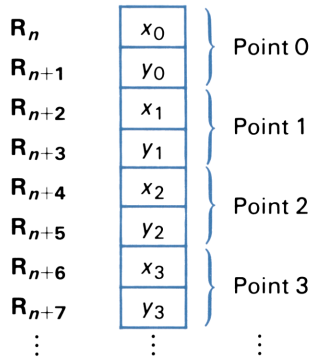
How to Access a Buffer. In order for PLTUXY to acquire coordinates from a buffer, you must use a buffer pointer for the NAME parameter. In a NAME buffer pointer, the integer identifies the initial buffer register (***iii***), the first three digits of the fraction identify the final buffer register (***fff***), and the fourth digit of the fraction identifies the buffer type (***t***). That is:



Buffer Types. The main purpose of specifying a buffer type is to indicate to PLTUXY whether the desired buffer contains both x- and y-coordinates or only y-coordinates. Where the buffer contains both x- and y-coordinates, the type specifier also indicates whether to plot using the character, pen, line type and buffer-filling elements specified by ***ccclp*** and ***bbbt*** in the PLTPRM parameter or to plot using only the pen and line type elements. The following paragraph describes in detail the buffer type distinctions.

When you want to use a buffer, you can choose either one of two x/y buffer types, or the y-only buffer type. The type specifiers and their corresponding buffers are:

- **Type 0.** Indicates to PLTUXY that the buffer contains both the x- and y-coordinates of each point to be plotted. The coordinates should be stored as shown in the following illustration.



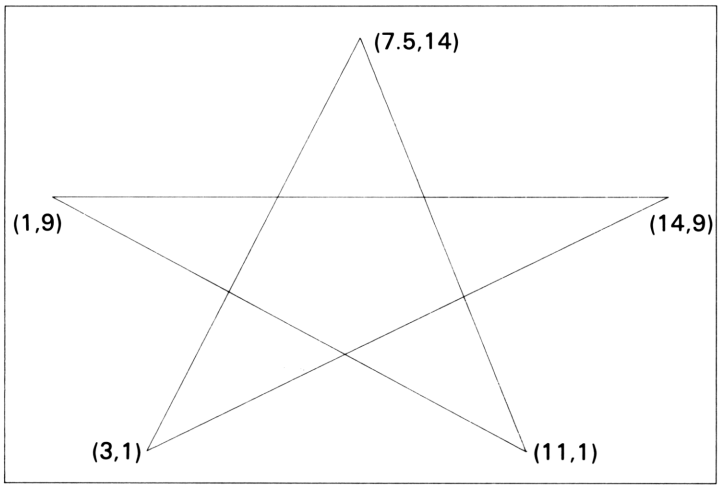
As in other types of plotting, PLTUXY uses the ***l*** and ***p*** elements in the integer portion of the PLTPRM parameter (***ccclp***) to determine line type and pen number. However, the ***ccc*** and ***bbbt*** elements are ignored. That is, even if ***ccc*** and/or ***bbbt*** are nonzero values, no Alpha characters will be printed at the plotted points and no buffer filling will occur. (For a description of ***ccclp.bbbt***, refer to The Character, Line Type, Pen, and Buffer-Filling Operations, page 30.)

Note: If the ***lp*** elements are zeroes, the plotter attempts to plot the buffer without using a pen. Notice also that, because type 0 buffer plotting uses the plotter module's PLREGX function to plot the buffer, autoscaling is not performed by type 0 buffer plotting.

- **Type 1.** Indicates to PLTUXY that the buffer contains only the y-coordinates of each point to be plotted. The y-coordinate of the first point is stored in R_i; the y-coordinate of the second point is stored in R_{i+1}, and so on. PLTUXY uses the XINC parameter to acquire the corresponding x-coordinates.
- **Type 2.** This type is interpreted by PLTUXY in the same way as type 0 except that the character (***ccc***), buffer-filling (***bbbt***), and autoscaling (***cccIp*=*0***) elements are not ignored by PLTUXY.*

The next two examples demonstrate how to plot using type 0 and type 1 buffers.

Example of Type 0 Buffer Plotting. Use NEWPLOT to define a plotting area 15 units square. Then store in R_{12} through R_{23} the coordinates for a series of points that, when plotted, form the five-point star shown below.



To plot the star, you will need, in addition to 12 data registers for the plotting data base and 26 unused memory registers for the I/O buffer, 12 data storage registers to contain the coordinates needed for plotting. The following keystrokes configure the HP-41's memory for these requirements.

Keystrokes	Display	
PCLBUF		Clears the I/O buffer if it exists. If the buffer does not exist, **PL:PLS PINIT** is displayed.
SIZE 024		Ensures that R_{00} through R_{11} are available for the plotting data base, and that R_{12} through R_{23} are available for a plotting buffer.
■ GTO · ·		Packs program memory.
PRGM	**00 REG *nn***	Switches HP-41 into Program mode. The number at the right of the display must be 26 or greater. (If it is not, refer to Ensuring that Enough Unused Registers Are Available, page 11.)
PRGM		Removes HP-41 from Program mode.

*PLTUXY operation with a type 0 buffer uses the PLREGX function described on page 83 in part II. PLTUXY operation with a type 2 buffer does not use PLREGX. When ***cccIp*** = 0, autoscaling occurs instead of plotting. Autoscaling is mentioned on page 30 and described later in this section, on page 55.

Now set up the plotting data base. For the NAME parameter, switch the HP-41 out of Alpha keyboard and key in the number 12.0230, which specifies a type 0 buffer using registers R_{12} through R_{23}. Later, when you generate the plot, [PLTUXY] will acquire both the x- and y-coordinates of each point from this buffer. For this reason, no x-coordinates will be acquired through the XINC parameter, and you can therefore ignore it.

Keystrokes	Display	
[NEWPLOT]	NAME?	Prompts you to key in the name of a subroutine.
[ALPHA] 12.023 [R/S]	XMIN=-1.000?	Switches HP-41 out of Alpha keyboard and enters a type 0 buffer pointer. HP-41 prompts you for the minimum x-axis value.
0 [R/S]	XMAX=1.000?	Enters XMIN value and prompts you for maximum x-axis value.
15 [R/S]	XINC=-11.000?	Enters XMAX value and prompts you for the x-increment value.
[R/S]	YMIN=-1.000?	Leaves XINC value unchanged and prompts you for y-minimum value. (As mentioned in the preceding paragraph, XINC is not used when a type 0 buffer is specified.)
0 [R/S]	YMAX=1.000?	Enters YMIN value and prompts you for y-maximum value.
15 [R/S]	PLOT?	Enters YMAX value and prompts you to generate a plot or edit a parameter.

Unless you specify otherwise, every plot includes a default set of major tics with corresponding labels. For this plot, eliminate the tics and labels by using zero for the ANNOT parameter.

Keystrokes	Display	
3 [R/S]	ANNOT=1000.01 T=1000.01000?	Uses register editing to access the ANNOT parameter. (Display scrolls.)
0 [R/S]	PLOT?	Enters ANNOT parameter and prompts you to generate a plot or edit a parameter.
11 [STO] 12	11.000	Stores in R_{12} the x-coordinate of first point of star.
1 [STO] 13	1.000	Stores in R_{13} the y-coordinate of first point of star.
[STO] 14 9 [STO] 15	9.000	Stores in R_{14} through R_{21} the x- and y-coordinates of the remaining points.
14 [STO] 16 9 [STO] 17	9.000	
3 [STO] 18 1 [STO] 19	1.000	
7.5 [STO] 20 14 [STO] 21	14.000	

To close the star plot, the pen must draw a line from the last point back to the first point. Thus, the last coordinate pair in the buffer is a duplicate of the first coordinate pair.

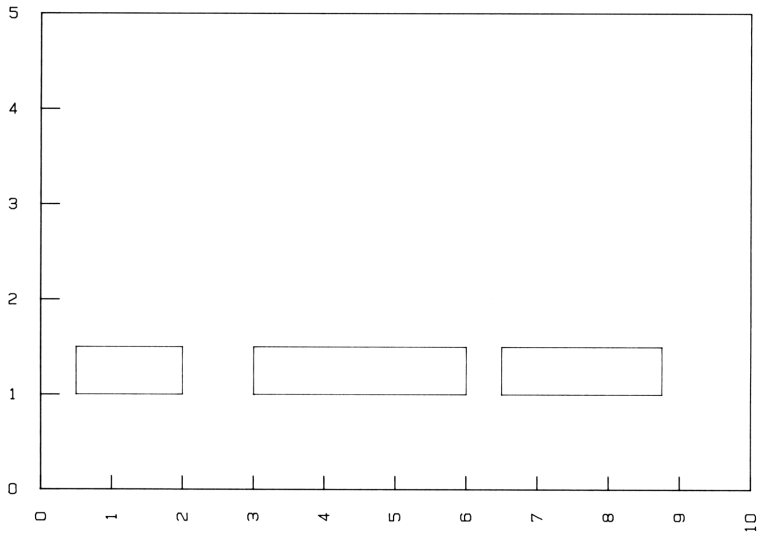
Keystrokes	Display	
11 [STO] 22 1 [STO] 23	1.000	Stores in R_{22} and R_{23} the coordinates of the first point.

Now let's automatically generate the complete star plot.

Keystrokes	Display	
REPLOT	PLOT?	HP-41 prompts you to generate a plot or edit a parameter.
R/S		Generates the star plot, then prompts you as above.
	PLOT?	

In the preceding example, the XINC parameter (R_{05}), whose sole purpose is to acquire x-coordinates, was ignored by PLTUXY because the x-coordinates for the plot were provided in the buffer specified by the type 0 buffer pointer used for the NAME parameter (R_{08}). In the next example, which demonstrates plotting with a type 1 buffer pointer (y-coordinates only), the XINC parameter must be used for acquiring x-coordinates.

Example of Type 1 Buffer Plotting. One use of y-only buffers is to plot shapes that remain constant in the y-dimension, but vary in the x-dimension. To demonstrate, use type 1 buffer plotting to plot three windows having the same height, but differing widths.



To begin, define a plotting area 5 units high and 10 units wide. Then, set up data registers R_{12} through R_{16} as a type 1 buffer by storing in these registers the series of y-coordinates needed for a series of points that form a rectangle. Use the X? subroutine for the XINC parameter so that you can key in variables for x-coordinates.

To execute this example you will need 12 data registers for the plotting data base, 5 data registers for the plotting buffer, and 26 unused memory registers for the I/O buffer.

Keystrokes	Display	
PCLBUF		Clears the I/O buffer if it exists. If it does not exist, PL:PLS PINIT is displayed.
SIZE 017		Ensures that R_{00} through R_{16} are available for the plotting data base and for the type 1 plotting buffer.

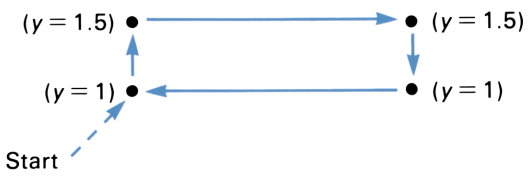
Use NEWPLOT to intitialize the plotting data base. Enter 12.0161 as a type 1 buffer pointer when prompted by NAME?. Because you want to be prompted for each x-coordinate used in the plot, use the X? subroutine label for the XINC parameter.

Keystrokes	Display	
NEWPLOT	NAME?	Prompts for a subroutine label or buffer pointer.
ALPHA 12.0161 R/S	XMIN=-1.000?	Removes HP-41 from Alpha keyboard and enters type 1 buffer pointer. HP-41 prompts for minimum x-axis value.
0 R/S	XMAX=-1.000?	Enters XMIN value. HP-41 prompts for maximum x-axis value.
10 R/S	XINC=-11.000?	Enters XMAX value. HP-41 prompts for x-increment.
ALPHA X? R/S	YMIN=-1.000?	Switches HP-41 to Alpha keyboard and enters X? subroutine label. HP-41 prompts for minimum y-axis value.
0 R/S	YMAX=1.000?	Enters YMIN value. HP-41 prompts for maximum y-axis value.
5 R/S	PLOT?	Enters YMAX value. HP-41 prompts you to generate a plot or edit a parameter.

The x-axis is 10 units long and the y-axis is 5 units long. Thus, the current annotation parameter (which is the default parameter set by NEWPLOT—ANNOT=1000.01000) will generate 11 x-axis tics with integer-only labels and 11 y-axis tics with fractional labels. Change the annotation parameter so that integer-only tics and labels will be generated for the y-axis. (Since the NAME parameter is a numeric value, it makes no difference whether the new ANNOT parameter is given a positive or negative sign.)

Keystrokes	Display	
3 R/S	ANNOT=1000.01 T=1000.01000?	Uses the register editing feature to access the annotation parameter in R_{03}. (Display scrolls.)
1000.005 R/S	PLOT?	Enters new annotation parameter. HP-41 prompts you to generate a plot or edit a parameter.

To create the type 1 buffer needed to plot the three windows, simply store the y-coordinates in registers R_{12} through R_{16}. Design the windows for a height of 0.5 units, with the lower corners at $y = 1$ and the upper corners at $y = 1.5$. Plan the sequence of points so that the plot for each window begins at the lower-left corner and proceeds in sequence through the upper-left, upper-right, and lower-right corners; then returns to the starting point.



Thus, you should use the following sequence when storing the y-coordinates in the plotting buffer: 1, 1.5, 1.5, 1, 1.

Keystrokes	Display	
1 STO 12	1.000	Creates y-only (type 1) plotting buffer in R_{12} through R_{16} by storing y-coordinates in these registers.
1.5 STO 13	1.500	
STO 14	1.500	
1 STO 15	1.000	
STO 16	1.000	

Now generate a plot of the three windows.

Keystrokes	Display	
REPLOT	PLOT?	HP-41 prompts you to generate a plot or edit a parameter.
R/S	X⟨0⟩= ?	Begins plot of first window by prompting you for the first x-coordinate.

The subroutine label (X?) used for the XINC parameter causes the HP-41 to prompt you for each point's x-coordinate. Each point's y-coordinate is retrieved, when needed, from the plotting buffer you created in R_{12} through R_{16}.

Keystrokes	Display	
.5 R/S	X⟨1⟩= ?	Enters x-coordinates and plots points for first window, then generates plot annotation.
.5 R/S	X⟨2⟩= ?	
2 R/S	X⟨3⟩= ?	
2 R/S	X⟨4⟩= ?	
.5 R/S		
	PLOT?	HP-41 prompts you to generate a plot or edit a parameter.

Because the plotter was initialized and the plot annotation was generated with the first window plot, it is not necessary to perform these operations again before plotting another window in the same plotting area. Thus, instead of pressing R/S (which initiates automatic execution of PLINIT and PLANOT as well as PLTUXY) to plot the next window, just execute PLTUXY.

Keystrokes	Display	
PLTUXY	X⟨0⟩= ?	Enters x-coordinates and plots points for second window, then halts.
3 R/S	X⟨1⟩= ?	
3 R/S	X⟨2⟩= ?	
6 R/S	X⟨3⟩= ?	
6 R/S	X⟨4⟩= ?	
3 R/S	PLOT?	Plot completed.*
	-or- 0.0000	

*Refer to footnote, page 33.

Now plot another window in the plotting area you used for the preceding two windows.

Keystrokes	Display	
PLTUXY	X⟨0⟩= ?	Enters x-coordinates and plots points for third window in same manner as preceding keystroke series.
6.5 R/S	X⟨1⟩= ?	
6.5 R/S	X⟨2⟩= ?	
8.75 R/S	X⟨3⟩= ?	
8.75 R/S	X⟨4⟩= ?	
6.5 R/S	PLOT?	
	-or-	
	0.0000	

The preceding examples demonstrate plotting with type 0 and type 1 buffers specified by an appropriate pointer for the NAME parameter. Plotting with a type 2 buffer specified by NAME operates the same as with a type 0 buffer except that character plotting (***ccc*** $\neq 0$) or autoscaling (***cccIp*** $= 0$), and buffer-filling (***bbbt*** $\neq 0$) will be performed, if specified. (Refer to Buffer Types, page 49.) Now that you have seen an example of plotting with coordinates entered singly from the keyboard, let's see how to handle such applications when it is not clear as to which axes minimums and maximums should be specified before entering coordinates.

The Autoscale Option

In plotting applications where the ranges of the x- and/or y-coordinates are not known before plotting, it is sometimes inconvenient to specify estimated maximum and minimum axis values. The autoscale (*automatic scaling*) option enables you to solve this problem by automatically resetting the maximum and minimum parameters on one or both axes to the appropriate limits—as determined by the ranges of the coordinates PLTUXY actually acquires through the XINC and/or NAME parameters. Because the axis parameters are not reset until all x- and y-coordinates have been processed, no points are plotted during an autoscaling operation. Once you have executed PLTUXY under autoscale control to determine the limits of a plot, you can immediately generate a properly scaled plot by just reexecuting PLTUXY.

How to Specify Autoscaling. A zero value representing the entire *integer* portion of the PLTPRM parameter (***cccIp*** $= 0$) causes PLTUXY to perform autoscaling. Thus, to specify autoscaling, execute REPLOT and use the register editing feature to replace the current PLTPRM parameter. (To simultaneously perform autoscaling and buffer-filling (which is described later, on page 59) the fractional portion of PLTPRM must also be zero.)

How Autoscaling Operates. When you execute PLTUXY under autoscale control, x- and y-coordinate pairs are generated by the XINC and/or NAME parameters in the same way as during any other execution of PLTUXY. However, when PLTPRM specifies autoscaling, PLTUXY does the following *instead* of plotting the coordinates as points:

1. Identifies the maximum and minimum coordinates for each axis.
   - If the XINC parameter is a subroutine label, or if the NAME parameter specifies a type 0 or 2 buffer (x, y buffer) the current XMIN and XMAX parameters will be reset to match the minimum and maximum x-axis coordinates. Otherwise, XMIN does not change, and XMAX is set to the x-coordinate in the rightmost point generated by PLTUXY. This is because the only remaining option for defining coordinates on the x-axis is to interpret the XINC parameter as specifying either the number of x-intervals ($-n$) between XMIN and XMAX or the interval between consecutive x-increments (int) bounded by XMIN and XMAX.
   - Regardless of the source of the y-coordinates, the YMIN and YMAX parameters will be reset to match the minimum and maximum y-axis coordinates.

2. After all coordinates have been generated, if the fractional portion of the PLTPRM parameter specified buffer filling (***bbbt*** $\neq 0$), PLTUXY then replaces the current NAME parameter with a buffer pointer that specifies the initial and final registers of the buffer and the type of buffer (***iii.ffft***). This allows you to simultaneously autoscale and fill a buffer in one operation, then immediately plot the buffer in a second operation.
3. Resets PLTPRM to the default value originally set by NEWPLOT —***cccIp.bbbt*** = (000)11.000. (That is, the null—***ccc*** = 0—character, line type 1, pen 1, and no buffer filling.)

Note: If you specify autoscaling, then select automatic plot generation by pressing R/S when prompted by **PLOT?**, autoscaling occurs as described in the preceding text instead of the plot of a function or series of points. However, because automatic plot generation includes execution of PLINIT and PLANOT, the plot annotation—frame, axes, tics, and labels—will be generated.

How to Use Autoscale. To use the autoscale feature:

1. Execute NEWPLOT or REPLOT and specify data base parameters as follows:
   - Enter the appropriate NAME parameter in R_{08}. (Program label or buffer number.)
   - If the XINC parameter you plan to use is a subroutine label, or if the NAME parameter specifies an x, y (type 0 or 2) buffer, ignore the XMIN and XMAX parameters, as they will be reset by the autoscaling process. Otherwise, enter these parameters.
   - Enter the desired XINC parameter. (If the NAME parameter specifies a type 0 or 2 buffer, you can skip this step because XINC will be ignored by PLTUXY.)
   - Ignore the current YMIN and YMAX parameters, as they will be reset by the autoscaling process.
   - Enter zero for the PLTPRM parameter to specify autoscale.*
2. Execute PLINIT, then PLTUXY. The autoscaling process described in the preceding text takes place. No plot is generated.

   Note: If you want to generate the plot annotation during this step instead of in the following step, execute automatic plot generation instead of executing PLINIT and PLTUXY individually. That is, when prompted by **PLOT?**, press R/S.

3. Generate a plot of the points specified by the XINC and/or NAME parameter either by executing PLTUXY, or by selecting automatic plot generation.

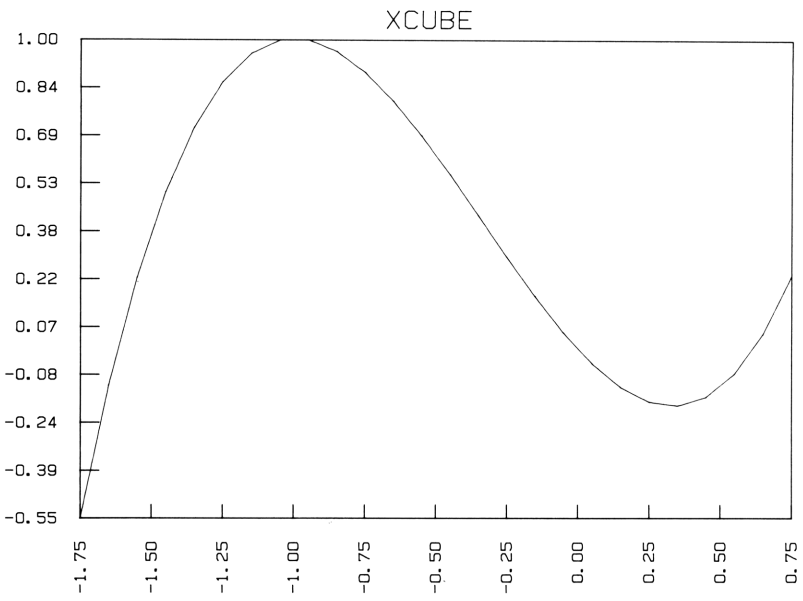
Keep in mind that between steps 2 and 3 you can change any data base parameter. In particular, you may want to use autoscaling to determine the limits of the x- and/or y-axes, then change the current axes intercepts (XAXAT and YAXAT) and annotation (ANNOT) to conform to the axes ranges resulting from the autoscaling. (Remember that if you change any of the scale parameters, you must reexecute PLINIT before proceeding with the plot.)

Example of Autoscaling. The function

$$f(x) = x^3 + x^2 - x$$

plots a curve that crosses the x-axis three times. Suppose that you wanted to plot this function so that the limits of the y-axis coincided with the maximum and minimum y-values between $x = -1.75$ and $x = .75$, as shown in the following illustration:

*If you wish to fill a buffer while autoscaling, the fractional portion of PLTPRM must contain the buffer data. This topic is discussed under Filling a Buffer on page 59.



You can easily determine these y-values using autoscaling. To do so, step through the following operations.

Keystrokes	Display	
PCLBUF		Clears the I/O buffer if it exists. Otherwise displays **PL:PLS PINIT**.
SIZE 012		
■ GTO · ·		Packs HP-41 program memory.
PRGM	**00 REG *nn***	If the value represented by ***nn*** is less than 29, there is not enough space in memory. Refer to Ensuring that Enough Unused Registers Are Available, page 11.

Key in the following program that calcualtes a y-value ($f(x)$) for the expression $x^3 + x^2 - x$.

Keystrokes	Display	
■ LBL ALPHA	**01 LBL _**	
XCUBE ALPHA	**01 LBL"XCUBE**	
ENTER↑	**02 ENTER↑**	
ENTER↑	**03 ENTER↑**	
ENTER↑	**04 ENTER↑**	
3	**05 3 _**	Calculates value of above-mentioned expression when given y in the X-register.
■ y^x	**06 Y↑X**	
$x \gtrless y$	**07 X<>Y**	
■ x^2	**08 X↑2**	
+	**09 +**	
$x \gtrless y$	**10 X<>Y**	
−	**11 −**	
PRGM		Removes HP-41 from Program mode.

Now initialize the plotter by executing [NEWPLOT]. Use the label of the preceding program as the NAME parameter. Set the x-minimum and x-maximum values to $x = -1.75$ and $x = 0.75$. To increase the plot's precision, use an equal x-increment of 0.1 (XINC = .1). Ignore the minimums and maximums for the y-axis. To specify autoscaling, replace the integer in the default PLTPRM parameter (R_{02}) with 0. (The default fractional portion of this parameter is also zero.)

Keystrokes	Display
[NEWPLOT]	NAME=?
XCUBE [R/S]	XMIN=-1.000?
1.75 [CHS] [R/S]	XMAX=1.000?
.75 [R/S]	XINC=-11.000?
.1 [R/S]	YMIN=-1.000?
[R/S]	YMAX=1.000?
[R/S]	PLOT?
2 [R/S]	PLTPRM=11.000 TPRM=11.0000?
0 [R/S]	PLOT?

Before you execute [PLTUXY] to begin autoscaling, initialize the plotter to the current plotting data base by executing [PLINIT].

Keystrokes	Display	
[PLINIT]	1.000 *-or-* PLOT?	Initializes plotter to conform to current plotting data base, then halts.*

Note: Failing to execute [PLINIT] before an autoscaling operation may cause the plot to appear in an undesirable location on the page. If [PLINIT] has not been executed at all, the I/O buffer may not exist, in which case executing [PLTUXY] results in a **PL:PLS PINIT** message.

You are now ready to execute the autoscaling operation, which occurs during execution of [PLTUXY].

Keystrokes	Display	
[PLTUXY]	0.995 *-or-* PLOT?	Executes autoscaling operation, then halts.*

To see the results of autoscaling, execute [REPLOT] and use register editing to inspect the y-axis maximum and minimum. Then, examine PLTPRM and notice that autoscaling changed the parameter from the zero you originally entered to 11 (which specifies pen 1 and line type 1 and allows you to immediately generate a plot of the function or data you just used for autoscaling). If your HP-41 is currently displaying the **PLOT?** prompt, you can bypass the first instruction ([REPLOT]) in the following keystroke series.

*Refer to the footnote on page 33.

Keystrokes	Display	
REPLOT	PLOT?	Prompts you to generate a plot or edit a parameter.
4 R/S	YMIN=-0.547?	Displays the *y*-axis minimum calculated by the autoscaling operation.
R/S	PLOT?	Prompts as above.
7 R/S	YMAX=0.995?	Displays the *y*-axis maximum calculated by the autoscaling operation.
R/S	PLOT?	Prompts as above.
2 R/S	PLTPRM=11.00 TPRM=11.0000?	Displays the default value which autoscaling inserted in place of the zero you used to originally specify the autoscaling operation. (Display scrolls.)
R/S	PLOT?	Prompts as above.

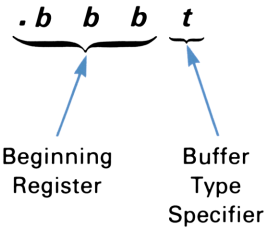
Before generating the plot resulting from the autoscaling operation, edit the annotation parameter so that the labels for the *y*-axis tics will be printed in FIX 2 display format instead of SCI 4. Then generate the plot by executing PLTUXY and PLANOT.

Keystrokes	Display	
3 R/S	ANNOT=1000.01 T=1000.01000?	Accesses annotation parameter. (Display scrolls.)
1000.210 R/S	PLOT?	Specifies 10 labels on each axis, with FIX 2 display format on the *y*-axis.
PLTUXY	0.000 *-or-* PLOT?	Plots function that was used by preceding execution of PLTUXY for autoscaling, then halts.*
PLANOT	0.0000 *-or-* PLOT?	Generates plot annotation, then halts.*

Filling a Buffer

As indicated earlier, *x*- and *y*-coordinates are acquired by PLTUXY using the XINC and/or NAME parameters. Using the buffer-filling option, you can direct PLTUXY to load these coordinates into a buffer. The operation can be designed to load either the *x*- and *y*-coordinates or only the *y*-coordinates. As you will see, buffer filling must always be combined with either a plotting or autoscaling operation.

The Buffer-Filling Pointer. This pointer, which replaces zero in the fractional portion of the PLTPRM parameter, contains (1) the beginning register of the buffer and (2) the buffer type. The pointer format is:



Because the plotting data base uses data registers R_{00} through R_{11}, the beginning register must be a register numbered 12 or higher. As there is no final register specified, the block of registers comprising the buffer is limited only by the number of coordinates that must be stored or by the number of data registers available.

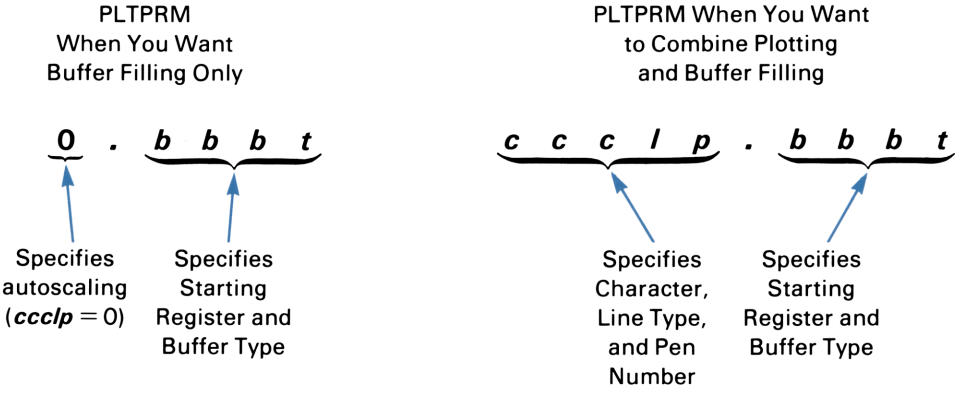
How Buffer Filling Affects the NAME and PLTPRM Parameters. When a buffer-filling pointer is used in PLTPRM, PLTUXY fills the specified buffer, then:

*Refer to footnote on page 33.

- Replaces the NAME parameter with an ***iii.ffft*** buffer pointer that corresponds to the filled buffer.*
- Replaces the buffer-filling pointer in PLTPRM (***.bbbt***) with zero. (Unless you simultaneously fill a buffer and autoscale, the ***ccclp*** elements of PLTPRM are not changed.)

If you simultaneously autoscale and fill a buffer, these automatic changes enable you to plot the filled buffer immediately by simply executing PLTUXY. (Otherwise, the buffer will be plotted as it is filled.)

How PLTUXY Uses the Buffer-Filling Pointer. As described in the preceding paragraphs, the buffer-filling pointer forms the fractional portion of the PLTPRM parameter. The integer portion of PLTPRM is formed by ***ccclp***, which determines whether plotting or autoscaling takes place when you execute PLTUXY. Thus, when you want to simultaneously plot points and load their coordinates into a buffer, specify a nonzero value for the ***ccclp*** portion of the PLTPRM parameter. When you want to load coordinates into a buffer without plotting any points (autoscaling), specify zero for the integer portion of PLTPRM.† Here are illustrations of both options:



The ***ccclp*** and ***bbbt*** parameters are unchanged by execution of PLTUXY *unless* ***ccclp*** is set to zero for autoscaling. (Refer to item 3 under How Autoscaling Operates on page 55.)

Buffer Type Specifiers. The type specifiers used in a PLTPRM buffer pointer to fill a buffer are the same as those used in a NAME parameter pointer to acquire coordinates from an existing buffer. (Refer to Buffer Types, page 49.) Thus, to build an *x, y* buffer, use 0 or 2 for the type specifier (***t***) in the buffer pointer. To build a *y*-only buffer use 1 for the type specifier.

> **Note:** The *only* purpose of a type specifier following ***bbb*** in the PLTPRM parameter is to distinguish *x, y*-buffer filling from *y*-only buffer filling. Thus, using 2 as a buffer specifier in PLTPRM produces results identical to using 0 as a buffer specifier. This is in contrast to the use of buffer specifiers in the NAME Parameter, where type 0 and type 2 differ.

---

*There is one exception. If the NAME Parameter initially contains an ***iii.fff*** **0** buffer pointer (specifies a type 0 buffer), the buffer-filling pointer in PLTPRM is ignored. (Refer to Buffer Types, page 49.)

†You can plot a special shape at each plotting point instead of actually plotting the specified point. Because this option requires that you *replace* the numeric PLTPRM parameter with an Alpha subroutine label, you cannot direct PLTPRM to simultaneously fill a buffer with coordinates and plot special shapes at the points defined by these coordinates. It is possible, however, to include in the subroutine the instructions necessary to store the coordinates in the desired buffer.

The PLANOT Routine

The PLANOT routine generates a frame, axes, tics, and labels according to the parameters in the plotting data base. PLANOT accesses all data base parameters except PLTPRM and XINC. When you automatically generate a complete plot (refer to The **PLOT?** Prompt, page 25), the REPLOT routine executes PLANOT after executing PLINIT and PLTUXY. However, you can execute PLANOT individually whenever you want to generate a plot annotation without unnecessarily reinitializing the plotter or plotting a function or series of points. (Execution of PLANOT leaves the plotter module's label origin setting at LORG 5. Refer to Changing the Label's Location, page 94.)

When to Use PLANOT

As indicated above, when you use automatic plot generation you don't need to think about executing PLANOT because it is automatically executed for you. If instead, you are plotting by individually executing routines in the Utility Plotting Program, use PLANOT whenever you need to annotate the plot. For example, you may want to annotate a new sheet of paper before you begin plotting on it. Also, remember to execute PLANOT whenever you reset XMIN, XMAX, YMIN, or YMAX. Otherwise, the scale on your plot will be incorrect.

> **Note:** Depending upon your plotting operations, it may be necessary to execute PLINIT before executing PLANOT. For further information, refer to The PLINIT Routine, page 38.

PLANOT Operation

PLANOT annotates a plot by performing the following:

- Selects pen 2.
- Uses line type 1 (refer to the LTYPE function, page 89).
- Frames the plotting area.
- Uses the ANNOT parameter to determine the number of major and minor tics on each axis.
- Uses a major tic length of 2% of the appropriate axis, and a minor tic length of 1% of the appropriate axis. (For information concerning tic length calculations, refer to the TICLEN function on page 100 and the PINIT function on page 68.)
- Labels major tics on each axis in the default or specified format indicated in the annotation parameter.

For more detailed information concerning PLANOT, refer to the PLANOT flowchart on page 179 and the PLANOT listing, which begins at line 474 of the Utility Plotting Program listing on page 165.

Increasing Your Plotting Skills

The Utility Plotting Program has numerous applications. Enhancements to these applications are also possible through the programs listed on pages 167 and 168, as well as through other programs you may wish to write. The material in this section can help you get started. However, one of the best ways to learn how to get the most performance from the Utility Plotting Program is to experiment with various combinations of parameters in the plotting data base. This will help you to gain deeper insights into how these parameters control plotting and, if you use the plotter module regularly, to develop an intuitive knowledge of the parameters needed to control any plotting situation.

This is the end of part I. If you wish to learn how the individual plotter module functions operate, turn to part II. If you wish to generate bar code either on your plotter or on the HP 82162A Thermal Printer, turn to section 7, Bar Code, which begins on page 108.

Part II
Plotter Module Functions

Section 3

The Plotting Area and Scale

Contents

Introduction 64
Graphic Limits 64
Graphic Scale and GUs 64
Plot Bounds 66
User Scale and UUs 66
GU Mode and UU Mode (SETGU , SETUU) 67
Initializing and Clearing the I/O Buffer (PINIT , PCLBUF) 68
Specifying Graphic Limits (LIMIT , RATIO) 69
Setting the User Scale (SCALE) 71
Revising the Plot Bounds (LOCATE , CLIPUU , UNCLIP) 73
Page Advance (GCLEAR) 75
Saving Steps 76
Initializing the HP-41 for Examples 76
Terminating Examples 77

Introduction

This section explains plotting boundary and scale concepts, and describes the plotter module functions you can use to implement these concepts.

Graphic Limits

A plotter has physically imposed limitations that restrict the region in which it can plot. For example, the HP 7470A Plotter is limited by the pen-carriage and paper transport mechanisms. Because of these physical limitations, the plotter module defines limits that restrict plotter operations. All pen movement is restricted to the area inside these *graphic limits*—sometimes called the "hard-clip" area.

Whenever a plotter is initialized, such as when it's turned on, it sets its graphic limits to its internal default values and stores these values internally. (Refer to the owner's manual for your plotter to determine its graphic limits.)

You can decrease (or slightly increase) the default graphic limits. This enables you to change the margin around the entire plot. You can change the graphic limits in two ways:

- Using the plotter module, as described in this section.
- Using the plotter's keyboard. This requires moving the pen to the lower left corner point P1 and pressing the appropriate keys (refer to your plotter owner's manual), then moving the pen to the upper right corner point P2 and again pressing the plotter keys. The points P1 and P2 define the graphic limits and a rectangular graphic area within.

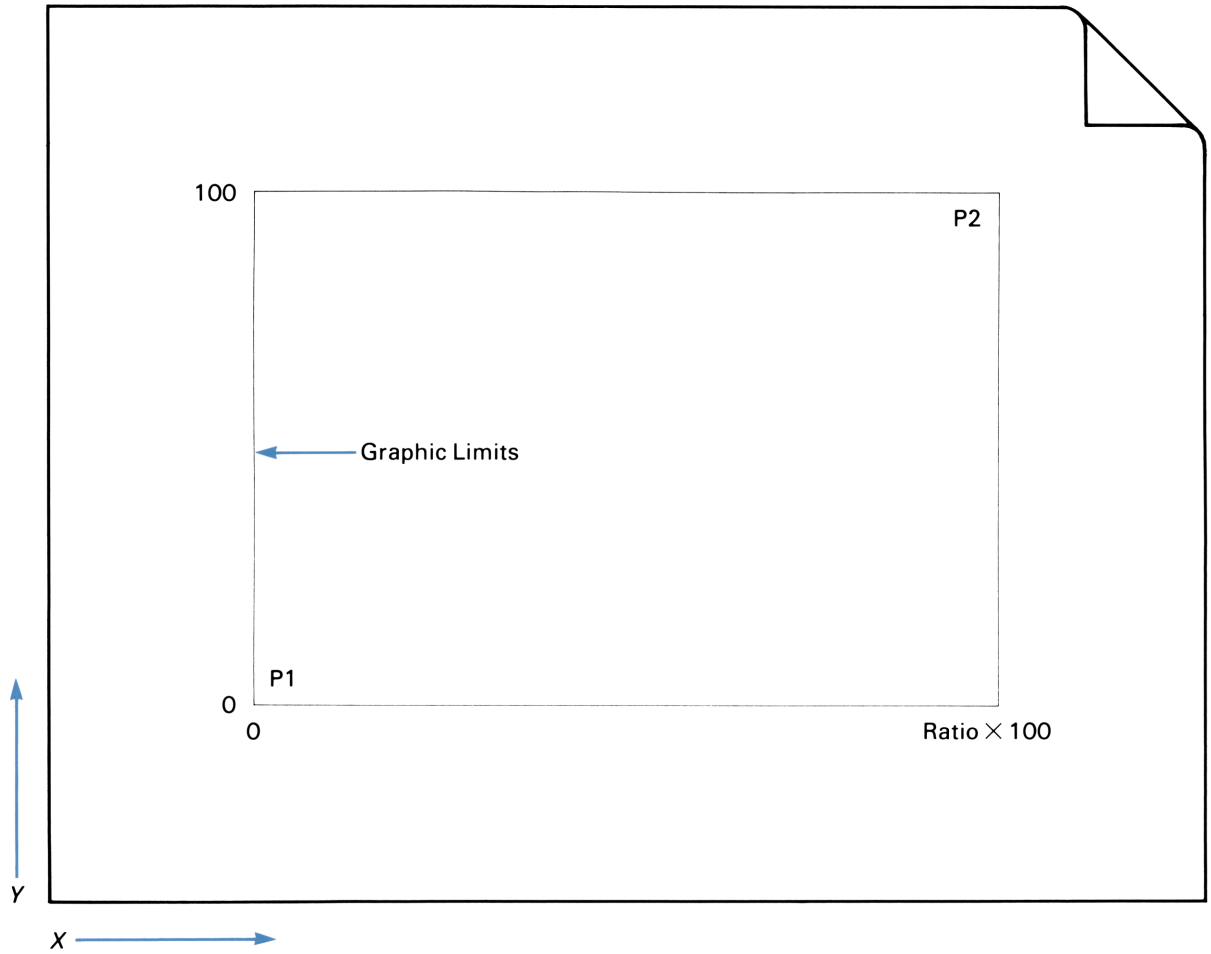
The revised limits are stored internally by the plotter.

Graphic Scale and GUs

Each point within the graphic limits can be represented by a pair of coordinates relative to the lower-left corner of this area. The *graphic scale* determines how such coordinates are measured within the graphic limits—it determines units with which you can measure distances.

The graphic area is characterized in terms of two dimensions: the x-dimension (horizontal) and the y-dimension (vertical). These dimensions are measured relative to the lower-left corner, which has the coordinates (0,0).

Regardless of the size of the graphic area, the shorter dimension is always scaled from 0 to 100 units. This dimension may be the x dimension or the y dimension, depending upon the shape of the graphic area. The longer dimension is scaled using units of the same size—naturally, there are more units along the longer dimension.



These units, used to scale the graphic area, are called *graphic units*, or simply GUs. GUs are useful for locating points using a percentage basis, since there are always 100 GUs in one dimension.

The ratio R is defined for the graphic area as the number of units in the x-dimension divided by the number of units in the y-dimension. This ratio, which can be computed using the plotter module, is helpful for finding the longer dimension of the graphic area.

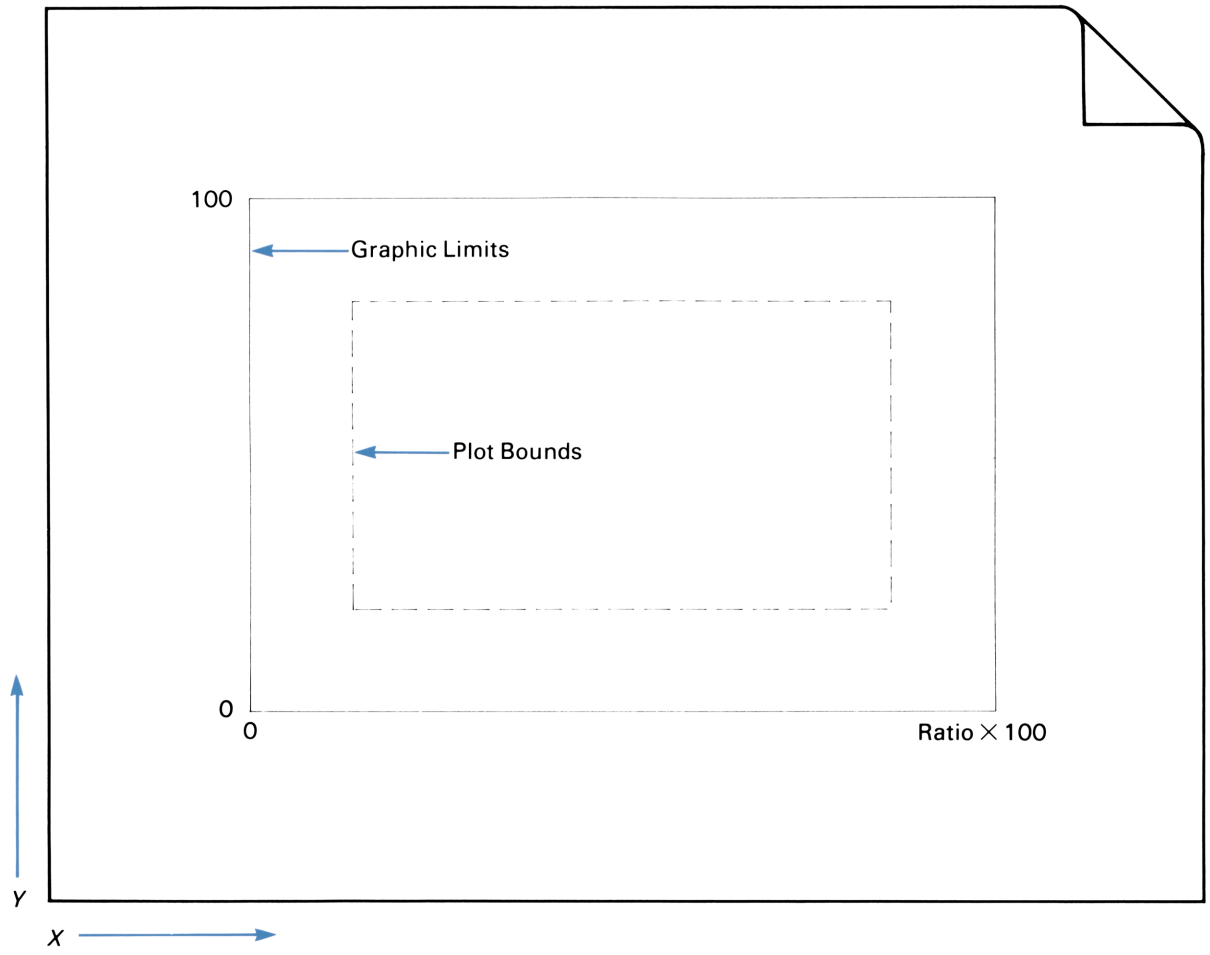
If the Longer Dimension Is	Then the Ratio Is	And the X-Dimension Is	And the Y-Dimension Is
$x > y$	R > 1	100 * R	100
$y > x$	R < 1	100	100/R
$x = y$	R = 1	100	100

For example, the graphic limits of the HP 7470A Plotter are 0 to 138.8888 GUs in the x-direction and 0 to 100 GUs in the y-direction when the plotter is first turned on. Thus, R is equal to 1.3888. You can specify any valid point using x and y coordinates within these ranges. Of course, if you redefine the graphic limits, both the ratio and the number of GUs along each axis may change.

Plot Bounds

The constraints imposed by the graphic limits can be further restricted for plotting operations. This enables you to reserve space within the graphic area for labels, for example.

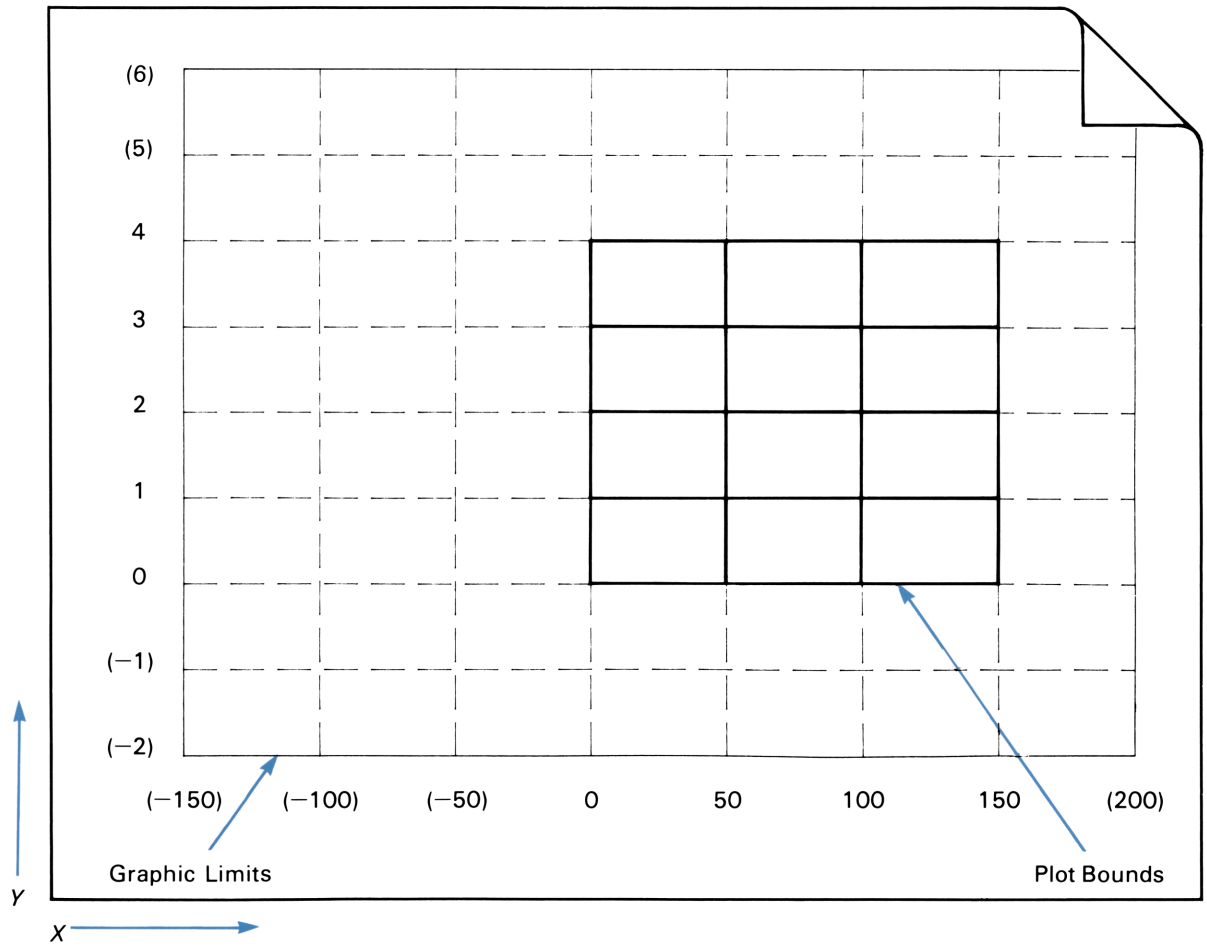
You can use the *plot bounds* to define a rectangular area that represents the usable plotting area—sometimes called the "window" or the "soft-clip" area. This area is intended for plotting, such as drawing lines and axes. The pen can move outside the plot bounds only for operations such as writing labels. (The HP-41 can never move the pen outside the graphic limits.)



User Scale and UUs

You can create your own units of measure that apply to the area inside the plot bounds. This enables you to scale this area using units that are appropriate for the values being plotted. For example, if you're plotting height versus weight, you might define the vertical scale of the plot bounds to be 0 to 4 (meters) and the horizontal scale to be 0 to 150 (kilograms).

The units that define the *user scale* are called *user units*, or UUs. Although you can't use the user scale to plot outside the plot bounds, the user scale does extend to the graphic limits, enabling you to specify label positions. If you don't define UUs for the plotting area, they remain equal to the range of GUs that fall within the plot bounds.



The UUs defined for the plot area remain in effect even if you change the plot bounds. In effect, the user scale remains the same over the entire graphic area—only the plot bounds shift. However, you can redefine the user scale for the new bounds.

GU Mode and UU Mode

Although you may have defined plot bounds, the use of those plot bounds is optional. You can still choose to plot and label anywhere within the graphic limits instead. Note that the plot bounds may coincide with the graphic limits, such as when you redefine the graphic limits. The plot bounds will never be outside the graphic limits.

GU mode is the operating mode in which you can access the entire area inside the graphic limits. Points are located using GUs.

UU mode is the operating mode in which you normally access only the area inside the plot bounds. Points are located using UUs. This is the mode that the plotter module starts in when it's initialized, as discussed in the next topic.

You can switch between GU mode and UU mode whenever you want to. This enables you to move the pen anywhere within the graphic limits (using GUs) or to limit pen movement (except for labeling) to within the plot bounds (using UUs).

Five functions in the plotter module automatically set the plotting mode. (These functions are described later in this section.) The table below shows how the plotting mode is affected by these functions.

Function	Sets UU Mode	Sets GU Mode
PINIT	X	
LIMIT	X	
SCALE	X	
SETUU	X	
SETGU		X

The last two functions in the preceding table affect only the plotting mode. They enable you to select the mode that is most useful for the operation being performed.

SETGU

The SETGU (*set GU mode*) function switches the plotter module to GU mode. SETGU is useful when you want to move the pen outside of the plot bounds or when you want to control pen movement using the GU scale instead of the UU scale.

SETUU

The SETUU (*set UU mode*) function switches the plotter module to UU mode. SETUU is useful when you want to restrict most pen movement to within the plot bounds and when you want to control pen movement using the UU scale instead of the GU scale.

When the plot bounds and the graphic limits are the same and their respective scales are equal, executing SETGU or SETUU has no effect on plotting operations.

The following functions (described in later sections) operate and interpret parameters according to the current mode—either GU mode or UU mode.

DGTIZE	IPLOT	PLOT	XAXIS
DRAW	LXAXIS	PLREGX	XAXISO
FRAME	LYAXIS	RPLOT	YAXIS
IDRAW	MOVE	WHERE	YAXISO
IMOVE			

Initializing and Clearing the I/O Buffer

As described under Memory Requirements and the I/O Buffer, page 10, the plotter module maintains an I/O (input/output) buffer in HP-41 memory. This buffer stores information used for plotting.

PINIT

The PINIT (*plotter initialize*) function reserves 26 unused memory registers for the I/O buffer if it does not already exist, and initializes the I/O buffer. If the I/O buffer does not exist, you must create it, by executing PINIT once, before you execute most other plotter module functions. (If the I/O buffer does not exist when needed, the HP-41 displays **PL:PLS PINIT**.)

Whenever you turn on the plotter, it stores default graphic limits in its internal memory.(If new limits are defined, these are stored instead of the default limits.) PINIT reads the graphic limits from the plotter's memory—either its default values or other values stored there by previous operations—and stores them in the I/O buffer.

In addition, PINIT sets the following default conditions (which are described in more detail in later sections of this manual):

- Sets the plot bounds equal to the graphic limits.
- Sets UUs equal to GUs and sets the HP-41 to UU mode.
- Selects pen 1.
- Sets the line type to type 1.
- Sets the character space height to 3 GUs.
- Sets the label origin to position 1 and the label direction to 0°.
- Sets your plotter's default tic length for the *x*- and *y*-axes.
- Sets the plotting rotation to 0°.
- Sets the parameters for plotting bar code to their default values if the buffer is being created (that is, if it did not already exist when PINIT was executed)—otherwise, the parameters are not changed.

Thus, executing PINIT alone does not affect the graphic limits, but does reset other plotting parameters to their default conditions.

Because the I/O buffer uses 26 memory registers, if the buffer does not exist when you execute PINIT *and* there are less than 26 unused registers available, the HP-41 displays **PL:NO ROOM**. If this occurs, refer to Ensuring that Enough Unused Registers Are Available, page 11.

The 26 registers reserved by PINIT and used by the plotter module can be returned to program memory by executing the following function.

PCLBUF

The PCLBUF (*clear plotter buffer*) function clears the I/O buffer, returning the 26 memory registers used by the buffer to available program memory.

(Additional information about the I/O buffer is provided in section 8, Plotting Conditions and the Input/Output Buffer.)

Specifying the Graphic Limits

You can change the graphic limits at any time using the LIMIT function. This enables you to work with several successive graphic areas on a single sheet or to use differently sized areas on successive sheets.

LIMIT

T	*x-minimum* (mm)
Z	*x-maximum* (mm)
Y	*y-minimum* (mm)
X	*y-maximum* (mm)

The LIMIT function replaces the current graphic limits with the graphic limits specified in the X-, Y-, Z-, and T-registers. (The limits maintained by the plotter—P1 and P2—and those in the I/O buffer are changed.) In addition, LIMIT sets the default conditions listed above for PINIT—except that bar code parameters aren't changed.

The HP-41 always interprets LIMIT parameters as millimeters, measured from the origin point (0, 0) at the lower-left physical limit of the plotter. The LIMIT parameters should be within the physical limits of the plotter since the plotter can't move outside its physical limits. For example, the physical limits for the

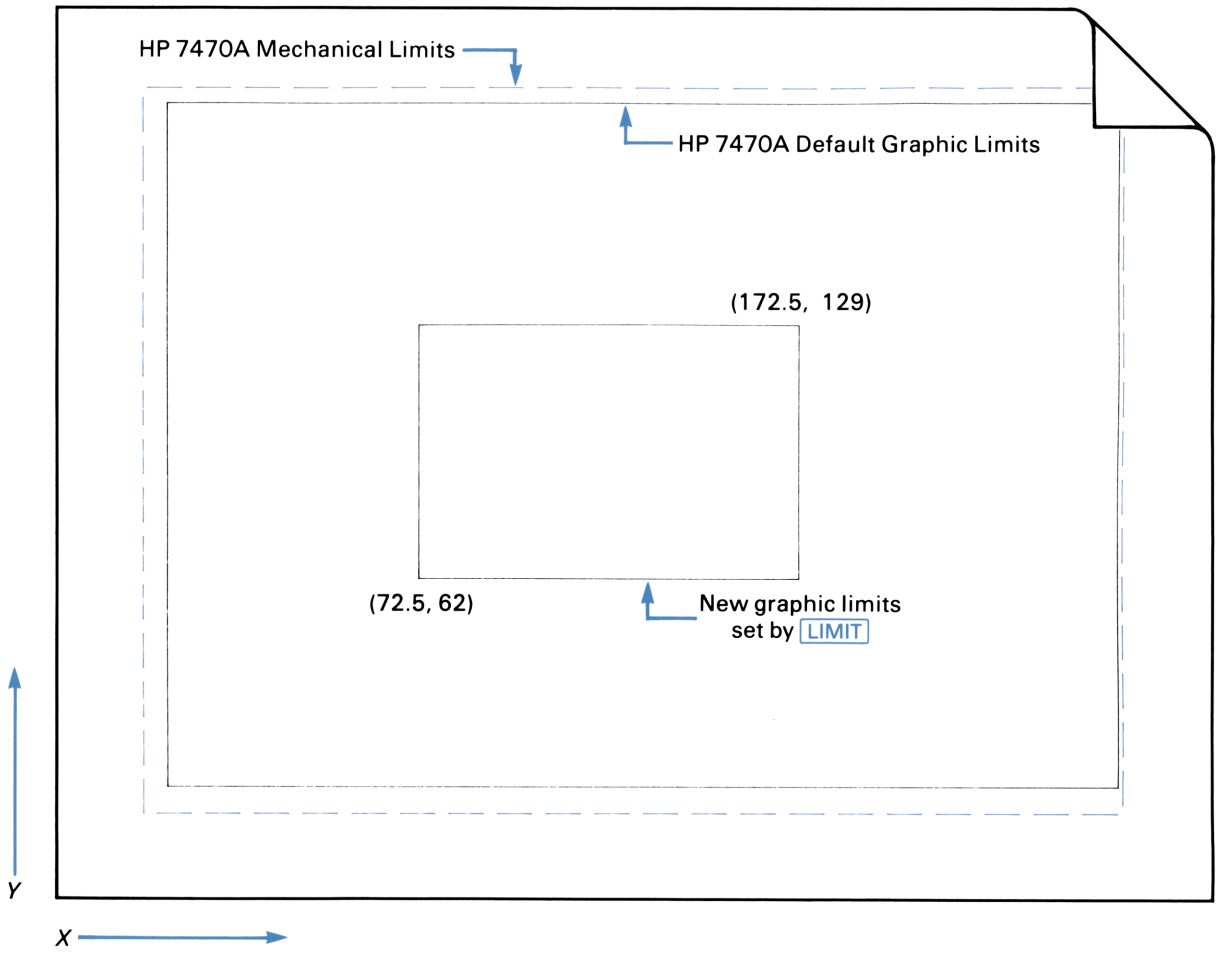
HP 7470A Plotter are 257.5 mm in the x direction and 191.25 mm in the y direction for "US" paper (272.5 mm × 191.25 mm for "A4" paper). When the HP 7470A Plotter is first turned on, the graphic limits stored internally are slightly less than the physical limits (6 to 256 mm in the x direction and 7 to 187 mm in the y direction.)

Once LIMIT specifies the graphic limits, these limits do not change unless you perform one of the following operations:

- Execute LIMIT with a new set of parameters.
- Manually set new limits using the plotter's keys, then execute PINIT.
- Turn the plotter off, then on, and execute PINIT.

Note: When the plotter has been turned off, then on, its internal limit points are reset to their default values. The next execution of PINIT resets the graphic limits maintained in the I/O buffer to the plotter's default values.

Example: Specify graphic limits having a 100-millimeter x-dimension, a 67-millimeter y-dimension, and a lower-left corner positioned 72.5 millimeters to the right of, and 62 millimeters above, the (0,0) mechanical limit of your plotter. Then use the FRAME function (discussed in section 4) to plot a frame, or outline, at the new graphic limits.



Keystrokes	Display	
PINIT		Initializes the buffer. (If the I/O buffer does not already exist, requires 26 unused memory registers.)
72.5 ENTER↑	72.5000	Enters *x*-minimum.
172.5 ENTER↑	172.5000	Enters *x*-maximum.
62 ENTER↑	62.0000	Enters *y*-minimum.
129	129_	Enters *y*-maximum.
LIMIT	129.0000	Sets the new graphic limits.
FRAME	129.0000	Frames the new graphic limits.

The frame drawn by your plotter should look like the one shown in the preceding illustration. (These new graphic limits are used in the next example.)

If you reverse the positions of *x*-minimum and *x*-maximum or *y*-minimum and *y*-maximum in the stack, LIMIT defines the corresponding graphic and user scales in the reverse direction (right-to-left or top-to-bottom).

RATIO

The RATIO function calculates the *x*-to-*y* ratio R—truncated to four decimal places—of the area within the graphic limits and enters the ratio into the X-register. That is, the ratio is equal to the number of units in the *x*-dimension divided by the number of units in the *y*-dimension. (You can work with either millimeters or GUs when calculating the ratio R.)

RATIO can be used find the length in GUs of the long side of the graphic limits. For instance, RATIO returns a value of $R = 1.3888$ (or 250 mm/180 mm) when the module is used with an HP 7470A Plotter set to its default limits. Thus, the maximum *x*-value (the longer dimension) for these limits is $100 \times R$, or 138.88 GUs. Remember that the length of the shorter side is always equal to 100 GUs. (Refer to the chart on page 65.)

Example: Find the ratio for the graphic limits set in the previous example and compute the length of the longer side in GUs. In this case, the longer side of the plotting area is the *x* side. Because the shorter side is always equal to 100 GUs, the length (in GUs) of the *x*-dimension is $100 \times R$. (The following keystrokes assume that the graphic limits defined in the previous example remain in memory.)

Keystrokes	Display	
RATIO	1.4925	The ratio of the *x*-dimension to the *y*-dimension.
100 X	149.2500	The length, in GUs, of the *x*-dimension.

RATIO can be particularly useful in determining the length of the longer side after you manually set the graphic limits using the plotter's keyboard.

Setting the User Scale

When the HP-41 is initialized for plotting by PINIT, the HP-41 is set to UU mode and the plot bounds define the current plotting area. (The plot bounds are set by PINIT and LIMIT and can be revised by several functions discussed under the next heading.) The user scale, which is associated with the plot bounds, is initially set to be the same as the graphic scale that lies within the plot bounds.

Often you want a user scale (UUs) that is more convenient than the graphic scale (GUs). The SCALE function enables you to define a user scale that is appropriate for your application.

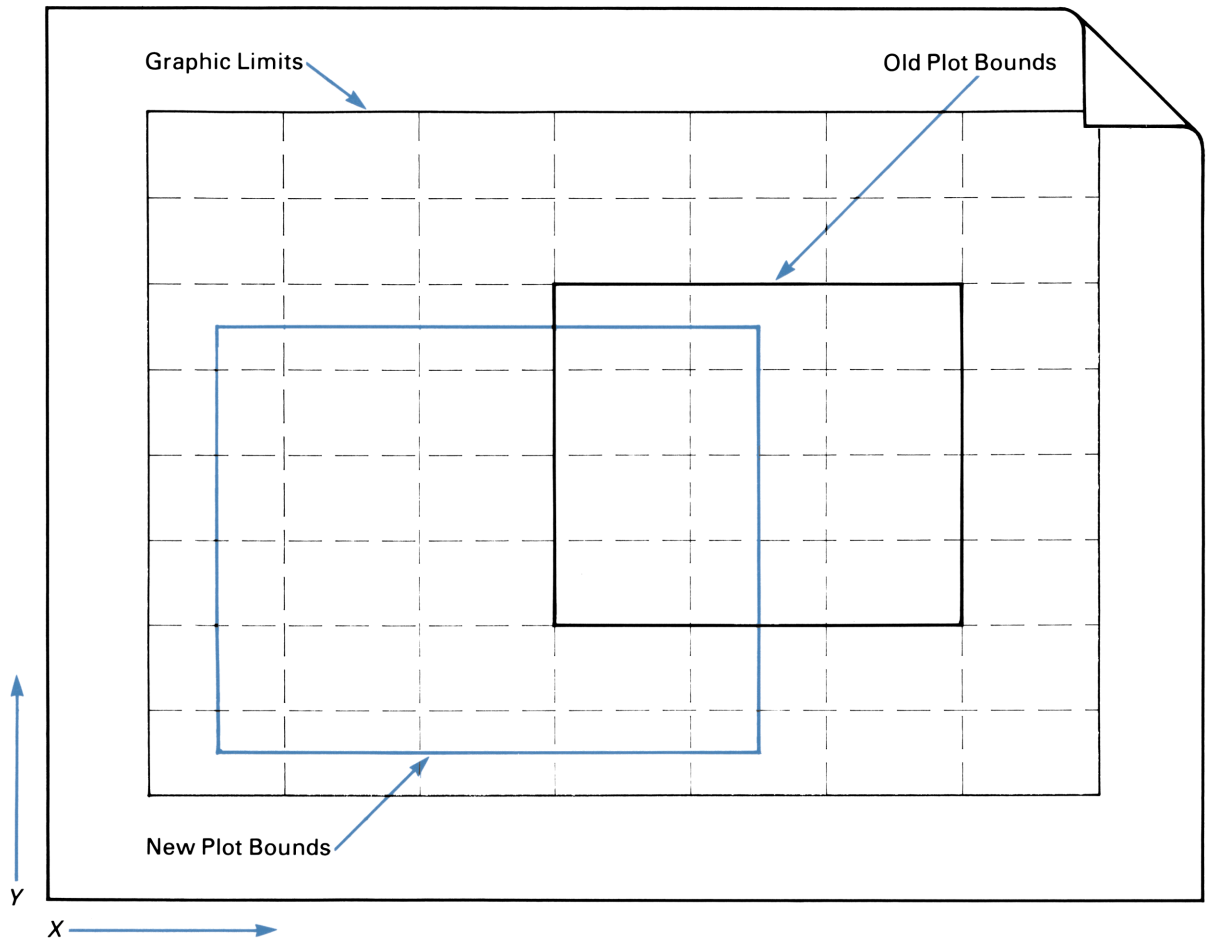
SCALE

T	*x-minimum* (UUs)
Z	*x-maximum* (UUs)
Y	*y-minimum* (UUs)
X	*y-maximum* (UUs)

The SCALE function sets the user scale from the x- and y-dimensions of the plot bounds and sets the plotter module to UU mode. The scaling for the x- and y-directions are independent of each other. Thus, plots are stretched or shrunk independently in the x- and y-directions to fit the plotting area. (This is sometimes called "anisotropic scaling.")

SCALE is particularly convenient for plots where the lengths of the x- and y-units differ. In the program for the annual rainfall plot (page 15), SCALE is used to specify years on the x-axis and inches on the y-axis. (Refer to program lines 33 through 40 in the listing of this program on page 161).

Although the user scale is defined by the x and y values at the plot bounds, the scale extends to the graphic limits—primarily for locating labels. If the plot bounds—but not the graphic limits—are subsequently revised (using functions described in the next topic), the user scale remains unchanged; the portion of the scale inside the new plot bounds becomes the usable plotting scale. Of course, you can use SCALE to define a new user scale for the new plot bounds.



The user scale—whether the default user scale or a scale you set using SCALE—remains in memory until it is cleared by one of the following operations:

- You execute SCALE to specify another user scale.
- You execute PINIT or LIMIT, which resets the graphic limits, plot bounds, and scales.

Note: Using the plotter keys to change P1 and P2 does not change the scale, but does affect the results of some plotting operations. Thus, if you use the plotter keys to change P1 and P2, it is recommended that you then reinitialize the I/O buffer by executing PINIT.

If the x or y parameters are reversed in the stack registers, the corresponding scale will be inverted relative to the normal left-to-right or bottom-to-top direction.

An example that uses SCALE is included in the next topic, which discusses revising (and scaling) the plot bounds.

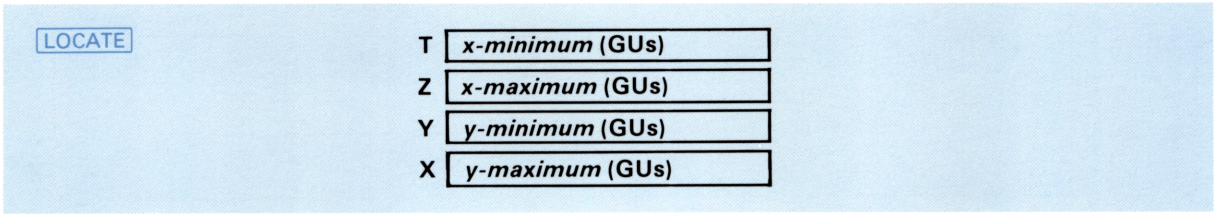
Revising the Plot Bounds

Whenever you create new graphic limits (by using PINIT or LIMIT), the plot bounds are set equal to the new limits. The plotter module provides you with three functions you can use for revising the plot bounds, enabling you to specify plot bounds anywhere within the graphic limits.

The first function, LOCATE, uses GUs to define the bounds (that is, it uses the graphic scale). The second function, CLIPUU, uses UUs to define the bounds (that is, it uses the user scale). Note that the graphic and user scales may be the same, as they are after PINIT or LIMIT is executed; or the scales may be different, as they would be if you defined a user scale (using SCALE). The third function, UNCLIP, sets the plot bounds at the graphic limits. These functions are described below.

LOCATE is often used to establish plot bounds *prior* to specifying a user scale, while CLIPUU is often used to establish plot bounds *after* a user scale has been defined using the SCALE function.

There are many uses for specially-sized plot bounds. Two of the most frequent uses are to allow space outside of a plot for labeling or to create a window effect to show only part of a plot.

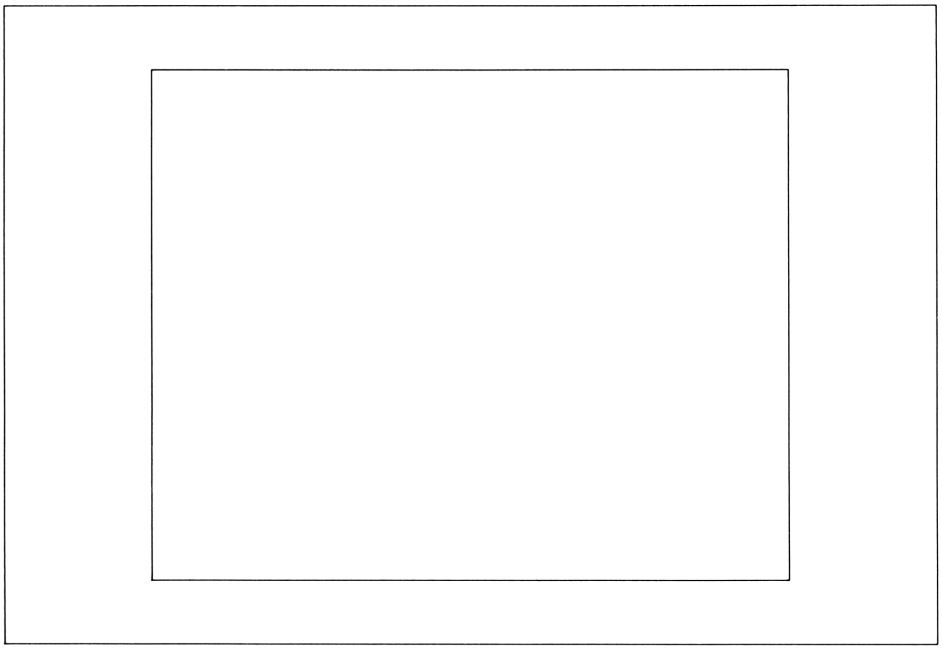


The LOCATE (*locate plot bounds*) function allows you to specify plot bounds that are anywhere within the graphic limits. The LOCATE parameters are interpreted as GUs. LOCATE does not change the user scale and can be executed while the plotter is in either UU mode or GU mode.

Example: Define graphic limits that are from 0 to 246 millimeters in the x-direction and from 16 to 184 millimeters in the y-direction. (The GU scale for these graphic limits is 146.2 GUs in the x-direction and 100 GUs in the y-direction.) Then define the plot bounds to be 100 GUs by 80 GUs and centered within the graphic limits.

Keystrokes	Display	
PINIT		Initializes the I/O buffer and sets UU mode. (If the I/O buffer does not already exist, requires 26 unused memory registers.)
0 ENTER↑ 246 ENTER↑ 16 ENTER↑ 184 LIMIT	246.0000 16.0000 184.0000	Defines the graphic limits.
FRAME	184.0000	Frames the graphic limits.
RATIO	1.4642	Displays GU ratio of x-dimension to y-dimension.
100 X	146.4200	Length in GUs of the x-dimension.
100 −	46.4200	Portion of x-dimension outside of desired plot bounds.

Keystrokes	Display	
2 [÷]	23.2100	Computes *x*-minimum for plot bounds.
[ENTER↑] [ENTER↑]	23.2100	Copies *x*-minimum in Y and Z.
100 [+]	123.2100	Computes *x*-maximum.
10 [ENTER↑] 90	90 _	Enters *y*-minimum and *y*-maximum for desired plot bounds.
[LOCATE]	90.0000	Sets plot bounds.
[FRAME]	90.0000	Frames the plot bounds.
0 [PEN]	0.0000	Returns pen to stall.



[CLIPUU]

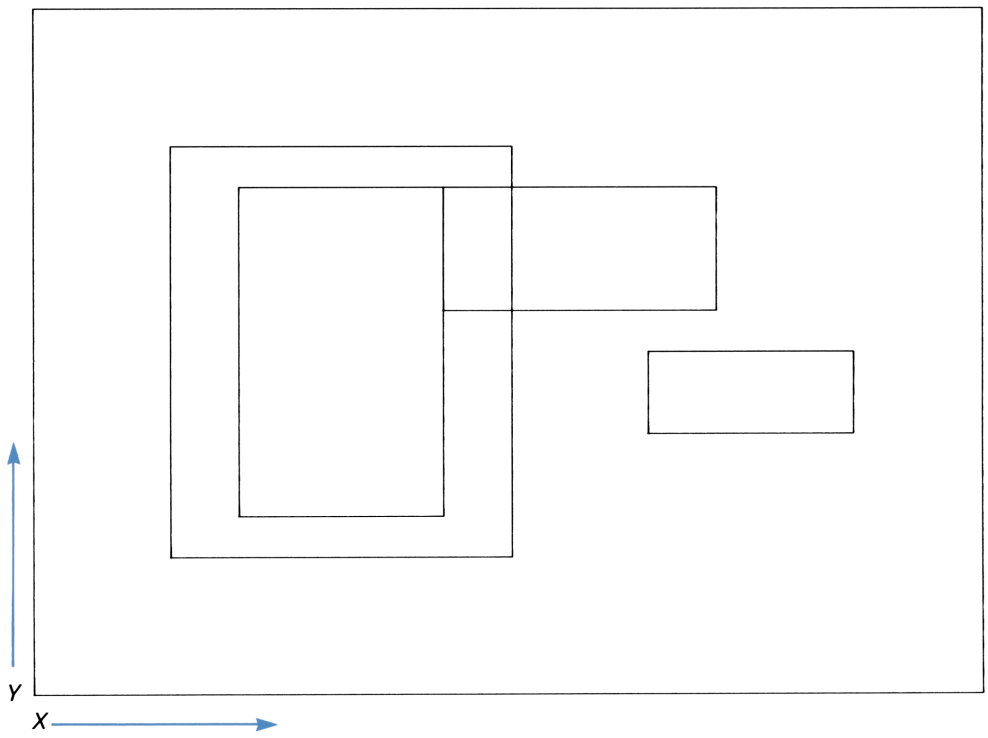
T	*x-minimum* (UUs)
Z	*x-maximum* (UUs)
Y	*y-minimum* (UUs)
X	*y-maximum* (UUs)

The [CLIPUU] (*clip-UUs*) function operates in the same way as the [LOCATE] function except that the parameters are interpreted as UUs—whether or not the HP-41 is currently set to UU mode.

[UNCLIP]

The [UNCLIP] function resets the plot bounds to match the graphic limits—the largest plot bounds possible without redefining the graphic limits. The user scale is not changed.

Example: Set the graphic limits to the default values by turning the plotter off and on, then executing [PINIT]. Use [LOCATE] and [SCALE] to create plot bounds and scale them. Then use [CLIPUU] to create various new plot bounds using the user scale. Finally reset the plot bounds to the graphic limits.



Keystrokes	Display	
PINIT		Initializes I/O buffer. (If the I/O buffer does not already exist, requires 26 unused memory registers.)
20 ENTER↑ 70 ENTER↑	70.0000	Revises and frames the plot bounds.
20 ENTER↑ 80	80 _	
LOCATE FRAME	80.0000	
0 ENTER↑ 5	5 _	Specifies user scale.
ENTER↑ 0 ENTER↑	0.0000	
10 SCALE	10.0000	
1 ENTER↑ 4 ENTER↑	4.0000	Revises and frames plot bounds based on user scale set in preceding step.
1 ENTER↑ 9	9 _	
CLIPUU FRAME	9.0000	
4 ENTER↑ 8 ENTER↑	8.0000	Revises and frames plot bounds based on user scale (outside plot bounds).
6 ENTER↑ 9	9 _	
CLIPUU FRAME	9.0000	
7 ENTER↑ 10 ENTER↑	10.0000	Revises and frames plot bounds.
3 ENTER↑ 5	5 _	
CLIPUU FRAME	5.0000	
UNCLIP FRAME	5.0000	Revises plot bounds to graphic limits and frames them.

Page Advance

GCLEAR

The GCLEAR (*graphics clear*) advances the page on plotters that have page feed mechanisms. GCLEAR has no effect on the operation of a plotter that does not have a page feed—such as the HP 7470A Plotter.

Saving Steps

The subroutines under the following two headings are designed to save you time by providing a standardized setup and termination for many of the examples in this manual. You may want to design similar subroutines to keep available in program memory for use in your plotting applications.

Initializing the HP-41 for Examples

Most of the remaining examples in this manual assume that the HP-41 is initialized to the plotting conditions set by the following SET program. To ensure that the desired conditions are active, these examples include execution of SET. To save you time and to help focus your attention on the functions illustrated in the examples, enter this program into your HP-41 now so that you can easily execute it when needed to set up your plotting system. (Bar code for SET is on page 205.)

Keystrokes	Display	
■ GTO · · PRGM	00 REG *nn*	Packs program memory. Switches HP-41 to program mode. If ***nn*** is less than 08, you must create additional memory space before entering SET.
■ LBL ALPHA SET	01 LBL SET _	
ALPHA	01 LBL"SET	
PINIT	02 PINIT	Initializes I/O buffer.*
6	03 6 _	Sets graphic limits.
ENTER↑	04 ENTER↑	
256	05 256 _	
ENTER↑	06 ENTER↑	
7	07 7 _	
ENTER↑	08 ENTER↑	
187	09 187 _	
LIMIT	10 LIMIT	
FRAME	11 FRAME	Frames graphic limits.
10	12 10 _	Sets plot bounds.
ENTER↑	13 ENTER↑	
90	14 90 _	
ENTER↑	15 ENTER↑	
10	16 10 _	
ENTER↑	17 ENTER↑	
90	18 90 _	
LOCATE	19 LOCATE	
CLST	20 CLST	Sets user scale.
100	21 100 _	
STO · Z	22 STO Z	
SCALE	23 SCALE	
FRAME	24 FRAME	Frames plot bounds.
■ FIX 0	25 FIX 0	Sets FIX 0 display.
DEG	26 DEG	Ensures Degrees mode.
■ CLX	27 CLX	Clears X-register.
END	00 REG *nn*	
PRGM		Removes HP-41 from Program memory.

Following execution of SET, the HP-41 is in UU mode with FIX 0 display and Degrees mode.

*As indicated in the discussion of PINIT on page 68, PINIT uses the graphic limits to which your plotter is currently set. Also, if the I/O buffer does not already exist when you execute SET, 26 unused memory registers must be available for PINIT to use for creating the buffer.

Terminating Examples

After you complete an example, you will usually want the plotter to display your work, return the pen to its stall, and return plotting parameters to their default values. The following TERM (for "termination") program performs these steps. Like the SET program, TERM is called as a subroutine by several of the program examples listed elsewhere in this manual. (Bar code for TERM is on page 205.)

Keystrokes	Display	
■ GTO · · PRGM	00 REG *nn*	Packs program memory. Switches HP-41 to Program mode. If ***nn*** is less than 03, you must create additional memory space before entering TERM.
■ LBL ALPHA TERM	01 LBL TERM _	
ALPHA	01 LBL"TERM	
PINIT	02 PINIT	Initializes the I/O buffer.
0	03 0 _	
ENTER↑	04 ENTER ↑	Displays plot.
MOVE	05 MOVE	
PEN	06 PEN	Returns pen to stall.
■ FIX 4	07 FIX 4	Sets FIX 4 display.
END	00 REG *nn*	
PRGM		Removes HP-41 from Program mode.

Section 4

Plotting

Contents

Framing the Plotting Area (FRAME) 78
Moving the Pen and Drawing Lines 78
Moving and Drawing to a Point (MOVE , DRAW) 78
Moving and Drawing by Increments (IMOVE , IDRAW) 79
Moving and Drawing Outside the Plotting Area 80
Using Plot-Option Functions 80
Pen Status 80
The Plot-Option Functions (PLOT , IPLOT , RPLOT , PLREGX) 81
Rotating the Plot Axes (PDIR) 86
Other Pen Control Functions (PEN , PENDN , PENUP , LTYPE , LTYPEO) 88

Framing the Plotting Area

FRAME

The FRAME function draws a box around the active plotting area. That is, if GU mode is active, FRAME draws a box at the graphic limits. If UU mode is active, FRAME draws a box at the plot bounds.

Moving the Pen and Drawing Lines

The following four functions control the basic operations of moving the pen—with or without drawing a line. They automatically ensure that the pen is "down" or "up" according to the operation you specify.

Moving and Drawing to a Point

MOVE Y *y-coordinate* (UUs or GUs)
X *x-coordinate* (UUs or GUs)

The MOVE function moves the pen to the specified (x,y) coordinate position without drawing a line. The coordinates are interpreted as the current units—either UUs or GUs. In UU mode the pen moves only within the plot bounds. In GU mode the pen moves anywhere within the graphic limits.

DRAW Y *y-coordinate* (UUs or GUs)
X *x-coordinate* (UUs or GUs)

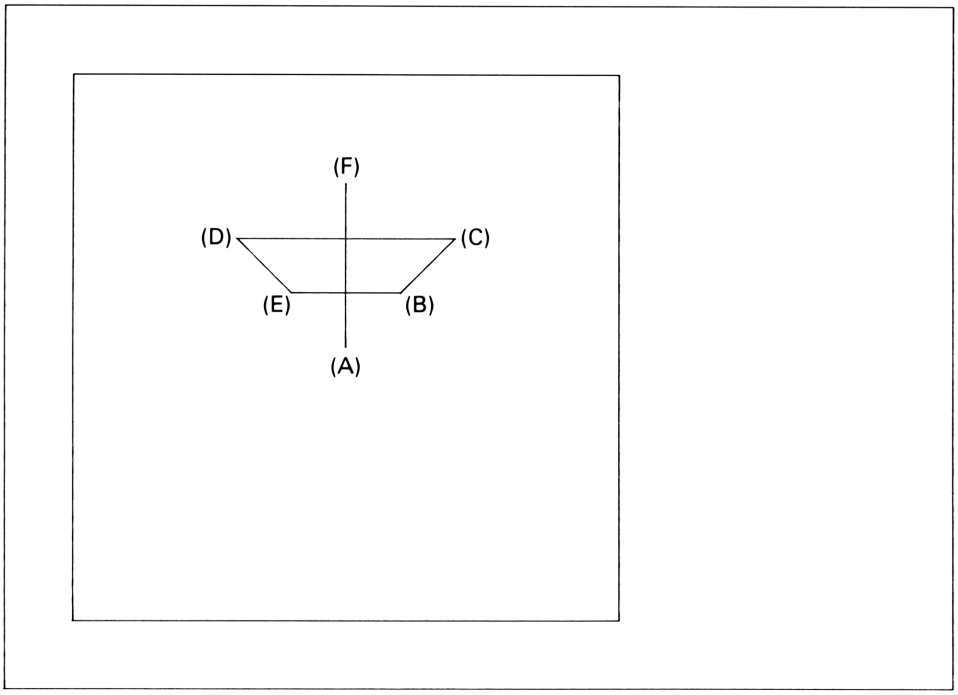
The DRAW function draws a line from the current pen position to the specified (x,y) coordinate position, then lifts the pen. The coordinates are interpreted as the current units—either UUs or GUs. The line type conforms to the currnt line type and repeat interval. When UU mode is active, you can draw lines only within the plot bounds. When GU mode is active, you can draw lines anywhere in the graphic limits.

Moving and Drawing by Increments

IMOVE	Y	*y-increment* (UUs or GUs)
	X	*x-increment* (UUs or GUs)
IDRAW	Y	*y-increment* (UUs or GUs)
	X	*x-increment* (UUs or GUs)

The IMOVE (*incremental move*) and IDRAW (*incremental draw*) functions operate in the same way as MOVE and DRAW except that pen travel is specified by x- and y-increments. The origin—point (0,0)—is assumed to be the most recent pen position. The direction of incremental pen travel can be rotated using PDIR. (Refer to Rotating the Plot Axes, page 86.)

Example: Use IMOVE and IDRAW to plot the following simple diagram where, after you initially position the pen, the coordinates of each pen movement are expressed in increments from the most recently specified pen position.



Keystrokes	Display	
XEQ ALPHA SET ALPHA	XEQ SET_ 0.	Initializes HP-41 and frames plotting area. (Refer to Initializing the HP-41 for Examples, page 76.)
50 ENTER↑ MOVE	50.	Moves pen to center of plotting area (point A).
10 ENTER↑	10.	Specifies a point B 10 units above and to the right of the point A and moves pen to point B.
IMOVE	10.	

Keystrokes	Display	
IDRAW	10.	Draws a line to a point (point C) that is 10 units above and to the right of point B.
0 ENTER↑ 40 CHS IDRAW	-40_ -40.	Specifies point D 40 units to the left of point C and draws line to point D.
10 CHS ENTER↑ CHS IDRAW	10. 10.	Specifies point E 10 units below and to the right of point D and draws line to point E.
0 ENTER↑ 20 IDRAW	20_ 20.	Specifies point B, which is 20 units to the right of point E and draws a line to point B.
10 CHS ENTER↑ IMOVE	-10. -10.	Specifies point A, which is 10 units below and to the left of point B, and moves pen to point A.
30 ENTER↑ 0 IDRAW	0_ 0.	Specifies point F 30 units above point A and draws line to point F.
XEQ ALPHA TERM ALPHA	XEQ TERM_ 0.0000	Terminates example. (Refer to Terminating Examples, page 77.)

In the preceding examples you have seen how the pen operates *inside* the plotting area. Before continuing with pen control functions, let's discuss pen movement to or from points *outside* the plotting area.

Moving and Drawing Outside the Plotting Area

The plotter module controls pen travel during "move" and "draw" operations so that it remains within the current plotting area—the graphic limits in GU mode or the plot bounds in UU mode. If you specify points that lie outside the plotting area, pen travel accurately reflects the portion of the intended travel that falls within the plotting area.

For example, if you execute a "move" or "draw" function using coordinates that are outside the plotting area, the pen travels to the point at the edge of the area in the direction of the point, but halts when it reaches the edge. If you then execute a "move" or "draw" function to a point within the plotting area, the pen first travels to the appropriate point at the edge where the line from the distant point to the new point should reenter the plotting area, then travels to the new point. If both points are outside the plotting area, the pen travels only on the portion of the line that lies within the plotting area.

If you use the HP-41 to specify a point outside the plotting area, repositioning the pen using the plotter's keyboard does not change the pen position maintained in the I/O buffer. That is, if you reposition the pen using the plotter's keyboard, then execute a "move" or "draw" function, the pen returns to the last position you specified using the HP-41, then executes the function.

Using Plot-Option Functions

There are four plot-option functions: PLOT, IPLOT, RPLOT, and PLREGX. These functions draw a line or simply move the pen according to the current line type and pen status.

Pen Status

The plotter module maintains an internal pen status to control the pen's "up" or "down" position during periods of pen activity. For the pen movement functions described previously, you need not be concerned with the current pen status—these functions automatically change the pen status as required.

If the pen is "down" when pen activity halts, the module temporarily lifts the pen to prevent the ink from blotting the paper—but doesn't change the internal pen status from "down." When pen activity resumes, the module reinstates the "down" status. (That is, the pen is lowered.)

The four plot-option functions described on the following pages use the pen status remaining from the preceding operation to determine whether to draw a line or to simply move the pen to the next point. If you use these functions, you will want to know how functions affect the pen status.

The following functions set the pen status to "down":

PENDN	PLOT	RPLOT
DRAW	IPLOT	PLREGX
IDRAW		

All other functions that use the pen set the pen status to "up." PINIT sets the pen status to "up ." Other functions that do not cause pen movement, such as SCALE and RATIO, are neutral. (A pen status indicator is maintained in the I/O buffer—refer to section 8.)

The pen status remaining from the last pen movement command affects the way the plotter responds to a new plot-option command:

- If the pen status is "down," the plotter draws a line to the specified point(s) and halts. The pen status remains "down" (although the pen is lifted to prevent blotting).
- If the pen status is "up," the plotter moves the pen to the specified point without drawing a line, sets the pen status to "down," and drops the pen (although the pen is lifted again to prevent blotting).

If the plot-option function is PLREGX, a series of points can be connected (that is, several lines drawn) with one execution of the function. If the pen status is "up" when PLREGX is executed, no line is drawn when the pen moves to the first point in the series—but at that point the pen status is set to "down."

The Plot-Option Functions

The four plot-option functions share these common properties:

- They define the specified position using current units—either UUs or GUs.
- They move the pen from its present position to the specified position using the existing pen status—"up" or "down."
- They set the pen status to "down" when they reach the point—except as provided by PLREGX.
- They move the pen only within the current plotting area—either the plot bounds (in UU mode) or the graphic limits (in GU mode).

The functions differ in the ways that they specify the points and the number of points that can be specified.

PLOT

Y *y-coordinate* (UUs or GUs)
X *x-coordinate* (UUs or GUs)

When you execute PLOT, the pen moves or draws to the actual (x,y) coordinate position.

IPLOT

Y *y-increment* (UUs or GUs)
X *x-increment* (UUs or GUs)

The IPLOT (*incremental plot*) function specifies pen movement by x- and y-increments. The origin (both increments 0) is the current pen position—the origin changes every time the pen moves.

The direction of incremental pen movement can be rotated using [PDIR] (page 86).

[RPLOT]	Y	*y-relative* (UUs or GUs)
	X	*x-relative* (UUs or GUs)

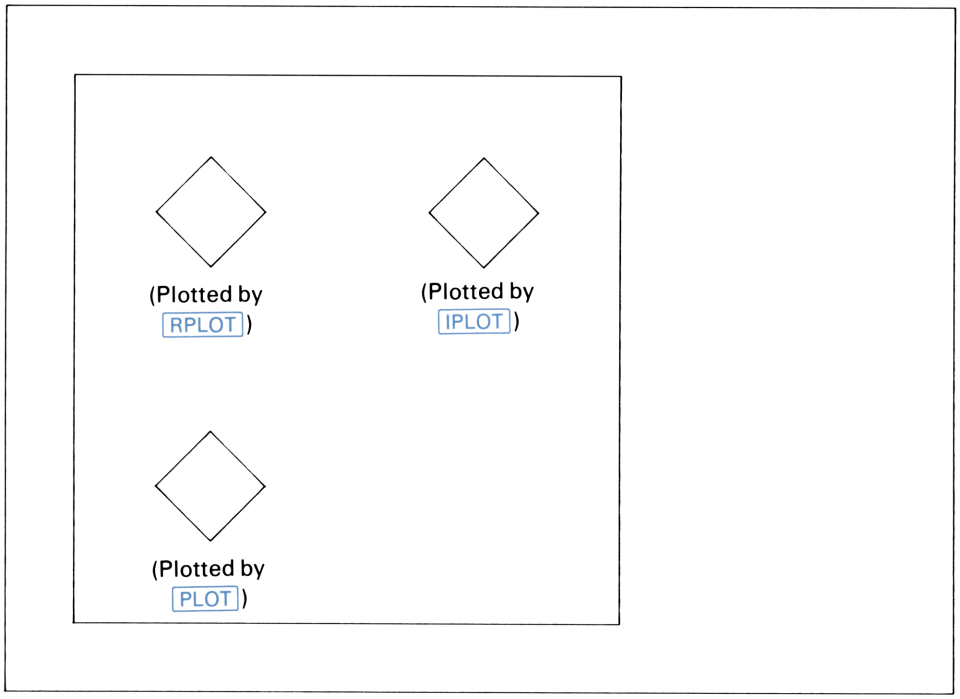
The [RPLOT] (*relative plot*) function specifies a point using its position relative to an assumed origin. The assumed origin is the last position determined by a function other than [RPLOT]—that is, [RPLOT] *doesn't* change the assumed origin, but any other pen movement *does* change the origin.

The direction of a relative pen movement can be rotated using [PDIR] (page 86).

Example of Absolute, Incremental, and Relative Plotting. Draw a diamond shape in each of three corners of the plot bounds using [PLOT], [IPLOT], and [RPLOT].

Keystrokes	Display	
[XEQ] [ALPHA] SET [ALPHA]	XEQ SET _ 0.	Initializes HP-41 (pen "up") and frames plotting area. (Refer to Saving Steps, page 76.)
25 [ENTER↑] 15 [PLOT]	15.	Moves pen to point (15, 25) and changes status to "down."
[x≷y] [PLOT]	25.	Specifies point (25, 15) and draws line, leaving pen "down."
35 [PLOT]	35.	Specifies point (35, 25) and draws line, leaving pen "down."
[x≷y] [PLOT]	25.	Specifies point (25, 35) and draws line, leaving pen "down."
15 [PLOT]	15.	Specifies point (15, 25) and draws line, leaving pen "down."
75 [ENTER↑] [MOVE]	75.	Moves pen to center point (75, 75) and sets pen status to "up."
5 [CHS] [ENTER↑] 0 [IPLOT]	0.	Moves pen an increment (0, − 5) and changes status to "down."
10 [ENTER↑] [IPLOT]	10.	Specifies increment (+ 10, + 10) and draws line, leaving pen "down."
[CHS] [IPLOT]	−10.	Specifies increment (− 10, + 10) and draws line.
[ENTER↑] [IPLOT]	−10.	Specifies increment (− 10, − 10) and draws line.
[CHS] [IPLOT]	10.	Specifies increment (+ 10, − 10) and draws line.
75 [ENTER↑] 25 [MOVE]	25.	Moves pen to center point (75, 25), establishes assumed origin for [RPLOT], and sets pen status to "up."
10 [CHS] [ENTER↑] 0 [RPLOT]	0.	Moves pen to relative position (0, − 10) and changes status to "down."
[x≷y] [CHS] [RPLOT]	10.	Specifies relative position (10, 0) and draws line, leaving pen "down."
[x≷y] [RPLOT]	0.	Specifies relative position (0, 10) and draws line.
[x≷y] [CHS] [RPLOT]	−10.	Specifies relative position (− 10, 0) and draws line.
[x≷y] [RPLOT]	0.	Specifies relative position (0, − 10) and draws line.

Save this plot for the next example.



PLREGX X *iii.fff* R_{iii} x_1 (UUs or GUs) ⋮ R_{fff} y_n (UUs or GUs)

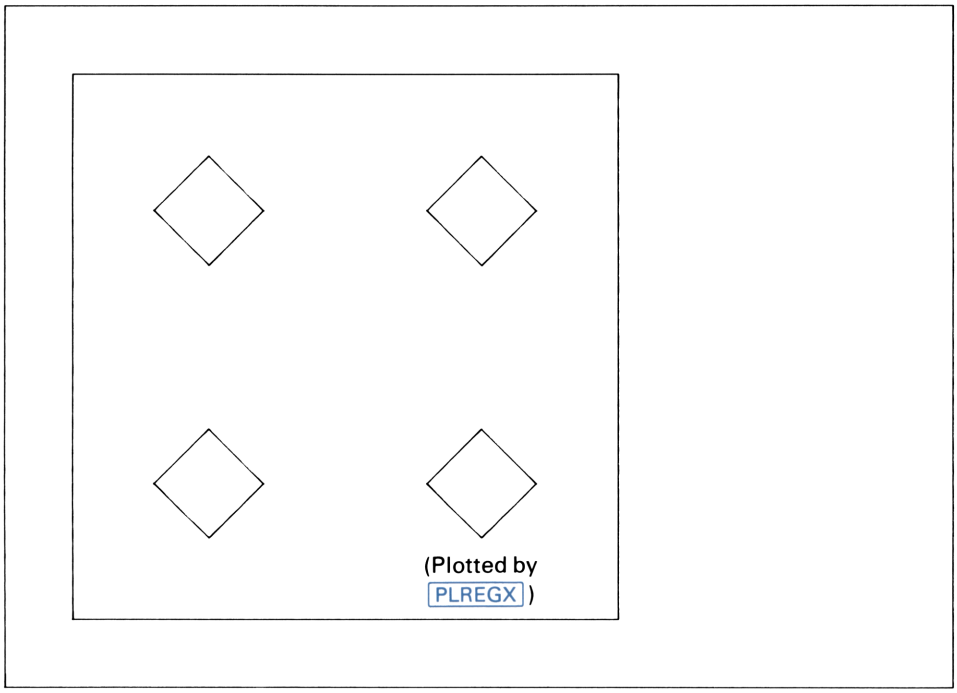
The PLREGX (*plot registers according to X*) function enables you to plot several lines (x, y coordinate pairs) from data you have previously stored in a series of data storage registers. This allows the plotting to be performed relatively fast since only one plotting function is performed. Coordinates are interpreted as current units—either UUs or GUs.

You specify the series of registers by placing in the X-register a number in the form ***iii.fff***, where ***iii*** indicates the initial register and ***fff*** indicates the final register. PLREGX interprets the values in the first and second registers of the series (R_{iii} and R_{iii+1}) as the x- and y-coordinates of the first point (P_1) of your plot. The values in the third and fourth registers of the series are interpreted as the x- and y-coordinates of the second point (P_2), and so on.

Diamond Example Continued. Use PLREGX to draw a diamond shape in the unused corner of the plot remaining from the preceding example. Use the data storage registers shown below.

R_{00}	75	x_1	Identifies first point.
R_{01}	15	y_1	
R_{02}	85	x_2	Identifies second point.
R_{03}	25	y_2	
R_{04}	75	x_3	Identifies third point.
R_{05}	35	y_3	
R_{06}	65	x_4	Identifies fourth point.
R_{07}	25	y_4	
R_{08}	75	x_1	Identifies first point (closes diamond).
R_{09}	15	y_1	

Keystrokes	Display	
75 [STO] 00	75.	Specifies first point. (Fix 0 display remains from execution of SET in preceding example.)
15 [STO] 01	15.	
85 [STO] 02	85.	Specifies second point.
25 [STO] 03	25.	
75 [STO] 04	75.	Specifies third point.
35 [STO] 05	35.	
65 [STO] 06	65.	Specifies fourth point.
25 [STO] 07	25.	
75 [STO] 08	75.	Specifies first point again.
15 [STO] 09	15.	
25 [ENTER↑] 75 [MOVE]	75.	Moves pen to center point of fourth diamond (25, 75) and sets pen status to "up."
.009 [PLREGX]	9. −03	Specifies R_{00} through R_{09} as registers containing coordinates of points to plot. Pen moves to first point, then sets pen status to "down" for remaining points.
[XEQ] [ALPHA] TERM [ALPHA]	0.0000	Terminates example. (Refer to Terminating Examples, page 77.)



If the pen status is set to "up" when [PLREGX] is executed, the pen moves to the first point, changes the pen status to "down," and begins plotting. If the pen status is set to "down" when [PLREGX] is executed, the pen status remains "down"—it draws a line from its current position to the first point, then plots the specified points. In other words, as soon as the pen moves to the first point, [PLREGX] sets and leaves the pen status as "down."

However, [PLREGX] provides a method for lifting the pen between points. If an Alpha string occupies either register representing a coordinate pair, that coordinate pair is ignored and the pen lifts—although the

internal pen status remains "down." The pen moves, in the "up" position, to the next point, then drops in preparation for plotting the next line. Note that an Alpha string causes the pen to move without drawing a line, but that the pen status remains "down."

Example: Draw a line from (0, 0) to (10, 20) to (20, 30) and a line from (30, 30) to (43, 35) to (60, 70). Use the storage registers shown below.

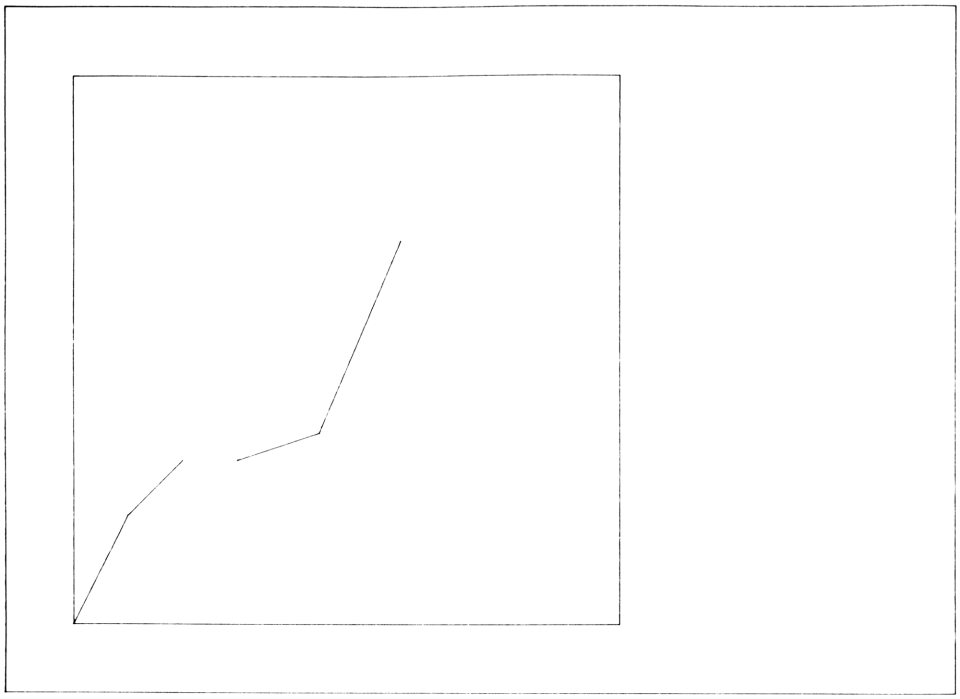
R_{01}	0	x_1	First point.
R_{02}	0	y_1	
R_{03}	10	x_2	Second point.
R_{04}	20	y_2	
R_{05}	20	x_3	Third point.
R_{06}	30	y_3	
R_{07}	A		Alpha character causes pen to lift and move to next point.
R_{08}	Ignored		
R_{09}	30	x_4	Fourth point.
R_{10}	30	y_4	
R_{11}	45	x_5	Fifth point.
R_{12}	35	y_5	
R_{13}	50	x_6	Sixth point.
R_{14}	30	y_6	

Keystrokes	Display	
0 STO 01	0.0000	Stores point 1 coordinates.
STO 02	0.0000	
10 STO 03	10.0000	Stores point 2 coordinates.
20 STO 04	20.0000	
STO 05	20.0000	Stores point 3 coordinates.
30 STO 06	30.0000	
ALPHA A	A_	Stores an Alpha character in R_{07}.*
■ ASTO 07	A	
ALPHA	30.0000	
STO 09	30.0000	Stores point 4 coordinates.
STO 10	30.0000	
45 STO 11	45.0000	Stores point 5 coordinates.
35 STO 12	35.0000	
60 STO 13	60.0000	Stores point 6 coordinates.
70 STO 14	70.0000	

Now place a new sheet of paper in your plotter and use PLREGX with registers R_{01} through R_{14} to draw the lines.

*Refer to the Alpha keyboard illustration on the HP-41's back label.

Keystrokes	Display	
XEQ ALPHA SET ALPHA	XEQ SET _ 0.	Initializes the HP-41 and sets the pen status to "up." (Refer to Initializing the HP-41 for Examples, page 76.)
1.014	1.014 _	Specifies R_{01} through R_{14}.
PLREGX	1.0140	Plots lines without connecting points 3 and 4.
XEQ ALPHA TERM ALPHA	XEQ TERM _ 0.0000	Terminates example. (Refer to Terminating Examples, page 77.)

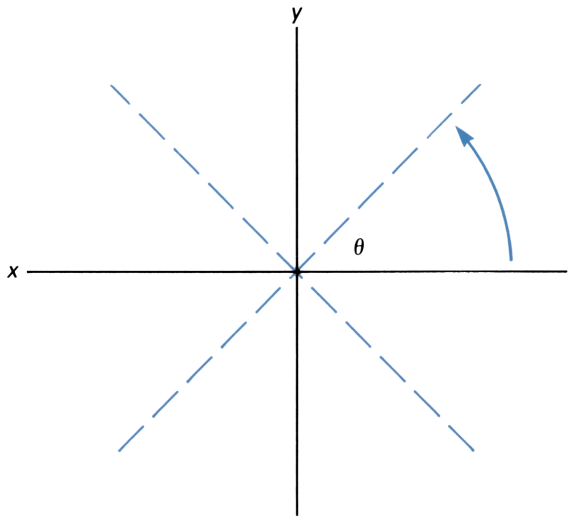


For another application of PLREGX, refer to the RAIN Program listed on pages 160 and 161. Lines 06 through 21 use a loop for programmed data input. The PLREGX function in line 84 of that program uses the input at line 83 for the necessary ***iii.fff*** number.

Rotating the Plot Axes

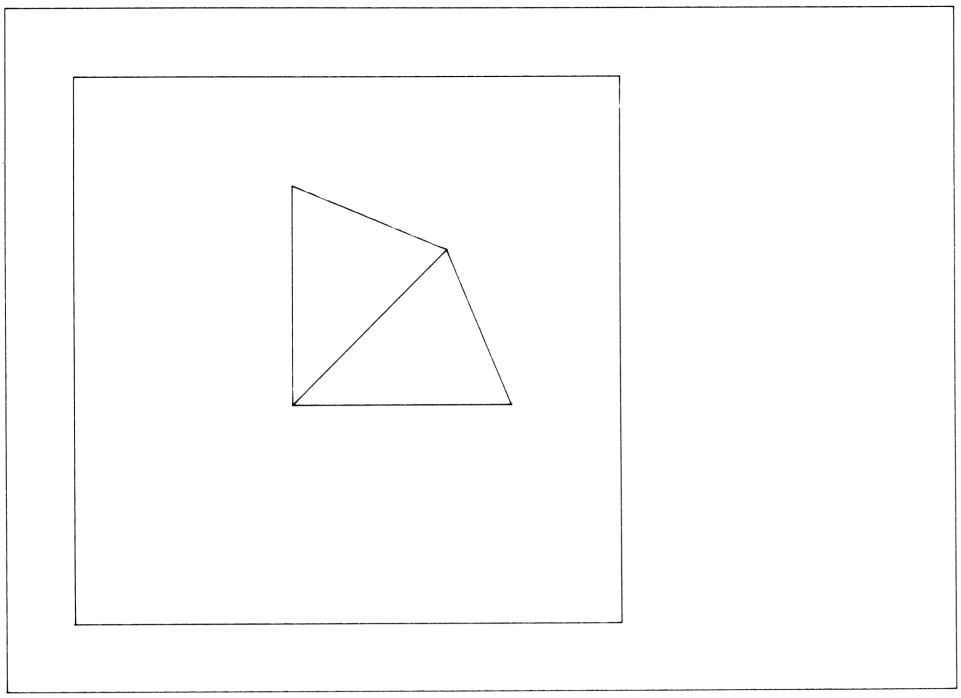
PDIR X *angle*

The PDIR (*plot direction*) function rotates the x- and y-axes to the specified angle and sets the new direction for incremental plotting (IMOVE, IDRAW, and IPLOT) and relative plotting (RPLOT) only. PDIR does not affect pen movement specified by absolute coordinates. The angle is specified as positive in the counterclockwise direction and is interpreted according to the HP-41's current angular mode (DEG, RAD, or GRAD).



The angles of all successive [PDIR] functions refer to the original, unrotated axes, not to a previously rotated pair of axes. The default angle (set by [PINIT] and [LIMIT]) is 0°.

Example: Use [PDIR] to plot the figure below.



Keystrokes	Display	
[XEQ] [ALPHA] SET [ALPHA]	XEQ SET_ 0.	Initializes HP-41 (pen "up" and zero rotation angle) and frames plotting area. (Refer to Initializing the HP-41 for Examples, page 76.)
5 [CHS] [ENTER↑] 10 [ENTER↑]	10.	Defines user scale.
5 [CHS] [ENTER↑]	-5.	
10 [SCALE]	10.	

Keystrokes	Display	
1 ENTER↑ PLOT	1.	Moves pen to point (1, 1). (Sets pen status to "down" and defines assumed origin for RPLOT.)
0 ENTER↑ 6 RPLOT	6.	Plots line from assumed origin (1, 1) along current x-axis.
1 ENTER↑ PLOT	1.	Plots line to point (1, 1).
45 PDIR	45.	Rotates axes to 45° orientation.
0 ENTER↑ 6 RPLOT	6.	Plots line along rotated x-axis.
1 ENTER↑ PLOT	1.	Plots line to point (1, 1).
90 PDIR	90.	Rotates axes to 90° orientation.
0 ENTER↑ 6 RPLOT	6.	Plots line along rotated x-axis.
45 PDIR	45.	Rotates axes to 45° orientation.
0 ENTER↑ 6 RPLOT	6.	Plots line to point on rotated x-axis.
0 PDIR	0.	Rotates axes to 0° orientation.
6 RPLOT	6.	Plots line to point on rotated x-axis.
XEQ ALPHA TERM ALPHA	XEQ TERM _ 0.0000	Terminates example. (Refer to Terminating Examples, page 77.)

Notice in this example that the assumed origin remained at (1, 1), the origin established by each PLOT function.

Other Pen Control Functions

The following functions enable you to control pen conditions and the type of line the pen draws. These functions give you flexibility for plotting lines of different styles and colors.

PEN X *pen number*

The PEN (*pen*) function selects the pen indicated by the number in the X-register. When PEN is executed, the actual pen that is selected depends upon the capability of the plotter that you're using. For example, the HP 7470A Plotter selects its left pen for odd pen numbers and its right pen for even numbers. Other plotters have only one pen, which they always use regardless of the pen number. (Refer to your plotter owner's manual for information about pen selection.)

After the specified pen is retrieved, the pen returns to the position it occupied when PEN was executed.

If the pen number is 0, executing PEN returns the current pen to its stall (if the plotter has that capability). PEN uses the absolute value of the integer portion of the pen number.

Whenever you execute PINIT or LIMIT, the pen number is set to "1" and the plotter selects pen number 1.

PENDN

The PENDN (*pen down*) function lowers the pen to the plotting surface. The pen remains in contact with the surface until another function causes it to lift.

PENUP

The PENUP (*pen up*) function lifts the pen from the plotting surface.

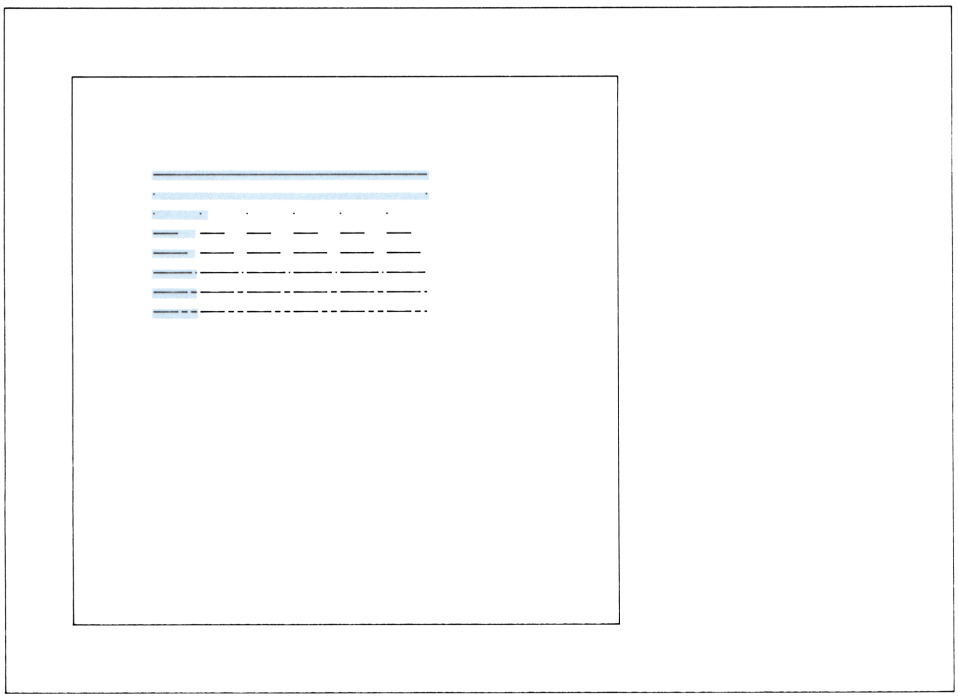
LTYPE X *line type*

The LTYPE (*line type*) function uses a line type number (integer value 1 through 8) to select one of eight solid or dashed line types for drawing or plotting. A type number of 0 yields the same line as type 1. If you specify a number larger than 8, the current line type is not changed.

Whenever you execute PINIT or LIMIT, the line type is set to type 1 (a solid line).

Example: The following program illustrates the line types selected by the type numbers and generated by the plotter. (A bar code listing of this program begins on page 99.)

Keystrokes	Display	
■ GTO · ·		Packs program memory.
PRGM	00 REG *nn*	Switches HP-41 to program mode. If ***nn*** is less than 10, you must create additional memory space before entering LINE. (Allow one data storage register—R_{00}—for use by the program.)
■ LBL ALPHA LINE	01 LBL LINE _	
ALPHA	01 LBL"LINE	Names program.
XEQ ALPHA SET ALPHA	02 XEQ"SET	Initializes HP-41. (Refer to Initializing the HP-41 for Examples, page 76.)
0	03 0 _	Specifies user scale.
ENTER↑	04 ENTER↑	
20	05 20 _	
ENTER↑	06 ENTER↑	
24 CHS	07-24 _	
ENTER↑	08 ENTER↑	
4	09 4 _	
SCALE	10 SCALE	
1.008	11 1.008 _	Stores line type/loop control counter.
STO 00	12 STO 00	
■ LBL 00	13 LBL 00	Begins line-drawing loop.
RCL 00	14 RCL 00	
INT	15 INT	Computes line type number.
CHS	16 CHS	Uses line type number for negative *y*-coordinate and 2 for *x*-coordinate, and moves pen to that position.
2	17 2 _	
MOVE	18 MOVE	
x⇄y	19 X<>Y	
CHS	20 CHS	Restores line type number.
LTYPE	21 LTYPE	Sets line type.
CHS	22 CHS	Uses pen data number to help position pen for drawing line.
3	23 3 _	
MOVE	24 MOVE	
0	25 0 _	
ENTER↑	26 ENTER↑	
10	27 10 _	
IDRAW	28 IDRAW	Draws line 10 units long.
■ ISG 00	29 ISG 00	Increments loop counter.
■ GTO 00	30 GTO 00	Returns to beginning of line-drawing loop.
1	31 1 _	Resets line type to default value.
LTYPE	32 LTYPE	
XEQ ALPHA TERM ALPHA	33 XEQ"TERM	Terminates example. (Refer to Terminating Examples, page 77.)
PRGM		Removes HP-41 from Program mode.
XEQ ALPHA LINE	XEQ LINE _	
ALPHA	0.0000	Executes LINE program to produce the following plot.



The screened portion of each line indicates the extent of the repeat pattern for each line type. LTYPE automatically uses a repeat length of 4 GUs. The next function enables you to control the length of the repeat pattern.

LYTPEO	Y	*repeat length* (GUs)
	X	*line type*

The LTYPEO (*line-type option*) function operates in the same way as LTYPE except that the number in the Y-register specifies the length of the repeat pattern as a percentage of the length of the diagonal between P1 and P2. (For an example of each line type, see the illustration above.)

The length of the repeat pattern is set to 4 GUs whenever you turn on the plotter or execute PINIT, LIMIT, or LTYPE.

Example: Alter the preceding program to generate longer repeat patterns. Then run the program and compare the result with the plot generated in the last example.

Keystrokes	Display	
■ GTO ALPHA LINE	GTO LINE _	Locate LINE program in memory.
ALPHA	0.0000	(Display from preceding example.)
PRGM	01 LBL^T LINE	
■ GTO .021	21 LTYPE	
←	20 CHS	Removes LTYPE function.
6	21 6 _	Specifies 6 GU repeat length.
x≷y	22 X<>Y	Places repeat length in Y-register and line type in X-register.
LTYPEO	23 LTYPEO	Enters LTYPEO function.
PRGM		

Before you run the program, place a clean sheet of paper in your plotter.

Keystrokes	Display	
XEQ ALPHA LINE	XEQ LINE _	
ALPHA	0.0000	Executes revised LINE program.

Section 5

Labels and Axes

Contents

Introduction 92
Using Labels 92
Printing a Label (LABEL) 92
Character Set 94
Changing the Label's Location (LORG , LDIR) 94
Changing the Character Size (CSIZE , CSIZEO) 96
Using Axes 99
Drawing an Axis (XAXIS , YAXIS , XAXISO , YAXISO) 99
Specifying Tic Marks (TICLEN) 100
Plotting a Grid 101
Drawing a Labeled Axis (LXAXIS , LYAXIS) 102

Introduction

The functions described in this section enable you to write titles or other labels on your plots, to draw axes and grids, and to label the axes. Using these functions, you can enhance your plots with important information presented in useful formats.

Using Labels

Labels can be drawn in almost any size, location, and rotation within the graphic limits, regardless of the active plotting mode—UU mode or GU mode.

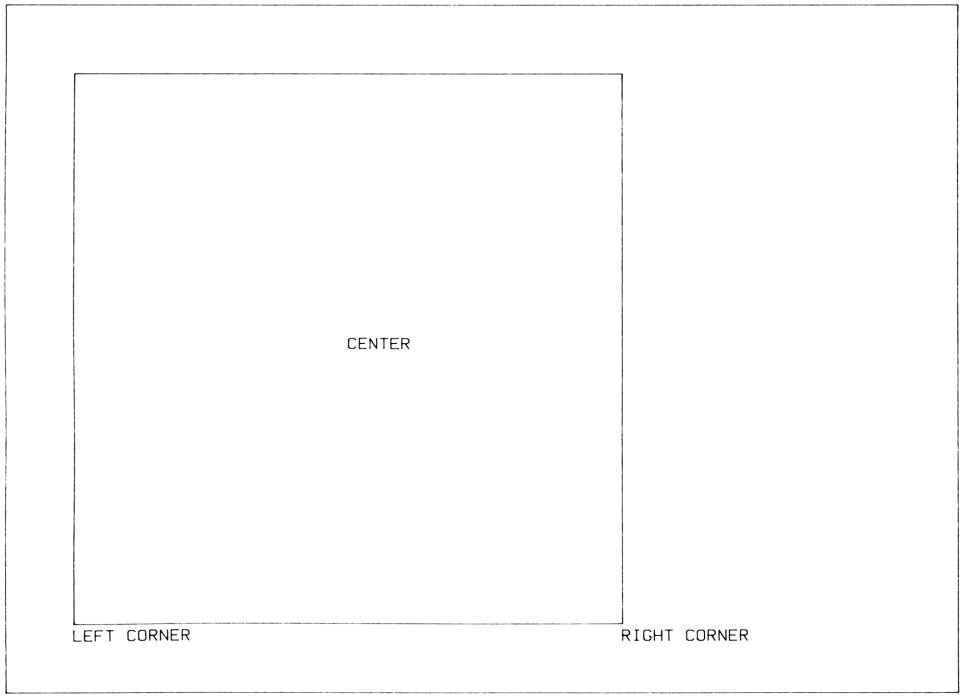
Printing a Label

LABEL **ALPHA** *alpha string*

The LABEL function prints the contents of the ALPHA register at the last pen position specified by the HP-41 and within the graphic limits. To use LABEL, first position the pen at the desired point, such as by executing one of the moving, drawing, or plot-option functions described in section 4. Then execute LABEL. (LABEL prints only at pen positions set by the HP-41; it ignores pen movement performed by the plotter's pen control keys.)

The LABEL function permits the pen to move outside the plot bounds in UU mode—although the pen can't move outside the graphic limits. This lets you print labels in the margin between the plot bounds and the graphic limits. However, you should note that when you position the pen prior to executing LABEL, it moves only as far as the plot bounds—the pen moves to the specified position when you execute LABEL (before the label is printed). The HP-41 remembers the actual point you specified for the label position. This ensures that the pen remains inside the plot bounds except for printing a label.

Example: Print labels that lie within and extend outside of the plot bounds.



Keystrokes	Display	
XEQ ALPHA SET	XEQ SET _	Initializes HP-41. (Refer to Initializing the HP-41 for Examples, page 76.)
ALPHA	0.	
50 ENTER↑ MOVE	50.	Moves pen to center of plot bounds.
ALPHA CENTER ALPHA	50.	Enters label.
LABEL	50.	Prints label.
ALPHA RIGHT SPACE CORNER	IGHT CORNER _	Enters label.
ALPHA	50.	
3 CHS ENTER↑ 100	100 _	Specifies pen movement to point outside plot bounds at lower right. Pen halts at plot bounds.
MOVE	100.	
LABEL	100.	Pen moves to specified point and prints label.
ALPHA LEFT SPACE CORNER	LEFT CORNER _	Enters label.
ALPHA	100.	
← MOVE	0.	Specifies pen movement to point outside plot bounds at lower left. Pen halts at plot bounds.
LABEL	0.	Pen moves to specified point and prints label.
XEQ ALPHA TERM	XEQ TERM _	Terminates example. (Refer to Saving Steps, page 77.)
ALPHA	0.0000	

The HP-41's flag 17 controls pen movement when the label is completed. If flag 17 is clear, the pen prints the desired label, then moves to a position one character space below the starting point of that label. If flag 17 is set, the pen prints the desired label, then stops at the first character space following the label just printed. Setting flag 17 enables you to print on a single line a label containing more than 24 characters (the size of the ALPHA register). Flag 17 is automatically cleared each time you turn on your HP-41.

Example: Use flag 17 and [LABEL] to print the long label indicated in the following keystrokes.

Keystrokes	Display	
■ [CF] 17		Ensures that flag 17 is cleared.
[XEQ] [ALPHA] SET [ALPHA]	XEQ SET _ 0.	Initializes HP-41. (Refer to Initializing the HP-41 for Examples, page 76.)
65 [ENTER↑] 25 [MOVE]	25.	Moves pen.
[ALPHA] THIS [SPACE] LABEL	THIS LABEL _	
[SPACE] HAS [SPACE] MO [ALPHA]	ABEL HAS MO _	Enters first part of label.
[LABEL]	25.	Prints label. Pen moves to next line.
■ [SF] 17	25.	Sets flag 17.
[LABEL]	25.	Prints preceding label. Pen remains on same line.
■ [CF] 17	25.	Clears flag 17.
[ALPHA] RE [SPACE] THAN [SPACE]	RE THAN _	
■ 2 ■ 4 [SPACE] CHARACTERS [ALPHA]	CHARACTERS _ 25.	Enters last part of label.
[LABEL]	25.	Prints label. Pen moves to next line.
[XEQ] [ALPHA] TERM [ALPHA]	XEQ TERM _ 0.0000	Terminates example. (Refer to Terminating Examples, page 77.)

Character Set

The HP-41 uses several characters that are not contained in the standard character set of most plotters. If any of these HP-41 characters are in the ALPHA register when [LABEL] is executed, they are printed by the plotter according to its own character set.

For example, the HP 7470A Plotter responds to the following HP-41 characters according to this table.

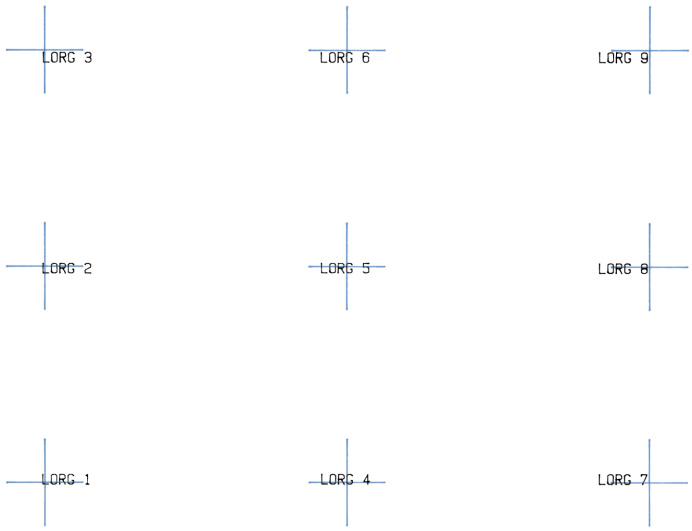
HP-41 Alpha Character	ASCII Character Code	Plotter Response
∡	13	Returns pen to beginning of same line.
≠	29	Ignored.
↑	94	Prints ^.
Σ	126	Prints ~.

Changing the Label's Location

You can use the following functions to change how the printed characters are oriented relative to the point at which the label is printed. By controlling the label's position and direction, you can locate labels in a variety of ways.

[LORG] X *label position*

The [LORG] (*label origin*) function sets the label origin position, which determines where labels are placed relative to the current pen position. The label position can have any value from 1 to 9—the value specifies whether the label is aligned at the left, right, top, or bottom, or is centered. The positions are defined in the following diagram, in which the cross represents the position specified by the HP-41 prior to executing [LABEL]. For example, a value of 8 centers the label vertically and aligns the right end at the pen position.



A value of 0 in the X-register sets the label position to 1. For noninteger values, [LORG] sets the label origin position to the unit digit of the number in the X-register (other digits, including fractions, are ignored). For integer values greater than 9, the [LORG] value is x modulo 10.

Whenever [PINIT] or [LIMIT] is executed, the label origin position is set to 1.

[LDIR] **X** ***angle***

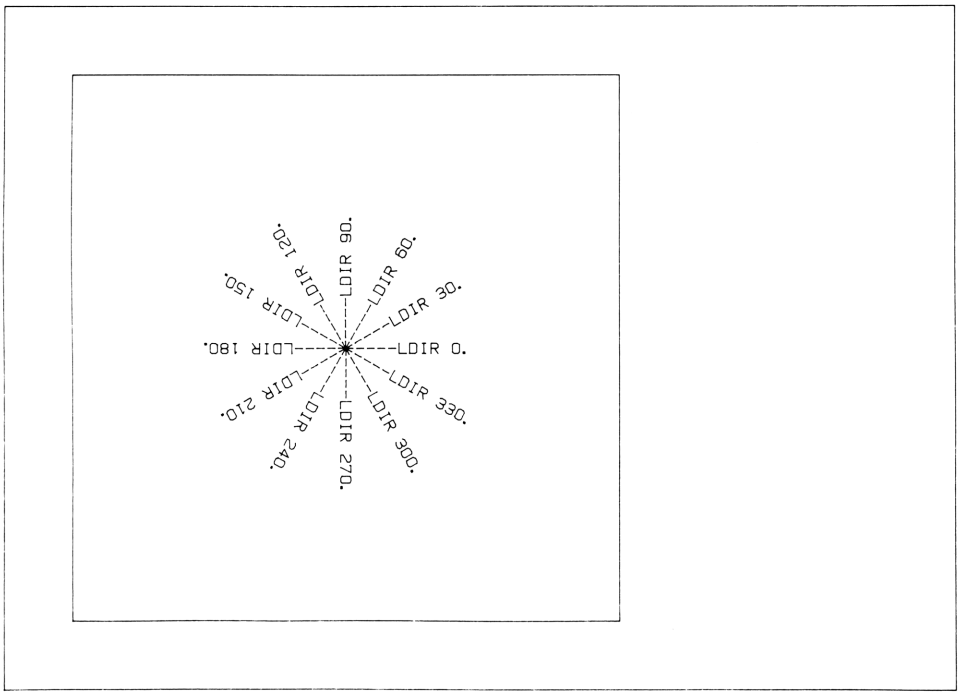
The [LDIR] (*label direction*) function sets the angle of rotation for printing labels. The angle is specified as positive in the counterclockwise direction and is interpreted according to the HP-41's current angular mode ([DEG], [RAD], or [GRAD]).

Whenever you execute [PINIT] or [LIMIT], the label direction is set to 0 degrees.

Example: Print a label at each 30° interval around a central point. (Bar code for the following program is on page 198.)

Keystrokes	Display	
■ [GTO] [·] [·]		Packs program memory.
[PRGM]	00 REG *nn*	Switches HP-41 to Program mode. If ***nn*** is less than 09, you must create additional memory space before entering LDR. (Allow at least one data storage register—R_{00}—for use by the program.)
■ [LBL] [ALPHA] LDR	01 LBL LDR _	
[ALPHA]	01 LBL"LDR	
[XEQ] [ALPHA] SET	02 XEQ SET _	Initializes HP-41. (Refer to Initializing the HP-41 for Examples, page 76.)
[ALPHA]	02 XEQ"SET	
.3303	03 .3303 _	Provides a loop counter and an [LDIR] integer.

Keystrokes	Display	
STO 00	04 STO 00	
2	05 2_	
LORG	06 LORG	Sets label origin to 2.
■ LBL 00	07 LBL 00	Begins labeling loop.
50	08 50_	
ENTER↑	09 ENTER↑	
MOVE	10 MOVE	Moves pen to point (50,50).
RCL 00	11 RCL 00	Recalls loop counter.
INT	12 INT	Truncates fractional portion.
LDIR	13 LDIR	Sets label direction.
ALPHA ■ - ■ - ■ - ■ - ■ - SPACE	14 - - - - - - - - - - - - _	Places label in ALPHA.
LDIR SPACE ALPHA	14ᵀ- - - - - - - - - - - - LDIR	
ALPHA ■ APPEND	15ᵀ⊢_	Appends label direction to label.*
■ ARCL · X ALPHA	16 ARCL X	
LABEL	17 LABEL	Prints label.
■ ISG 00	18 ISG 00	Increments loop counter.
■ GTO 00	19 GTO 00	Returns to LBL 00.
XEQ ALPHA TERM	20 XEQ TERM_	Terminates program. (Refer to Terminating Examples, page 77.)
ALPHA PRGM	20 XEQᵀTERM	
XEQ ALPHA LDR ALPHA	0.0000	Executes program.



Changing the Character Size

Printed characters are defined by several parameters that determine the actual shape and size of the characters. Also, a character is composed of both a symbol and the space that surrounds and separates it from other characters.

*Refer to Alpha keyboard illustration on the HP-41's back label.

The *height* of a character space specifies the vertical distance between the baselines of successive lines. The actual height of the printed character is one-half of the height parameter—the remaining height provides blank space between lines.

The *width* of a character space specifies the horizontal distance between the left edge of each character. The actual width of the printed character is two-thirds of the width parameter—the remaining width provides blank space between characters. (That is, the overall character space is always 1½ times as wide as the actual character width.)

The *aspect ratio* specifies the width-to-height ratio of the printed character—a small value specifies a narrow character, a large value specifies a wide character. Changing the aspect ratio changes the width of the character and the character space:

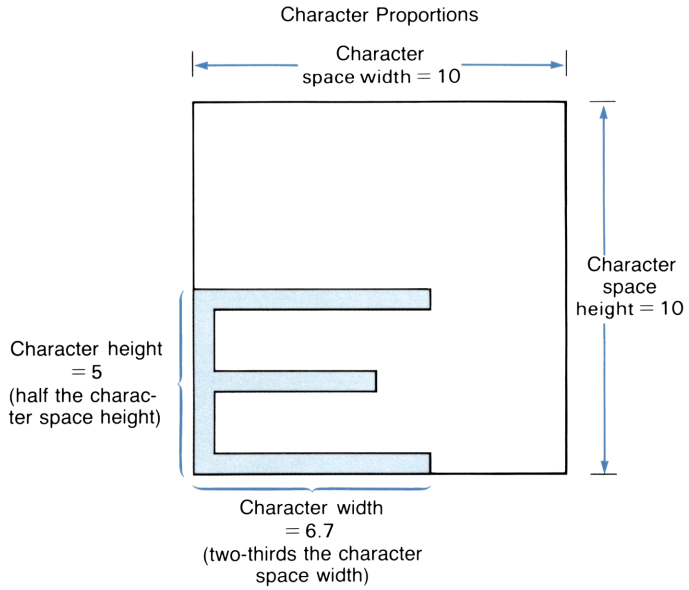
$$\text{character width} = \text{aspect ratio} \times \text{character height}$$

$$\text{character space width} = \tfrac{3}{4} \times \text{aspect ratio} \times \text{character space height}$$

The *slant* specifies the angle from vertical at which the character "leans" forward—a positive slant specifies a forward "lean," a negative slant specifies a backward "lean." Slant affects vertical strokes in a character, but not horizontal strokes.

Whenever you execute PINIT or LIMIT, the character parameters are set to these values: *height* = 3 GUs, *aspect ratio* = 0.7, and *slant* = 0. These *height* and *aspect ratio* parameters define a *width* of 1.575 GUs.

The following diagram illustrates a character with *height* = 10, *width* = 10, and *slant* = 0. The corresponding *aspect ratio* is 1.3333, which would be used to specify this shape.



CSIZE

X *height* (GUs)

The CSIZE (*character size*) function sets the height of the character space for subsequent labels—printed characters are half that height. The height is interpreted as GUs, regardless of the current plotting mode.

CSIZEO

Z *slant*

Y *aspect ratio*

X *height* (GUs)

The [CSIZEO] (*character size—option*) function sets the height, aspect ratio, and slant parameters. These parameters also determine the character space width. The height is interpreted as GUs, regardless of the current plotting mode. The slant angle is interpreted according to the HP-41's current angular mode ([DEG], [RAD], or [GRAD]). (The aspect ratio has no units.)

[CSIZEO] uses the absolute value of the height and aspect ratio parameters. The slant angle range allowed by the plotter module is between −90° and 90°. Within this range the actual maximum slant angle depends upon your plotter. (Angles outside the ±90° range are repeatedly changed by 180° until they fall into this range.)

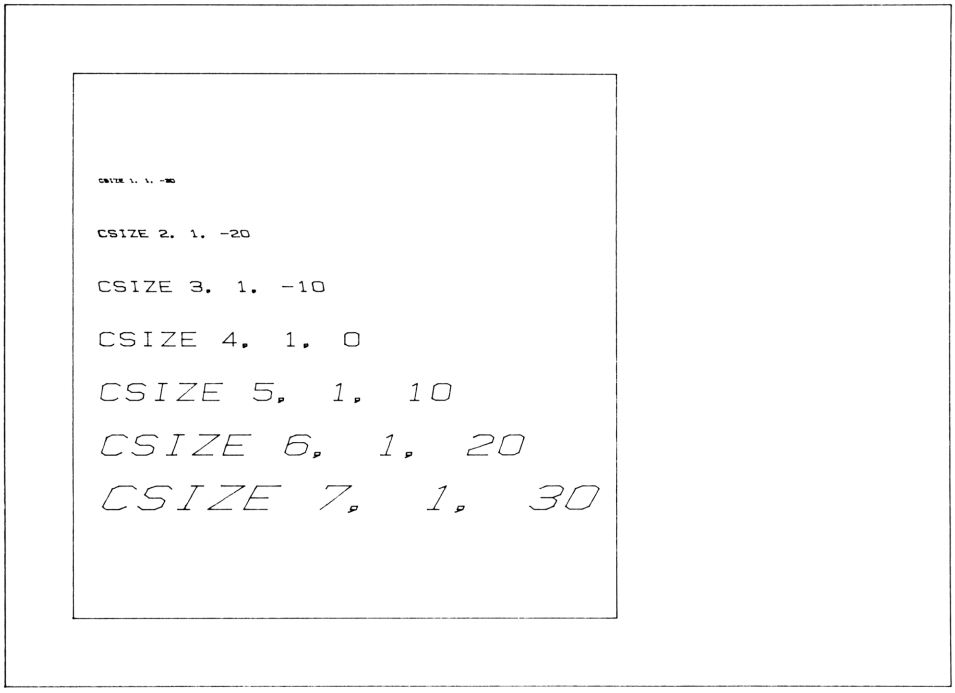
Example: Write a program that prints seven different character sizes and slants using an aspect ratio of 1. (A bar code listing for the following program begins on page 198 in appendix D.)

Keystrokes	Display	
[PRGM] [■] [GTO] [·] [·]		Packs program memory.
[PRGM]	00 REG *nn*	Switches HP-41 to Program mode. If ***nn*** is less than 13, you must create additional memory space before entering LABL. (Allow four data storage registers—R_{00} through R_{03}—for use by the program.)
[■] [LBL] [ALPHA] LABL	01 LBL LABL _	
[ALPHA]	01 LBL ᵀLABL	
[XEQ] [ALPHA] SET	02 XEQ SET _	Initializes HP-41. (Refer to Initializing the HP-41 for Examples, page 76.)
[ALPHA]	02 XEQ ᵀSET	
80.0101	03 80.0101 _	
[STO] 03	04 STO 03	Stores counter for label position.
1.008	05 1.008 _	
[STO] 00	06 STO 00	Stores height counter.
30.0401 [CHS]	07 −30.0401 _	
[STO] 01	08 STO 01	Stores slant counter.
[■] [CF] 29	09 CF 29	
[■] [LBL] 00	10 LBL 00	
[ALPHA] CSIZE [SPACE]	11 ᵀCSIZE _	Creates label.
[ALPHA]	11 ᵀCSIZE	
[RCL] 01	12 RCL 01	Sets slant value from integer of slant counter.
[INT]	13 INT	
1	14 1 _	Sets aspect ratio.
[RCL] 00	15 RCL 00	Recalls height value (h = INT (height counter)).
[CSIZEO]	16 CSIZEO	Sets slant, aspect ratio, and height.
[ALPHA] [■] [ARCL] [·] X	17 ARCL X	Creates label for current slant, aspect ratio, and height.*
[■] [APPEND], [SPACE] [■] 1, [SPACE],	18 ⊢, 1, _	
[■] [ARCL] [·] Z [ALPHA]	19 ARCL Z	Specifies current label position.
[RCL] 03	20 RCL 03	
[INT]	21 INT	
5	22 5 _	
[MOVE]	23 MOVE	Moves pen to label position.
[LABEL]	24 LABEL	Prints label.
[■] [ISG] 00	25 ISG 00	Increments height counter.
[■] [ISG] 01	26 ISG 01	Increments angle counter.
[DSE] 03	27 DSE 03	Decrements position counter.

*Refer to Alpha keyboard illustration on the HP-41's back label.

Keystrokes	Display	
■ GTO 00	28 GTO 00	Returns to start of loop.
■ SF 29	29 SF 29	
XEQ ALPHA TERM	30 XEQ TERM _	Terminates program. (Refer to Terminating Examples, page 77.)
ALPHA	31 XEQ^T TERM	
PRGM	0.0000	
XEQ ALPHA LABL ALPHA	0.0000	Executes LABL.

Executing LABL generates the following plot:



Using Axes

You can use the following functions to draw axes, determine their lengths, control the tic marks, and label the axes.

Drawing an Axis

XAXIS X *y-intercept* (UUs or GUs)

The XAXIS (*x-axis*) function draws a horizontal axis at the specified *y*-intercept. The *y*-intercept is interpreted according to the active plotting mode—either in UUs or in GUs. The axis extends completely across the plotting area—across the plot bounds in UU mode or across the graphic limits in GU mode.

YAXIS X *x-intercept* (UUs or GUs)

The YAXIS (*y-axis*) function draws a vertical axis at the specified *x*-intercept. The *x*-intercept and the extent of the axis is determined by the active plotting mode—either UU mode or GU mode.

The next four functions draw axes that don't necessarily extend across the entire plotting area and can also include tic marks and labels.

XAXISO

T *x-maximum* (UUs or GUs)
Z *x-minimum* (UUs or GUs)
Y *tic-spacing* (UUs or GUs)
X *y-intercept* (UUs or GUs)

The XAXISO (*x-axis option*) draws an x-axis between the specified minimum and maximum x-values and at the specified y-intercept, including tic marks at the specified spacing (described below).

YAXISO

T *y-maximum* (UUs or GUs)
Z *y-minimum* (UUs or GUs)
Y ±*tic-spacing* (UUs or GUs)
X *y-intercept* (UUs or GUs)

The YAXISO (*y-axis option*) function draws a y-axis between the specified y-values and the specified x-intercept, including tic marks at the specified spacing (described next).

Specifying Tic Marks

Tic Spacing. The tic-spacing parameter specifies the spacing between plotted tic marks. If the tic-spacing number in the y-register is positive, tic marks begin at the smallest value on the axis and are spaced equally along the axis in the positive direction. If the tic-spacing number is negative, tic marks begin at the highest value on the axis and are spaced equally in the negative direction. For example, consider these axes plotted from 0 to 10:

If the axis extends outside the plotting area, tic marks are located according to the intended starting point, rather than where the axis actually enters the plotting area.

When you want to plot an axis with a specified maximum and minimum, but you don't want any tics to appear on the axis, use a tic spacing value that exceeds the length of the axis.

Tic Length. The length of each plotted tic mark is related to the length of the plotting area. For example, tic marks on a y-axis are horizontal, and their length is proportional to the horizontal dimension of the plotting area. Similarly, the vertical tic marks on an x-axis are proportional to the vertical dimension of the plotting area. You can change the tic length using the following function.

TICLEN

X *tic length* (%)

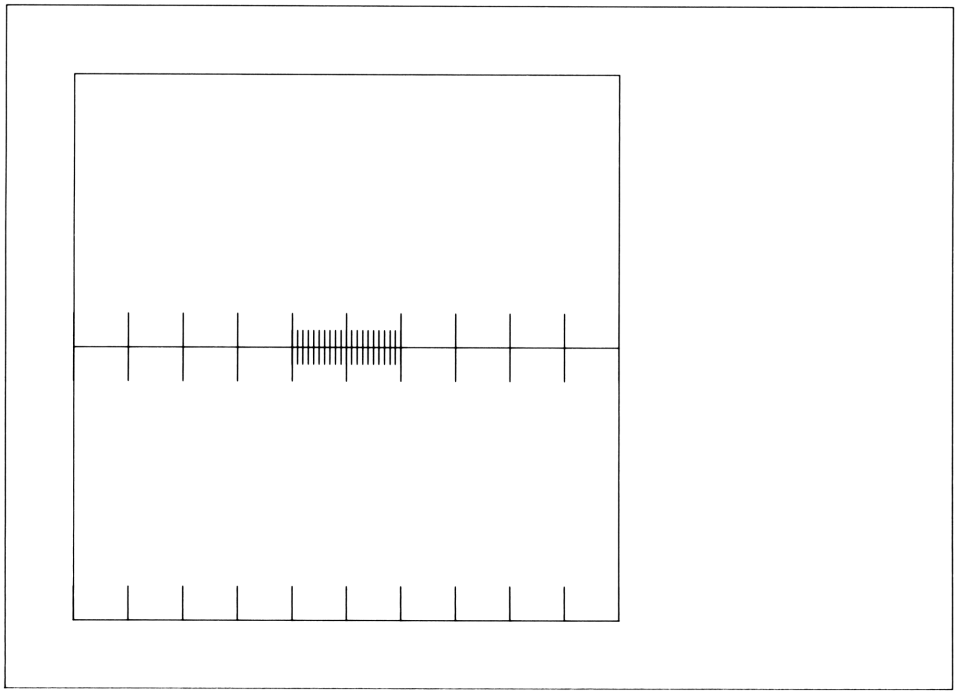
The TICLEN (*tic length*) function sets the tic length to the specified percentage of the graphic limits. It sets the vertical and horizontal tic lengths equal to a percentage of the vertical and horizontal dimensions of the graphic limits—the dimensions of the plot bounds don't affect the tic length, nor does the plotting mode. This percentage applies to both the x- and y-axes. (If the graphic limits have unequal dimensions, vertical and horizontal tic marks have unequal lengths.)

Tic marks are drawn with the specified length on each side of the axis. For example, a specified tic length of 2 percent produces tic marks with an actual length of 4 percent (2 percent on each side of the axis). However, if the axis is drawn at the edge of the plotting area, the tic marks are drawn only inside the area (2 percent long, in the same example).

TICLEN uses the absolute value of the tic length parameter. Whenever you execute PINIT or LIMIT, the tic length is set to your plotter's default tic length value.

Example: Set up a plotting area and draw x-axes at $y = 0$ and $y = 50$, each having large-size tic marks. Then draw minor tic marks on a portion of the upper x-axis.

Keystrokes	Display	
XEQ ALPHA SET ALPHA	XEQ SET _ 0.	Initializes HP-41. (Refer to Initializing the HP-41 for Examples, page 76.)
5 TICLEN	5.	Sets tic length to 5 percent.
100 ENTER↑ 0 ENTER↑ 10 ENTER↑ 50 XAXISO	0. 50 _ 50.	Draws x-axis at $y = 50$.
← XAXISO	0.	Changes y-intercept to 0 and redraws x-axis.
2.5 TICLEN	3.	Sets tic length at 2.5 percent.
60 ENTER↑ 40 ENTER↑ 1 ENTER↑ 50 XAXISO	40. 50 _ 50.	Draws minor tics between $x = 40$ and $x = 60$ on the x-axis at $y = 50$.
XEQ ALPHA TERM ALPHA	TERM _ 0.0000	Terminates example. (Refer to Terminating Examples, page 77.)



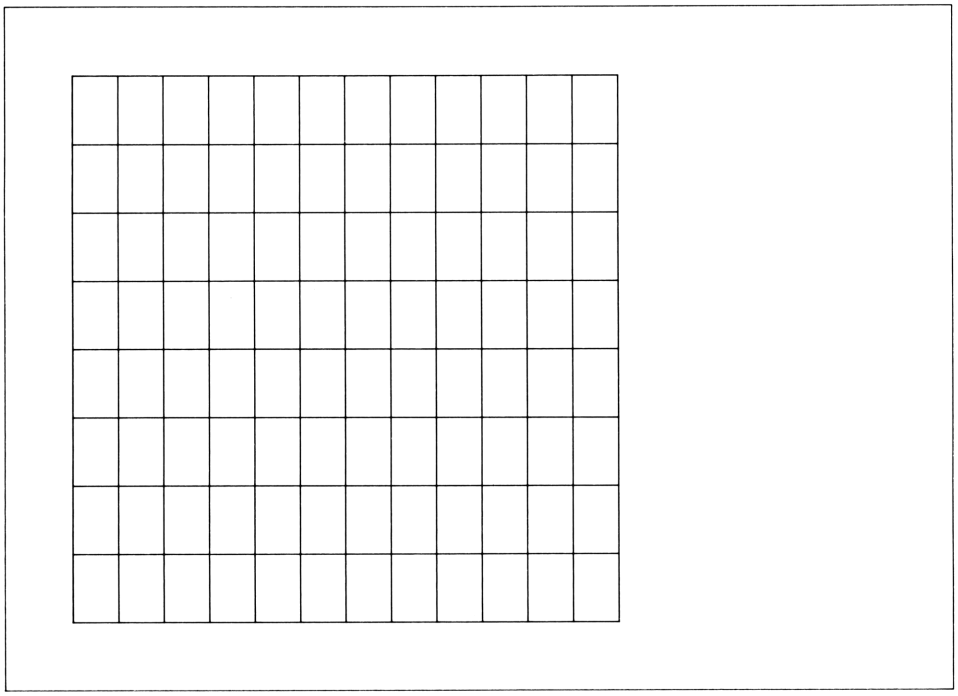
Plotting a Grid

You can plot a full grid using the axis-drawing functions. Simply set the tic length to 100 percent, then execute XAXISO and YAXISO.

Example. Create and scale plot bounds having 12 units in the x-dimension and 8 units in the y-dimension. Use TICLEN, XAXISO, and YAXISO to plot a grid.

Keystrokes	Display	
XEQ ALPHA SET ALPHA	XEQ SET _	Initializes HP-41. (Refer to Initializing the HP-41 for Examples, page 76.)
0 ENTER↑ 12 ENTER↑ 0 ENTER↑ 8 SCALE	12. 8.	Scales plotting area.
100 TICLEN	100.	Sets tic length to 100 percent.

Keystrokes	Display	
12 [ENTER↑] 0 [ENTER↑] 1 [ENTER↑] 0 [XAXISO]	0. 0.	Draws the *x*-axis portion of the grid.
8 [ENTER↑] 0 [ENTER↑] 1 [ENTER↑] 0 [YAXISO]	0. 0.	Draws the *y*-axis portion of the grid.
[XEQ] [ALPHA] TERM [ALPHA]	XEQ TERM _ 0.0000	Terminates the example. (Refer to Terminating Examples, page 77.)



You can draw various types of partial grids by:

- Specifying an [XAXISO] or [YAXISO] range that is smaller than the extent of the axis.
- Using [TICLEN] to set a tic length that is shorter than the length of the axis.

Drawing a Labeled Axis

Two functions, [LXAXIS] and [LYAXIS], provide automatic labeling for your axis plotting operations.

These functions operate in the same way as [XAXISO] and [YAXISO], except that the plotter also prints a numeric label at each tic mark it draws on the axis. The labels are printed in the area between the plot bounds and the graphic limits. Be sure to allow enough room around the plot bounds so that the labels are not clipped at the graphic limits.

Axis labels are formatted acccording to the HP-41 display format. The labels use the current character size (height, aspect ratio, and slant) and set the label direction (which may have been set by [LDIR]) to 0. The current [LORG] position isn't changed or used. After the axis is drawn, the ALPHA register contains the last label printed.

[LXAXIS]

T	*x-maximum* (UUs or GUs)
Z	*x-minimum* (UUs or GUs)
Y	*tic spacing* (UUs or GUs)
X	*y-intercept* (UUs or GUs)

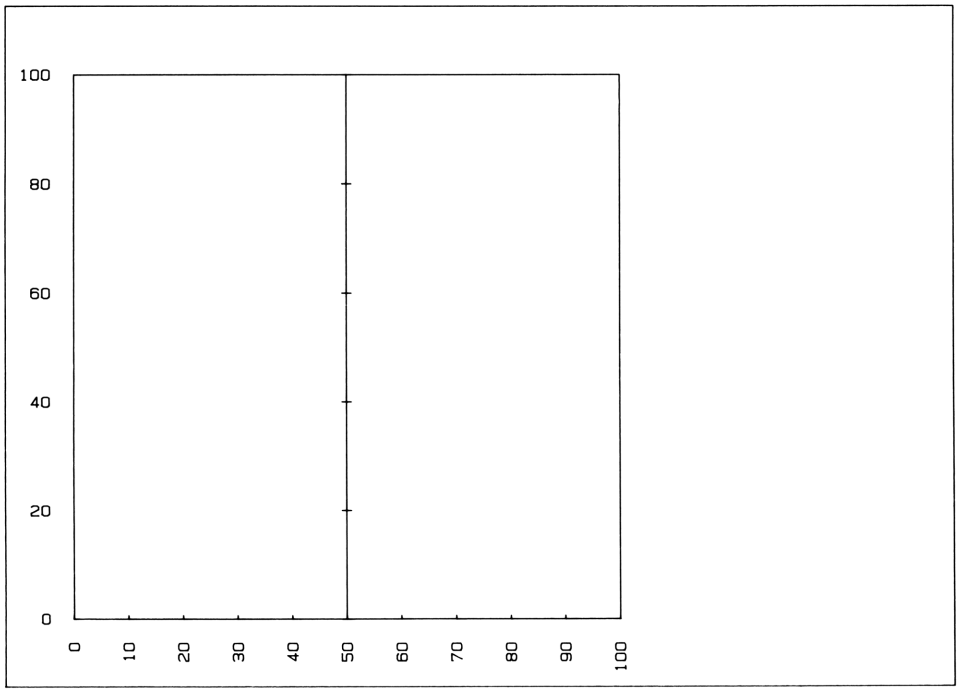
The LXAXIS (*label x-axis*) function draws the specified x-axis and labels its tic marks *below the plot bounds*. If the tic spacing parameter is positive, the labels are printed vertically (perpendicular to the x-axis); if the tic spacing parameter is negative, the labels are printed horizontally (parallel to the x-axis). Be sure to check the format, size, and spacing of horizontal labels to avoid overlap.

LYAXIS	T	*y-maximum* (UUs or GUs)
	Z	*y-minimum* (UUs or GUs)
	Y	*tic spacing* (UUs or GUs)
	X	*x-intercept* (UUs or GUs)

The LYAXIS (*label y-axis*) function draws the specified y-axis and labels its tic marks *to the left of the plot bounds*. The labels are always printed horizontally (perpendicular to the y-axis).

Example: Using the graphic limits and plot bounds defined by the SET program (page 76), draw an x-axis at $y = 0$ and label this axis at intervals of 10 UUs. Also draw a y-axis at $x = 50$ and label it at intervals of 20 UUs.

Keystrokes	Display	
XEQ ALPHA SET	XEQ SET _	Initializes HP-41. (Refer to Initializing the HP-41 for Examples, page 76.)
ALPHA	0.	
100 ENTER↑ 0 ENTER↑	0.	Enters x-maximum and minimum.
10 ENTER↑ 0	0 _	Enters tic spacing and y-intercept.
LXAXIS	0.	Draws and labels x-axis.
R↓ R↓	0.	Recalls y-maximum and minimum.
20 ENTER↑ 50	50 _	Enters tic spacing and x-intercept.
LYAXIS	50.	Draws and labels y-axis.
XEQ ALPHA TERM	XEQ TERM _	Terminates example. (Refer to Terminating Examples, page 77.)
ALPHA	0.0000	



For another application of LXAXIS, refer to lines 46 through 53 of the RAIN program listed on pages 160 and 161.

Section 6

Digitizing

Contents

Introduction 104
Selecting a Point (DGTIZE) 104
Identifying a Point (WHERE) 106
Digitizing Plot Bounds (LOCATD) 106

Introduction

The functions described in this section enable you to use your plotter's manual pen control keys to obtain x, y coordinate data from points on the plotting area. This process is called *digitizing* and is essentially the inverse of plotting. That is, plotting sends x,y coordinate values to the plotter, which then moves the pen to the specified location on the plotting area; digitizing involves moving the plotter's pen to a point on the plotting area and returning the x,y coordinates of that point to your HP-41. This feature allows you to "trace" an illustration by determining the coordinates of several of its points, and to use those coordinates as inputs to replot the illustration. By changing the scale, you can reproduce the same illustration in various sizes. The first example in this section uses the PLREGX function in a program that plots an illustration from data accumulated by digitizing.

When you execute the examples in this section, use the pen arm control keys on your plotter. To enter a point during a digitizing operation, press the *plotter's* ENTER key. The x, y coordinate values and the pen status will be returned to the HP-41.

On plotters equipped with a digitizing sight, the sight can be placed in the pen holder as if it were a pen. The digitizing procedure is the same as that for a plotter without a digitizing sight.

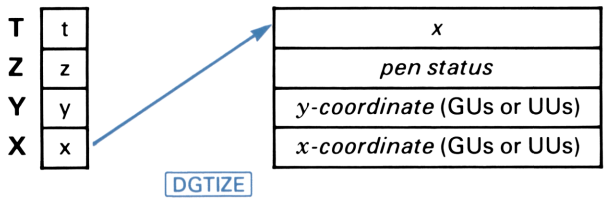
Selecting a Point

DGTIZE

The DGTIZE (*digitize*) function prompts you with the message **ENTER POINT**. Then:

1. If the pen is not already at the point you want to enter into the HP-41, move the pen to the desired point.
2. If necessary, use PENUP, PENDN, or the pen elevation keys on your plotter to place the pen in the desired up or down position.
3. Press your plotter's ENTER key.

DGTIZE enters the x- and y-coordinates of the pen's position—in the current scale units—into the HP-41's X- and Y-registers, and enters the current pen status (0 = "up", 1 = "down") in the Z-register.



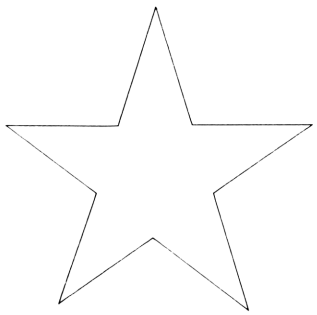
If [DGTIZE] is executed in a running program, program execution resumes after you press the plotter's ENTER key. If you do not press this key within approximately 10 minutes of executing [DGTIZE], the HP-41 turns itself off.

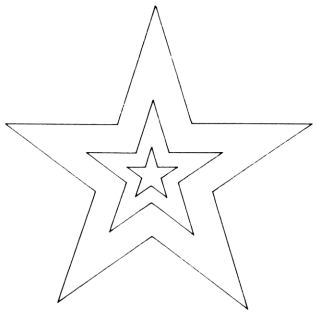
The *pen status* parameter is useful when you are digitizing under program control. That is, if you are storing digitized coordinates in HP-41 data registers that will later be accessed by [PLREGX], you can design the program to place an Alpha character in a register whenever a pen "up" status must be communicated to the [PLREGX] function. (Refer to the [PLREGX] function description on page 83.)

Example: Use the following program to digitize a simple design that connects 10 points. (Ensure that data registers R_{00} through R_{22} in your HP-41 are available.)

Keystrokes	Display	
■ [GTO] [·] [·] [PRGM]	00 REG *nn*	Switches HP-41 to Program mode. If ***nn*** is less than 06, you must create additional memory space. (Allow at least one data storage register—R_{00}—for use by the program.)
■ [LBL] [ALPHA] DIG [ALPHA]	01 LBL DIG _ 01 LBL"DIG	
[XEQ] [ALPHA] SET [ALPHA]	02 XEQ"SET	Initializes HP-41. (Refer to Initializing the HP-41 for Examples, page 76.)
1.022	03 1.022 _	Specifies loop counter and storage register pointer.
[STO] 00	04 STO 00	Stores preceding value in R_{00}.
■ [LBL] 00	05 LBL 00	Begins digitizing loop.
[DGTIZE]	06 DGTIZE	Digitizes current pen position.
[STO] ■ 00	7 STO IND 00	Stores *x*-coordinate in register indicated by integer portion of value in R_{00}.
■ [ISG] 00	08 ISG 00	Increments value in R_{00}.
[R↓]	09 RDN	Rolls down *y*-coordinate.
[STO] ■ 00	0 STO IND 00	Stores *y*-coordinate in register indicated by integer portion of value in R_{00}.
■ [ISG] 00	11 ISG 00	Increments value in R_{00}.
■ [GTO] 00	12 GTO 00	Returns program execution to [LBL] 00.
[XEQ] [ALPHA] TERM [ALPHA] [PRGM]	13 XEQ"TERM	Terminates example. (Refer to Terminating Examples, page 77.)

Using the preceding program, digitize the outline of the star shown below to the left (or a facsimile). Then, by changing the [LIMIT] parameters, plot on another sheet the series of stars shown below to the right.





The procedure for this example is:

1. Execute the SET program to initialize the plotter and HP-41. (Refer to Initializing the HP-41 for Examples, page 76.)
2. Move pen to (50,50) and place a facsimile of the star on the platen so that it is centered under the pen.
3. Execute DIG and digitize the indicated points on the star. (You should begin and end at the same point.)
4. Execute [SETGU], move the pen to point (0, 0), and place a new sheet of paper in the plotter. (You may then have to retrieve the pen from its stall.)
5. Place the number 1.022 in the X-register (to indicate data registers R_{01} through R_{22}) and execute [PLREGX] to plot the points that you digitized in step 3.
6. Reduce the graphic limits by 50 mm on all sides of point (50, 50) by entering 56, 206, 57, and 137, and executing [LIMIT]. Then perform step 5 again.
7. Further reduce the graphic limits by 25 mm on all sides of point (50, 50) by entering 81, 181, 82, and 112, and executing [LIMIT]. Then perform step 5 again.
8. Press 0 [ENTER↑], then execute [PEN] to store the pen and [MOVE] to display the stars.

The preceding information is only a sample of how data accumulated by [DGTIZE] might be used. The procedures you use for your own applications are likely to vary according to the nature of each problem.

Identifying a Point

The [DGTIZE] function recalls the point to which *you* moved the pen using the plotter's keys. The [WHERE] function recalls the point to which *the HP-41* last moved the pen.

[WHERE]

The [WHERE] (*where is pen*) function enters into the HP-41's X- and Y-registers the *x*- and *y*-coordinates of the point specified by the last pen movement function performed by the HP-41.* (Pen positions resulting from movement using the plotter's pen control keys are ignored.) The pen status is placed in the Z-register; it can either be used as described for the preceding [DGTIZE] function or ignored. The stack lifts in the same way that it does when you execute [DGTIZE]. [WHERE] does not terminate program execution.

[WHERE] allows you to easily determine the pen position, store it for future reference, and then perform some other plotting task. When you are ready, you can recall the stored data and return to the original point. This may be particularly useful for determining the pen position at the end of a [LABEL] operation.

Digitizing Plot Bounds

You can define the plot bounds by moving the pen to the desired corner points. For example, this enables you to locate your plot properly on a printed sheet.

[LOCATD]

The [LOCATD] (*locate by digitizing*) function allows you to specify the plot bounds by digitizing the two opposite corners. The plotter sends the coordinates of the specified corner points to the HP-41. The HP-41 interprets the coordinates as GUs, so be sure that the HP-41 is set to GU mode (or, if in UU mode, that the user scale is the same as the graphic scale) before using [LOCATD].

*If the last intended point was far outside the physical limits of the plotter, [WHERE] may return an *x*- or *y*-coordinate that is limited according to the numerical range of the plotter.

When you execute LOCATD, **ENTER POINT** appears in the display. You then use your plotter's keyboard to:

1. Position the pen at the lower left-hand point of the desired plot bounds and press the *plotter's* ENTER key.
2. When **ENTER POINT** again appears in the display, move the pen to the upper right-hand point of the desired plot bounds and again press the plotter's ENTER key.

After you complete the preceding two steps the plotter module sets the specified plot bounds and places in the stack the maximum and minimum for each axis, as shown in the following illustration.

T	***x-minimum*(GUs)**
Z	***x-maximum*(GUs)**
Y	***y-minimum*(GUs)**
X	***y-maximum*(GUs)**

Plot bounds you set using LOCATD are the same as the ones you set using the LOCATE function described on page 73. If you execute LOCATD with the corner points of the plot bounds outside the graphic limits, the resulting plot bounds are truncated where they intersect the graphic limits. (The x- and y-maximums and minimums placed in the stack will be those of the corner points you attempted to enter as plot-bound parameters.)

If you do not enter a point within approximately 10 minutes of the time an **ENTER POINT** prompt appears, the HP-41 turns itself off.

Section 7

Bar Code

Contents

Introduction 108
Terminology 109
Controlling Bar and Space Proportions 109
Measuring the Pen Width (PWIDTH) 110
Changing the Bar Code Size and Type (BCSIZE) 111
Using a Program to Generate Bar Code 114
Using PLOTBC (PLOTBC) 115
PLOTBC Operating Notes 119
Bar Code Subroutines 120
General Operating Parameters 121
Use of Registers 121
X-Register Data Bar Code Subroutines (XBC, XSBC) 121
ALPHA Register Data Bar Code Subroutines (ABC, AABC) 123
Generating Individual Rows of Program Bar Code (PBC) 124
Basic Bar Code Functions 127
Learning How to Plot or Print Bar Code 127
Setup for Examples 127
Standard HP-41 Data and Program Bar Code (BC , BCX , BCKS) 127
Alpha Data Bar Code Functions (BCA , BCAA) 130
Program Bar Code Function (BCP) 131
Printing HP-41 Bar Code on the HP 82162A Thermal Printer 133
Using BCO to Print Standard Data and Program Bar Code (BCO) 134
Printing Paper Keyboard and Direct Execution Bar Code 136
Utility Bar Code Functions 136
Generating the Bit Pattern and Checksum Separately (BCREGX , BCCKSM) 136
Four-Bit Mirror Checksum 138
Four-Bit Summation Checksum 139
BC and Null Bytes 142
Eight-Bit Summation Checksum 142
Plotting Alternative (Non HP-41) Bar Code (BCO) 143

Introduction

The plotter module enables you to plot standard HP-41 bar code* for the HP 82153A Optical Wand and three types of bar code for scanning devices used in other bar code systems. The module also enables you to print HP-41 bar code on the HP 82162A Thermal Printer. If you will be using only the printer to generate bar code, read the information under Terminology, page 109, then turn to Printing HP-41 Bar Code on the HP 82162A Thermal Printer, page 133. If you will be using a plotter to generate bar code, you should read sequentially through the topics in this section. If you have not already read the *HP 82153A Wand Owner's Manual,* you should do so before proceeding in this section.

*That is, all bar code you can read using the HP 82153A Optical Wand. This bar code includes the following types: program, numeric data, sequenced data, Alpha-Replace, Alpha-Append, paper keyboard, and direct execution. You can also generate special bar code (for use with the wand's WNDSCN function).

The easiest and most direct way to generate HP-41 data and program bar code is to use the PLOTBC and XBC programs. PLOTBC interacts with you to determine what HP-41 bar code you want, then uses XBC to plot the bar code in sequential, left-justified rows. This method enables you to generate HP-41 program bar code and most types of HP-41 data bar code in a default format. When you want to generate any of these same bar code types at selective pen positions, you can use the preprogrammed subroutines described under Bar Code Subroutines on page 120. To generate unlabeled bar code, paper keyboard or direct execution bar code, or bar code that is designed for scanning by devices other than the HP 82153A Optical Wand, refer to the functions and procedures described under Basic Bar Code Functions (page 127) and Utility Bar Code Functions (page 136).

Terminology

It is not necessary to understand the details of bar code terminology before you begin printing program or data bar code. However, to help you more easily use the material in this section, you may want to read through the following brief definitions of the more commonly used bar code terms as they apply to the HP 82184A Plotter Module:

Absolute Plotting Units: The pen movement parameters used by your plotter are internally calculated in absolute plotter units (APUs). There are 40 APUs per millimeter. The bar code width and height parameters used by the plotter module are expressed in APUs.

Alternate Bar Code: Any type of non HP-41 bar code in which the spaces and/or the bars represent data. In the plotter module, type numbers 1 through 3 designate alternate bar codes.

Bit: A binary digit of data. (That is, a 0 or a 1 digit.)

Bit Pattern: A grouping of bits in the ALPHA register that represent a particular data item.

Byte: An eight-bit unit of data held in a computer register. Also, a unit of memory space. (Refer to your HP-41 owner's manual.)

Checksum: A parameter used for error checking—usually the sum of all data bytes in a row of bar code.

HP-41 Bar Code: The bar code designed by Hewlett-Packard to be read by the HP 82153A Optical Wand.

Running Checksum: A parameter used for error checking in program bar code. The checksum for each row represents the sum of the current row and all previous rows in the program.

Controlling Bar and Space Proportions

The width of a pen you use to plot bar code affects the proportions between the widths of the bars and the widths of the spaces. Because wear increases pen width, you must periodically determine pen width and recalibrate the plotter module according to the current pen width. Otherwise, the bar/space width proportions can become distorted and therefore make the bar code difficult for the wand to read.

The plotter module specifies a bar or space width by the number of APUs a pen of a given width must move to travel across that bar or space. The plotter module's computation of how many APUs to allow for a bar or space depends upon the *bar code size parameter* maintained in the I/O buffer. This parameter, which specifies pen width and several bar and space widths, is automatically set to a default value when you first create the I/O buffer by executing PINIT. This default value is compatible with the 0.3-mm (12-APU) pen. As long as you generate HP-41 bar code with a pen having this width, the bars and spaces in your bar code will be correctly proportioned. However, if either the actual width of your pen differs from

the width contained in the size parameter or you want to generate non HP-41 bar code, the bar/space proportions in your bar code may be distorted. This can render the bar code difficult or impossible to read with your scanning device. To eliminate this distortion, use the PWIDTH program (provided in this manual) and the plotter module's BCSIZE function.

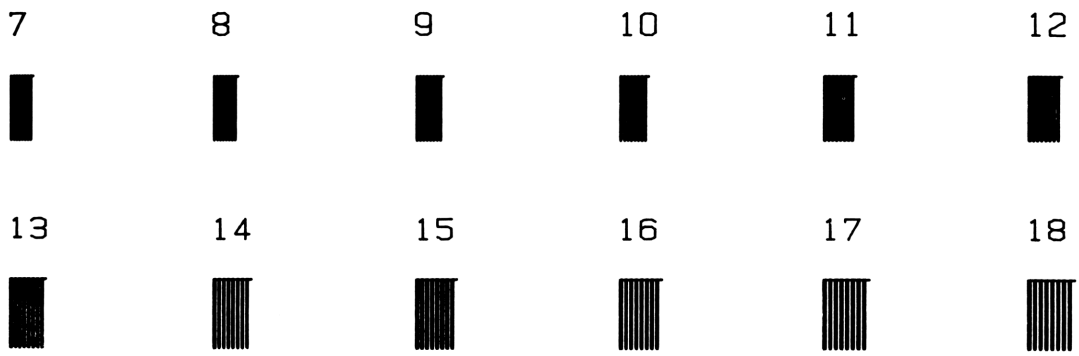
Measuring the Pen Width

PWIDTH

The PWIDTH (*pen width*) program uses the pen in stall 1 to plot a series of null bytes of bar code. The bytes are plotted in two rows, the first of which begins near the lower-left corner of the plotting area. Each byte is plotted using a different pen width value in the bar code size parameter. PWIDTH begins with a pen width of 7 APUs and terminates with a pen width of 18 APUs. Each byte is labeled to indicate the pen width value used for that byte. The first few bytes usually appear as solid blocks. The last few bytes show increasingly wider spaces between the bars. The first byte having a visible space between any of the bars indicates the approximate pen width. PWIDTH uses data registers R_{00} through R_{02}.

The PWIDTH program helps you to approximate the width of the pen you want to use for plotting bar code. Using the pen width value indicated by PWIDTH, you then execute the BCSIZE function (which is described on the next page) to calibrate the plotter module to the width of your pen. (If you are generating HP-41 bar code, you can usually perform this calibration by placing zero in the Y-register and pen width—from executing PWIDTH—in the X-register, then executing BCSIZE.)

The following illustration shows the result of executing PWIDTH. In this case, PWIDTH indicates a pen width of approximatly 13 APUs.



PWIDTH Must Always Be Followed By BCSIZE. Executing PWIDTH changes the bar code size parameter maintained by the I/O buffer. Thus, when you execute PWIDTH, you must then reset the bar code size parameter using the BCSIZE function. If you are generating HP-41 bar code, you can usually reset the size parameter by simply placing zero in the Y-register, placing the pen width indicated by PWIDTH in the X-register, and executing BCSIZE.

Example of PWIDTH Execution: Use the bar code on page 203 or the annotated program listing on page 169 to enter the PWIDTH program into your HP-41. Place in stall 1 a pen you can use for plotting bar code. Then:

1. Turn your plotter off and on (to set it to its default graphic limits).
2. Place a new sheet of paper in the plotter.

3. Execute PWIDTH, examine the resulting plot, and identify the pen width. (PWIDTH executes PINIT, which, in this case, creates a new I/O buffer.)
4. Calibrate the plotter module with the new pen width by placing zero in the Y-register, placing the pen width from step 3 in the X-register, then executing BCSIZE.

Keystrokes	Display	
XEQ ALPHA PWIDTH ALPHA	XEQ PWIDTH _	Plots and labels 12 null bytes. After last byte is plotted, pen returns to stall.
	0.0000	

Examine the bytes generated by PWIDTH. The lowest-numbered byte showing a gap between any two bars indicates the approximate width in APUs of the pen you are using.

> **Note:** PWIDTH results are subject to visual interpretation. For this reason you may find that, with some pens, you cannot clearly distinguish which of two or three adjacent width values you should use with BCSIZE. In such cases you should experiment with each of the values to identify the optimum width value to use. To do so, generate a separate row of bar code at each possible width value, then compare the rows visually and by reading with your bar code scanning device.

To calibrate your plotter module to a pen width indicated by PWIDTH, place 0 in the Y-register and the pen width (*pp*) in the X-register, then execute BCSIZE. After you use this method to calibrate the plotter module to the pen you are using, you are ready to begin plotting bar code.

Keystrokes	Display	
0 ENTER↑	0.0000	Enters 0 in the Y-register.
pp (Key in pen width.)	*pp*.0000	Enters pen width in APUs.
BCSIZE	*pp*.0000	Sets bar code size parameter.

(Leave the page in your plotter for use in the next example.)

Periodic Adjustments. As indicated earlier, pen wear is a factor that can gradually introduce distortion into the bar and space proportions of your bar code. Thus, you should periodically reexecute PWIDTH and BCSIZE to ensure that the bar code size parameter remains calibrated to the current pen width.

Changing the Bar Code Size and Type

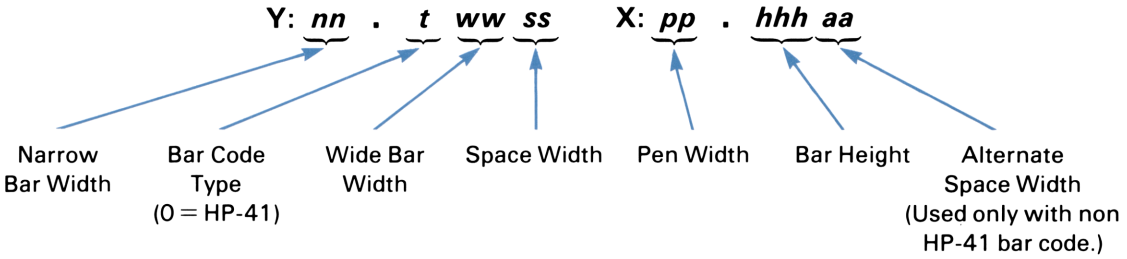
In the preceding discussion of PWIDTH you saw that you can reset the bar code size parameter for HP-41 bar code by simply keying in the pen width and executing BCSIZE. However, if you want to adjust the individual bar and space widths for HP-41 bar code, or if you want to specify an alternate bar code type, you will need to specify several elements of the bar code size parameter when you execute BCSIZE. If you are new to plotting bar code, you may not need to understand such details of BCSIZE operation until you have gained more experience. In this case, go on to Using a Program to Generate Bar Code on page 114 and resume reading. Otherwise, continue with the following description of BCSIZE.

BCSIZE

Y *nn.twwss*
X *pp.hhhaa*

The BCSIZE (*bar code size*) function uses the absolute value of the parameter elements in the X- and Y-registers to adjust the bar and space proportions. This compensates for variance in pen width and/or the requirements of an alternate bar code type that you may want to generate. With the exception of the bar code type number (*t*), the parameters are interpreted in absolute plotter units (APUs).

The Bar Code Size Parameter. The bar code size parameter, which consists of the elements illustrated below, controls bar width, space width, and bar code type.



The elements in this parameter are set to a series of default values when you execute PINIT to create the I/O buffer. However, if the I/O buffer already exists, the elements in the bar code size parameter are not affected by executing PINIT. The default element values are designed to produce correctly proportioned HP-41 bar code when you use a pen having a 0.3-mm (12-APU) width to plot the bar code. The PINIT column in the following chart shows the default value for each element.

When you execute BCSIZE, the parameter elements are set according to the values in the X- and Y-registers. However, where zeros are used to represent elements in the bar code size parameter, BCSIZE automatically sets those elements to values that are compatible with HP-41 bar and space width proportions. The BCSIZE column in the following chart shows how BCSIZE calculates such values.

Thus, as indicated in the PWIDTH discussion on page 111, you can set the bar code size parameter for HP-41 bar code by executing BCSIZE with zero in the Y-register and the correct pen width element in the X-register. Also, you can execute BCSIZE with zero in both the X- and Y-registers to set all elements in the bar code size parameter to their PINIT default values.

Elements of the HP-41 Bar Code Size Parameter

Register	Element	Application	Default When PINIT Creates I/O Buffer	BCSIZE Calculation For Any Unspecified Element	Element Range
Y-Register	*nn*	Narrow Bar Width	18 APUs	$nn = pp + pp/2$	0-99 APUs (0-2.5 mm)
	t	Bar Code Type (Used by BCO)	Type 0	0	0 thru 3
	ww	Wide Bar Width	30 APUs	$ww = 2nn - pp/2$	0-99 APUs (0-2.5 mm)
	ss	Space Width (For Types 0, 1 and 3) Narrow Space Width (For Type 2)	21 APUs	$ss = nn + 3$	0-99 APUs (0-2.5 mm)
X-Register	*pp*	Pen Width	12 APUs	12	0-99 APUs (0-2.5 mm)
	hhh	Bar Height	350 APUs	350	0-999 APUs (0-25 mm)
	aa	Alternate Space Width (For Types 1 and 2)	0	0	0-99 APUs (0-2.5 mm)

Example of How to Use BCSIZE**:** Compare the results of using three different bar code size parameters. Do this by generating the same bar code row three times; once for each parameter setting. The rows you generate will be similar to the following three rows. However, the width of the pen you use will control the actual width of your bars and spaces.



Keystrokes	Display	
PINIT		Initializes the I/O buffer. If the buffer already exists, this function does not affect the current bar code size parameter.
1 PEN 35 MOVE	35.0000	Selects and positions pen for first row.

For your first setting, use the default bar code size parameter. This parameter assumes the HP-41 bar code type (type 0), a bar height of 350 APUs, a pen width of 12 APUs, and a series of default bar and space values based on the pen width. (Refer to the PINIT column in the chart on page 112.)

Keystrokes	Display	
0 ENTER↑	0.0000	Places zero in the X- and Y-registers.
BCSIZE	0.0000	Sets the elements in the bar code size parameter.
123 BCX BC	3.0000	Uses bar code functions to generate a row of bar code. (The "3" results from the BCX function, which is described later in this section.)

For your second setting, use a pen width element of 10 APUs and a bar height element of 300 APUs. Use zeros in the remaining element positions. (This causes BCSIZE to set these elements in proportion to the specified pen width—refer to the BCSIZE column in the chart on page 112.)

Keystrokes	Display	
40 ENTER↑ 35 MOVE	35.0000	Positions pen for second row.
0 ENTER↑	0.0000	Specifies 0 for the ***nn***, ***t***, ***ww***, and ***ss*** parameter elements.
10.3	10.3_	Specifies 10 for the ***pp*** element and 300 for the ***hhh*** element.
BCSIZE	10.3000	Sets the elements in the bar code size parameter.
123 BCX BC	3.0000	Uses bar code functions to generate a row of bar code.

For your third setting, use a pen width element of 16 APUs and a bar height element of 225 APUs. For the ***nn*** and ***ss*** elements, add 5 APUs to the value that BCSIZE would calculate for each of these if they were unspecified. Similarly, add 10 APUs to the ***ww*** element value. (That is, add 5 to the results of the ***nn*** and ***ss*** equations in the chart on page 112, and add 10 to the result of the ***ww*** equation.)

If $\quad pp = 16$

then $\quad nn = pp + pp/2$
$\qquad = 24$

$\qquad ww = 2 \times nn - pp/2$
$\qquad = 40$

$\qquad ss = nn + 3$
$\qquad = 27$

and $nn + 5 = 29$
$ww + 10 = 50$
$ss + 5 = 32$

Because this example is intended for HP-41 type bar code, use zero for both the ***t*** and the ***aa*** elements.

Keystrokes	Display	
70 [ENTER↑] 35 [MOVE]	35.0000	Positions pen for next row.
29.0	29.0 _	Keys in ***nn*** and ***t***.
50	29.050 _	Keys in ***ww***.
32	29.05032 _	Keys in ***ss***.
[ENTER↑]	29.0503	Enters ***nn.twwss*** in the Y-register.
16.	16. _	Keys in ***pp***.
225	16.225 _	Keys in ***hhh***.
[BCSIZE]	16.2250	Sets the elements in the bar code size parameter.
123 [BCX] [BC]	3.0000	Uses bar code functions to generate a row of bar code.

This concludes the [BCSIZE] example. Because this example altered the bar code size parameter in your plotter module, use PWIDTH and [BCSIZE] to recalibrate the size parameter before you move on to the next topic.

Keystrokes	Display	
[XEQ] [ALPHA] PWIDTH [ALPHA]	XEQ PWIDTH _ 3.0000	Executes PWIDTH. You should now determine approximate pen width (*pp*) from resulting plot for use in next step.
pp [BCSIZE]	*pp*.0000	Keys in pen width indicated by preceding step and calibrates plotter module for that width.

Further [BCSIZE] Operating Details. When you execute [BCSIZE], if the integer portion of the number in either the X- or Y-register (the pen width or narrow bar parameter element) contains more than two digits, only the rightmost two digits of such an integer will be used. Any combination of ***pp*** and/or ***nn*** parameters that results in an overflow in ***nn*** or ***ww*** causes the computer to display the **DATA ERROR** message.

> **Note:** When the bar code size parameter has been calibrated to a large pen width, the plotting area may not be wide enough for a bar code row. For example, when the bar code size parameter is set to a pen width of 18 APUs, rows can be no longer than 12 bytes (where the default plotting area is used).
>
> The [BCSIZE] function is designed to calibrate the plotter module for pen widths in the range of 7 through 18 APUs. Specifying pen widths outside of this range can reduce the accuracy of the bar and space proportions in your bar code.

Using a Program to Generate Bar Code

Many of your bar code generation projects may require only that you produce several rows of data or program bar code on a sheet of paper. This manual provides you with a pair of programs, PLOTBC and XBC, that you can use together to complete such tasks. PLOTBC is an interactive program that prompts you to select a bar code type and to key in the information needed to plot the desired bar code. PLOTBC then executes the subroutine in XBC that plots and labels the bar code. PLOTBC automatically positions the pen for each row and prompts you to insert a fresh page when needed.

PLOTBC cannot be used without XBC. (The XBC program contains the subroutines that plot and label individual rows or bar code and can be used independently of PLOTBC. Independent use of XBC is

described under Bar Code Subroutines on page 120.) PLOTBC generates the following five types of bar code.:

- X-Register Data **D**
- X-Register Sequential Data **SD**
- ALPHA Register Data **A**
- Alpha-Append Data **AA**
- Program **P**

Using an HP 82153A Optical Wand, you can enter PLOTBC and XBC into your HP-41 by:

1. Scanning the PLOTBC program bar code on pages 199 through 201.
2. Executing [■] [GTO] [·] [·].
3. Scanning the XBC program bar code on page 207 through 209.

If you do not have a wand, you can load the programs from the keyboard by using the PLOTBC program listing on page 161 and the XBC program listing on page 171. PLOTBC uses 56 memory registers and data storage registers R_{00} through R_{03}. XBC uses 52 memory registers and data storage registers R_{00} and R_{01}.

Using PLOTBC

Key Assignments. PLOTBC operation is controlled by the *main prompt* and the top row keys, as shown below:

The Main Prompt and Corresponding Top Row Keys

D	SD	A	AA	P
[Σ+]	[1/x]	[√x]	[LOG]	[LN]

When the main prompt is displayed, you select the desired bar code type by pressing the corresponding top row key.* The program then prompts you for the appropriate data or program information, generates the bar code, and prompts you for either the next data entry or the next bar code type.

> **Note:** If you assign a function or program to any top row key, pressing that key while the main prompt is displayed (and the User keyboard is active) executes the assigned function or program instead of continuing with PLOTBC bar code operations.

Pens. PLOTBC uses pen number 1 to plot bar code and pen number 2, if available, to print labels.

Example of PLOTBC Operation: Enter PLOTBC and XBC into program memory. Then plot bar code for the four data types and for the SET program you used earlier in this manual (refer to Initializing the HP-41 for Examples, page 76). To ensure that the current graphic limits are large enough for the bar code you are about to generate, turn your plotter off, then on, before you begin.

Keystrokes	Display	
[XEQ] [ALPHA] PLOTBC	XEQ PLOTBC _	Executes PLOTBC.
[ALPHA]	INSERT PAGE	Prompts you to insert a sheet of paper into the plotter. Press [R/S] when you are ready to proceed.
[R/S]	D SD A AA P	Prompts you to select a bar code type.
[Σ+]	DATA?	Selects X-register data bar code type and prompts you to key in your data.
12.34 [R/S]	DATA?	Enters data and plots labeled bar code, positions pen to begin next row, and prompts you for another data entry.

*Line 04 of the PLOTBC program sets flag 27, which is the HP-41's User keyboard flag. The top row keys are used with PLOTBC to specify the desired bar code type in the same way that they are used with the Utility Plotting Program to specify a plotting routine. Refer to User Keyboard, Key Assignments, and Keyboard Overlay, page 21.

Keystrokes	Display	
[ALPHA] ABC [R/S]	DATA?	Enters data, plots labeled bar code, positions pen to next row, and prompts for next data entry.
[R/S]	D SD A AA P	When no data entered, returns you to main prompt.
[1/x]	SEQ = 0?	Selects sequential data bar code and prompts you to verify that 0 is the sequence number for this row of data bar code.
[R/S]	DATA?	When no value keyed in, assumes that 0 is the correct sequence number and prompts you for data input.
56.78 [R/S]	SEQ = 1?	Enters data and plots labeled bar code, positions pen to next row, and prompts you to verify that 1 is the sequence number for the next data bar code row.
[R/S]	DATA?	Assumes that 1 is the correct sequence number and prompts you for data input.
9.01 [R/S]	SEQ = 2?	Enters data and plots labeled data bar code. Positions pen to next row and prompts you to verify next sequence number.
[R/S]	DATA?	Assumes that 2 is the correct sequence number and prompts you for data input.
3.23 [R/S]	SEQ = 3?	Enters data and plots labeled bar code. Positions pen to next row and prompts you to verify next sequence number.
[R/S]	DATA?	Assumes that 3 is the correct sequence number and prompts you for data input.
[R/S]	D SD A AA P	When no data entered, returns you to main prompt.
[√x]	A DATA?	Selects ALPHA register bar code type and prompts you to key in your data.
XXX [R/S]	A DATA?	Enters alpha data. Plots labeled bar code, positions pen to next row, and prompts for next data entry.
[R/S]	D SD A AA P	When no data entered, returns you to main prompt.
[LOG]	AA DATA?	Selects Alpha-Append bar code type and prompts you to key in your data.
YYY [R/S]	AA DATA?	Enters alpha data. Generates labeled bar code, positions pen to next row, and prompts for next data entry.
[R/S]	D SD A AA P	When no data entered, returns you to main prompt.
[LN]	NAME?	Selects program bar code type and prompts you to key in your data.
SET [R/S]	ROW = 1.16?	Selects the SET program and prompts you to verify that 1 is the row number to start with and 16 is the maximum byte length for each row.
[R/S]	⟩- ⟶ ⟩-	Verifies row and byte number. Plots program bar code (in approximately 6 minutes using the HP 7470A Plotter) and, because no space remains on the page, prompts you to insert a page.
	INSERT PAGE	
[←] [←] [■] [FIX] 4	0.0000	Clears display.
[■] [GTO] [·] [·]	0.0000	Removes HP-41 from PLOTBC program.

Keystrokes	Display	
[SF] 29	0.0000	Sets digit grouping flag, which is cleared by some PLOTBC data operations. (This topic is discussed later in this section, under X-Register Data Bar Code Subroutines on page 121.)

The preceding keystrokes generate the following bar code.

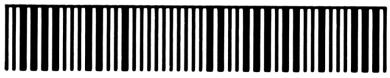
D: 12.34

D: "ABC"

SD 0: 56.78

SD 1: 9.01

SD 3: 3.23



A: "XXX"

AA: "YYY"

SET

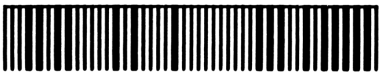
ROW 1: LINES 1-5

ROW 2: LINES 5-12

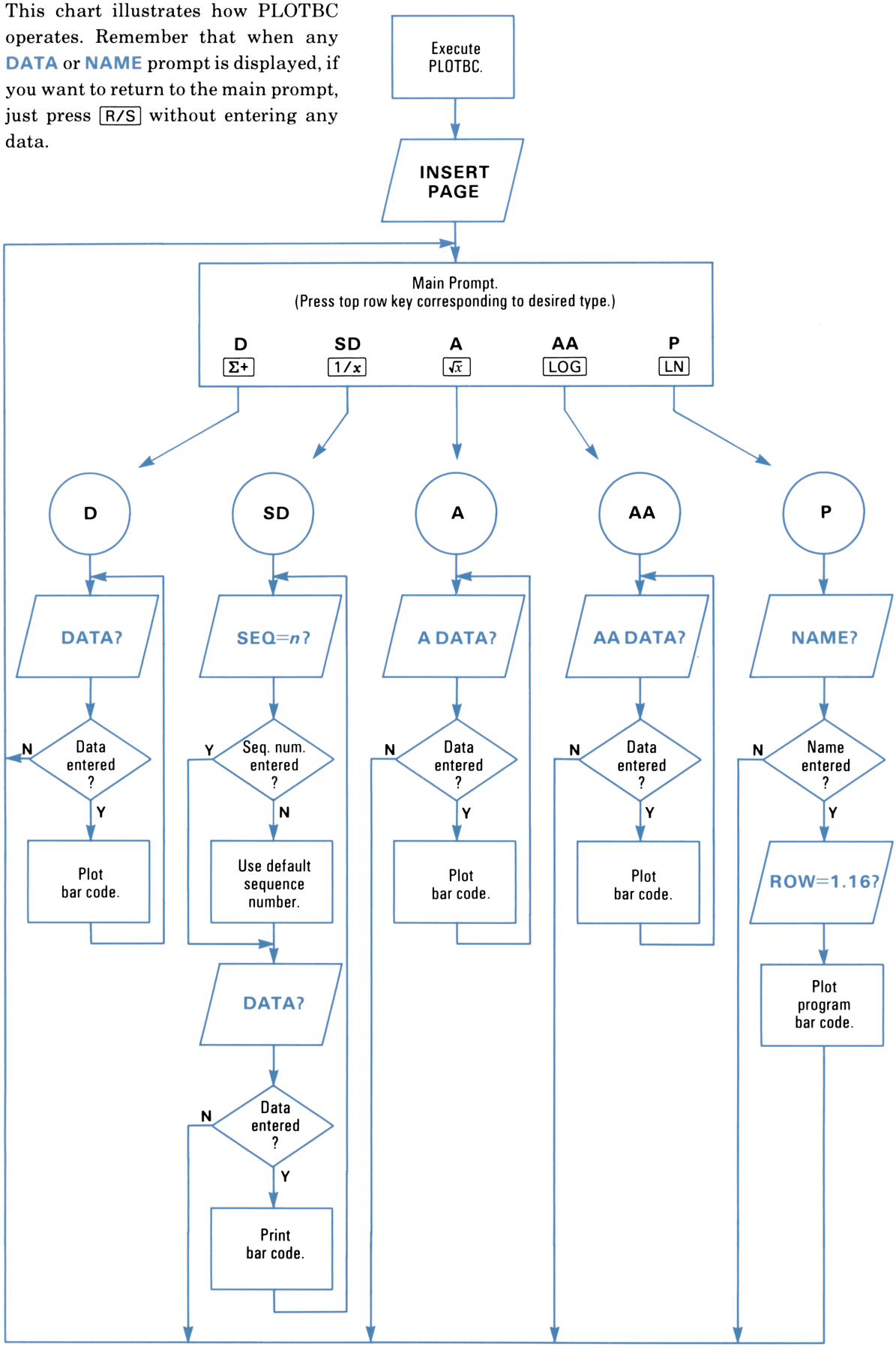
ROW 3: LINES 13-21

ROW 4: LINES 21-28

ROW 5: LINES 28-28



This chart illustrates how PLOTBC operates. Remember that when any **DATA** or **NAME** prompt is displayed, if you want to return to the main prompt, just press [R/S] without entering any data.



Whenever the HP-41 is set to any line in the PLOTBC program and the User keyboard is active, the following key assignments are also active:

- [■] [a]: Executes PLOTBC. (For keys to press, refer to the HP-41's back label.)
- [■] [b]: Returns you to the main prompt.
- [■] [c]: Moves pen to the next bar code row.
- [■] [d]: Moves pen to the preceding bar code row.
- [■] [e]: Replot a specific row of bar code. When you press [■] [e], the X- and Y-registers must contain the same data as is shown for the PBC subroutine (page 124), and the pen must be positioned to plot the row. (If you have to move the pen, use either [■] [c] or [■] [d], as described below, under Changing Rows.)

The PLOTBC program provides you with an easy method for producing consecutive rows of bar code. If this program meets your bar code generation needs, you may not need to use the plotter module's individual bar code subroutines and functions. The following information describes PLOTBC operating conditions that you may need to remember when you use the program in your applications.

PLOTBC Operating Notes

Ensuring Adequate Graphic Limits. PLOTBC executes [PINIT], which maintains the current graphic limits, but returns all other plotter, module, and I/O buffer settings (except the bar code size parameter*) to their default values. Thus, the plotting area and user units (UUs) are set to the dimensions of the current graphic limits and graphic units (GUs). Because the character spaces for the labels automatically default to 3 GUs, the current dimensions of the graphic limits determine the physical height of your bar code label characters. The dimensions of the bar code are specified in plotter units and are not affected by the dimensions of the graphic limits. To ensure labels of the size generated in the preceding example, set the plotter module to your plotter's default graphic limits by turning the plotter off, then on, before you begin a bar code plotting session with PLOTBC.

> **Note:** If PLOTBC encounters an edge of the plotting area while plotting bar code, the portion of the bar code row that is outside the boundary is not plotted. In such cases, the plotter will appear to pause because the HP-41 will still perform all calculations for the current row, even though some part of the row is not being plotted.

Pen Positioning. Each time you execute PLOTBC the pen moves to the first row position on the paper. If the paper is already in the plotter, ignore the **INSERT PAGE** prompt and press [R/S]. The pen then moves to the first row position. If, when **INSERT PAGE** is displayed, you want to go to the main prompt without moving the pen, press [■] [b]. Likewise, if the HP-41 is set to any line in PLOTBC and is set to the User keyboard, you can go directly to the main prompt—and therefore bypass the **INSERT PAGE** prompt and any pen movement—by pressing [■] [b].

Rows Per Page. When started from the top of an 8½ by 11-inch page and used without interruption, PLOTBC allows you to plot 13 standard HP-41 bar code rows to a page and prompts you when it is time to insert a new page. (If you plot data bar code, then shift to program bar code, a total of only 12 rows are plotted on the page.) If you have to insert a new page, press [R/S] when you are ready for bar code plotting to resume.

Changing Rows. Whenever the HP-41 is set to any line in the PLOTBC program, you can move the pen from the current row to the beginning of the next row by pressing [■] [c]. To move to the beginning of the preceding row, press [■] [d]. Whenever PLOTBC is in program memory, but the HP-41 is not set to any line in this program (for example, when it is set to one of the bar code subroutines), you can shift to the next row or the preceding row by executing [NXTROW] or [LSTROW], respectively.

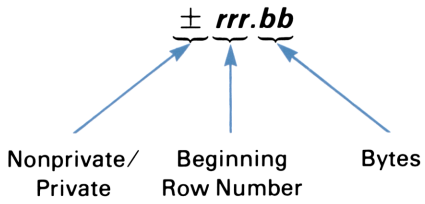
---

*Unless [PINIT] creates the I/O buffer. Refer to the last item in the list at the top of page 69.

Flags. PLOTBC uses the following HP-41 user flags: 9, 17, 22, 23, 27, and 29. The "numeric input" flag (flag 22) is used to detect your numeric data inputs for data bar code generation. This flag is not affected by data entered by recall from a storage register. Thus, if you enter Alpha or numeric data by recalling it from a storage register, then pressing [R/S], the HP-41 responds as if no data was entered. (That is, it returns to the main prompt without plotting any bar code.)

Packing Memory. If a program you want to plot is not packed, PLOTBC automatically packs memory and prompts you to reexecute the program-plotting operation. Thus, if you want to avoid an interruption for packing, you should pack the HP-41's program memory before you begin plotting program bar code. (Refer to your HP-41 owner's manual.)

Program Bar Code Generation. The ROW = 1.16? prompt allows you to accept or change the default beginning row and length parameters. (Default plotting begins with row 1. Default row length is 16 bytes—the maximum allowed.) The format to use for keying in your own starting row number and byte parameter is:



PLOTBC uses the absolute value of ***rrr.bb*** to determine row and byte length parameters. A negative ***rrr.bb*** value results in "private" program bar code. (A private bar code program can be scanned and executed like any other bar code program. However, a private bar code program cannot be viewed, listed, altered, recorded, or replotted through normal operations.)

Row Parameters. When you want to begin plotting program bar code at row 1, but do not want to print the program's name, key in zero for ***rrr***. (An ***rrr*** of 0 defaults to 1 for purposes of selecting the starting row number.) If you select an ***rrr*** other than 0 or 1, there will be a pause between the moment you press [R/S] to begin plotting bar code and the moment that the plotter actually begins plotting. This is because the system cannot start plotting at the row you want until it has computed the bar code checksums for all rows that precede the desired row.

Byte Parameters. Byte parameters of 00 through 04 and byte parameters exceeding 16 default to 16. Always key in byte parameters as two-digit numbers (05, ... 16).

Halting Bar Code Generation. If, for any reason, you want to halt a bar code operation that is already underway, press the [R/S] key. The plotter halts and lifts the pen. If you use this method to halt bar code generation, but want to continue the operation at a later time, you should resume by replotting the row that precedes the one in which you halted. This procedure helps to avoid a possibility of error in your desired bar code sequence.

> **Note:** If you halt the system while a row of bar code is being plotted, the HP-41 will be set to the XBC program instead of to PLOTBC. Thus, to subsequently move to the next row position, you will have to execute NXTROW instead of pressing ■ [c]. To move to the last row position you will have to execute [LSTROW] instead of pressing ■ [d].

Bar Code Subroutines

The bar code subroutines in the XBC program are provided for use in conjunction with PLOTBC when you want to plot several consecutive, left-justified rows of bar code on a page. However, you can also use these subroutines when you want to plot isolated rows of bar code at various positions on a page. To use any of the subroutines, enter the XBC program into your HP-41, use a plotter module function such as [MOVE] or [IMOVE] to position the pen, and execute the desired subroutine.

Note: The bar code subroutines use UUs to position bar code rows relative to their labels. For this reason, initialize your plotter to its default graphic limits and plotting area parameters before you use the bar code subroutines. An easy way to perform this operation is to turn your plotter off, then on, and execute PINIT.

General Operating Parameters

When you execute any of the bar code subroutines:

- Plot direction resets to 0°.
- Label direction resets to 90°.
- The current CSIZE or CSIZE0 setting determines the character size for labels. (The PINIT default of 3 GUs is convenient for many applications.)
- Pen 2 prints the labels.
- Pen 1 plots the bar code, then returns to its stall.

When you execute a bar code subroutine, the label is printed at the last pen position specified by a plotter module function. The bar code row is plotted 2 UUs below the label. The default bar code height is 350 APUs. (You can change the height value using the BCSIZE function described under Changing the Bar Code Size and Type on page 111.)

Numeric data labels always contain all digits of the bar code they identify, and they are not affected by the current display setting.

Use of Registers

The bar code subroutines affect the stack and LAST X registers in various ways that are described separately below for each subroutine. The subroutines also use various bar code functions (described later in this section, beginning on page 127) that replace the contents of the ALPHA register with bit string data used to control bar code printing. If you set your HP-41 to Alpha mode after executing a bar code subroutine, you will see most of this bit string data displayed as nonstandard, unintelligible characters.

In most cases, the values remaining in the stack and LAST X registers after bar code subroutine execution are of no interest to the user.

X-Register Data Bar Code Subroutines

The following two subroutines generate numeric or Alpha bar code from corresponding data in the X-register. Both subroutines alter flag 29 (the digit grouping flag) and the display setting so that numeric data labels contain all significant digits of the value in the X-register. Alpha data strings in the X-register and their corresponding labels can contain up to six characters. (For most Alpha data applications, you will probably want to use the Alpha data bar code subroutines—described under the next heading—instead of the X-register data bar code subroutines.)

Note: The bar code subroutine examples on the following pages assume that you have executed PINIT and that the plotter module is set to your plotter's default graphic limits and plotting area.

XBC **X** *data* **(Numeric or Alpha)**

The XBC (*Data Bar Code from X*) subroutine plots and labels a row of data bar code containing the data currently in the X-register. XBC labels always begin with a **D:** followed by the number or Alpha characters contained in the corresponding bar code. Scanning a row of numeric data generated by XBC

enters that row's data in the X-register. Scanning a row of Alpha XBC data replaces the contents of the ALPHA register with the Alpha data contained in the Alpha XBC row and switches the HP-41 to the Alpha keyboard.

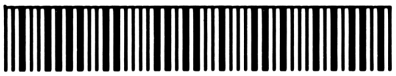
Following XBC execution, the X-register indicates the number of bytes generated (and, for numeric data, the number of fractional digits). The Y-register contents remain unchanged. The Z- and T-registers contain variables produced during subroutine execution. The LAST X register contains a copy of the data plotted in bar code by XBC.

Example. Plot on the same line a row of numeric data and a row of Alpha data.* (This example assumes you have entered the XBC program from the bar code on page 207 or the program listing on page 171.)

Keystrokes	Display	
1 [ENTER↑] 3 [MOVE]	3.0000	Positions pen for next row.
1.23456	1.23456	Enters numeric data in X.
[XEQ] [ALPHA] XBC [ALPHA]	XEQ XBC _	Plots label and bar code for numeric data.
	5.00000	Number of bytes in row.
50 [ENTER↑] 3 [MOVE]	3.00000	Positions pen for next row.
[ALPHA] SAMPLE [ASTO]	ASTO __	Places Alpha message in X.†
[·] X [ALPHA]	SAMPLE	
[XEQ] [ALPHA] XBC [ALPHA]	XEQ XBC _	Plots label and bar code for SAMPLE Alpha data.
	8.00000	Number of bytes in row.
[FIX] 4	8.0000	Sets [FIX] 4 display mode.
[SF] 29	8.0000	Resets the digit grouping flag.

(Keep the current page in your plotter for the next example.)

D: 1.23456



D: "SAMPLE"

XSBC

Y *sequence number*
X *data*

The XSBC (*sequential data bar code from X*) subroutine plots and labels a row of sequential data bar code containing the data currently in the X-register. XSBC labels always begin with **SD *n***: followed by the number or Alpha characters included in the corresponding bar code. To scan a row of bar code generated by XSBC, first execute the HP 82153A Optical Wand's [WNDDTX] function, then scan the sequential data rows in their proper order. The data in each scanned row is placed in the appropriate data storage register. (Refer to the [WNDDTX] function in the *HP 82153A Wand Owner's Manual.*) A sequence number greater than 999 results in a **NONEXISTENT** error message *after* the row label is printed.

Following XSBC execution, the X-register contains the *next* sequence number in the series. The value in the Y-register indicates the number of bytes generated. The Z- and T-registers contain variables produced during subroutine execution. The LAST X register contains a copy of the data printed in bar code by XSBC.

*Use the default parameters as described in the first note on page 121.

†For [ASTO], refer to the Alpha Keyboard illustration on the back of the HP-41.

Example. On the same line plot two rows of sequential numeric data bar code from X and one row of sequential Alpha data bar code from X. Use the default plotting parameters as described in the note at the top of page 121. (This example assumes you have entered the XBC program from the bar code on page 207 or the listing on page 171.)

Keystrokes	Display	
1 [ENTER↑] 15 [MOVE]	15.0000	Positions pen for next row.
1 [ENTER↑] 1.414	1.414 _	Enters sequence number and value to be plotted.
[XEQ] [ALPHA] XSBC [ALPHA]	XEQ XSBC _	Plots label and bar code.
	2.000	Next logical sequence number.
50 [ENTER↑] 15 [MOVE]	15.000	Positions pen for next row. (Also moves sequence number to Z-register.)
[R↓] [R↓]	2.000	Returns next sequence number to X.
1.732	1.732 _	Enters next value to be plotted in bar code.
[XEQ] [ALPHA] XSBC [ALPHA]	XEQ XSBC _	Plots label and bar code.
	3.000	Next logical sequence number.
21 [ENTER↑] 27 [MOVE]	27.000	Positions pen for next row.
[R↓]	21.000	Rolls down next sequence number (3) from Z to Y.
[ALPHA] X [■] [-] SEQ	X - SEQ _	Enters into X the Alpha data for plotting. (The Alpha data writes over the contents of X.)
[■] [ASTO] [·] X [ALPHA]	X - SEQ	
[XEQ] [ALPHA] XSBC [ALPHA]	XEQ XSBC _	Plots label and bar code.
	4	Next logical sequence number.
[←] [■] [FIX] 4	0.0000	Clears display and sets [FIX] 4 display mode.
[■] [SF] 29	0.0000	Resets the digit grouping flag.

(Leave the current page in your plotter for the next example.)

SD 1: 1.414

SD 2: 1.732

SD 3: "X-SEQ"

Alpha Register Data Bar Code Subroutine

ABC **ALPHA** *Alpha data*

The ABC (*Alpha data bar code*) subroutine plots and labels a row of Alpha data bar code containing the first 14 characters in the ALPHA register.* Any characters beyond the 14th character will be truncated (deleted from the ALPHA register). ABC labels always begin with **A**: followed, within quotation marks,

*Standard HP-41 Alpha characters as indicated on the HP-41's back label. (There are four exceptions, which are indicated in the chart on page 94.) Trailing nulls and nulls enclosed by Alpha characters will be printed in the bar code, but not in the corresponding label.

by the Alpha characters included in the corresponding bar code. Scanning a row of ABC bar code replaces the current contents of the ALPHA register with the scanned data and switches the HP-41 to the Alpha keyboard.

Following ABC execution, the X-register contains the number of bytes generated. The Y-, Z-, T-, and LAST X registers contain variables produced during subroutine execution.

Example: Place an Alpha string in the ALPHA register and plot a row of bar code containing this data. Use the default plotting parameters as described in the note at the top of page 121. (This example assumes you have entered the XBC Program from the bar code on page 207 or the listing on page 171.)

Keystrokes	Display	
[ALPHA] TEST [SPACE] ROW [ALPHA]	TEST _ 0.0000	Enters the Alpha data in the ALPHA register.
1 [ENTER↑] 39 [MOVE]	39.0000	Positions pen for next row.
[XEQ] [ALPHA] ABC [ALPHA]	12.0000	Plots Alpha data bar code from data you entered in the ALPHA register.
0 [PEN]	0.0000	Returns pen to stall.

(Leave the current page in your plotter for the next example.)

A: "TEST ROW"

AABC	ALPHA	*Alpha data*

The AABC (*Alpha-append bar code*) subroutine operates in the same way as the ABC subroutine, except that the bar code generated by AABC is in the Alpha-append format. (That is, scanning a row of AABC bar code appends the scanned data to the contents of the ALPHA register and switches the HP-41 to the Alpha keyboard.)

Generating Individual Rows of Program Bar Code

When you want to plot the bar code for an entire program, or from a particular row to the end of the program, you can use the PLOTBC program described under Using PLOTBC on page 115. However, when you want to plot a single program row or several rows without being restricted to a sequential row order, you can do so using PBC.

PBC	Y	*program name*
	X	*rrr.bb*

The PBC (*program bar code*) subroutine plots and labels row ***rrr*** of the program named in the Y-register. ***bb*** determines the number of bytes in the row. PBC then increments ***rrr*** by 1 and halts. The program name remains in the Y-register. Thus, by using the data remaining in the X- and Y-registers after executing

PBC, you can print the next sequential program row by simply reexecuting PBC. You can select a random row by changing the ***rrr*** value in the X-register after PBC execution halts. PBC indicates that it has printed the last row in a program by replacing ***rrr.bb*** with **0.0000**. As with the PLOTBC Program, before you use PBC to plot program bar code, you should pack the program by executing ■ GTO · ·.

For programs plotted in bar code it is usually preferable to use the same byte parameter—***bb***—for each row. The topic of row length is discussed on the next page, under The Row and Byte Parameters.

PBC uses data storage registers R_{00} and R_{01} to temporarily store the pen's location when you execute PBC. The contents of the Z-, T-, and LAST X registers are altered by variables produced during PBC execution.

The PBC subroutine switches the display setting to FIX 3.

Example: Plot a pair of sequential 10-byte rows and a nonsequential 10-byte row from the SET program used earlier in this manual. If SET is not currently in your HP-41, you can enter it using either the bar code listing on page 205 or the annotated listing on page 76. (This example assumes you have entered the XBC program from the bar code on page 207 or the listing on page 171, and that you are using the default parameters described in the note at the top of page 121.)

Keystrokes	Display	
1 ENTER↑ 51 MOVE	51.0000	Positions pen for next row.
ALPHA SET ■ STO · X ALPHA	ASTO _ SET	Enters SET in X.
2.10	2.10 _	Specifies row 2 and 10 bytes in X and lifts SET to Y.
XEQ ALPHA PBC ALPHA	PBC _ 3.100	Plots and labels a 10-byte version of row 2. After plotting row, PBC automatically increments row counter by 1. SET remains in the Y-register.
1 ENTER↑ 63 MOVE	63.000	Positions pen for next row.
R↓ R↓	3.100	Returns row parameters to X- and Y-registers.
XEQ ALPHA PBC	PBC _	Plots and labels a 10-byte version of row 3.
ALPHA	4.100	Increments row counter by 1.
1 ENTER↑ 75 MOVE	75.000	Positions pen for next row.
R↓ R↓	4.100	Returns program label to Y-register and row parameter to X-register.
← 8.1	8.1 _	Switches row parameter from 4 to 8 and maintains same byte length (10).
XEQ ALPHA PBC ALPHA	PBC _	Plots and labels row 8. Because last program line occurs before size specified for row (10 bytes) is reached, the row is less than 10 bytes long.
	0.000	Indicates completion of the last row of the program.
■ FIX 4	0.0000	

Note: Because the rows generated by this example do not begin with row 1 and are, in the case of row 8, nonsequential, a checksum error message is generated when you attempt to scan these rows. To bypass a checksum error that results from scanning a row that is not in the numeric sequence expected by the wand, press SST one or more times, as needed, until the number of the row you want to scan is displayed. (Refer to Recovery: Program bar code, under **W:CKSUM ERR** in appendix A of the *HP 82153A Wand Owner's Manual*.)

ROW 2: LINES 2-5

ROW 3: LINES 6-10

ROW 8: LINES 26-28

Computation Time. When using PBC, if the first row you plot is not row 1, or if the rows you select are not sequential, several seconds or more may elapse between the time you execute PBC and when the plotter begins plotting the selected row. This is because the system must compute the checksums* for all preceding rows in order to determine the running checksum for the row you want to plot. For this reason, if you generate a series of nonsequential bar code rows, you should proceed from the lowest-numbered to the highest-numbered row.

The Row and Byte Parameters. The row number (***rrr***) and byte length (***bb***) conditions used by the PBC subroutine are identical to those described for printing PLOTBC program bar code.† If you specify "private" bar code by using a negative ***rrr.bb*** value, ***rrr*** is decremented when you execute PBC. If the absolute value of ***rrr*** exceeds the number of rows computed by the module for the program named in the Y-register, no plotting takes place. Instead, PBC replaces the ***rrr.bb*** value in the X-register with a zero and enters a zero in the Y-register. (The program name is lifted to the Z-register.)

When you are using PBC to generate a series of rows, you can vary the length of successively higher-numbered rows by changing the byte parameter (***bb***) between generation of any given row and the succeeding row.

> **Note:** When plotting rows of program bar code in nonsequential order, you can plot rows numbered lower than rows you have already plotted as long as you do not change the number of bytes per row. When plotting rows in sequential order, you can change the byte length for rows that have not yet been plotted. However, if you wish to change the number of bytes for rows you have already plotted or rows numbered lower than other rows that you have already plotted, you must replot all rows in the program. Otherwise a checksum error will result when you scan the rows. This is because the checksum for any row is based on the running checksum accumulated for all sequentially earlier rows in the program (even if those rows have not actually been plotted).

The Program Name Parameter. A number or an Alpha null string (a string of blank Alpha characters) used in the Y-register instead of a program name specifies the last program in HP-41 memory. (If you pack program memory by pressing ■ GTO . . after entering the last program in memory, the permanent •END• placed at the end of memory by ■ GTO . . becomes the "last program.")

---

*The system always maintains the running checksum computed to the last row specified by execution of PBC. Thus, if you print any series of rows in ascending order, the system has only to compute the running checksum from the last row specified by PBC to the next row specified by the next successive execution of PBC. If any two successive executions of PBC result in a descending order of row numbers, the system must recompute the running checksum from the beginning of the program, which means added calculation time.

†Refer to the information on page 120 under the following three headings: Program Bar Code Generation, Row Parameters, and Byte Parameters.

Editing and Packing a Program Between Executions of PBC. You can edit any line in a program only if that line comes later in memory than the last row plotted. When you do so, execute PACK or ■ GTO ·· to pack program memory before you resume generating bar code. If you edit a part of a program that is within or prior to the highest-numbered row already plotted, you must plot the row containing the edited material and all subsequent rows. Otherwise, the running checksum will be in error for all rows numbered higher than the row containing the edited material.

When you execute PBC, if the program you specify is not already packed, the HP-41 packs the program. (Refer to the PACK function in your HP-41 owner's handbook.) If this occurs, simply reexecute PBC when **TRY AGAIN** is displayed.

Basic Bar Code Functions

The following text describes the bar code functions used by PLOTBC and the bar code subroutines. Using these functions to generate bar code is a two-step process. The first step determines and stores in the ALPHA register a bit pattern for the desired row. The second step plots the row at the current physical pen position (or prints the row if the HP 82162A Thermal Printer is being used with the plotter module). The basic bar code functions do not generate labels for bar code rows.

Learning How to Plot or Print Bar Code

Five of the next six functions described in this section are used to generate both plotter and printer versions of HP-41 bar code. The examples provided with the descriptions of these functions apply to plotter operation. Examples that use these functions for printer operation are provided later, under Printing HP-41 Bar Code on the HP 82162A Thermal Printer. Thus, if you are using a plotter, you can read through the descriptions and perform the examples in the same way that you have done elsewhere in this manual. If you are using an HP 82162A Thermal Printer to generate bar code, you should read the function descriptions (except that for BC) but can ignore the plotter examples.

Setup for Examples

Before you use the bar code functions with your plotter, ensure that there is sufficient space in the plotting area to print the bar code. The easiest way to do this is to set the default graphic limits by turning your plotter off, then on, and executing PINIT. The following keystrokes set up the plotter system for subsequent examples in this section.

Keystrokes	Display	
PINIT	0.0000	Initializes plotter module.
0 PEN	0.0000	Returns pen to stall.
■ FIX 4	0.0000	Sets display mode to FIX 4.

Standard HP-41 Data and Program Bar Code

BC ALPHA *bit pattern*

The BC (*plot standard HP-41 barcode*) function uses the bit pattern placed in the ALPHA register by the next five functions (BCX, BCXS, BCA, BCAA, or BCP) to plot a single row of HP-41 data or program bar code. When BC plots the row, it plots the standard HP-41 directional bars at the beginning and end of the row.

BC does not alter the contents of the stack, ALPHA, or LAST X registers. (The descriptions of the following five functions include examples of BC operation.)

Depending on pen width, [BC] can plot bar code rows having up to 128 bars (16 bytes), which is the maximum row length for HP-41 applications.

The [BC] function also plots HP-41 paper keyboard and direct execution bar code. Generating these two types of bar code requires more advanced bar code techniques, which are discussed later in this section.

[BCX]	X *data*	Numeric or Alpha

The [BCX] (*bar code for data in X*) function places in the ALPHA register a bit pattern for nonsequenced bar code representing the data in the X-register. Following [BCX] execution, the X-register shows the number of bytes in the bit pattern. The LAST X register contains a copy of the data originally in the X-register. The contents of the Y-, Z-, and T-registers are unaffected. Alpha data is limited by the X-register to a maximum of six characters and is printed as Alpha-Replace (type 7)*† bar code. Numeric data is printed as numeric data (type 6)† bar code.

Example: Place a fresh page in your plotter and generate one row each of numeric and Alpha data bar code for the X-register. Use [BCX] to create the bit patterns and use [BC] to actually generate the bar code. (This example assumes that the graphic limits and plotting area are set as described under Setup for Examples on page 127.)

Keystrokes	Display	
45 [SIN]	0.7071	Places data in X.
[BCX]	7.0000	Places bit pattern for 0.7071‡ in ALPHA and replaces data in X with a value indicating the number of bytes in the bit pattern.
[ALPHA]	▒▒▒7▒▒	Displays bit pattern placed in ALPHA by [BCX].
[ALPHA]	7.0000	Returns display to X.
1 [PEN]	1.0000	Selects pen 1.
5 [ENTER↑] [MOVE]	5.0000	Positions pen for next row.
[BC]	5.0000	Generates row of numeric data bar code from bit pattern in ALPHA.
[ALPHA] TEST [ASTO]	ASTO __	Places ALPHA data in X.
[·] X [ALPHA]	TEST	
[BCX]	6.0000	Places the bit pattern for TEST in ALPHA. Displayed X shows the number of bytes generated.
5 [ENTER↑] 15 [MOVE]	15.0000	Positions pen for next row.
[BC]	15.0000	Generates row of Alpha data bar code from bit pattern in ALPHA.
0 [PEN]	0.0000	Returns pen to stall.

(Leave the current page in your plotter for the next example.)

*When you scan a row of Alpha-Replace bar code, the data contained in the row replaces the current contents of the ALPHA register.

†You can ignore the type number unless you need to use the utility bar code functions described later in this section and the charts in appendix F to design the byte structure of bar code for specialized applications.

‡Because the HP-41 computes this value to nine fractional digits—five of which are not displayed due to the current [FIX] 4 display setting—the actual number coded in the bit pattern contains nine fractional digits.

Note for Advanced Users: When required, BCX automatically places *filler bits*—1010—at the beginning of numeric bar code bit strings. (Filler bits are required if the data in the X-register produces a bit pattern that is not a multiple of 8 bits.)

BCXS	Y	*sequence number*	$0 \leqslant s \leqslant 999$
	X	*data*	Numeric or Alpha

The BCXS (*sequenced bar code for data in X*) function places in the ALPHA register a bit pattern for sequenced bar code representing the data in the X- and Y-registers. Following BCXS execution the X-register contains a value indicating the number of bytes in the bit pattern. The Y-register contains the *next* sequence number in the series. The LAST X register contains a copy of the data originally in the X-register. The Z- and T-registers are unaffected.

Note: BCXS-generated bar code is designed to be scanned by revision F and later versions of the HP 82153A Optical Wand. The first marketed version of the wand was revision E. There is no functional difference between revisions E and F except the addition of the sequenced data bar code type. To determine the revision letter of your wand, remove all modules from your HP-41, then plug in the wand and press ■ CATALOG 2. The revision letter will appear in the display. Attempting to read sequenced data bar code with a revision E wand results in a **TYPE ERROR** or improper operation that includes nonstandard data.

Example: Generate two rows of numeric data bar code that you can store sequentially in your HP-41's data storage registers. (This example uses the plotting area established by the keystrokes under Setup for Examples on page 127.)

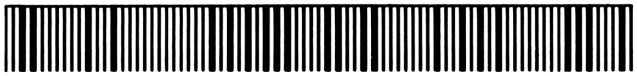
Keystrokes	Display	
1 PEN	1.0000	Selects pen 1.
10 ENTER↑ 25 MOVE	25.0000	Moves pen to next position.
0 ENTER↑ 30 COS	0.8660	Places a number in X.
BCXS	9.0000	Places bit pattern for 0.8660* in ALPHA, places sequence number in Y, and replaces data in X with a value indicating the number of bytes in the bit pattern.
BC	9.0000	Generates row of numeric-sequential data bar code from bit pattern in ALPHA.
x≷y	1.0000	Displays incremented sequence number.
60 COS	0.5000	Enters data in X and lifts sequence number to Y.

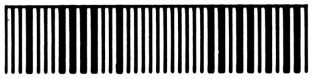
*Refer to the third footnote on page 128.

Keystrokes	Display	
BCXS	4.0000	Places bit pattern for 0.5000 in ALPHA, increments sequence number in Y, and replaces data in X (0.5000) with size of bit pattern.
10 ENTER↑ 35 MOVE	35.0000	Moves pen to next position.
BC	35.0000	Generates row of numeric-sequential data bar code from bit pattern in ALPHA.
R↓ R↓ R↓	2.0000	Displays incremented sequence number.
0 PEN	0.0000	Returns pen to stall.

(Leave the current page in your plotter for the next example.)

When you want the data from BCXS-generated bar code to be stored in a series of data storage registers, use the wand's WNDDTX function. When you want such data placed in the X-register, simply scan the bar code.





Alpha Data Bar Code Functions

BCA ALPHA *Alpha data*

The BCA (*Alpha-Replace bar code*) function converts the current data in the ALPHA register to an Alpha-Replace bar code bit pattern. If there are more than 14 characters in ALPHA when you execute BCA, only the last 14 are used. (If the ALPHA register contains only nulls, BCA generates a row of bar code containing a two-byte header and a one-byte null string.) BCA also copies the X-register value into LAST X and places in X a value indicating the number of bytes in the bit pattern. The contents of the Y-, Z-, and T-registers are unaffected. Scanning BCA-generated bar code replaces the contents of the ALPHA register with the ALPHA data in the bar code.

Example: Generate a row of Alpha-Replace bar code. (This example uses the plotting area established by the keystrokes under Setup for Examples on page 127.)

Keystrokes	Display	
1 PEN	1.0000	Selects pen 1.
15 ENTER↑ 45 MOVE	45.0000	Moves pen to next row position.
ALPHA YOUR SPACE	YOUR _	Places data in ALPHA.
DATA ALPHA	45.0000	
BCA	11.0000	Replaces data in ALPHA with corresponding bit pattern, and replaces contents of X with size (bytes) of bit pattern.
BC	11.0000	Generates a row of Alpha-Replace bar code from bit pattern in ALPHA.
0 PEN	0.0000	Returns pen to stall.

(Leave the current page in your plotter for the next example.)

BCAA	ALPHA	*alpha data*

The BCAA (*Alpha-Append bar code*) function operates in the same way as the preceding BCA function. However, scanning BCAA-generated bar code appends the data contained in the bar code to the current contents of the ALPHA register instead of replacing the contents.

Program Bar Code Function

BCP	Y	*program name*
	X	±*rrr.bb*

The BCP (*program barcode*) function generates for row ***rrr*** of the specified program in the ALPHA register a bar code bit pattern ***bb*** bytes in length. A negative ***rrr.bb*** value specifies "private" bar code. The absolute value of the current ***rrr*** is then incremented to the next sequential row number. The numbers of the first and last program lines included in that row are placed in the Z-register in an ***fff.lll*** format (includes lines carried over from the last row or to the next row). The number of bytes generated in the bit pattern is placed in the T-register.

T	*bb*	ALPHA	*Bit Pattern*
Z	*fff.lll*		
Y	*Program Name*		
X	±*rrr.bb*		

If ***rrr*** specifies the last row of the program, BCP replaces ***rrr.bb*** with zero. If ***rrr*** specifies a nonexistent row, BCP places zeroes in the X-, Z-, and T-registers, and places a one-byte null string in the ALPHA register. The program name remains in the Y-register.

As with the PLOTBC program and the PBC subroutine, before you use BCP to generate a program row bit pattern, you should first pack the program by executing ■ GTO · ·.

Example: Plot, one by one, the rows of program bar code for the TERM program used earlier in this manual. (If this program is not currently in your HP-41, you can load it using the bar code on page 205 or the annotated listing on page 77.)

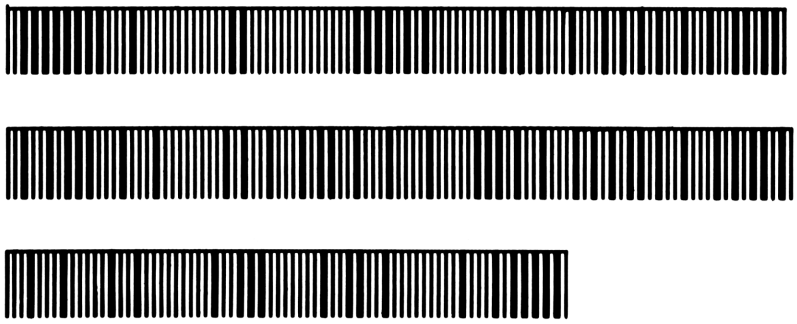
Keystrokes	Display	
1 PEN	1.0000	Selects pen 1.
5 ENTER↑ 60 MOVE	60.0000	Positions pen for next row.
ALPHA TERM ■ ASTO	ASTO _ _	Places program name in X.
· X ALPHA	TERM	
1.11	1.11 _	Enters ***rrr.bb*** in X and program name in Y.

You are now ready to execute [BCP] to generate in the ALPHA register the bit pattern for the first row. When you do so, the stack will contain the values described and illustrated on the preceding page.

Keystrokes	Display	
[BCP]	2.1100	Generates in ALPHA the bit pattern for row 1 of TERM program. Displays next row number and row length.
[R↓]	TERM	Rolls down stack to show program label from Y-register.
[R↓]	1.0010	Rolls down stack to show data from Z indicating bit pattern contains data for program line 1.
[R↓]	11.0000	Rolls down stack to show data from T indicating there are 11 bytes in the bit pattern.
[R↓]	2.1100	Rolls down stack to return data for next row to X. (All values returned to their original stack registers.)

Now plot the bar code for row 1 from the bit pattern currently in the ALPHA register by executing [BC]. Then reexecute [BCP] as needed to generate in the ALPHA register the bit patterns for rows 2 and 3. Because row 3 is the final row of the TERM program, 0.0000 will be displayed to indicate that no higher-numbered rows can be generated for this program. Also, because this last row does not require a full 11 bytes to complete the program, the T-register will contain a number indicating fewer than 11 bytes.

Keystrokes	Display	
[BC]	2.1100	Plots row 1 of TERM program. Does not affect display or any values in the stack.
5 [ENTER↑] 70 [MOVE]	70.0000	Positions pen for next row. Values remaining in Z and T for row 1 bit pattern are lost.
[R↓] [R↓]	2.1100	Rolls down ***rrr.bb*** and program name to X and Y, respectively.
[BCP]	3.1100	Generates in ALPHA the bit pattern for row 2 of TERM program. Displays next row number and row length.
[R↓]	TERM	Program label remains in stack.
[R↓]	2.0060	Indicates that bit pattern for this row contains data for program lines 2 through 6.
[R↓]	11.0000	Indicates that bit pattern contains 11 bytes.
[R↓]	3.1100	Returns all values to their original stack registers.
[BC]	3.1100	Plots row 2 of TERM program.
5 [ENTER↑] 80 [MOVE]	80.0000	Positions pen for next row.
[R↓] [R↓]	3.1100	Rolls down ***rrr.bb*** and program name to X and Y.
[BCP]	0.0000	Generates in ALPHA the bit pattern for row 3 of TERM program. Displayed zeroes indicate that row 3 bit pattern contains the program's last row.
[BC]	0.0000	Plots program's last row.
[PEN]	0.0000	Returns pen to stall.



BCP is used in the PBC subroutine described on page 124. Thus, BCP uses parameters in the same way as PBC and generates single rows in a manner similar to that of PBC and to the program bar code aspects of PLOTBC—which itself uses PBC.

To generate bit patterns for random rows, change the ***rrr*** portion of the ***rrr.bb*** value remaining in the X-register after each execution of BCP. This feature is useful if you have to replot a row that was damaged or plotted with a pen having a depleted ink supply.

When printing nonsequential program bar code rows, the time considerations are the same as those described under Computation Time on page 126.

Printing HP-41 Bar Code on the HP 82162A Thermal Printer

The HP 82184A Plotter Module enables you to print any type of HP-41 bar code on an HP 82162A Thermal Printer. To print bar code, you need only your HP-41, the plotter module, and the following three items:

- HP 82162A Thermal Printer.
- HP 82175A Black Print Thermal Paper.
- HP 82160A HP-IL Module.

To print most types of HP-41 bar code, use the basic bar code functions described earlier in this section (pages 127 through 133) and the printer option of the BCO function described following this introductory information. To see how easy it is to print bar code, connect your HP-41 and printer to the interface loop, place a roll of black thermal paper in the printer, and step through the following example. (This example is intended to demonstrate the plotter module's bar code printing capability rather than instruct in how to print bar code. The material following the example describes the details of bar code printing.)

> **Note:** Unless you are printing program bar code, the I/O buffer is not used for bar code printing operations on the HP 82162A Thermal Printer. This topic is described on the next page, under Using BCO to Print Standard HP-41 Program Bar Code.

Bar Code Printing Example. Print a row of numeric data bar code containing the approximate speed of light in a vacuum (kilometers/second) and a row of Alpha data bar code containing the characters "HP-41 bar code".

Keystrokes	Display	
299792.5	299,792.5 _	Keys in the approximate speed of light.
BCX	6.0000	Places bit pattern in ALPHA and leaves number of bit pattern bytes in X.
CHS	-6.0000	Converts byte value to a negative number. (The negative sign specifies thermal printer bar code.)
BCO	-6.0000	Prints a row of bar code containing the speed of light.
ALPHA HP ■ -	HP- _	
■ 4 ■ 1 SPACE	HP-41 _	Enters the characters "HP-41" in ALPHA.

Keystrokes	Display	
BAR [SPACE] CODE	41 BAR CODE_	
[ALPHA]	-6.0000	
[BCA]	16.0000	Replaces characters in ALPHA with a bit pattern containing the characters. Places in X the number of bytes in the bit pattern.
[CHS]	-16.0000	Converts byte value to a negative number.
[BCO]	-16.0000	Prints a row of bar code containing the characters "HP-41 BAR CODE".

As you can see, printing bar code on the HP 82162A Thermal Printer is a short, quick procedure. Now let's learn more about the [BCO] function you used in the preceding example to print the bar code.

Using [BCO] to Print Standard Data and Program Bar Code

The [BCO] function provides you with a *printer option* for printing HP-41 bar code, and a *plotter option* for plotting both HP-41 bar code and non HP-41 bar code. The following material describes how to use the [BCO] function's printer option to print standard HP-41 data and program bar code.

[BCO] (Printer Option) X [*-bb*] ALPHA [*bit pattern*]

Executing the printer option of the [BCO] (*bar code options*) function prints on the HP 82162A printer a row of HP-41 bar code corresponding to the bit pattern in the ALPHA register. The absolute value of the negative number in the X-register (|-***bb***|) tells [BCO] how many bytes of HP-41 bar code to print. The value |-***bb***| must be in the range $1 \leqslant |-bb| \leqslant 16$.

Using [BCO] to Print Standard HP-41 Data Bar Code. The four data bar code functions ([BCX], [BCXS], [BCA], and [BCAA]) described on pages 128 through 133 each place a bit pattern in the ALPHA register and the number of bytes (***bb***) for the bit pattern in the X-register. After you execute one of these functions to generate the bit pattern and number of bytes, perform the following two steps to print on the HP 82162A Thermal Printer the row of bar code corresponding to that bit pattern:

1. Change the byte number (***bb***) in the X-register to a negative value by executing [CHS].
2. Execute [BCO].

The preceding example demonstrates how to use this procedure.

Using [BCO] to Print Standard HP-41 Program Bar Code. The [BCP] function described on pages 131 through 133, generates the data used by [BCO] to print program bar code. Since [BCP] uses the I/O buffer (described on pages 10 and 11), create the buffer by executing [PINIT] before you use [BCP]. *Unless a plotter is plugged into the interface loop, you must set flag 25 (the Error Ignore flag) before executing* [PINIT]. Otherwise, the **NO PLOTTER** message appears and the buffer will not be created.

To generate on the HP 82162A Thermal Printer a row of program bar code:

1. Generate a bit pattern for the desired row by executing [BCP].
2. Recall from the T-register the byte number (***bb***) generated by [BCP].
3. Change ***bb*** to a negative value by executing [CHS].
4. Execute [BCO].

Printed Program Bar Code Example. Print the bar code for the TERM program listed on page 77 of this manual. (If the TERM program is not currently in your calculator, enter it before you proceed with this example.)

Keystrokes	Display	
■ GTO ··		Ensures that program is packed before you begin generating bit patterns.
■ SF 25 PINIT		Creates I/O buffer.

The first step is to set up the HP-41 for execution of BCP, which generates in the ALPHA register the bit pattern for the program row. To do so, place the program label in the Y-register and the beginning row number and byte length (***rrr.bb***) in the X-register. Then execute BCP.

Keystrokes	Display	
ALPHA TERM	TERM _	
■ ASTO · X	TERM	Enters program name in X.
ALPHA	TERM	
1.12	1.12 _	Specifies program row 1 with a length of 12 bytes.
BCP	2.1200	Generates row 1 bit pattern and increments ***rrr*** for row 2.

Following execution of BCP, the ***bb*** value needed to execute BCO is in the T-register. Recall ***bb***, change it to a negative value, and print the row by executing BCO. Then return ***rrr.bb*** to the X-register in preparation for printing the next row.

Keystrokes	Display	
RCL · T	12.0000	Recalls ***bb*** from T.
CHS	-12.0000	Converts ***bb*** to ***-bb***.
BCO	-12.0000	Prints first row of bar code.
R↓	2.1200	Returns ***rrr.bb*** to X.

Now generate the rest of the program's bar code.

Keystrokes	Display	
BCP	3.1200	Generates row 2 bit pattern and increments ***rrr*** for row 3.
RCL · T	12.0000	Recalls ***bb*** from T.
CHS	-12.0000	Converts ***bb*** to ***-bb***.
BCO	-12.0000	Prints second row of bar code.
R↓	3.1200	Returns ***rrr.bb*** to X.
BCP	0.0000	Generates row 3 bit pattern. Because row 3 is the last row in the program, value in X is replaced by zero.
RCL · T	6.0000	Recalls ***bb*** from T.
CHS	-6.0000	Converts ***bb*** to ***-bb***.
BCO	-6.0000	Prints last row of bar code.
←	0.0000	Clears display.

Note: When you execute BCO to print a row, if you use for |***-bb***| a value other than that computed by that row's basic bar code function, the resulting bar code may be unreadable or interpreted incorrectly by the wand. That is, if INT|***-bb***| is not equal to the number of bytes in the ALPHA register when you execute BCO, the length of the printed row will not correspond to the number of bytes in ALPHA. Because the checksums in such rows are not likely to correspond to the printed bit pattern, the wand will usually not read such rows except under control of the WNDSCN or WNDTST functions. However, if such a row should correspond to the bit pattern of any legitimate bar code other than that specified in the ALPHA register, the row will be readable without executing WNDSCN or WNDTST. Except under controlled operating conditions, this could introduce an error that would not be detected by the HP-41. If there are more than |***-bb***| bytes of data in the ALPHA register, only the last |***-bb***| bytes will be printed.

Applications Program for Printer-Generated Program Bar Code. The PRBC program included in this manual prompts you for the name of a program you have entered in the HP-41's memory, then prints the bar code for that program (including row labels). PRBC limits bar code row length to nine bytes so that the rows can be conveniently mounted on standard 8½-inch pages. For a listing and description of PRBC, refer to pages 172 and 173. Bar code for PRBC is on pages 202 and 203.

Printing Paper Keyboard and Direct Execution Bar Code

HP-41 paper keyboard and direct execution bar code is printed using the printer option of the BCO function in the same way that it is used for printing HP-41 data and program bar code. However, determining the bit pattern for paper keyboard and direct execution bar code is a more complex process that involves the use of appendix F, Bar Code Charts, and execution of the BCREGX and BCCKSM functions, which are described next.

Utility Bar Code Functions

The utility bar code functions enable you to plot paper keyboard and direct execution bar code. They also give you the flexibility to plot specially designed HP-41 bar code (used with WNDSCN) and design alternative bar code for devices used in other scanning systems.

The bar code functions described in the following text involve, on the user level, some of the more technical aspects of bar code generation. For this reason you may need more time to study these functions than was necessary for the bar code functions described earlier.

In the following discussion, "first byte" always refers to the leftmost byte displayed in the ALPHA register; the "last byte" refers to the rightmost byte.

Generating the Bit Pattern and Checksum Separately

The following two functions are designed for plotting HP-41 paper keyboard and direct execution bar code, but can also be used to plot non HP-41 bar code rows.

BCREGX X [*iii.fff*]

The BCREGX (*bar code from registers according to X*) function transforms the values in the specified storage registers to characters in the ALPHA register. These characters hold the bit patterns which correspond to the values in those registers. The leftmost character holds the bit pattern for the value in the lowest numbered register (***iii***), and the rightmost character holds the bit pattern for the value in the highest numbered register (***fff***). BCREGX uses the control value in the X-register, ***iii.fff***, to build bit patterns in the ALPHA register corresponding to the values in registers ***iii*** through ***fff***.

- If ***fff*** > 0, BCREGX uses the data in R_{iii} through R_{fff}.
- If ***fff*** = 0, BCREGX uses the data in R_{01} through R_{iii}.
- If ***iii*** > ***fff***, BCREGX uses only the data in R_{iii}.
- If ***iii*** = 0 and ***fff*** > 0, then BCREGX uses the data in R_{00}.

Each value in a data register corresponds to one byte in the bit pattern. Since these values are converted to eight-bit binary code in ALPHA, the values you use with BCREGX must be in the range 0 through 255. (Fractional portions of values in the data registers are ignored.) Unlike the bar code generation functions covered earlier, BCREGX does *not* generate a checksum for the bit pattern placed in the ALPHA register. The function described next, BCCKSM, is designed specifically for this purpose. Executing BCREGX does not alter the values currently in the stack.

Example: Using [BCREGX] with [BC], plot a five-byte row of arbitrary HP-41 bar code. Then, if you have an HP 82153A Optical Wand, plug it into your HP-41, execute [WNDSCN], and scan the row. (Because a bar code type and checksum are not included in this row, you will be able to read the row only by executing [WNDSCN] or [WNDTST].) Before you begin, position the pen at a convenient location for plotting a row of five bytes.*

Keystrokes	Display	
1 [PEN]	1.0000	Selects pen 1.
5 [ENTER↑] [MOVE]	5.0000	Moves pen to plotting position.
10 [STO] 06	10.0000	Stores 10 in R_{06}.
20 [STO] 07	20.0000	Stores 20 in R_{07}.
30 [STO] 08	30.0000	Stores 30 in R_{08}.
40 [STO] 09	40.0000	Stores 40 in R_{09}.
50 [STO] 10	50.0000	Stores 50 in R_{10}.
6.01	6.01 _	Specifies R_{06} as the initial (*iii*) register, and R_{10} as the final (*fff*) register.
[BCREGX]	6.0100	Places in ALPHA a bit pattern for the data you stored; one byte for each data register position.
[BC]	6.0100	Plots the data as HP-41 bar code.
0 [PEN]	0.0000	Returns pen to stall.
[WNDSCN]	W: READY	Prompts you to scan a row of bar code. (This step requires that a wand be plugged into your HP-41.)
(scan the row)	5.0000	Displays the number of bytes read.
[RCL] 01	10.0000	Displays decimal equivalents of the five bytes in the scanned row.
[RCL] 02	20.0000	
[RCL] 03	30.0000	
[RCL] 04	40.0000	
[RCL] 05	50.0000	

(If you are continuing on to the next example, leave the current page in your plotter.)

Note: If there are more than 16 bytes of data in the ALPHA register, [BC] generates a bar code row containing only the last (rightmost) 16 bytes. (This is the maximum row length that the HP 82153A Optical Wand can read. However, you can use the plotter option of the [BCO] function—described later in this section—to generate alternate types of bar code containing up to 24 bytes in a row. Refer to Plotting Alternative (Non HP-41) Bar Code, page 143.)

[BCCKSM] X ***bb*** ALPHA ***bit pattern***

The [BCCKSM] (*compute bar code checksum*) function computes the checksum of the bit pattern represented by the characters in the ALPHA register. To use [BCCKSM], place the desired bit pattern in the ALPHA register (using [BCREGX]), key the number of bytes (***bb***) of that pattern into the X-register, and execute [BCCKSM]. [BCCKSM] computes the checksum for ***bb*** bytes and places it as a bit pattern in byte ***bb*** (*counting from the right*).

[BCCKSM] can be used with [BCREGX] and [BC] to plot HP-41 paper keyboard and direct execution bar code. (Other bar code functions, such as [BCX], automatically include a checksum in the bit pattern.) However, you can also use [BCCKSM] with the plotter option of the [BCO] function (described later in this section) to plot non HP-41 bar code. Executing [BCCKSM] alters only the contents of the ALPHA register.

*To print the bar code from this example on the HP 82162A Thermal Printer, ignore the pen movement instructions and replace the [BC] function with the following keystrokes: 5 [CHS] [BCO].

The value of ***bb*** placed in the X-register to produce standard HP-41 paper keyboard and direct execution bar code determines which of the following three types of checksum are calculated:

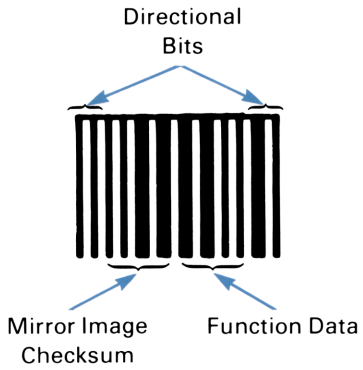
- Four-bit mirror checksum if ***bb*** = 1.
- Four-bit summation checksum with end-around carry if ***bb*** = 2.
- Eight-bit summation checksum with end-around carry if ***bb*** ⩾ 3.

The type of checksum calculated depends upon the row length of the bar code you are generating.

Four-Bit Mirror Checksum

The four-bit mirror image checksum is normally used with one-byte paper keyboard bar code. The structure of this type of bar code is described in the One-Byte Paper Keyboard chart in appendix F, Bar Code Specification Charts.*

To specify the checksum for a one-byte paper keyboard bit pattern in the ALPHA register, place 1 in the X-register and execute [BCCKSM]. A mirror image of the rightmost four bits will be placed in the leftmost four bits as the checksum.



The four-bit mirror checksum is placed in the leftmost four bits of the rightmost byte in the ALPHA register.† (These four bits don't hold any other information essential to the paper keyboard function.)

Example: Plot the one-byte paper keyboard bar code for the HP-41 [CHS] function. To generate the [CHS] bit pattern, convert the binary representation of [CHS] (1100—found in the One-Byte Paper Keyboard Bar Code chart in appendix F) to its decimal value (12) and store it in R_{01}. Enter 1 into the X-register and execute [BCREGX]. Then, with 1 in the X-register to specify the four-bit mirror checksum, execute [BCCKSM]. To see the effect of [BCCKSM] on the bit pattern, plot the bit pattern both before and after you generate the checksum.‡

---

*This type is also described in section 1, Description of Bar Code Types, in *Creating Your Own HP-41 Bar Code*, part number 82153-90019. This Hewlett-Packard technical manual provides a base for generating HP-41 bar code on computer printer/plotter systems. For further information, contact your authorized Hewlett-Packard dealer. In the U.S.A., refer to Dealer and Product Information on page 159.

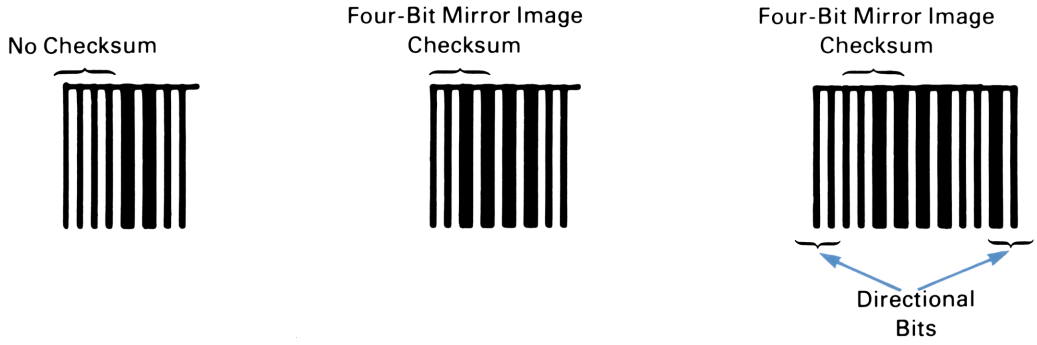
†Executing [BCCKSM] with 1 in the X-register always generates a four-bit mirror checksum for the rightmost byte, regardless of how many non-null bytes are currently in the ALPHA register.

‡To print the bar code from this example on the HP 82162A Thermal Printer, use the following keystrokes instead of those shown in the example: 12 [STO] 01 1 [BCREGX] [BCCKSM] [CHS] [BCO]. (Using a plotter for this example produces three rows; using a printer produces one row.)

Keystrokes	Display	
1 [PEN]	1.0000	Selects pen 1.
5 [ENTER↑] 15 [MOVE]	15.0000	Position pen to new row.
12 [STO] 01	12.0000	Stores decimal representation of [CHS] in R_{01}.
1 [BCREGX]	1.0000	Computes and stores in ALPHA a bit pattern representing the data in R_{01}.
0 [ENTER↑] [ENTER↑]	0.0000	
1 [BCO]	1.0000	Plots the bar code pattern before the checksum is calculated.
[BCCKSM]	1.0000	Calculates the four-bit mirror image checksum and inserts it into the existing pattern.
15 [ENTER↑] [MOVE]	15.0000	Positions pen for next row.
0 [ENTER↑] [ENTER↑]	0.0000	
1 [BCO]	1.0000	Plots bar code for [CHS] bit pattern in the ALPHA register. The bar code has the checksum included, but has no directional bits.
25 [ENTER↑] 15 [MOVE]	15.0000	Positions pen for next row.
[BC]	15.0000	Plots bar code for [CHS] bit pattern in ALPHA with directional bits.
0 [ENTER↑] [MOVE] [PEN]	0.0000	Removes pen from plotting area.

(If you are continuing on to the next example, leave the current page in your plotter.)

The rows you plotted in the preceding example should look like this:



Note: If you use this type of checksum for bar code other than that described under One-Byte Paper Keyboard Bar Code on page 216 in appendix F, the checksum will not be correct. However, such bar code can be scanned using the [WNDSCN] function.

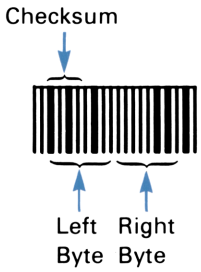
Four-Bit Summation Checksum

The four-bit summation checksum is normally used with two-byte paper keyboard bar code. The structure of this type of bar code is described in the Two-Byte Paper Keyboard chart in appendix F, Bar Code Specification Charts.*

Four-Bit Summation Checksum Procedure. To generate the checksum for a two-byte paper keyboard bit pattern in the ALPHA register, enter 2 in the X-register and execute [BCCKSM]. This creates a four-bit summation checksum (with end-around carry) that is placed in the leftmost four bits of the left byte. †

*Refer to the first footnote on page 138.

† [BCCKSM] calculates the checksum for the rightmost ***bb*** bytes in the bit pattern. For example, if there are five bytes in the ALPHA register and 2 (***bb***) is in the X-register when [BCCKSM] is executed, only the two rightmost bytes will be used for calculating the checksum. The checksum is then placed in the second (***bb***) byte from the right.



This type of checksum replaces the leftmost four bits in the leftmost byte with the checksum of the two-byte bit pattern. As with the four-bit mirror image checksum, the four-bit summation checksum replaces bits that, in functions included in the HP-41 paper keyboard, are zeroes.* If more than two bytes are in the ALPHA register, then the checksum will be placed in the second byte from the right. In such cases, the resulting HP-41 bar code can be read only by executing [WNDSCN] or [WNDTST].

> **Note:** The first byte should have a decimal value less than or equal to 15 to ensure that its leftmost four bits are 0. When you execute [BCCKSM], these four bits are added into and replaced by the checksum. If these four bits are 0, the checksum is correct. If these four bits have some other value, the checksum will be incorrect.

The following two examples demonstrate how to plot two-byte paper keyboard bar code with four-bit summation checksums.

Example: Plot the bar code for the HP-41 [BEEP] function. From item 1 (A through C) in the Two-Byte Paper Keyboard chart in appendix F (page 216), zero is the value of the first byte. Usually the decimal value of the first byte should be stored in the first data register of the series. But since, in this case, the first byte is a null, it need not be stored.

Item 1D of the Two-Byte Paper Keyboard chart indicates that the second byte must contain the value (function code) of the particular function. The decimal value of the [BEEP] function is 134.* Key in this value and store it in R_{02}. Then key in 2.002 and execute [BCREGX] to generate the bit pattern in the ALPHA register.†

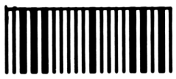
Keystrokes	Display	
1 [PEN]	1.0000	Selects pen 1.
5 [ENTER↑] 25 [MOVE]	25.0000	Positions pen for new row.
134 [STO] 02	134.0000	Stores 134 in R_{02} for the decimal equivalent of the rightmost byte in the bit pattern.
2.002 [BCREGX]	2.0020	Generates bit pattern in ALPHA from contents of R_{02}.
2 [BCCKSM]	2.0000	Calculates a four-bit summation checksum and adds it into the leftmost four bits of the left byte. (Prior to executing [BCCKSM] the left byte was an undisplayed null byte.)
[BC]	2.0000	Plots two-byte paper keyboard bar code for [BEEP] function.
0 [PEN]	0.0000	Returns pen to stall.

(If you are continuing on to the next example, leave the current page in your plotter.)

---

*Found in hexadecimal form in the HP-41 Function table on page 219. (The same table is also printed in section 4, Algorithms, in the *Creating Your Own HP-41 Bar Code* manual described in the first footnote on page 138.)

†To print the bar code from this example on the HP 82162A Thermal Printer, ignore the pen movement instructions and replace the [BC] function with the following keystrokes: [CHS] [BCO].



Example: Plot the bar code for the plotter module's BC function. As indicated under item 5 in the Two-Byte Paper Keyboard chart (page 216), you must first convert the BC function's XROM number (18,11—from the table on page 210) into a binary representation of the two-byte row you want to generate (using zeroes for the unknown checksum bits). You then convert this binary value into a pair of decimal values you can use with BCREGX to generate the necessary bit pattern in ALPHA. Before you begin, position the pen at a location convenient for plotting a two-byte row.*

	Checksum Bits	1	18					11					
		1	*f*	*f*	*f*	*g*	*g*	*h*	*h*	*h*	*h*	*h*	*h*
Binary:	0 0 0 0	1	1	0	0	1	0	0	0	1	0	1	1
Hexadecimal:	0		C					8		B			
Decimal:	12							139					

Keystrokes	Display	
1 PEN	1.0000	Selects pen 1.
5 ENTER↑ 35 MOVE	35.0000	Positions pen for new row.
12 STO 01	12.0000	Stores 12 in R_{01} for the decimal equivalent of the left byte in the bit pattern.
139 STO 02	139.0000	Stores 139 in R_{02} for the decimal equivalent of the right byte in the bit pattern.
2 BCREGX	2.0000	Generates bit pattern in ALPHA from contents of R_{01} and R_{02}.
BCCKSM	2.0000	Calculates four-bit summation checksum and adds it into the leftmost four bits of the left byte with a four-bit summation checksum.
BC	2.0000	Plots two-byte paper keyboard BC (XROM) function.
0 PEN	0.0000	Returns pen to stall.

(If you are contiuing on to the next example, leave the current page in your plotter.)

*To print the bar code from this example on the HP 82162A Thermal Printer, ignore the pen movement instructions and replace the BC function with the following keystrokes: CHS BCO.

BC and Null Bytes

BC uses the leftmost non-null (nonblank) character in the ALPHA register as the leftmost data byte in a bar code row.* Thus, BC cannot produce a row in which the first byte is null. (In all but one case, standard HP-41 bar code has a non-null checksum in the first byte. The single exception is the one-byte row which the HP 82153A Optical Wand interprets as the paper keyboard zero value. To generate this row, clear the ALPHA register, then execute BC.)

Eight-Bit Summation Checksum

A row of direct execution bar code contains three or more bytes plus an eight-bit summation checksum. This type of bar code is described in appendix F under Direct Execution Bar Code. When generating direct execution bar code, you may need to refer to one or more of the following three tables in appendix F:

- HP-41 Function Table.
- Numeric Values for A through J, a through e, X, Y, Z, T and L.
- Programmable Function Derivation.

Eight-Bit Checksum Procedure. To generate the checksum for a direct execution bit pattern of ***bb*** bytes, enter a value of ***bb*** + 1 in the X-register, then execute BCCKSM. This procedure inserts an eight-bit summation checksum as the leftmost byte in the bit pattern. Thus, to generate the checksum for a three-byte direct execution bit pattern, enter 4 in the X-register and execute BCCKSM.

Example: Plot the direct execution bar code for the PLOTBC program described at the beginning of this section. Referring to the Direct Execution Bar Code chart on page 216, you can see that direct execution bar code rows include a checksum, type indicator (40_{16}), and a function code. The function code must include the XEQ α function and the Alpha characters PLOTBC. Thus, the desired bit pattern's components are:

Checksum	Type	Unused	XEQ α	PLOTBC
8 bits	4	0	8 bits	48 bits
Byte 1	Byte 2		Byte 3	Bytes 4 through 9

The checksum is computed last. The type indicator in the second byte is 40_{16}, or 64_{10}. The remainder of the bit pattern can be computed using row 1 of the Programmable Function Derivation chart on page 218 as a guide. The HP-41 Function Table (page 219) lists XEQ α in row 1, column 14, meaning that the hexadecimal value is 1E and the decimal value is 30. The remaining values we need are the decimal values for each character in "PLOTBC". The bit pattern to place in the ALPHA register is shown below. The checksum is omitted because it is inserted later by BCCKSM.†

*The ALPHA register always contains 24 bytes of data. New data enters the ALPHA register from the right. BC (and the plotter option of the BCO function—described later in this section) ignores all null bytes to the left of the first non-null byte. Such null bytes do not appear when the ALPHA register is displayed. Each null byte to the right of the leftmost non-null byte appears as a ⁻ (overbar character) when you display the ALPHA register. If all bytes in the ALPHA register are nulls, BC generates a row of the paper keyboard bar code for zero.

†To print the bar code from this example on the HP 82162A Thermal Printer, ignore the pen movement instructions and replace the BC function with the following keystrokes: CHS BCO.

Type	XEQ α	P	L	O	T	B	C
01000000	00011110	01010000	01001100	01001111	01010100	01000010	01000011
4 0	1 E	5 0	4 C	4 F	5 4	4 2	4 3
64	30	80	76	79	84	66	67

Keystrokes	Display	
1 PEN	1.0000	Selects pen 1.
5 ENTER↑ 45 MOVE	45.0000	Positions pen for new row.
64 STO 01	64.0000	Stores 64 in R_{01} for type indicator byte.
30 STO 02	30.0000	Stores 30 in R_{02} for XEQ α function byte.
80 STO 03	80.0000	Stores Alpha character values in R_{03} through R_{08}.
76 STO 04	76.0000	
79 STO 05	79.0000	
84 STO 06	84.0000	
66 STO 07	66.0000	
67 STO 08	67.0000	
8 BCREGX	8.0000	Generates bit pattern in ALPHA from contents of R_{01} through R_{08}.
ALPHA	PLOTBC	Displays bit pattern without checksum.
ALPHA	8.0000	
9 BCCKSM	9.0000	Generates eight-bit checksum as leftmost byte and moves all other bytes one place to the right.
ALPHA	PLOTBC	Displays bit pattern with checksum. Notice that the checksum byte has been inserted.
ALPHA	9.0000	
BC	9.0000	Plots direct execution bar code for XEQ PLOTBC.
0 PEN	0.0000	Returns pen to stall.

(If you are continuing to the next example, leave the current page in your plotter.)

With the PLOTBC program loaded into program memory and the HP-41 switched out of Program mode, scan the row you just plotted—the program should begin running.

Plotting Alternative (Non HP-41) Bar Code

When plotting alternative bar code types, it may be desirable to use different directional bits than those normally obtained with BC. The plotter option of the BCO function allows you to plot bar code using bit patterns that vary widely in their geometry and interpretation.

BCO
(Plotter Option)

Z ***number of bits*** (leftmost byte)
Y ***number of bits*** (rightmost byte)
X ***bb***

ALPHA ***bit pattern***

The plotter option of the BCO (*bar code options*) function operates the same as BC except that:

- HP-41 directional bars are not automatically added to the row.
- You can specify how many rightmost bits from the first and last bytes to plot.
- Allows you to interpret bit patterns in alternative ways, depending upon the current bar code type. (Refer to the discussion of the BCSIZE function on page 111.)

Note: The *printer option* of the BCO function allows you to print HP-41 bar code on the HP 82162A Thermal Printer. (Refer to page 133 for more information.)

Thus, BCO allows you to determine your own leading and trailing bits for a plotted bar code row. This feature is most useful when you are designing non HP-41 bar code. Using BCO you can plot bar code rows having up to 192 bars (24 bytes).

Procedure. To use BCO (plotter option):

1. For a row containing ***bb*** bytes (not counting the directional bits) store in successive registers the decimal values for those bytes. (We will call these registers R_{iii} through R_{fff}.)
2. Determine the extra bits you need at the *beginning* (left end) of a bar code row containing the ***bb*** bytes.
3. Determine the decimal value of those bits.
4. Repeat steps 2 and 3 for the extra bits you need at the right end of the same row.
5. Store in R_{iii-1} (but not R_{00}) the decimal value of the byte you designed to contain the extra leading bits, and store in R_{fff+1} the decimal value of the byte you designed to contain the extra trailing bits.
6. Place in the X-register a number having the form ***hhh.ggg***, where ***hhh*** = ***iii*** − 1 and ***ggg*** = ***fff*** + 1.
7. Execute BCREGX to place the bit pattern in the ALPHA register.
8. Enter in the Z- and Y-registers the number of rightmost bits you want plotted from the bytes in R_{iii-1} and R_{fff+1}, then enter in the X-register the total number of bytes contained in R_{iii-1} through R_{fff+1} (***f*** − ***i*** + 3 bytes), and execute BCO.

(For bar code that has no leading or trailing bits you should enter 0 in both the Y- and Z-registers and the number of bytes in R_{iii} through R_{fff} (***f*** − ***i*** + 1) in the X-register.)

Example: Plot a row of bar code containing the bit pattern for "ABCD." Instead of plotting the standard HP-41 directional bits use BCO to plot the row with 101 as alternative leading directional bits and 11 as alternative trailing directional bits. To determine the values to load into your HP-41:

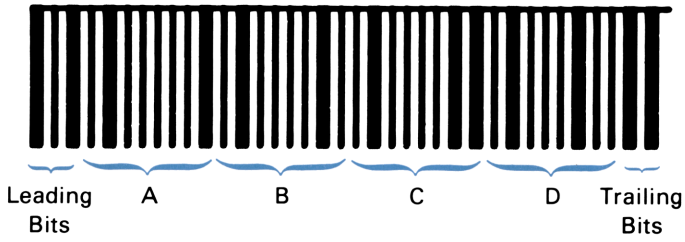
	Leading Extra Bits	A	B	C	D	Trailing Extra Bits
Binary:	(00000)101	01000001	01000010	01000011	01000100	(000000)11
Hexadecimal:	0 5	4 1	4 2	4 3	4 4	0 3
Decimal:	5	65	66	67	68	3

As indicated in the above chart, the three rightmost bits in the first byte and the two rightmost bits in the last byte are the only parts of those two bytes that should be plotted in the desired row. *Even though this is type "0" bar code, it cannot be read by the wand because it does not have the standard HP-41 directional bits.*

Keystrokes	Display	
1 [PEN]	1.0000	Selects pen 1.
5 [ENTER↑] 55 [MOVE]	55.0000	Positions pen for new row.
5 [STO] 01	5.0000	Stores in R_{01} the byte containing the desired leading bits.
65 [STO] 02	65.0000	Stores in R_{02} through R_{05} the bytes containing the Alpha character values for "ABCD".
66 [STO] 03	66.0000	
67 [STO] 04	67.0000	
68 [STO] 05	68.0000	
3 [STO] 06	3.0000	Stores in R_{06} the byte containing the desired trailing bits.
6 [BCREGX]	6.0000	Generates bit pattern in ALPHA from contents of R_{01} through R_{06}.
3 [ENTER↑]	3.0000	Specifies that three bits of leftmost byte should be plotted.
2 [ENTER↑]	2.0000	Specifies that two bits of rightmost byte should be plotted.
6 [BCO]	6.0000	Plots bar code for bit pattern in ALPHA.
0 [PEN]	0.0000	Returns pen to stall.

(If you are continuing on to the next example, leave the current page in your plotter.)

The bit pattern in the row you just plotted should match the pattern in the following row.



General Operating Procedure. Using the same general procedure described earlier in this section, you can generate any type of bar code. To do this you (1) store the decimal equivalent of each byte into successive storage registers, (2) use [BCREGX] to generate the appropriate bit pattern in the ALPHA register, (3) calculate the checksum of the bit pattern using [BCCKSM], and (4) execute [BC] to plot rows having standard HP-41 directional bits, or execute the plotter option of the [BCO] function to plot different bar codes having either directional bits that you specify or no directional bits at all.*

The bar code type used thus far in this manual (and all HP-41 bar code) is type "0" bar code which consists of uniformly spaced wide and narrow bars (as "0"s and "1"s). When you want to generate another type of bar code, reset the plotter module to that type by executing [BCSIZE] before you begin plotting the bar code rows (refer to Changing the Bar Code Size and Type, page 111).

For some types of bar code you may also want to adjust the bar code size parameters to achieve the correct width ratio between bars and spaces. In some cases this involves a process of experimenting with various parameters.

Type 1. Type 1, or *alternating*, bar code uses both spaces and bars in the bit pattern. The components of this type are:

- Wide bars or wide spaces equal "1".
- Narrow bars or narrow spaces equal "0".

*You can use the printer option of [BCO] to generate nonstandard HP-41 bar code on the HP 82162A Thermal Printer. If nonstandard bar code has the standard HP-41 directional bits, it can be read by the wand using [WNDSCN] or [WNDTST].

All bit patterns begin with a bar which may be wide ("1") or narrow ("0") and use alternating bars and spaces to show the bit pattern. That is, a bit can be either a bar or a space.

Type 2. Type 2, or *proportional*, bar code represents each bit as a bar and a space. Thus, a "1" could be represented as a 2-unit bar and a 1-unit space, and a "0" represented as a 1-unit bar and a 2-unit space—each bit is 3 units wide.

Type 3. This type is a utility bar code provided to allow advanced users to create any type of bar code. The components of this type are:

- Narrow bars equal "1".
- Narrow spaces equal "0".

It is possible to use this type designator to produce any of the preceding bar code types, as well as bar code that resembles Universal Product Code.

Example: The previous example plots a row of type 0 bar code. Now plot the same row in types 1, 2, and 3. The following table shows the [BCSIZE] parameters to use if your pen's actual width is 10 APUs. (If you are unsure of your pen's width, execute the PWIDTH program and use the indicated pen width instead of the 10 APU pen width shown in the table. However, do not alter any of the other values specified by the table.)

[BCSIZE] Parameters

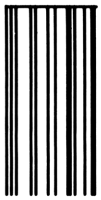
Type	Y-Register				X-Register		
	nn	*t*	*ww*	*ss*	*pp*	*hhh*	*aa*
1	15.	1	30	15	10.	350	30
2	15.	2	30	15	10.	350	30
3	12.	3	00	13	10.	999	

Before you begin, ensure that the bit pattern generated in the preceding example is in the ALPHA register.

Keystrokes	Display	
1 [PEN]	1.0000	Select pen 1.
5 [ENTER↑] 65 [MOVE]	65.0000	Position pen to new row.
15.13015 [ENTER↑]	15.1302	Sets new [BCSIZE] parameter for type 1 bar code.
10.3503 [BCSIZE]	10.3503	
3 [ENTER↑] 2 [ENTER↑]	2.0000	Enters [BCO] parameters.
6 [BCO]	6.0000	Plots the bit pattern as a row of type 1 bar code.

Keystrokes	Display	
25 [ENTER↑] 65 [MOVE]	65.0000	Moves pen to the right.
15.23015 [ENTER↑]	15.2302	Sets new [BCSIZE] parameter for type 2 bar code.
10.3503 [BCSIZE]	10.3503	
3 [ENTER↑] 2 [ENTER↑]	2.0000	Enters [BCO] parameters.
6 [BCO]	6.0000	Plots the bit pattern as a row of type 2 bar code.

Keystrokes	Display	
60 ENTER↑ 65 MOVE	65.0000	Moves pen to the right.
12.30013 ENTER↑	12.3001	Sets new BCSIZE parameter for type 3 bar code.
10.999 BCSIZE	10.9990	
3 ENTER↑ 2 ENTER↑	2.0000	Enters BCO parameters.
6 BCO	6.0000	Plots the bit pattern as a row of type 3 bar code.
0 PEN ENTER↑ BCSIZE	0.0000	Returns pen to stall and resets bar code parameter to default value.



Section 8

Plotting Conditions and the Input/Output Buffer

Contents

Introduction 148
The 26-Register I/O Buffer (PRCL) 148
Default Plotting Conditions 150

Introduction

In order to perform plotter scaling in your HP-41, as well as several other plotting functions, a variety of scale factors and other data are automatically maintained in HP-41 memory. To protect these values from being inadvertently cleared or altered by normal operations, they are placed in a buffer composed of 26 HP-41 storage registers. (Refer to Minimum Memory Requirements, page 11, and Initializing and Clearing the I/O Buffer, page 68.) You can inspect these registers using the PRCL function described on the next page. However, since direct inspection of the I/O buffer contents is not essential for most users and applications, you may want to skip the remainder of this section unless your applications require you to know specific I/O buffer information.

The 26-Register I/O Buffer

The following diagram describes the contents of the I/O buffer. (Certain operations, such as bar code functions, use these registers differently than shown below.) Because coordinate information is converted into APUs ("absolute plotter units" used by plotters) before it is sent to a plotter, the I/O buffer stores information in terms of APUs. Pen positions expressed in APUs are restricted to the range −32768 to +32767 to be compatible with the numerical range of plotters.

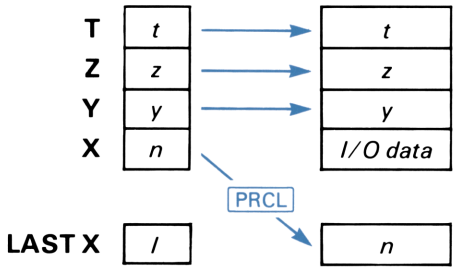
BR_{00}	Status Information	
BR_{01}	$P1_x$	Lower left-hand and upper right-hand corners of graphic limits in APUs.
BR_{02}	$P2_x$	
BR_{03}	$P1_y$	
BR_{04}	$P2_y$	
BR_{05}	x_1	Lower left-hand and upper right-hand corners of plot bounds in APUs.
BR_{06}	x_2	
BR_{07}	y_1	
BR_{08}	y_2	
BR_{09}	Last x	Intended pen position in APUs after most recent plotting or LABEL command.
BR_{10}	Last y	
BR_{11}	Misc x	Miscellaneous storage.
BR_{12}	Misc y	
BR_{13}	Last x'	Actual pen position in APUs after most recent plotting or LABEL command.
BR_{14}	Last y'	

BR_{15}	Misc x'	Miscellaneous storage.
BR_{16}	Misc y'	
BR_{17}	Factor 1 x	GU Scaling Factors (GUs → APUs).
BR_{18}	Factor 2 x	
BR_{19}	Factor 1 y	
BR_{20}	Factor 2 y	
BR_{21}	Factor 1 x'	UU Scaling Factors (UUs → APUs).
BR_{22}	Factor 2 x'	
BR_{23}	Factor 1 y'	
BR_{24}	Factor 2 y'	
BR_{25}	BCSIZE Parameters	→ (Recalled as alpha string.)

PRCL **X** *register number*

The PRCL (*recall I/O parameter*) function recalls to the X-register the contents of the I/O buffer register specified by the absolute value of the number in the X-register. The I/O register number must be from 0 through 25. The function ignores fractional portions of such numbers.

The values stored in buffer registers 01 through 25 are described in the illustration above.



If you specify a register number of 0, PRCL returns a number representing several values stored in BR_{00} and encoded as

$$m \,.\, n_1\, n_2\, n_3\, n_4\, n_5\, n_6\, n_7$$

where

m : LORG value. (Refer to page 94.)

n_1 : PDIR angle θ indicator. (Refer to page 86.)

n_1	Conditions		Quadrant
0	$\cos\theta \geqslant 0$	$\sin\theta \geqslant 0$	I
1	$\cos\theta < 0$	$\sin\theta \geqslant 0$	II
2	$\cos\theta \geqslant 0$	$\sin\theta < 0$	IV
3	$\cos\theta < 0$	$\sin\theta < 0$	III

$n_2 - n_5$: Cosine of PDIR angle θ in ***x.xxx*** format (without sign).

n_6 : RPLOT indicator. (Refer to page 82.)

n_6	Condition
0	RPLOT not last function
2	RPLOT was last function

n_7: Plotting status.

n_7	Plot Mode	Pen Status	Last Point Specified
0	GU	Up	In Bounds
1	GU	Up	Out of Bounds
2	GU	Down	In Bounds
3	GU	Down	Out of Bounds
4	UU	Up	In Bounds
5	UU	Up	Out of Bounds
6	UU	Down	In Bounds
7	UU	Down	Out of Bounds

Example of PRCL Execution. Determine the default plotting conditions that the plotter module sets in BR_{00}.

Keystrokes	Display	
PINIT		Initializes plotting system.
0 PRCL	1.0100	Displays contents of I/O buffer status register (BR_{00}).
■ FIX 7	1.0100004	Sets FIX 7 display mode to show all digits in the buffer register.
■ FIX 4	1.0100	Restores FIX 4 display.

The indicated default conditions for the HP 7470A Plotter are

- LORG value: 1.
- PDIR quadrant: I.
- Cosine PDIR angle: 1.000.
- RPLOT not last function.
- UU mode.
- Pen up.
- Last point in bounds.

Default Plotting Conditions

Executing PINIT or LIMIT sets the following default conditions:

- Reads the P1 and P2 coordinates from the plotter (except that LIMIT first sets P1 and P2 from data in the stack) and uses the corresponding points as the diagonal end points of the graphic limits.
- Determines the GU scale factors $1x$, $2x$, $1y$, $2y$ to generate a GU scale. (Refer to RATIO on page 71.)
- Sets the UU scale factors $1x'$, $2x'$, $1y'$, and $2y'$ equal to the GU scale factors described above and sets the plotter to UU mode.
- Selects pen 1.
- Selects line type 1.
- Selects label origin 1.

- Sets character space height to 3 GUs.
- Sets label direction as left-to-right.
- Sets angular rotation of axes for incremental and relative plotting to zero.
- Sets the tic length to the plotter's default value.
- Clears any error conditions.

Executing PINIT when an I/O buffer does not already exist sets default BCSIZE parameters for bar code geometry—otherwise, BCSIZE parameters remain unchanged.

APPENDICES

Appendix A

Error Messages

This appendix contains a list of messages and errors that are related to plotter module *functions*. For each error message listed below, the possible causes are listed according to the plotter functions that generate that message.

For error conditions that occur during execution of the user-language plotting routines (NEWPLOT, REPLOT, PLINIT, PLTUXY, PLANOT), the bar-code subroutines, PLOTBC, or other programs, set the HP-41 to Program mode. The function in the displayed program line is the one that caused the error. If it's a plotter function, refer to the list below. Otherwise, refer to the owner's manual for the HP-41 extension that provides that function.

Display	Functions	Meaning
ALPHA DATA	***-all-***	Alpha characters are in a register where a number is required—either a stack register or a data storage register.
DATA ERROR	BCCKSM	$x = 0$.
	BCO	$x \geqslant 25$ or $x \leqslant -17$.
	BCP	$x \geqslant 1000$.
	BCREGX	For any specified register, $\lvert contents \rvert \geqslant 256$.
	BCSIZE	$nn \neq 0$ and $nn \leqslant pp$, $ww \neq 0$ and $ww \leqslant pp$, $pp + ss > 97$, $pp + aa > 97$, *nn* not specified and $pp > 66$, *ww* not specified and $pp > 38$, $0 < \lvert nn \rvert < \lvert pp \rvert$, or $0 < \lvert ww \rvert < \lvert pp \rvert$, where $x =$ ***pp.hhhaa*** and $y =$ ***nn.twwss***.
	RATIO	*y*-minimum = *y*-maximum for the graphic limits.
	SCALE	*x*-minimum = *x*-maximum, or *y*-minimum = *y*-maximum.
NO PLOTTER	***-all-***	A standard plotter (accessory types 96 through 111) is not in the interface loop. (Occurs in Auto mode only—flag 32 clear.)
NONEXISTENT	BCREGX	$\lvert x \rvert \geqslant 1000$.
	BCO	For $x < 0$, an HP-IL module is not connected, its Print Function Switch is not set to ENABLE, or an HP 82162A Thermal Printer is not connected (or is not the primary device in Manual interface mode—flag 32 set).
	BCP	Function specifies a nonexistent program.
	BCXS	$\lvert y \rvert \geqslant 1000$.
	PLREGX	Function specifies a nonexistent data register.
	PRCL	Function specifies a register numbered higher than 25.
PACKING TRY AGAIN	BCP	Specified program had not been packed.

Display	Functions	Meaning
PL:NO HPIL	***-all-***	An HP 82160A HP-IL module is not plugged into the HP-41.
PL:NO ROOM	PINIT	There are less than 26 memory registers available to build the I/O buffer.
PL:PLS PINIT	***-all-***	The I/O buffer does not exist.
PL:RANGE ERR	LIMIT	A plotting area larger than the plotter's mechanical limits has been specified.
	LXAXIS LYAXIS XAXIS XAXISO YAXIS YAXISO	Function specifies a tic spacing of zero, x-maximum $\leqslant$ x-minimum or y-maximum $\leqslant$ y-minimum. A specified value converts to a point beyond the physical limits of the plotter.
ROM	BCP	Specified program is in ROM (read-only memory) in a plug-in module. (Use COPY first.)
TRANSMIT ERR	***-all-***	Interface loop is not connected or a device is turned off. In Manual interface mode, the primary device may not be able to perform the operation; select the proper device.

Appendix B

Care, Warranty, and Service Information

Contents

Module Care 156
Limited One-Year Warranty 156
What We Will Do 156
What Is Not Covered 156
Warranty for Consumer Transactions in the United Kingdom 157
Obligation to Make Changes 157
Warranty Information 157
Service 158
Obtaining Repair Service in the United States 158
Obtaining Repair Service in Europe 158
International Service Information 158
Technical Assistance 159
Dealer and Product Information 159

Module Care

CAUTION

Always turn off the HP-41 before connecting or disconnecting any module or peripheral. Failure to do so could result in damage to the HP-41 or disruption of the system's operation.

- Keep the contact area of the module free of obstructions. Should the contacts become dirty, carefully brush or blow the dirt out of the contact area. Do not use any liquid to clean the contacts.
- Store the module in a clean, dry place.
- Always turn off the computer before installing or removing any module or peripherals.
- Observe the following temperature specifications:

  Operating: 0°C to 45°C (32°F to 113°F).

  Storage: -40°C to 75°C (-40°F to 167°F).

Limited One-Year Warranty

What We Will Do

The HP 82184A Plotter Module is warranted by Hewlett-Packard against defects in materials and workmanship affecting electronic and mechanical performance, but not software content, for one year from the date of original purchase. If you sell your unit or give it as a gift, the warranty is automatically transferred to the new owner and remains in effect for the original one-year period. During the warranty period, we will repair or, at our option, replace at no charge a product that proves to be defective, provided you return the product, shipping prepaid, to a Hewlett-Packard service center.

What Is Not Covered

This warranty does not apply if the product has been damaged by accident or misuse or as the result of service or modification by other than an authorized Hewlett-Packard service center.

No other express warranty is given. The repair or replacement of a product is your exclusive remedy. **ANY OTHER IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS IS LIMITED TO THE ONE-YEAR DURATION OF THIS WRITTEN WARRANTY.** Some states, provinces, or countries do not allow limitations on how long an implied warranty lasts, so the above limitation may not apply to you. **IN NO EVENT SHALL HEWLETT-PACKARD COMPANY BE LIABLE FOR CONSEQUENTIAL DAMAGES.** Some states, provinces, or countries do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

This warranty gives you specific legal rights, and you may also have other rights which vary from state to state, province to province, or country to country.

Warranty for Consumer Transactions in the United Kingdom

This warranty shall not apply to consumer transactions and shall not affect the statutory rights of a consumer. In relation to such transactions, the rights and obligations of Seller and Buyer shall be determined by statute.

Obligation to Make Changes

Products are sold on the basis of specifications applicable at the time of manufacture. Hewlett-Packard shall have no obligation to modify or update products once sold.

Warranty Information

If you have any questions concerning this warranty or service, please contact an authorized Hewlett-Packard dealer or a Hewlett-Packard sales and service office. Should you be unable to contact them, please contact:

- In the United States:

Hewlett-Packard
Corvallis Division
1000 N.E. Circle Blvd.
Corvallis, OR 97330
Telephone: (503) 758-1010
Toll-Free Number: (800) 547-3400 (except in Oregon, Hawaii, and Alaska)

- In Europe:

Hewlett-Packard S.A.
7, rue du Bois-du-Lan
P.O. Box
CH-1217 Meyrin 2
Geneva
Switzerland
Telephone: (022) 83 81 11

Note: Do *not* send units to this address for repair.

- In other countries:

Hewlett-Packard Intercontinental
3495 Deer Creek Rd.
Palo Alto, California 94304
U.S.A.
Telephone: (415) 857-1501

Note: Do *not* send units to this address for repair.

Service

Obtaining Repair Service in the United States

The Hewlett-Packard United States Service Center for handheld and portable computer products is located in Corvallis, Oregon:

Hewlett-Packard Company
Corvallis Division Service Department
P.O. Box 999/1000 N.E. Circle Blvd.
Corvallis, Oregon 97330, U.S.A.
Telephone: (503) 757-2000

Obtaining Repair Service in Europe

Service centers are maintained at the following locations. For countries not listed, contact the dealer where you purchased your unit.

AUSTRIA
HEWLETT-PACKARD GES.m.b.H.
Kleinrechner-Service
Wagramerstrasse-Lieblgasse 1
A-1220 WEIN (Vienna)
Telephone: (0222) 23 65 11

BELGIUM
HEWLETT-PACKARD BELGIUM SA/NV
Woluwedal 100
B-1200 BRUSSELS
Telephone: (02) 762 32 00

DENMARK
HEWLETT-PACKARD A/S
Datavej 52
DK-3460 BIRKEROD (Copenhagen)
Telephone: (02) 81 66 40

EASTERN EUROPE
Refer to the address listed under Austria.

FINLAND
HEWLETT-PACKARD OY
Revontulentie 7
SF-02100 ESPOO 10 (Helsinki)
Telephone: (90) 455 02 11

FRANCE
HEWLETT-PACKARD FRANCE
Division Informatique Personnelle
S.A.V. Calculateurs de Poche
F-91947 Les Ulis Cedex
Telephone: (6) 907 78 25

GERMANY
HEWLETT-PACKARD GmbH
Kleinrechner-Service
Vertriebszentrale
Berner Strasse 117
Postfach 560 140
D-6000 FRANKFURT 56
Telephone: (611) 50041

ITALY
HEWLETT-PACKARD ITALIANA S.P.A.
Casella postale 3645 (Milano)
Via G. DiVittorio, 9
I-20063 CERNUSCO SUL NAVIGLIO (Milano)
Telephone: (2) 90 36 91

NETHERLANDS
HEWLETT-PACKARD NEDERLAND B.V.
Van Heuven Goedhartlaan 121
N-1181 KK AMSTELVEEN (Amsterdam)
P.O. Box 667
Telephone: (020) 472021

NORWAY
HEWLETT-PACKARD NORGE A/S
P.O. Box 34
Desterndalen 18
N-1345 OESTERAAS (Oslo)
Telephone: (2) 17 11 80

SPAIN
HEWLETT-PACKARD ESPANOLA S.A.
Calle Jerez 3
E-MADRID 16
Telephone: (1) 458-2600

SWEDEN
HEWLETT-PACKARD SVERIGE AB
Skalholtsgatan 9, Kista
Box 19
S-163 93 SPANGA (Stockholm)
Telephone: (08) 750 20 00

SWITZERLAND
HEWLETT-PACKARD (SCHWEIZ) AG
Kleinrechner-Service
Allmend 2
CH-8967 WIDEN
Telephone: (057) 31 21 11

UNITED KINGDOM
HEWLETT-PACKARD Ltd.
King Street Lane
GB-WINNERSH, WOKINGHAM
BERKSHIRE RG11 5AR
Telephone: (734) 784 774

International Service Information

Not all Hewlett-Packard service centers offer service for all models of HP products. However, if you bought your product from an authorized Hewlett-Packard dealer, you can be sure that service is available in the country where you bought it.

If you happen to be outside of the country where you bought your module, you can contact the local Hewlett-Packard service center to see if service is available for it. If service is unavailable, please ship the module to the address listed above under Obtaining Repair Service in the United States. A list of service centers for other countries can be obtained by writing to that address.

All shipping, reimportation arrangements, and customs costs are your responsibility.

Technical Assistance

The keystroke procedures and program material in this manual are supplied with the assumption that the user has a working knowledge of the concepts and terminology used. Hewlett-Packard's technical support is limited to explanation of operating procedures used in the manual and verification of answers given in the examples. If you have technical problems involving this manual, consult your HP-41 owner's manual. Should you need further assistance, write to:

Hewlett-Packard
Corvallis Division Customer Support
1000 N.E. Circle Blvd.
Corvallis, OR 97330

Dealer and Product Information

For U.S.A. dealer locations, product information, and prices, please call (800) 547-3400. In Oregon, Alaska, or Hawaii, call (503) 758-1010.

Plotter supplies are available from many authorized Hewlett-Packard dealers and from Hewlett-Packard sales offices. In the U.S.A. you can also order plotter supplies through the *Personal Computer Supplies and Accessories* catalog (part number 5953-2010) or the *Computer Users' Catalog* (part number 5953-2450), either of which you can receive without charge by writing to:

Hewlett-Packard
Computer Supplies Operation
P.O. Box 60008
Sunnyvale, California 94088

Appendix C

Program Documentation

Contents

Annotated Program Listings 160
RAIN Program to Produce a Graph 160
KWH Program to Produce a Bar Chart 161
Utility Plotting Program 162
The BAR Subroutine 167
The TITLES Subroutine 168
Pen Width Calibration 169
PLOTBC Program 169
Bar Code Subroutines (XBC Program) 171
Printing a Program on the HP 82162A Thermal Printer 172
The PLPLOT/PLPLOTP Subroutine 173
Flowcharts 174
NEWPLOT 174
REPLOT 175
PLINIT 175
PLTUXY 176
PLANOT 179
BAR 181
TITLES 182
PWIDTH 185
PLOTBC 186
XBC 190
PLPLOT/PLPLOTP 194

Annotated Program Listings

Page references are included in the headings that precede programs described elsewhere in this manual.

RAIN Program to Produce a Line Graph (Page 15)

Fixes display format for labeling.

```
01♦LBL "RAIN"
02 FIX 0
03 CF 29
```

Stores loop counter and buffer pointer.

```
04 1.022
05 STO 00
```

Stores year as *x*-coordinate in buffer.

```
06♦LBL 01
07 RCL 00
08 2
09 /
10 INT
11 1968
12 +
13 STO IND 00
14 ISG 00
15 CLA
```

Prompts user with year to input rainfall.

```
16 ARCL X
17 "⊢=?"
18 PROMPT
```

Stores rainfall as *y*-coordinate in buffer.

```
19 STO IND 00
```

Loops until buffer is full.

```
20 ISG 00
21 GTO 01
```

Initializes plotter. Frames hard clip.

```
22 PINIT
23 FRAME
```

Sets and frames soft clip area.

```
24 20
25 ENTER↑
26 120
27 ENTER↑
28 20
29 ENTER↑
30 90
31 LOCATE
32 FRAME
```

Scales soft clip with years in T and Z, and rainfall range in Y and X.

```
33 1968
34 ENTER↑
35 1978
36 ENTER↑
37 0
38 ENTER↑
39 60
40 SCALE
```

Labels *y*-axis from 0 to 60 with a labeled tic every 5 units; intercept at 1968.

```
41 X<>Y
42 5
43 ENTER↑
44 1968
45 LYAXIS
```

Label *x*-axis from 1968 to 1978 with a labeled tic for every year. Intercept at $y = 0$.

```
46 1978
47 ENTER↑
48 1968
49 ENTER↑
50 1
51 ENTER↑
52 0
53 LXAXIS
```

Moves to label chart.

```
54 -15
55 ENTER↑
56 1970
57 MOVE
58 1
59 LORG
```

Sets character size and slant.

```
60 10
61 ENTER↑
62 .5
63 ENTER↑
64 5
65 CSIZE0
```

Disables CR/LF (carriage return, line feed) for LABEL.

```
66 SF 17
```

Selects pen 2.

```
67 2
68 PEN
```

Labels chart.

```
69 "ANNUAL RAINFALL"
70 "⊢ IN CORV"
71 LABEL
72 "ALLIS, OREGON"
73 LABEL
```

Moves to label *y*-axis.

```
74 25
75 ENTER↑
76 1967
77 MOVE
```

Rotates labels.

```
78 DEG
79 90
80 LDIR
```

Labels *y*-axis.

```
81 "INCHES"
82 LABEL
83 1.022
```

Plots rainfall.

```
84 PLREGX
```

Switches to GU mode, moves pen arm to display chart, and stores pen.

```
85 CLX
86 ENTER↑
87 SETGU
88 MOVE
89 PEN
90 FIX 4
91 END
```

KWH Program to Produce a Bar Chart (Page 17)

Sets label display format and loop counter.

```
01♦LBL "KWH"
02 FIX 0
03 CF 29
04 1.012
05 STO 00
```

Prompts user for each month's power usage and stores user entry.

```
06♦LBL 13
07 XEQ IND 00
08 PROMPT
09 STO IND 00
10 ISG 00
11 GTO 13
```

Sets up plotter. Uses LIMIT to adjust hard clip area.

```
12 PINIT
13 0
14 ENTER↑
15 250
16 ENTER↑
17 0
18 ENTER↑
19 175
20 LIMIT
```

Locates a soft clip area.

```
21 20
22 ENTER↑
23 120
24 ENTER↑
25 15
26 ENTER↑
27 85
28 LOCATE
```

Sets scale, labels *y*-axis, draws *x*-axis and tics, sets small character size.

```
29 0
30 ENTER↑
31 13
32 ENTER↑
33 0
34 ENTER↑
```

```
35 3600
36 SCALE
37 X<>Y
38 200
39 ENTER↑
40 0
41 LYAXIS
42 13
43 ENTER↑
44 0
45 ENTER↑
46 1
47 ENTER↑
48 0
49 XAXIS0
```

Labels *x*-axis.

```
50 1.012
51 STO 00
52 5
53 LORG
```

Selects pen 2 and sets character size for chart title.

```
54 2.5
55 CSIZE
```

Labels chart.

```
56♦LBL 14
57 -110
58 RCL 00
59 MOVE
60 XEQ IND 00
61 LABEL
62 ISG 00
63 GTO 14
```

Sets loop counter for bars.

```
64 2
65 PEN
66 5
67 CSIZE
```

Labels chart.

```
68 3800
69 ENTER↑
70 6.5
71 MOVE
72 "1981 KWH ELECTR"
73 "HICITY USE"
74 LABEL
```

```
75 1.012
76 STO 00
```

Locates center point of bar and computes XMIN as $x - 0.2$ and XMAX as $x + 0.2$. Gets user input for YMAX.

```
77♦LBL 15
78 RCL 00
79 .2
80 -
81 RCL 00
82 .2
83 +
84 0
85 RCL IND 00
```

Clips and frames defined area. Loops until plot is completed, then stores pen and halts.

```
86 CLIPUU
87 FRAME
88 ISG 00
89 GTO 15
90 CLX
91 ENTER↑
92 SETGU
93 MOVE
94 PEN
95 FIX 4
96 RTN
```

```
97♦LBL 01
98 "JAN"
99 RTN
100♦LBL 02
101 "FEB"
102 RTN
103♦LBL 03
104 "MAR"
105 RTN
106♦LBL 04
107 "APR"
108 RTN
109♦LBL 05
110 "MAY"
111 RTN
112♦LBL 06
113 "JUN"
114 RTN
115♦LBL 07
116 "JUL"
117 RTN
118♦LBL 08
119 "AUG"
120 RTN
121♦LBL 09
122 "SEP"
123 RTN
124♦LBL 10
125 "OCT"
126 RTN
127♦LBL 11
128 "NOV"
129 RTN
130♦LBL 12
131 "DEC"
132 END
```

Utility Plotting Program (Page 19)

Sets User mode, prompts for function name, sets default values.

```
01♦LBL "NEWPLOT"
02♦LBL A
03 SF 27
04 8
05 ENTER↑
06 " "
07 ASTO X
08 XEQ 00
09 1000.01
10 STO 03
11 0
12 -1
13 STO 04
14 XEQ 00
15 STO 09
16 1
17 STO 07
18 ENTER↑
19 XEQ 00
20 5
21 11
22 STO 02
23 CHS
```

Prompts for XMAX, XMIN, XINC.

```
24 XEQ 00
25 4
26 XEQ 01
27 STO 06
28 7
29 XEQ 01
```

Plots chart, edits data base, or executes user routine.

```
30♦LBL "REPLOT"
31♦LBL B
32 CF 22
33 CF 23
34 "PLOT?"
```

```
35 TONE 7
36 PROMPT
37 AOFF
38 FS?C 23
39 GTO 03
40 FS?C 22
41 GTO 02
42 XEQ "PLINIT"
43 XEQ "PLTUXY"
44♦LBL E
45 XEQ "PLANOT"
46 GTO B
47♦LBL C
48 XEQ "PLINIT"
49 GTO B
50♦LBL D
51 XEQ "PLTUXY"
52 GTO B
53♦LBL 03
54 ASTO X
55 XEQ IND X
56 GTO B
57♦LBL 02
58 XEQ 01
59 GTO B
60♦LBL 00
61 STO IND Y
62 RDN
63♦LBL 01
64 STO 11
65 10
66 X<=Y?
67 X<>Y
68 XEQ IND Y
```

```
69 RCL IND 11
70 SIGN
71 X=0?
72 AON
73 LASTX
74 XEQ 14
75 STO IND 11
76 RTN
```

Ensures R_{nn} never has a "." in the prompt.

```
77♦LBL 14
78 CF 22
79 CF 23
80 CF 29
81 "⊢="
82 ARCL X
83 "⊢?"
84 PROMPT
85 AOFF
86 FIX 3
87 FS? 23
88 ASTO X
89 RTN
```

```
90♦LBL 00
91 "XMIN"
92 RTN
93♦LBL 01
94 "XMAX"
95 RTN
96♦LBL 02
97 "PLTPRM"
98 FIX 4
99 RTN
100♦LBL 03
101 "ANNOT"
102 FIX 5
103 RTN
104♦LBL 04
105 "YMIN"
106 RTN
107♦LBL 05
108 "XINC"
109 RTN
110♦LBL 06
111 "XAXAT"
112 RTN
113♦LBL 07
114 "YMAX"
115 RTN
116♦LBL 08
117 "NAME"
118 FIX 4
119 RTN
120♦LBL 09
121 "YAXAT"
122 RTN
123♦LBL 10
124 "R"
125 FIX 0
126 ARCL X
127 FIX 3
128 RTN
```

Gets PLTPRM type. If $x = 0$ executes user routine. If $x = 1$ or 3, performs point plotting. If $x = 2$, performs autoscale.

```
129♦LBL 11
130 RCL 02
131 ENTER↑
132 SIGN
133 X=0?
134 RTN
135 X<>Y
136 INT
137 X≠0?
138 X<>Y
139 2
140 +
141 RTN
```

Initializes plotting area regardless of x/y ratio.

```
142♦LBL "PLINIT"
143 PINIT
144 CLST
145 1 E2
146 STO Z
147 SCALE
```

Sets user scale.

```
148♦LBL 12
149 20
150 95
151 LORG
152 25
153 95
154 CLIPUU
155 RCL 00
156 RCL 01
157 RCL 04
158 RCL 07
159 SCALE
160 RTN
```

Plot Routine. If PLTPRM is Alpha data, goes to [LBL] 00. Otherwise gets desired pen and line type.

```
161♦LBL "PLTUXY"
162 PENUP
163 XEQ 11
164 X=0?
165 GTO 00
166 RCL 02
167 1 E2
168 MOD
169 10
170 /
171 INT
172 LTYPE
173 RCL 02
174 INT
175 10
176 MOD
177 PEN
```

Sets x_0 in R_{10} and changes number of increments to increment value, if needed.

```
178♦LBL 00
179 RCL 00
180 STO 10
181 RCL 05
182 SIGN
183 X=0?
184 GTO 00
185 RCL 01
186 RCL 00
187 -
188 RCL 05
189 X=0?
190 GTO 00
191 X>0?
192 ST- 10
193 /
194 X>0?
195 GTO 00
196 ABS
197 ST- 10
198 X<> 05
199 ABS
```

If plotting buffer, stores point count in R_{11}. Otherwise sets to 1,000 points.

```
200♦LBL 00
201 RCL 08
202 SIGN
203 X=0?
204 GTO 00
205 LASTX
206 INT
207 LASTX
208 FRC
209 1 E3
210 *
211 INT
212 -
213 RCL 08
214 1 E4
```

Clears "done" flag.

```
215 *
216 10
217 MOD
218 X≠0?
219 /
220 ENTER↑
221♦LBL 00
222 RDN
223 INT
224 X=0?
225 999
226 .1
227 %
228 STO 11
229 CF 17
```

Go to [LBL] 04 if not plotting buffer. Otherwise, get buffer type and go to that buffer.

```
230♦LBL 13
231 RCL 08
232 SIGN
233 X=0?
234 GTO 04
235 LASTX
236 1 E4
237 *
238 10
239 MOD
240 GTO IND X
```

Plot type 0 buffer.

```
241♦LBL 00
242 RCL 08
243 PLREGX
244 GTO 03
```

If type 1 buffer, get x_i from XINC and y_i from buffer.

```
245♦LBL 01
246 XEQ 05
247 STO 10
248 RCL 11
249 INT
250 RCL 08
251 INT
252 +
253 RCL IND X
254 GTO 01
```

If type 2 buffer, get x_i and y_i from buffer.

```
255♦LBL 02
256 RCL 08
257 INT
258 RCL 11
259 INT
260 2
261 *
262 +
263 RCL IND X
264 STO 10
265 RDN
266 1
267 +
268 RCL IND X
269 GTO 01
```

Non-buffered plotting. Get x_i from XINC and y_i from NAME.

```
270♦LBL 04
271 XEQ 05
272 STO 10
273 FS?C 17
274 GTO 04
275 XEQ IND 08
```

PLTPRM end of loop.

```
276♦LBL 01
277 FS?C 17
278 GTO 04
279 XEQ 08
280 ISG 11
281 GTO 13
```

```
282♦LBL 04
283 RCL 02
284 SIGN
285 X=0?
286 GTO 03
287 LASTX
```

Clean up. If buffer was just built, store pointer and buffer type in R_{08} and insert zero for buffer-building parameters.

```
288 FRC
289 STO Y
290 1 E3
291 *
292 X=0?
293 GTO 00
294 INT
295 +
296 STO 08
297 1 E4
298 *
299 10
300 MOD
301 X=0?
302 2
303 RCL 11
304 INT
305 *
306 1
307 -
308 1 E3
309 /
310 ST+ 08
```

If autoscale was just done, resets PLTPRM to 11, resets XAXAT and YAXAT, sets scale just identified, and returns.

```
311♦LBL 00
312 RCL 02
313 INT
314 STO 02
315 X≠0?
316 GTO 03
317 11
318 STO 02
319 RCL 00
320 STO 09
321 RCL 01
322 RCL 04
323 STO 06
324 RCL 07
325 SCALE
326 RTN
```

Returns pen, presents plot, and ends.

```
327♦LBL 03
328 RCL 04
329 RCL 00
330 MOVE
331 0
332 PEN
333 CLA.
334 RTN
```

If XINC is alpha, gets x_i from user routine. If R_{05} is numeric, $x_i = x_{i-1} + \text{XINC}$.

```
335♦LBL 05
336 RCL 05
337 SIGN
338 X=0?
339 GTO IND 05
340 LASTX
341 RCL 10
342 +
343 RTN
```

Test PLTPRM parameter. If numeric, go to [LBL] 00. If PLTPRM is alpha, go to routine specified by label in R_{02} (PLTPRM).

```
344♦LBL 08
345 X<> 02
346 SIGN
347 X≠0?
348 GTO 00
349 CLX
350 LASTX
351 X<> 02
352 RCL 10
353 GTO IND 02
```

If not filling a buffer, go to [LBL] 04. Otherwise go to [LBL] number corresponding to buffer type.

```
354♦LBL 00
355 LASTX
356 X<> 02
357 RCL 02
358 FRC
359 X=0?
```

Stores x and y for type 0 or type 2 buffer.

Stores y for type 1 buffer.

Plots point. Plots character if desired (and printer code is available).

```
360 GTO 04
361 1 E4
362 *
363 10
364 MOD
365 GTO IND X
366♦LBL 00
367♦LBL 02
368 X<>Y
369 RCL 11
370 INT
371 2
372 *
373 RCL 02
374 FRC
375  E3
376 *
377 INT
378 +
379 RCL 10
380 STO IND Y
381 CLX
382 1
383 +
384 X<>Y
385 STO IND Y
386 ENTER↑
387 GTO 04
388♦LBL 01
389 RCL 02
390 FRC
391 1 E3
392 *
393 RCL 11
394 INT
395 +
396 RCL Z
397 STO IND Y
398 ENTER↑

399♦LBL 04
400 RCL Y
401 XEQ 11
402 RCL Z
403 GTO IND Y

404♦LBL 01
405♦LBL 03
406 RCL 10
407 PLOT
408 RCL 02
409 1 E2
410 /
411 INT
412 X=0?
413 RTN
414 0
415 X<>Y
416 SF 25
417 BLDSPEC
418 FC?C 25
419 RTN
420 CLA
421 ARCL X
422 CLX
423 PRCL
424 1 E7
425 *
426 2
427 MOD
428 X≠0?
429 RTN
430 RDN
```

Autoscale. Update maximum and minimum.

On first time through autoscale sets XMIN and YMIN to 10^{99} and XMAX and YMAX to -10^{99}.

Selects pen 2, resets linetype, frames plotting area.

Gets XMAX-XMIN. Gets number of major X-tics. I zero major X-tics, goes to [LBL] 00.

Sets tic length to 1, gets number of minor X-tics. I zero minor X-tics, goes to [LBL] 00.

```
431 LABEL
432 PLOT
433 RTN
434♦LBL 02
435 RCL 11
436 INT
437 X=0?
438 XEQ 02
439 RDN
440 SIGN
441 X=0?
442 RTN
443 LASTX
444 RCL 10
445 SIGN
446 X=0?
447 RTN
448 RCL 04
449 RCL Z
450 X<Y?
451 STO 04
452 RCL 07
453 X<>Y
454 X>Y?
455 STO 07
456 RCL 00
457 RCL 10
458 X<Y?
459 STO 00
460 RCL 01
461 X<>Y
462 X>Y?
463 STO 01
464 RTN
465♦LBL 02
466 1 E99
467 STO 00
468 STO 04
469 CHS
470 STO 01
471 STO 07
472 RDN
473 RTN
474♦LBL "PLANOT"
475 2
476 PEN
477 1
478 LTYPE
479 FRAME
480 RCL 01
481 RCL 00
482 -
483 STO 10
484 RCL 03
485 XEQ 07
486 X=0?
487 GTO 00
488 XEQ 05
489 CF 29
490 XEQ 02

491♦LBL 00
492 1
493 TICLEN
494 RCL 03
495 INT
496 ABS
497 1 E2
498 MOD
499 X=0?
500 GTO 00
501 ST/ 10
502 SF 29
```

If flag 29 is set, plots minor tics. Otherwise plot major tics with labels.

```
503♦LBL 02
504 RCL 01
505 RCL 00
506 RCL 10
507 RCL 06
508 FS? 29
509 XAXIS
510 FC? 29
511 LXAXIS
512 FC? 29
513 RTN
```

Gets YMAX-YMIN, gets number of major Y-tics. If zero, goes to [LBL] 00.

```
514♦LBL 00
515 RCL 07
516 RCL 04
517 -
518 STO 10
519 RCL 03
520 FRC
521 1 E5
522 *
523 INT
524 STO 11
525 XEQ 07
526 X=0?
527 GTO 00
528 XEQ 05
529 CF 29
530 XEQ 03
```

Sets tic length to 1, gets number of minor Y-tics. If zero, goes to [LBL] 00.

```
531♦LBL 00
532 1
533 TICLEN
534 RCL 11
535 ABS
536 1 E2
537 MOD
538 X=0?
539 GTO 00
540 ST/ 10
541 SF 29
```

If flag 29 is set, plots minor tics. Otherwise plot major tics with labels.

```
542♦LBL 03
543 RCL 07
544 RCL 04
545 RCL 10
546 RCL 09
547 FS? 29
548 YAXIS
549 FC? 29
550 LYAXIS
551 FC? 29
552 RTN
```

Plot function name as chart title if annotation parameter is "+" and R_{08} contains alpha data.

```
553♦LBL 00
554 5
555 LORG
556 CSIZE
557 FIX 4
558 RCL 03
559 X<=0?
560 GTO 04
561 RCL 08
562 SIGN
563 X≠0?
564 GTO 04
565 CLA
566 ARCL 08
567 UNCLIP
568 CLST
569 LDIR
570 1 E2
571 STO Z
572 SCALE
573 98
574 57
575 MOVE
576 LABEL
577 XEQ 12
```

Termination. Puts pen away, moves to [LBL] 00, returns.

```
578♦LBL 04
579 SETGU
580 3
581 CSIZE
582 CLST
583 MOVE
584 PEN
585 SETUU
586 RTN
```

Gets increment, stores in R_{10}, sets tic length to 2. If you gave [FIX] parameter, sets the display and returns. Otherwise sets [FIX] or [SCI] display mode, depending on increment.

```
587♦LBL 05
588 CF 29
589 /
590 STO 10
591 2
592 TICLEN
593 X<> Z
594 1 E4
595 /
596 INT
597 FIX IND X
598 X≠0?
599 RTN
600 CLX
601 1 E4
602 X<=Y?
603 SF 29
604 1/X
605 X>Y?
606 SF 29
607 SCI 4
608 CLX
609 .005
610 RDN
611 FS? 29
612 RTN
```

```
613♦LBL 15
614 STO Y
615 FIX IND T
616 RND
617 X=Y?
618 RTN
619 X<>Y
620 ISG T
621 GTO 15
622 SCI 4
623 RTN
```

```
624♦LBL 07
625 ABS
626 INT
627 STO Z
628 1 E2
629 /
630 LASTX
631 MOD
632 INT
633 RTN
```

Prompts for y_i.

```
634♦LBL "Y?"
635 " "
636 ASTO X
637 "Y("
638 GTO 00
```

Prompts for x_i.

```
639♦LBL "X?"
640 " "
641 ASTO X
642 "X("
```

Recalls *i* from R_{11}, prompts for value, then XEQ 14.

```
643*LBL 00
644 FIX 0
645 RCL 11
646 INT
647 ARCL X
648 "├)"
649 RDN
650 XEQ 14
```

If no input, sets "done" flag (17) and returns. Otherwise, returns.

```
651 FC?C 22
652 FS?C 23
653 RTN
654 SF 17
655 END
```

The BAR Subroutine

BAR generates histograms when used in conjunction with the PLTPRM parameter in the Utility Plotting Program described in section 2. BAR increases the number of reserved registers in the NEWPLOT data base to 13 (R_{00} through R_{12}). Register 12 is used to store a parameter that you must enter in the form of ***pff.ww*** where

p determines the pen number.

ff determines the fill density by designating the number of lines used to shade the bar.

ww represents a percentage of the *x*-axis used to determine the width of the bars.

BAR plots bars only within the plotting limits.

Bar code for the following listing is on pages 195 and 196.

Tests *y* against YMAX AND YMIN and replaces *y* if out of bounds. Stores *y* in Z. Sets tic length to 100% and selects specified pen.

```
01*LBL "BAR"
02 RDN
03 RCL 07
04 X>Y?
05 X<>Y
06 RCL 04
07 X<Y?
08 X<>Y
09 STO 02
10 1 E2
11 TICLEN
12 RCL 12
13 1
14 %
15 INT
16 PEN
17 RCL 01
18 RCL 00
19 -
20 RCL 12
21 FRC
22 *
23 2
24 /
25 ENTER↑
26 ENTER↑
27 RCL 10
28 RCL Y
29 -
30 RCL 00
31 X<Y?
```

Gets bar start and end point and clips against XMIN and XMAX.

```
32 RDN
33 RCL Z
34 RCL 10
35 +
36 RCL 01
37 X<Y?
38 X<>Y
39 RDN
40 RCL 06
41 RCL 02
```

Clips and frames bar.

```
42 CLIPUU
43 FRAME
```

```
44 RDN
45 RDN
46 X<>Y
47 STO 02
48 RDN
49 RCL 01
50 RCL 00
51 -
52 RCL 12
53 FRC
54 *
55 RCL 12
56 INT
57 1 E2
58 MOD
59 X=0?
60 GTO 01
61 /
```

Fills bar.

```
62 RCL 02
63 X<>Y
64 RCL 06
65 XAXIS0
```

Restores old soft clip limits.

```
66*LBL 01
67 RCL 00
68 RCL 01
69 RCL 04
70 RCL 07
71 CLIPUU
72 CLA
73 "BAR"
74 ASTO 02
75 END
```

The TITLES Subroutine

TITLES is an independent subroutine that initially prompts you for a plot title using a **PTITL?** prompt. Entering a number terminates interactive prompting. TITLES interprets a numeric input as a pointer to 12 consecutive registers containing a plot title in the first four registers, an *x*-units title in the next four registers, and a *y*-units title in the last four registers. TITLES accumulates only Alpha data during the scan of these registers (that is, if you want to omit specific titles, store numeric data in the corresponding registers).

An Alpha response is limited to 24 characters and appears centered above the plot using pen 2 and CSIZE 5. A R/S with no data entry suppresses the main title. Subsequent prompts of **XTITL?** and **YTITL?** may be answered with Alpha data or no data entry. Conditions for these labels are PEN 1 and CSIZE 3. After printing the label, the pen moves to (GU) point (0,0). The pen is then put away and the plotter is set to UUs.

TITLES *assumes your plotter is already initialized and scaled.*

Bar code for the following listing is on pages 206 and 207.

If numeric register pointer is entered, gets data from registers.

If Alpha data entered, plots it.

Gets data from registers.

```
01♦LBL "TITLES"
02 CF 22
03 CF 23
04 "PTITL?"
05 AON
06 PROMPT
07 AOFF
08 FC?C 22
09 GTO 05
10 CLA
11 .1
12 %
13 +
14 .003
15 +
16 STO 10
17 XEQ 00
18 XEQ 02
19 .003
20 ST+ 10
21 XEQ 00
22 XEQ 03
23 .003
24 ST+ 10
25 XEQ 00
26 XEQ 04
27 RTN
28♦LBL 05
29 FS?C 23
30 XEQ 02
31 "XTITL?"
32 AON
33 PROMPT
34 AOFF
35 FS?C 23
36 XEQ 03
37 "YTITL?"
38 AON
39 PROMPT
40 AOFF
41 FS?C 23
42 XEQ 04
43 GTO 06
44♦LBL 00
45 RCL IND 10
46 SIGN
47 X=0?
48 ARCL IND 10
```

Regardless of P1, P2 setting, positions pen and labels chart title.

Positions pen for labeling *x*-axis.

```
49 ISG 10
50 GTO 00
51 RTN
52♦LBL 02
53 4
54 PRCL
55 24
56 PRCL
57 -
58 23
59 PRCL
60 /
61 RCL 01
62 RCL 00
63 +
64 2
65 PEN
66 /
67 MOVE
68 6
69 ENTER↑
70 5
71 GTO 07
72♦LBL 03
73 3
74 PRCL
75 24
76 PRCL
77 -
78 23
79 PRCL
80 /
81 RCL 01
82 RCL 00
83 +
84 2
85 /
86 MOVE
87 1
88 PEN
89 4
90 ENTER↑
91 3
92♦LBL 07
93 CSIZE
94 RDN
95 LORG
```

Positions pen for labeling *y*-axis.

```
 96 LABEL
 97 CLA
 98 RTN
 99♦LBL 04
100 RCL 07
101 RCL 04
102 +
103 2
104 /
105 1
106 PRCL
107 22
108 PRCL
109 -
110 21
111 PRCL
112 /
```

Presents plot, stores pen, sets UU mode, and halts.

```
113 MOVE
114 1
115 PEN
116 ASIN
117 LDIR
118 6
119 ENTER↑
120 3
121 XEQ 07
122♦LBL 06
123 CLST
124 LDIR
125 SETGU
126 MOVE
127 PEN
128 SETUU
129 END
```

Pen Width Calibration (Page 110)

Sets display for labels, initializes plotter, sets label direction to 90°, sets *x*-coordinate and counter in R_{01}, sets pen width parameter in R_{02}. Clears ALPHA, and sets *y*-coordinate and counter in R_{00}.

```
01♦LBL "PWIDTH"
02 CF 29
03 FIX 0
04 PINIT
05 CLX
06 ACOS
07 LDIR
08 5.02015
09 STO 01
10 7.1
11 STO 02
12 CLA

13♦LBL 01
14 5.08015
15 STO 00
```

Gets *y* and *x* and moves to that point. Places current pen width in ALPHA for label. Sets narrow bars to 26 plotter units and space to 4 plotter units. Sets pen width to current R_{02} value and labels this pen size. Plots 1 null byte using [BCO]. Increments pen parameter and value in Y, then goes to R_{02}, or increments value in X and goes to R_{01}, or returns pen and halts.

```
16♦LBL 02
17 RCL 00
18 RCL 01
19 MOVE
20 ARCL 02
21 .00004
22 RCL 02
23 INT
24 BCSIZE
25 LABEL
26 CLA
27 CLST
28 1
29 BCO
30 ISG 02
31 ISG 00
32 GTO 02
33 ISG 01
34 GTO 01
35 CLX
36 PEN
37 FIX 4
38 END
```

PLOTBC Program (Page 114)

Ensures that [LABEL] generates CR/LF ([CF] 17), sets user mode ([SF] 27), sets flag so that data entry is signalled by flags 22 and 23 ([CF] 09). Puts plotter in known state ([PINIT]). Puts pen away, sets *y*-coordinate and character size to 3 and performs [XEQ] 13.

```
01♦LBL "PLOTBC"
02♦LBL a
03 CF 17
04 SF 27
05 CF 09
06 CLST
07 PINIT
08 PEN
09 3
10 STO 02
11 CSIZE
12 XEQ 13
```

Displays prompts above local labels.

```
13♦LBL b
14 " D SD A AA P"
15 PROMPT
16 GTO b
```

Reprint a specific row, fall into NXTROW.

```
17♦LBL e
18 XEQ "PBC"
```

Stores *x* and *y*. Gets *x* and *y* position from R_{02} and R_{03}. Increments X by 10. If X > 131, [XEQ] 13.

```
19♦LBL "NXTROW"
20♦LBL c
21 STO 00
22 RDN
23 STO 01
24 RCL 02
25 RCL 03
26 10
27 +
28 131
29 X<=Y?
30 XEQ 13
```

Moves to new position, stores *x*- and *y*-coordinates. Recover previous contents of X and Y from R_{00} and R_{01}.

```
31♦LBL 05
32 RDN
33 MOVE
34 STO 03
35 RCL 01
36 RCL 00
37 RTN
```

Stores X and Y. Gets x and y position from R_{02} and R_{03}. Decrements X by 10. If $X \leqslant Y$ then $X = 5$. Go to [LBL] 05.

```
38♦LBL "LSTROW"
39♦LBL d
40 STO 00
41 RDN
42 STO 01
43 RCL 02
44 RCL 03
45 10
46 -
47 5
48 X>Y?
49 STO Y
50 GTO 05
```

Gets data, then labels and plots data, moves to next row, and repeats.

```
51♦LBL A
52 XEQ 04
53 XEQ "XBC"
54 XEQ c
55 GTO A
```

Sets first sequence number equal to zero.

```
56♦LBL B
57 0
```

Prompts to verify sequence number. Gets data, then labels and plots data, moves to next row, and repeats.

```
58♦LBL 00
59 FIX 0
60 CF 29
61 "SEQ="
62 ARCL X
63 "⊢?"
64 PROMPT
65 STO Y
66 XEQ 04
67 XEQ "XSBC"
68 XEQ c
69 GTO 00
```

Sets ABC flag ([CF] 22), gets alpha data, then labels and plots data, moves to next row, repeats.

```
70♦LBL C
71 CF 22
72 XEQ 14
73 XEQ "ABC"
74 XEQ c
75 GTO C
```

Sets AABC flag (22), gets Alpha data, then labels and plots data, moves to next row, repeats.

```
76♦LBL D
77 SF 22
78 XEQ 14
79 XEQ "AABC"
80 XEQ c
81 GTO D
```

Clears data input flags, gets program to be plotted. If flag 23 not set, returns to main prompt. Call [BCP] with row 0 to enable easy user recovery if program is unpacked. Prompts for starting row number and number of bytes per row. If integer part of row number equals zero, set flag 09 and don't plot program title. Otherwise, plot program title.

```
82♦LBL E
83 CF 22
84 CF 23
85 "NAME?"
86 AON
87 PROMPT
88 AOFF
89 FC?C 23
90 GTO b
91 ASTO X
92 0
93 BCP
94 RDN
95 "ROW=1.16?"
96 PROMPT
97 FC? 22
98 1
99 CF 09
100 INT
101 X=0?
102 SF 09
103 X<> L
104 FC? 09
105 XEQ 12
```

Plots and labels one row, moves to next row. Is next row equal to zero? If so, returns to main prompt. If at top of page, [XEQ] 12, do again.

```
106♦LBL 10
107 XEQ "PBC"
108 XEQ c
109 X=0?
110 GTO b
111 CF 29
112 RCL 03
113 5
114 X=Y?
115 SF 29
116 RDN
117 RDN
118 FS? 29
119 XEQ 12
120 GTO 10
```

If flag 09 set, does not print program title. Otherwise, prints title and moves to next row.

```
121♦LBL 12
122 FS? 09
123 RTN
124 CLA
125 2
126 PEN
127 CLX
128 LORG
129 ACOS
130 LDIR
131 LABEL
132 ARCL Z
133 LABEL
134 RDN
135 GTO c
```

Prompts for page, sets x-coordinate equal to 5, recalls y, moves to top of page, and returns.

```
136♦LBL 13
137 "INSERT PAGE"
138 PROMPT
139 RCL 02
140 5
141 MOVE
142 ENTER↑
143 STO 03
144 RTN
```

If ABC flag set, prompts with **A DATA?**. Otherwise prompts with **AA DATA?**. If no data is input, returns to main prompt.

```
145♦LBL 14
146 CF 23
147 "A"
148 FS? 22
149 "AA"
150 "⊢ DATA?"
151 AON
152 PROMPT
153 AOFF
154 FC?C 23
155 GTO b
156 RTN
```

Prompts for data. If Alpha data, stores in X and returns. If flag 09 is set, returns. (Treats contents of X as data). If flag 22 is clear, returns to main prompt. Otherwise executes [RTN].

```
157♦LBL 04
158 CF 22
159 CF 23
160 "DATA?"
161 PROMPT
162 AOFF
163 FS? 23
164 ASTO X
165 FS?C 23
166 RTN
167 FC? 09
168 FS?C 22
169 RTN
170 GTO b
171 END
```

Bar Code Subroutines (Page 121)

Sets up ALPHA with data prefix, [XEQ] 00, and generates bit pattern.

```
01♦LBL "XBC"
02 "D:"
03 XEQ 00
04 BCX
```

Plots bit pattern.

```
05♦LBL 05
06 BC
```

Puts pen away, restores stack, and returns.

```
07♦LBL 11
08 0
09 PEN
10 RDN
11 RTN
```

Sets ALPHA with sequence data prefix, [XEQ] 00, goes to [LBL] 05 routine.

```
12♦LBL "XSBC"
13 FIX 0
14 CF 29
15 "SD "
16 ARCL Y
17 "├:"
18 XEQ 00
19 BCXS
20 X<>Y
21 GTO 05
```

If X is numeric, goes to [LBL] 01; otherwise appends data to ALPHA.

```
22♦LBL 00
23 SF 25
24 X=0?
25 CLD
26 FS?C 25
27 GTO 01
28 "┤""
29 ARCL X
30 "┤""
```

[XEQ] 10, labels data with pen 2, moves pen to bar code location, selects pen 1, restores stack, and returns.

```
31♦LBL 07
32 XEQ 10
33 CLX
34 2
35 PEN
36 LABEL
37 CHS
38 RDN
39 STO L
40 CLX
41 R↑
42 IMOVE
43 CLX
44 1
45 PEN
46 CLA
47 RDN
48 RDN
49 RTN
```

Sets label origin to zero, sets incremental plot direction to 0°, sets label direction to 90° (regardless of HP-41 trigonometric setting), and returns.

```
50♦LBL 10
51 0
52 PDIR
53 LORG
54 LASTX
55 X<>Y
56 ACOS
57 LDIR
58 RTN
```

If $x = 0$, goes to [LBL] 06; otherwise, if $|x| \geq 1E11$ or if $|x| < 1E\text{-}5$, sets flag 29; sets a loop counter of 0.009 in T.

```
59♦LBL 01
60 FIX 0
61 CF 29
62 X=0?
63 GTO 06
64 ENTER↑
65 ABS
66 1 E11
67 X<=Y?
68 SF 29
69 RDN
70 1 E-5
71 X>Y?
72 SF 29
73 *
74 CLX
75 .009
76 RDN
```

Makes a copy of the value in Y, sets display according to index in T and flag 29 status, rounds one copy to display setting. If rounded copy ≠ value in Y, then continues.

```
77♦LBL 02
78 STO Y
79 FIX IND T
80 FS? 29
81 SCI IND T
82 RND
83 X≠Y?
84 GTO 03
```

Recalls value in X to ALPHA showing *all significant digits*, and finishes with [LBL] 07 routine.

```
85♦LBL 04
86 RDN
87♦LBL 06
88 ARCL X
89 GTO 07
```

Recovers unrounded value to X, increments index and starts again at [LBL] 02. If [ISG] fails, sets display to [SCI] 9, puts value in Y and finishes with [LBL] 04 routine.

```
90♦LBL 03
91 X<>Y
92 ISG T
93 GTO 02
94 SCI 9
95 X<>Y
96 GTO 04
```

Sets ABC flag ([CF] 22) and goes to [LBL] 09.

```
97♦LBL "ABC"
98 CF 22
99 GTO 09
```

Sets AABC flag.

```
100♦LBL "AABC"
101 SF 22
```

Truncates Alpha strings longer than 14 characters to 14, stores string in stack (X, Y, LAST X). If flag 22 is clear, do ABC. Otherwise do AABC. Finish in [LBL] 05 subroutine.

```
102♦LBL 09
103 ASTO L
104 ASHF
105 ASTO X
106 ASHF
107 ASTO Y
108 "    "
109 ARCL Y
110 ASTO Y
111 "  "
112 ARCL Y
113 ASHF
114 ASTO Y
115 "A:""
116 FS? 22
117 "AA:""
118 ARCL L
119 ARCL X
120 ARCL Y
121 "┤""
122 XEQ 07
```

If the integer portion of a given row number = 0, sets integer portion to 1 or -1, depending on the sign of the row number.

```
123 ARCL L
124 ARCL X
125 ARCL Y
126 FC? 22
127 BCA
128 FS?C 22
129 BCAA
130 GTO 05
131♦LBL "PBC"
132 ENTER↑
133 INT
134 X≠0?
135 GTO 08
136 RDN
137 ENTER↑
138 SIGN
139 +
140 R↑
141♦LBL 08
142 RDN
143 STO L
144 RDN
145 WHERE
146 STO 00
147 X<>Y
148 STO 01
149 R↑
150 XEQ 10
151 CLX
152 ENTER↑
153 1
154 IMOVE
155 PEN
156 RDN
157 RDN
```

Gets pen position and saves in R_{00} and R_{01}. XEQ 10, moves pen to bar code position. If number of bytes returned by BCP is zero, then returns; otherwise plots row. Assembles row label with information returned by BCP. Recovers label position and moves to plot row label with pen 2. Restores stack, sets display to FIX 3. Finishes with LBL 11 routine.

```
158 STO L
159 BCP
160 R↑
161 X=0?
162 RTN
163 RDN
164 BC
165 FIX 0
166 CF 29
167 "ROW "
168 LASTX
169 INT
170 ARCL X
171 RDN
172 "├: LINES "
173 ARCL Z
174 "├-"
175 RCL Z
176 FRC
177 1 E3
178 *
179 ARCL X
180 RDN
181 RCL 01
182 RCL 00
183 MOVE
184 CLX
185 2
186 PEN
187 LABEL
188 RDN
189 RDN
190 FIX 3
191 GTO 11
192 END
```

Printing a Program on the HP 82162A Thermal Printer (Page 136)

The PRBC program prompts you for a program name and prints the program in bar code form on the HP 82162A Thermal Printer. PRBC assumes that the HP 82162A printer is connected to the system (*and that an HP 82143A Printer is not plugged into the HP-41*). It is unnecessary to have a plotter in the interface loop when you execute PRBC.

Set flag 25 (Error Ignore Flag—refer to your HP-41 owner's manual) so that PINIT does not cause a **NO PLOTTER** error condition.*

Clears flag 29 and sets FIX 0 display mode so that row label has no decimal part.

```
01♦LBL "PRBC"
02 SF 25
03 PINIT
04 CF 29
05 FIX 0
```

Gets program name. If no name is entered (or if flag 23 is clear), executes RTN.

Starts at row 1 and makes each row nine bytes long.

Prints row label.

```
06 CF 23
07 "NAME?"
08 AON
09 PROMPT
10 AOFF
11 FC?C 23
12 RTN
13 1.09
14 ASTO Y
15♦LBL 01
16 "ROW: "
17 ARCL X
18 PRA
```

*BCP requires that the 26-register I/O buffer be present. PINIT creates this buffer and also ensures that the HP-IL module is plugged in and that a plotter is in the loop. PINIT produces an error message if there is no plotter or if the HP-IL module is not plugged into the HP-41. However, PINIT creates the buffer *before* it checks for these conditions. To print HP-41 bar code on the HP 82162A Thermal Printer when a plotter is *not* plugged into the loop, set flag 25 *before* executing PINIT.

Gets bit pattern of current row.

Gets number of bytes from T.

Value in X must be negative.

Prints row.

Restores stack order.

```
19 BCP
20 R↑
21 CHS
22 BCO
23 RDN
```

If finished with program (next row is zero), halts execution. Otherwise, goes to LBL 01 and prints next row.

```
24 X≠0?
25 GTO 01
26 END
```

The PLPLOT/PLPLOTP Subroutine

PLPLOT/PLPLOTP allows you to execute a function plot on the HP 82162A Thermal Printer, then translate the data base and generate the function plot on an HP 7470A Plotter.

Rearranges data base and goes to **PLOT?** prompt. Lets user adjust data base.

```
01♦LBL "PLPLOT"
02 XEQ 01
03 GTO "REPLOT"
```

Rearranges data base and plots function.

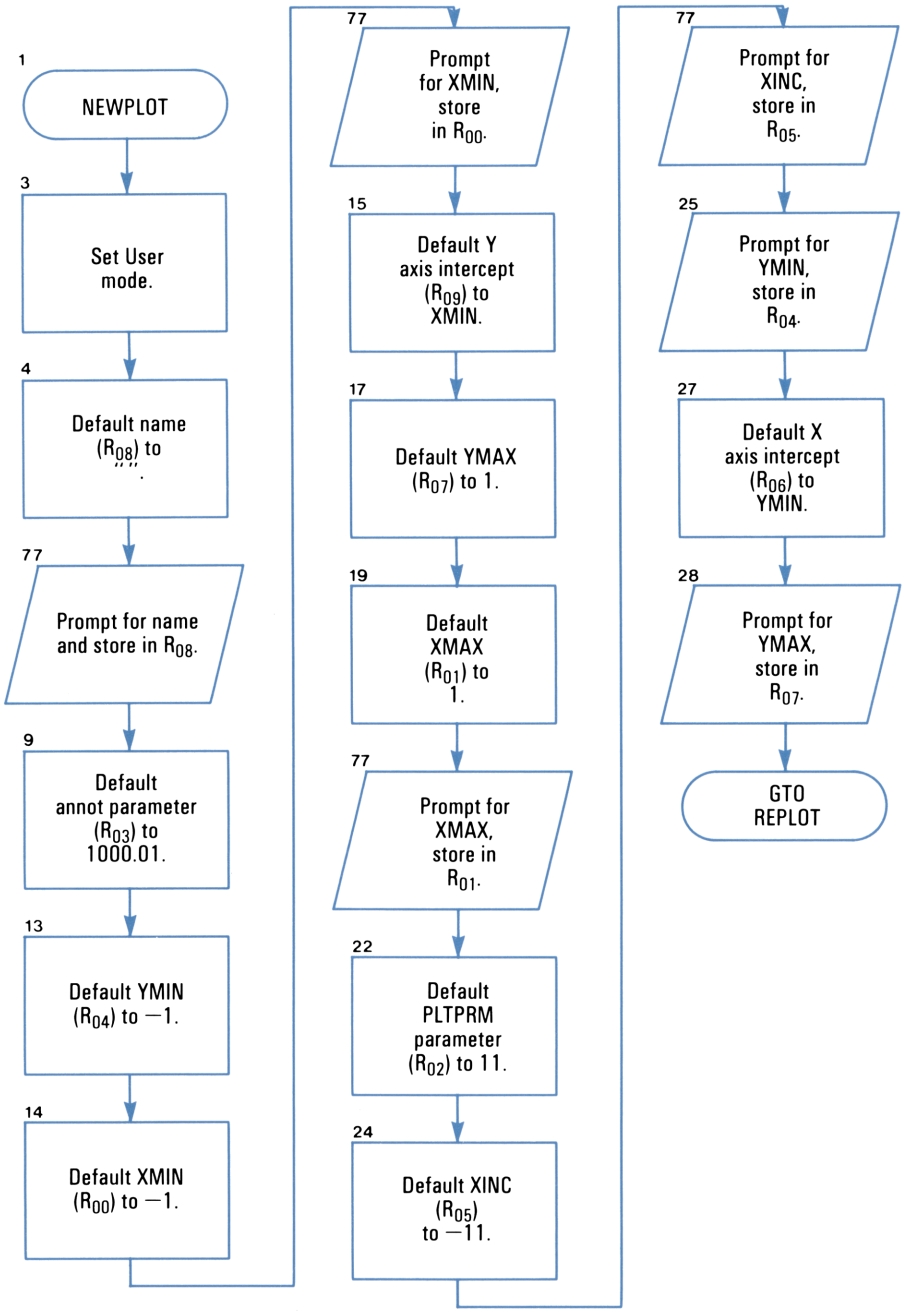
```
04♦LBL "PLPLOTP"
05 XEQ 01
06 XEQ "PLINIT"
07 XEQ "PLTUXY"
08 XEQ "PLANOT"
09 RTN
```

Rearranges data base from PRPLOT for NEWPLOT.

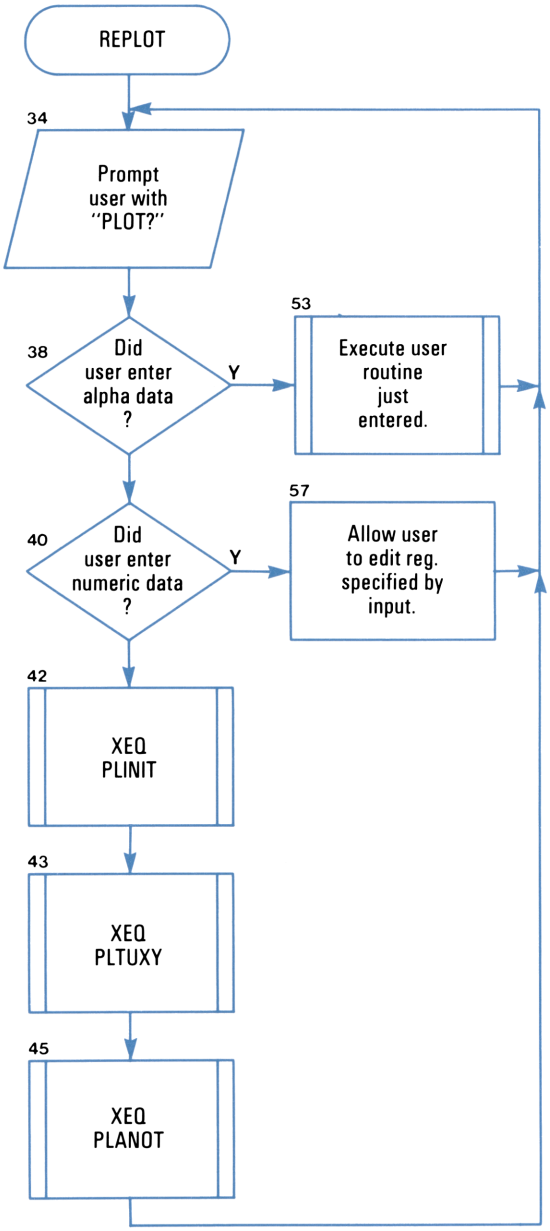
```
10♦LBL 01
11 RCL 01
12 STO 07
13 RCL 10
14 STO 05
15 RCL 00
16 RCL 04
17 ENTER↑
18 SIGN
19 X=0?
20 RDN
21 RDN
22 STO 06
23 RCL 00
24 STO 04
25 RCL 09
26 STO 01
27 RCL 08
28 STO 09
29 STO 00
30 RCL 11
31 STO 08
32 1000.01
33 STO 03
34 11
35 STO 02
36 END
```

Flowcharts

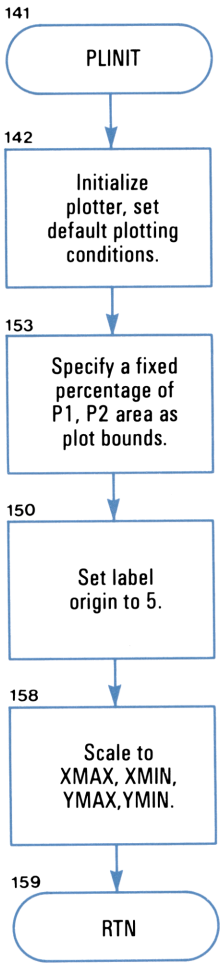
NEWPLOT



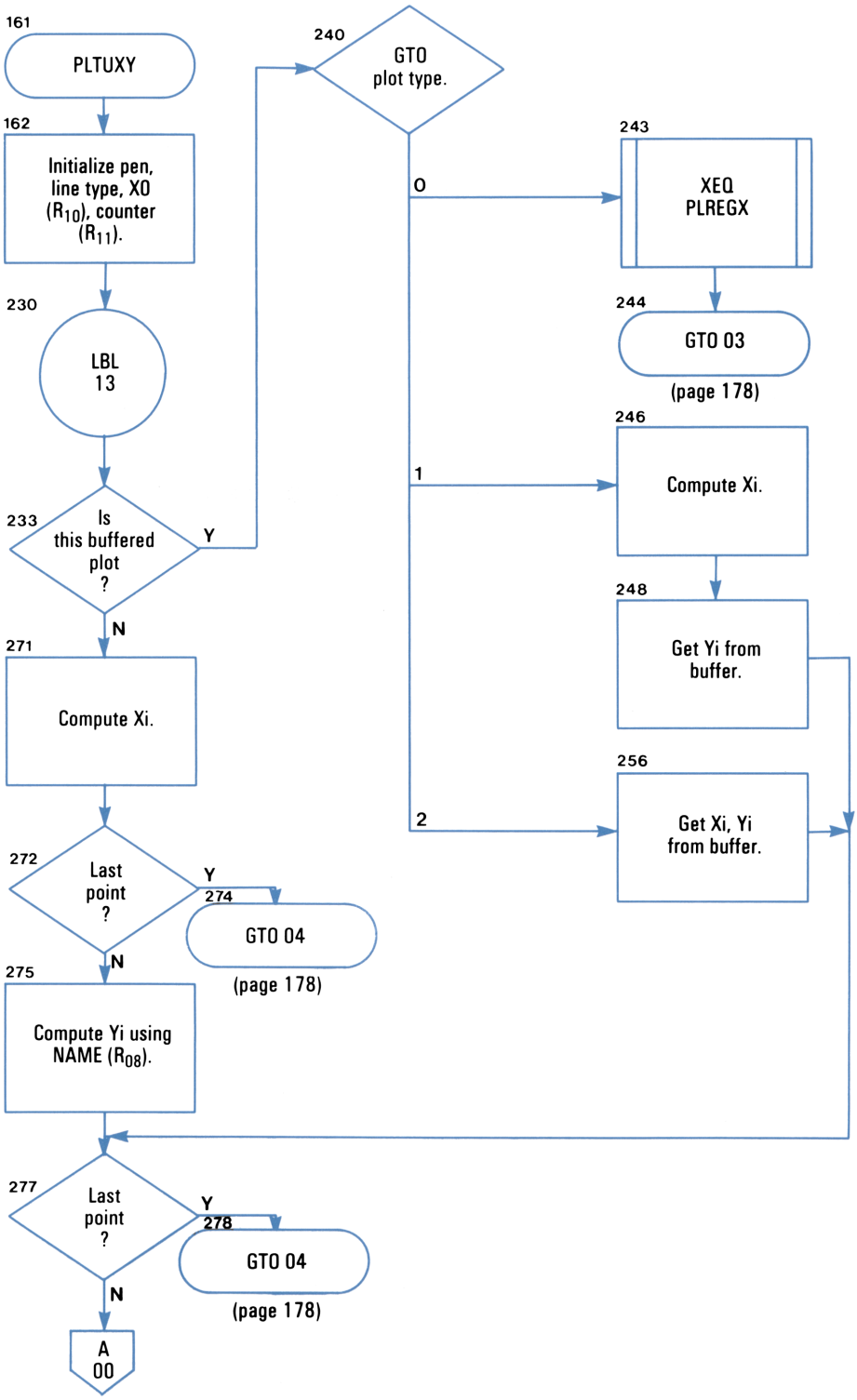
REPLOT



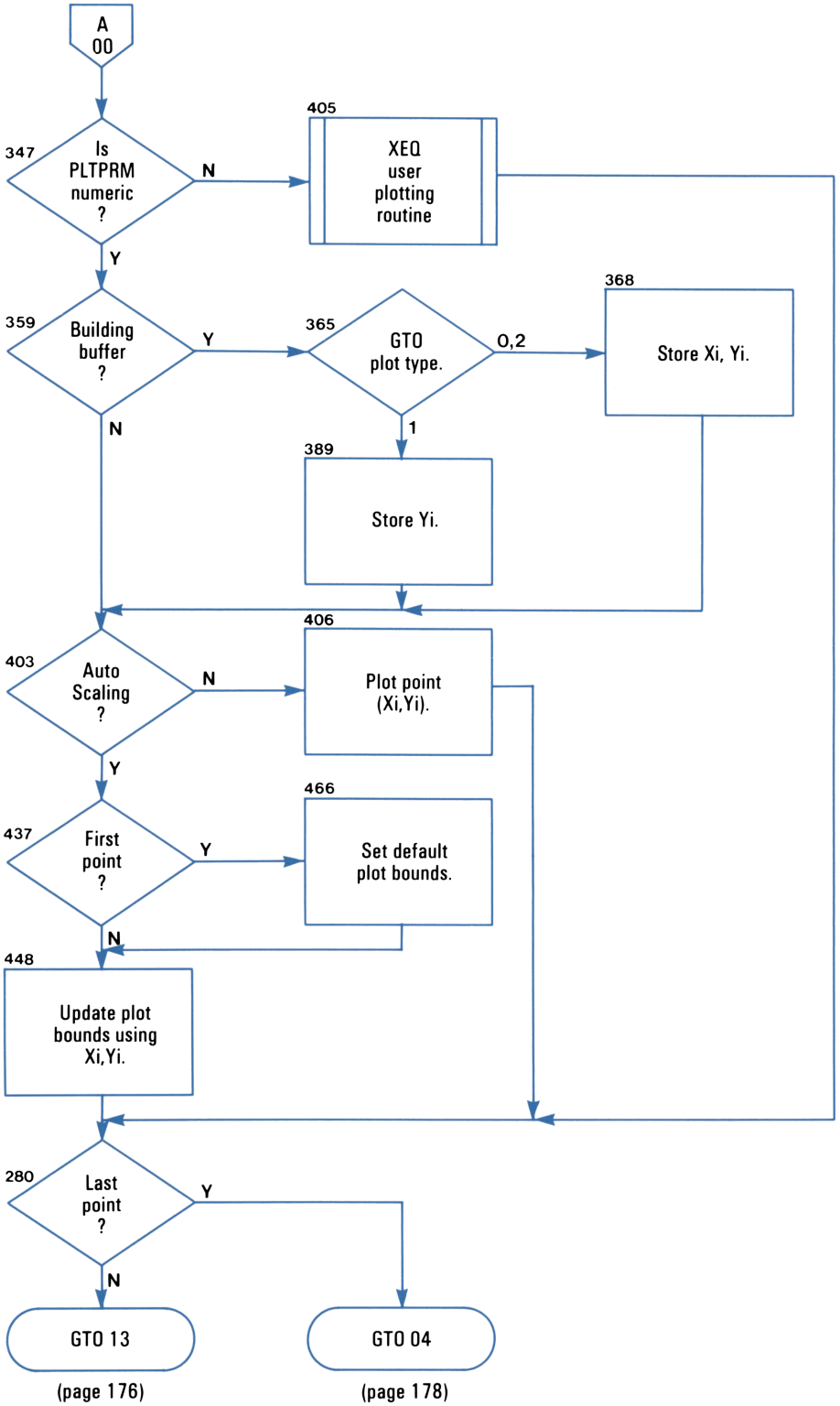
PLINIT



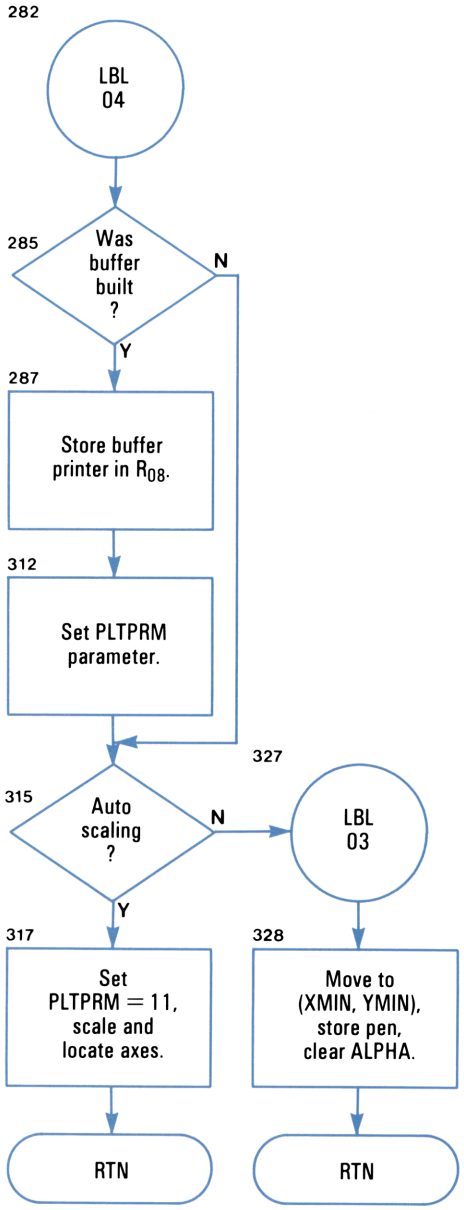
PLTUXY


161
PLTUXY
162
Initialize pen, line type, XO (R_{10}), counter (R_{11}).
230
LBL 13
233
Is this buffered plot ?
Y
N
271
Compute Xi.
272
Last point ?
Y
274
GTO 04
(page 178)
N
275
Compute Yi using NAME (R_{08}).
277
Last point ?
Y
278
GTO 04
(page 178)
N
A 00
240
GTO plot type.
0
243
XEQ PLREGX
244
GTO 03
(page 178)
1
246
Compute Xi.
248
Get Yi from buffer.
2
256
Get Xi, Yi from buffer.

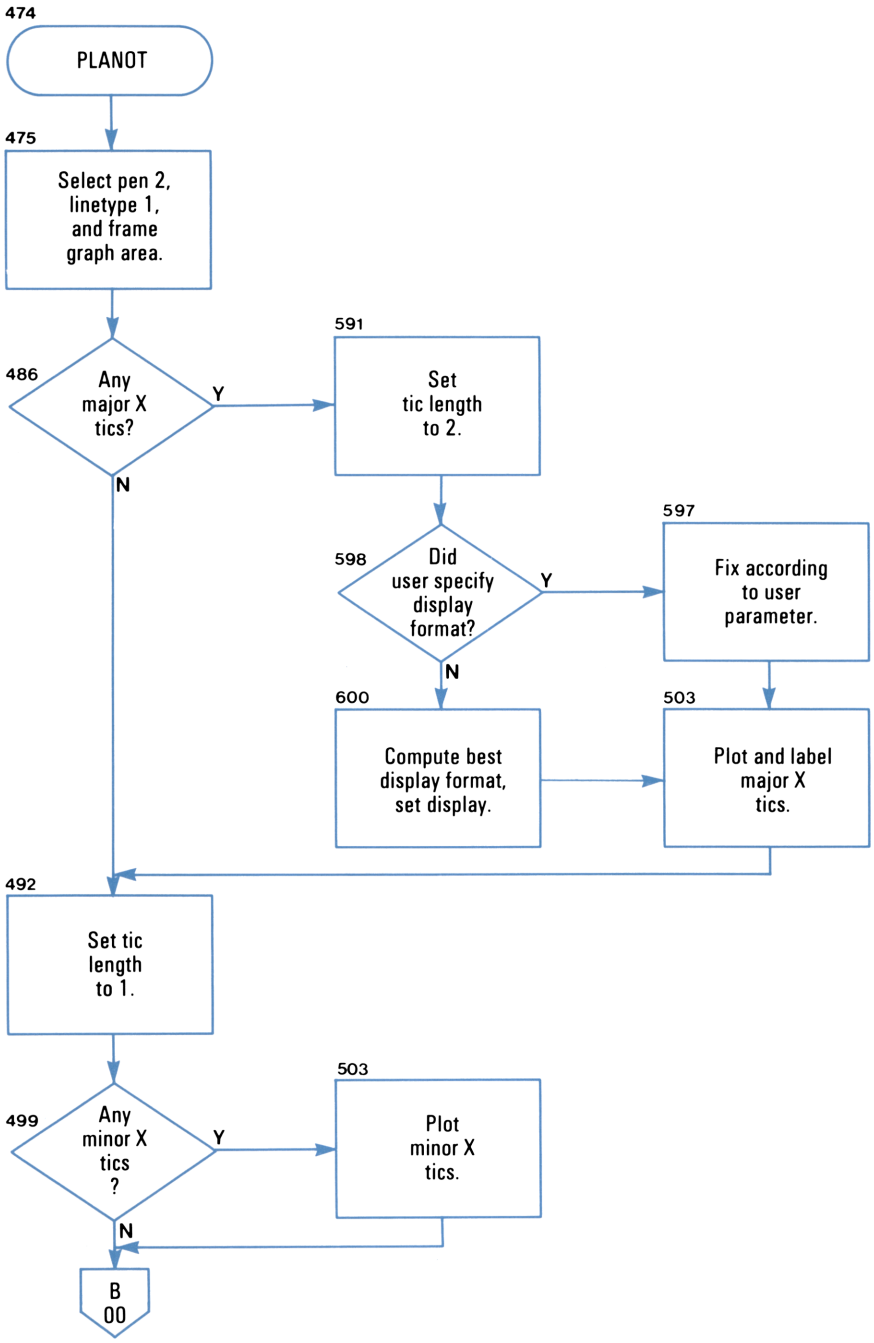
PLTUXY cont.


A
00
347
Is
PLTPRM
numeric
?
N
405
XEQ
user
plotting
routine
Y
359
Building
buffer
?
Y
365
GTO
plot type.
0,2
368
Store Xi, Yi.
N
1
389
Store Yi.
403
Auto
Scaling
?
N
406
Plot point
(Xi,Yi).
Y
437
First
point
?
Y
466
Set default
plot bounds.
N
448
Update plot
bounds using
Xi,Yi.
280
Last
point
?
Y
N
GTO 13
(page 176)
GTO 04
(page 178)

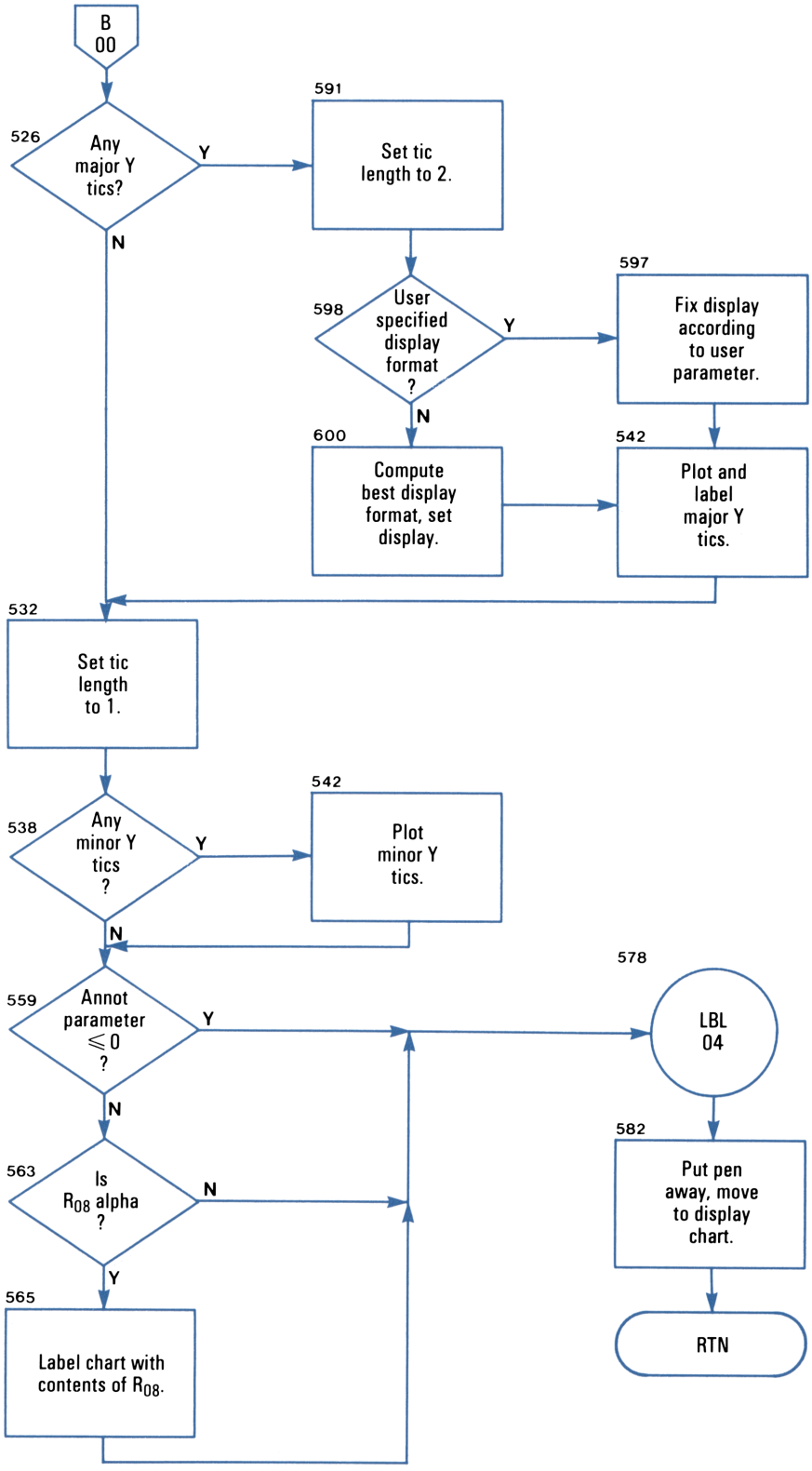
PLTUXY cont.


282
LBL
04
285
Was
buffer
built
?
N
Y
287
Store buffer
printer in R_{08}.
312
Set PLTPRM
parameter.
327
315
Auto
scaling
?
N
LBL
03
Y
317
Set
PLTPRM = 11,
scale and
locate axes.
328
Move to
(XMIN, YMIN),
store pen,
clear ALPHA.
RTN
RTN

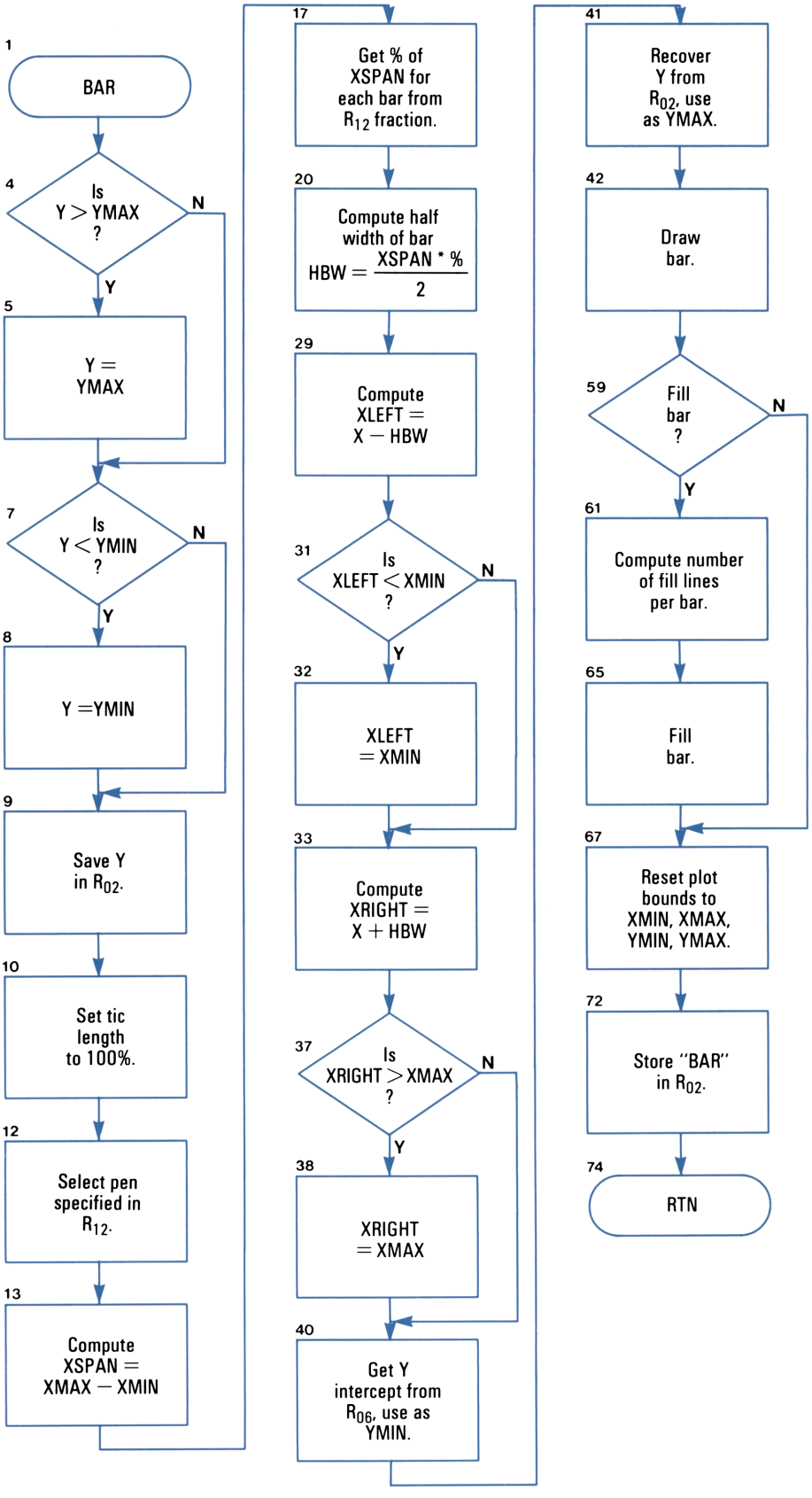
PLANOT


474
PLANOT
475
Select pen 2, linetype 1, and frame graph area.
486
Any major X tics?
Y
N
591
Set tic length to 2.
598
Did user specify display format?
Y
N
597
Fix according to user parameter.
600
Compute best display format, set display.
503
Plot and label major X tics.
492
Set tic length to 1.
499
Any minor X tics ?
Y
N
503
Plot minor X tics.
B
00

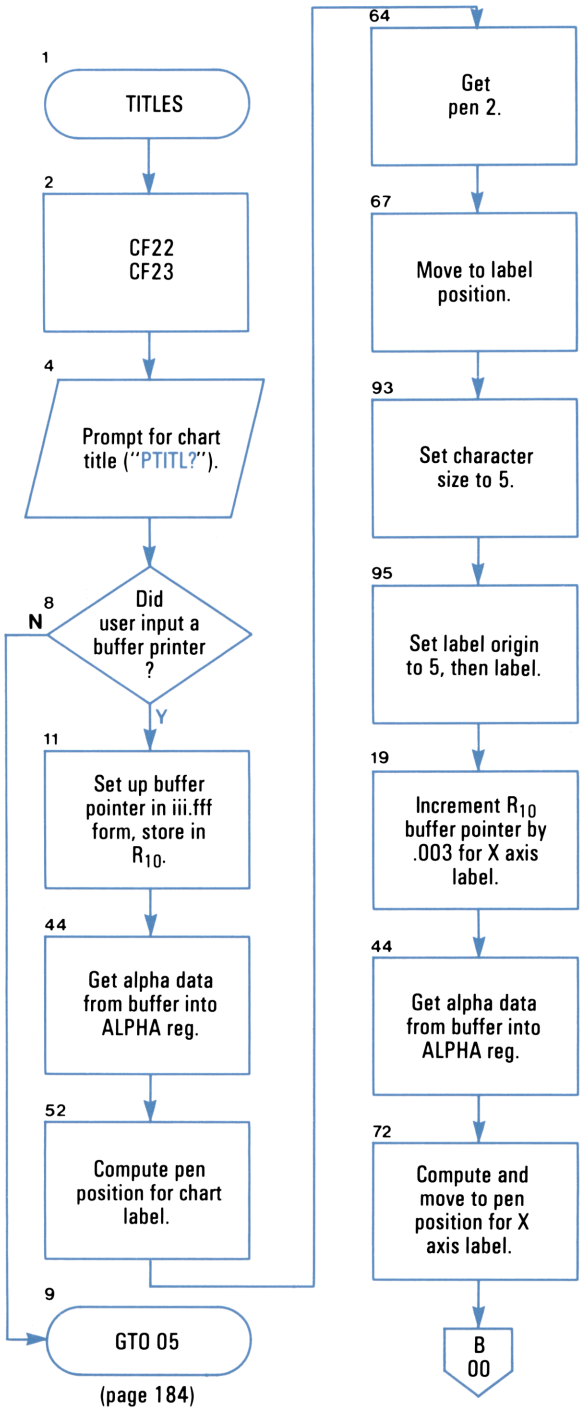
PLANOT cont.


B
00
526
Any major Y tics?
Y
N
591
Set tic length to 2.
598
User specified display format ?
Y
N
597
Fix display according to user parameter.
600
Compute best display format, set display.
542
Plot and label major Y tics.
532
Set tic length to 1.
538
Any minor Y tics ?
Y
N
542
Plot minor Y tics.
559
Annot parameter ≤ 0 ?
Y
N
578
LBL 04
563
Is R_{08} alpha ?
N
Y
582
Put pen away, move to display chart.
565
Label chart with contents of R_{08}.
RTN

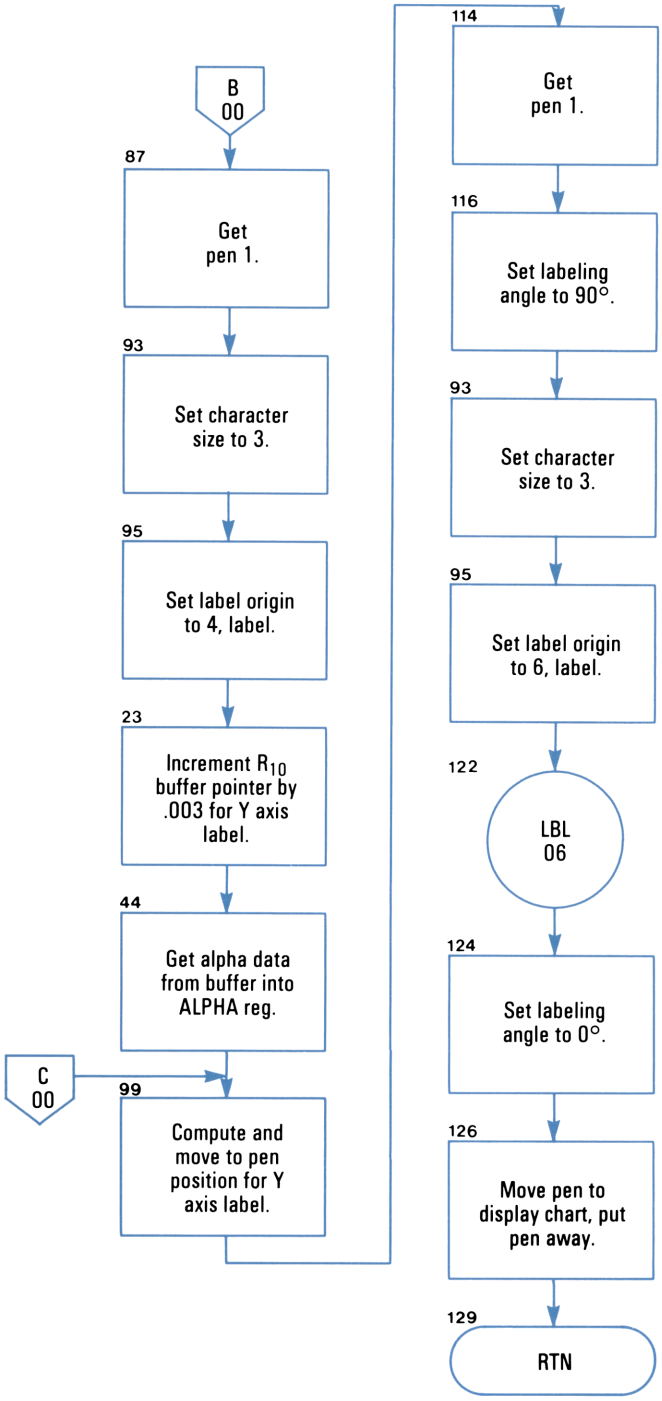
BAR


1
BAR
4
Is Y > YMAX ?
N
Y
5
Y = YMAX
7
Is Y < YMIN ?
N
Y
8
Y =YMIN
9
Save Y in R_{02}.
10
Set tic length to 100%.
12
Select pen specified in R_{12}.
13
Compute XSPAN = XMAX − XMIN
17
Get % of XSPAN for each bar from R_{12} fraction.
20
Compute half width of bar $HBW = \frac{XSPAN * \%}{2}$
29
Compute XLEFT = X − HBW
31
Is XLEFT < XMIN ?
N
Y
32
XLEFT = XMIN
33
Compute XRIGHT = X + HBW
37
Is XRIGHT > XMAX ?
N
Y
38
XRIGHT = XMAX
40
Get Y intercept from R_{06}, use as YMIN.
41
Recover Y from R_{02}, use as YMAX.
42
Draw bar.
59
Fill bar ?
N
Y
61
Compute number of fill lines per bar.
65
Fill bar.
67
Reset plot bounds to XMIN, XMAX, YMIN, YMAX.
72
Store "BAR" in R_{02}.
74
RTN

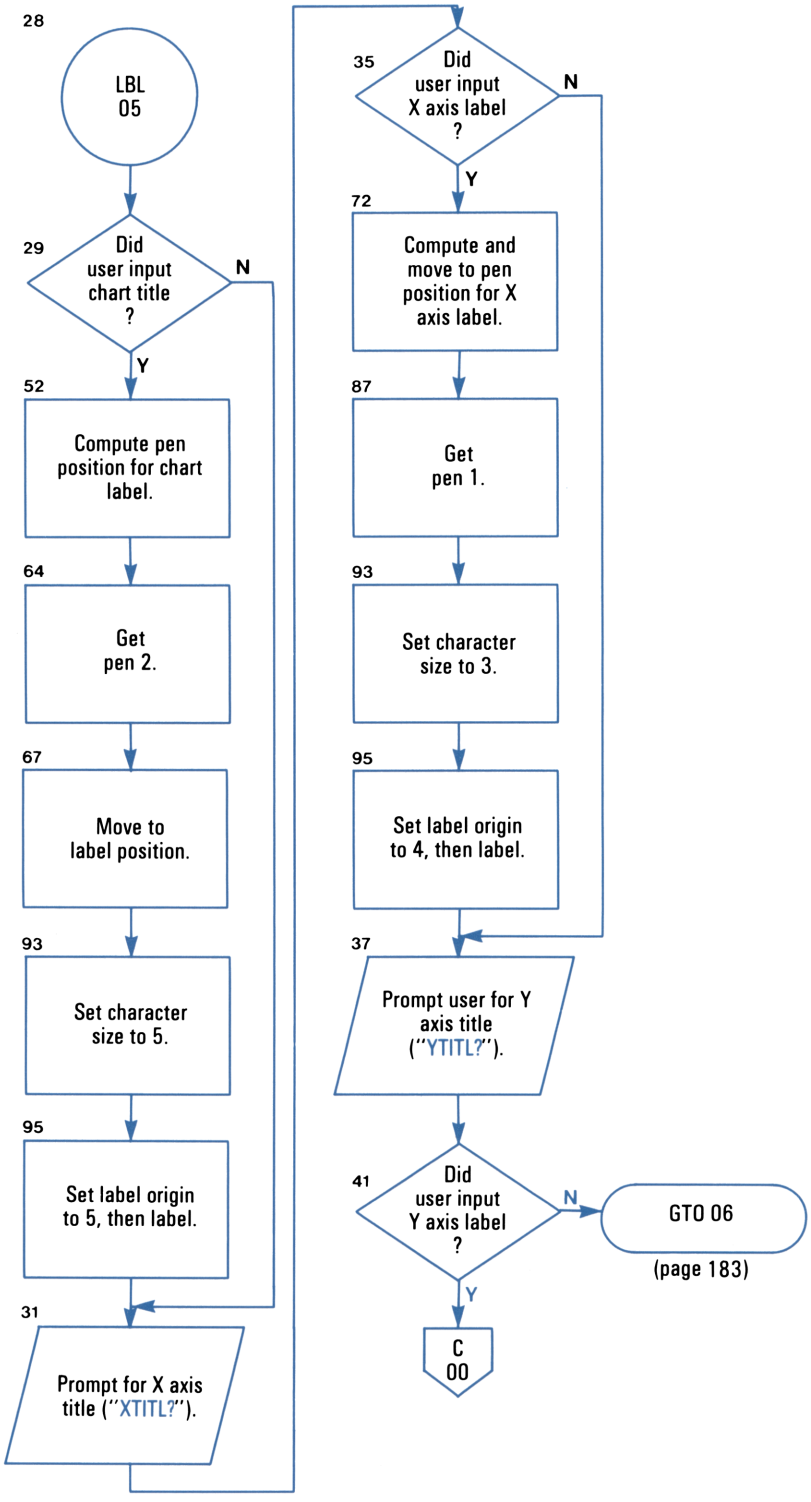
TITLES


1
TITLES
2
CF22
CF23
4
Prompt for chart title ("PTITL?").
8
Did user input a buffer printer ?
N
Y
11
Set up buffer pointer in iii.fff form, store in R_{10}.
44
Get alpha data from buffer into ALPHA reg.
52
Compute pen position for chart label.
9
GTO 05
(page 184)
64
Get pen 2.
67
Move to label position.
93
Set character size to 5.
95
Set label origin to 5, then label.
19
Increment R_{10} buffer pointer by .003 for X axis label.
44
Get alpha data from buffer into ALPHA reg.
72
Compute and move to pen position for X axis label.
B
00

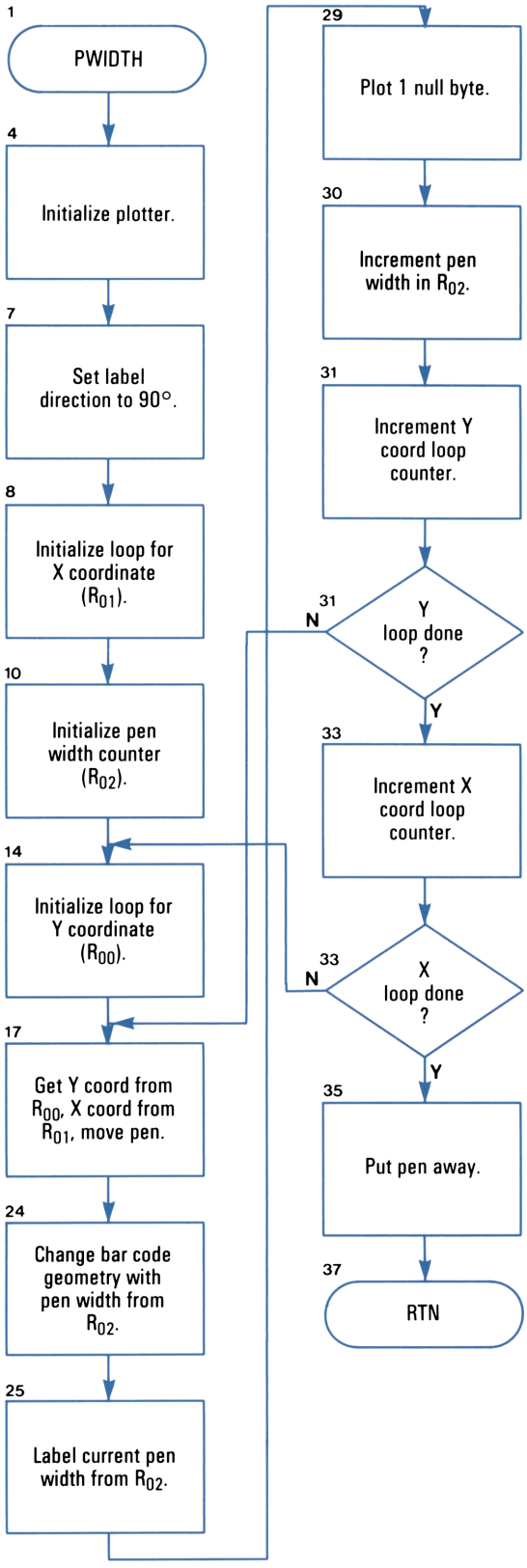
TITLES cont.


B
00
87
Get
pen 1.
93
Set character
size to 3.
95
Set label origin
to 4, label.
23
Increment R_{10}
buffer pointer by
.003 for Y axis
label.
44
Get alpha data
from buffer into
ALPHA reg.
C
00
99
Compute and
move to pen
position for Y
axis label.
114
Get
pen 1.
116
Set labeling
angle to 90°.
93
Set character
size to 3.
95
Set label origin
to 6, label.
122
LBL
06
124
Set labeling
angle to 0°.
126
Move pen to
display chart, put
pen away.
129
RTN

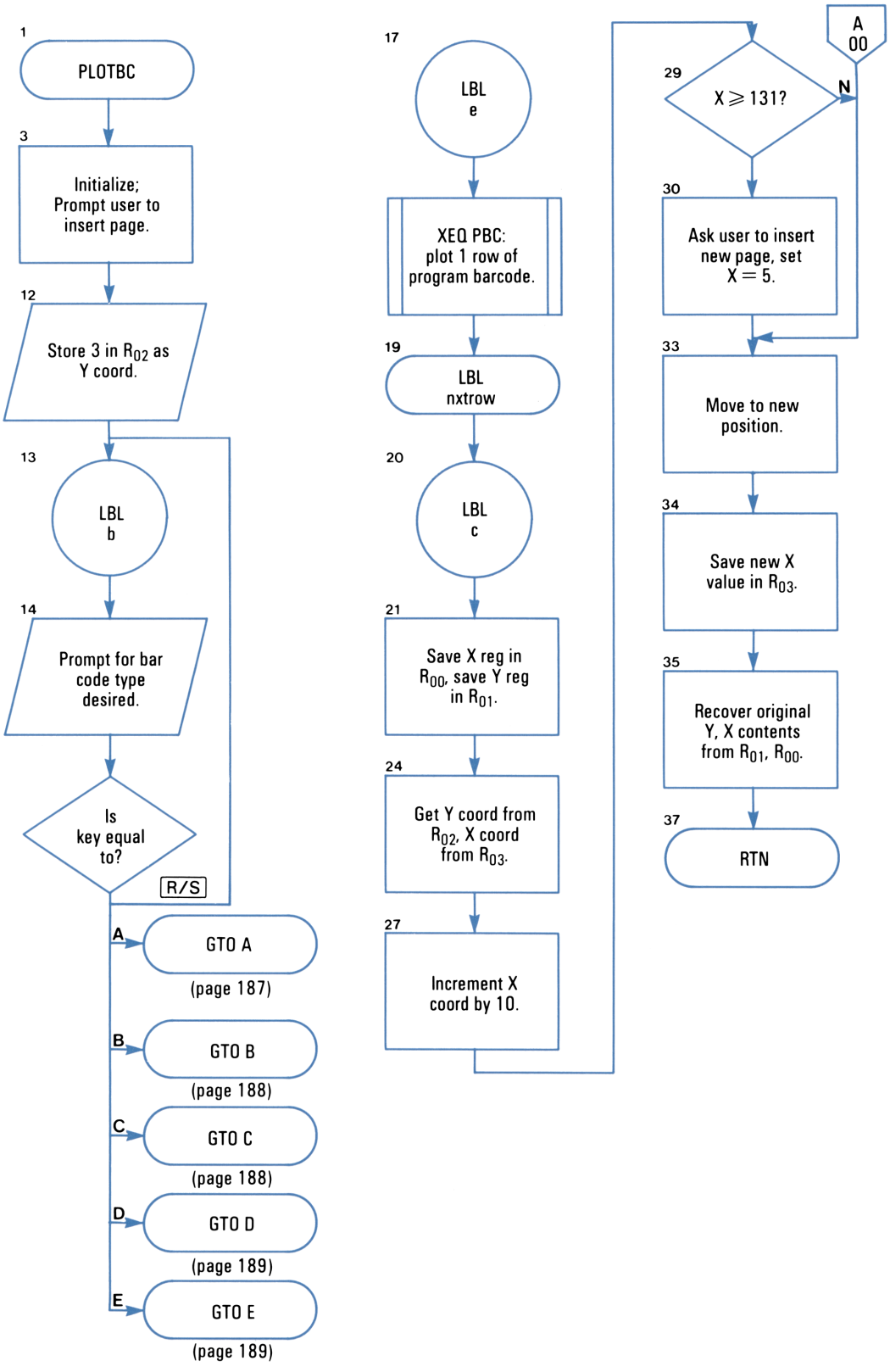
TITLES cont.


28
LBL
05
29
Did
user input
chart title
?
N
Y
52
Compute pen
position for chart
label.
64
Get
pen 2.
67
Move to
label position.
93
Set character
size to 5.
95
Set label origin
to 5, then label.
31
Prompt for X axis
title ("XTITL?").
35
Did
user input
X axis label
?
N
Y
72
Compute and
move to pen
position for X
axis label.
87
Get
pen 1.
93
Set character
size to 3.
95
Set label origin
to 4, then label.
37
Prompt user for Y
axis title
("YTITL?").
41
Did
user input
Y axis label
?
N
Y
GTO 06
(page 183)
C
00

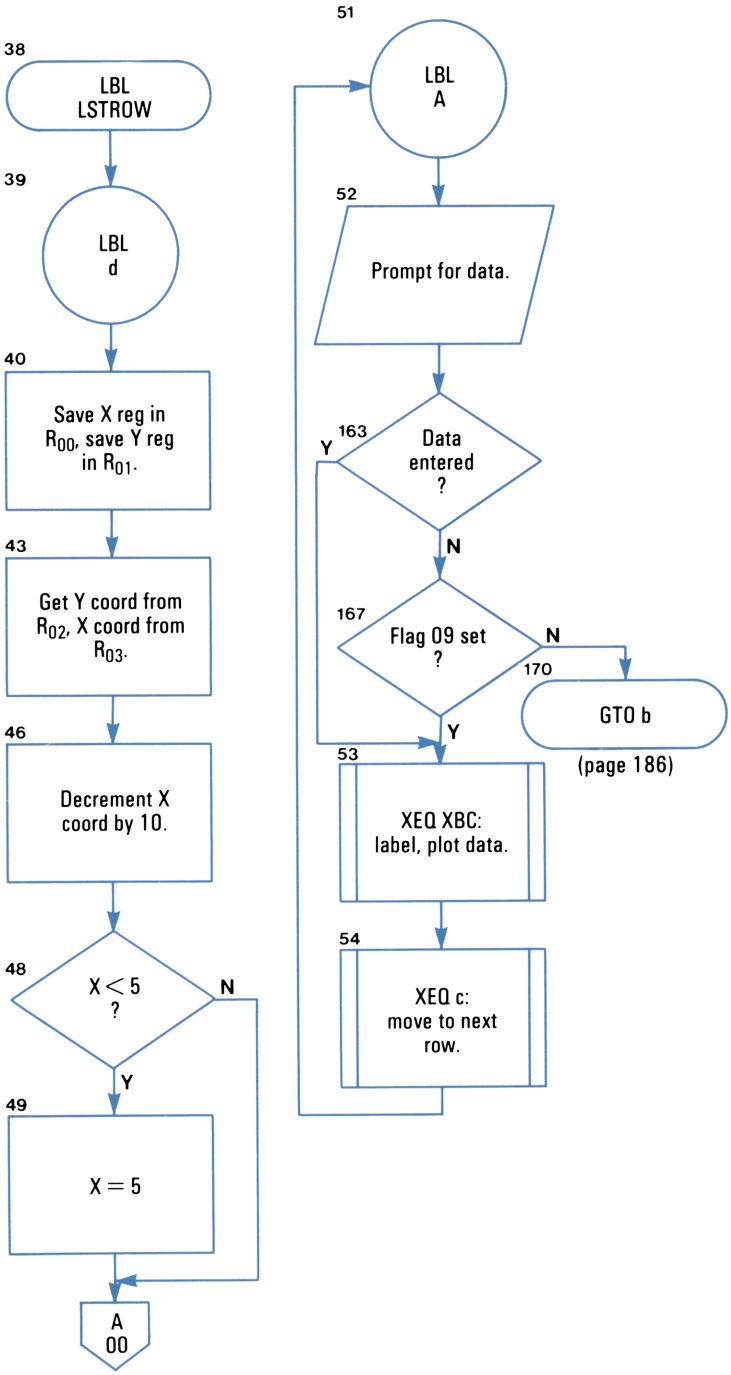
PWIDTH


1
PWIDTH
4
Initialize plotter.
7
Set label direction to 90°.
8
Initialize loop for X coordinate (R_{01}).
10
Initialize pen width counter (R_{02}).
14
Initialize loop for Y coordinate (R_{00}).
17
Get Y coord from R_{00}, X coord from R_{01}, move pen.
24
Change bar code geometry with pen width from R_{02}.
25
Label current pen width from R_{02}.
29
Plot 1 null byte.
30
Increment pen width in R_{02}.
31
Increment Y coord loop counter.
31
Y loop done ?
N
Y
33
Increment X coord loop counter.
33
X loop done ?
N
Y
35
Put pen away.
37
RTN

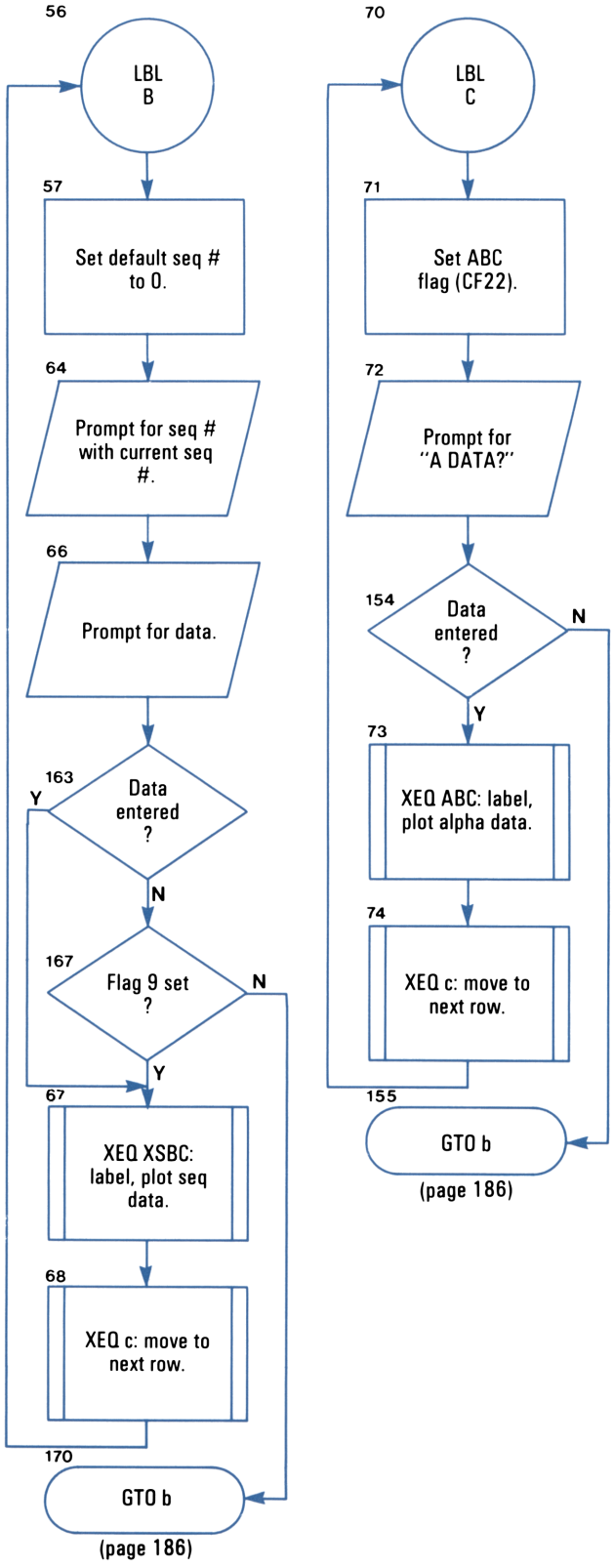
PLOTBC


1
PLOTBC
3
Initialize; Prompt user to insert page.
12
Store 3 in R_{02} as Y coord.
13
LBL b
14
Prompt for bar code type desired.
Is key equal to?
R/S
A
GTO A
(page 187)
B
GTO B
(page 188)
C
GTO C
(page 188)
D
GTO D
(page 189)
E
GTO E
(page 189)
17
LBL e
XEQ PBC: plot 1 row of program barcode.
19
LBL nxtrow
20
LBL c
21
Save X reg in R_{00}, save Y reg in R_{01}.
24
Get Y coord from R_{02}, X coord from R_{03}.
27
Increment X coord by 10.
29
X ⩾ 131?
N
A
00
30
Ask user to insert new page, set X = 5.
33
Move to new position.
34
Save new X value in R_{03}.
35
Recover original Y, X contents from R_{01}, R_{00}.
37
RTN

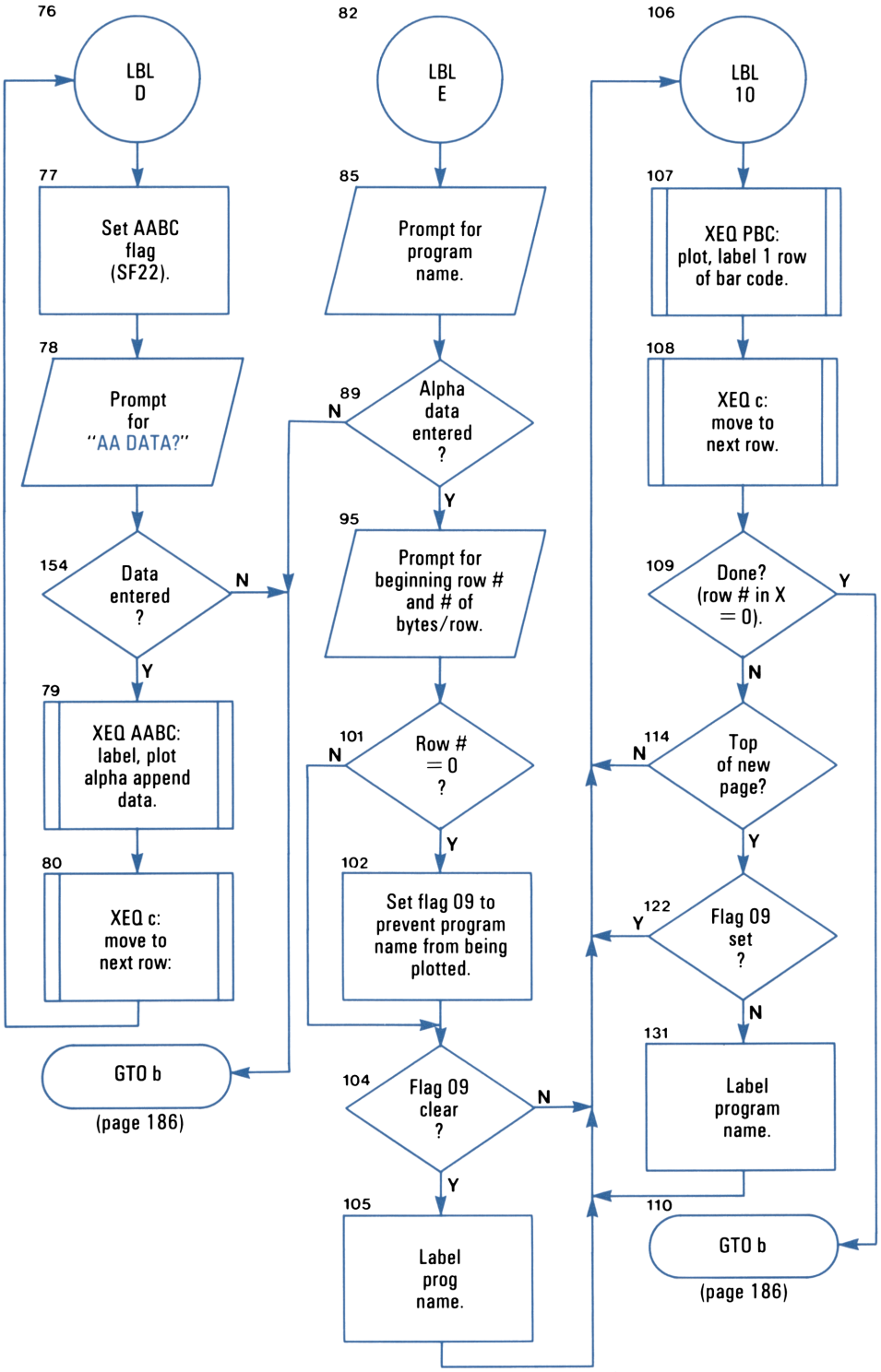
PLOTBC cont.


38
LBL
LSTROW
39
LBL
d
40
Save X reg in
R00, save Y reg
in R01.
43
Get Y coord from
R02, X coord from
R03.
46
Decrement X
coord by 10.
48
X < 5
?
N
Y
49
X = 5
A
00
51
LBL
A
52
Prompt for data.
163
Data
entered
?
Y
N
167
Flag 09 set
?
N
170
GTO b
(page 186)
Y
53
XEQ XBC:
label, plot data.
54
XEQ c:
move to next
row.

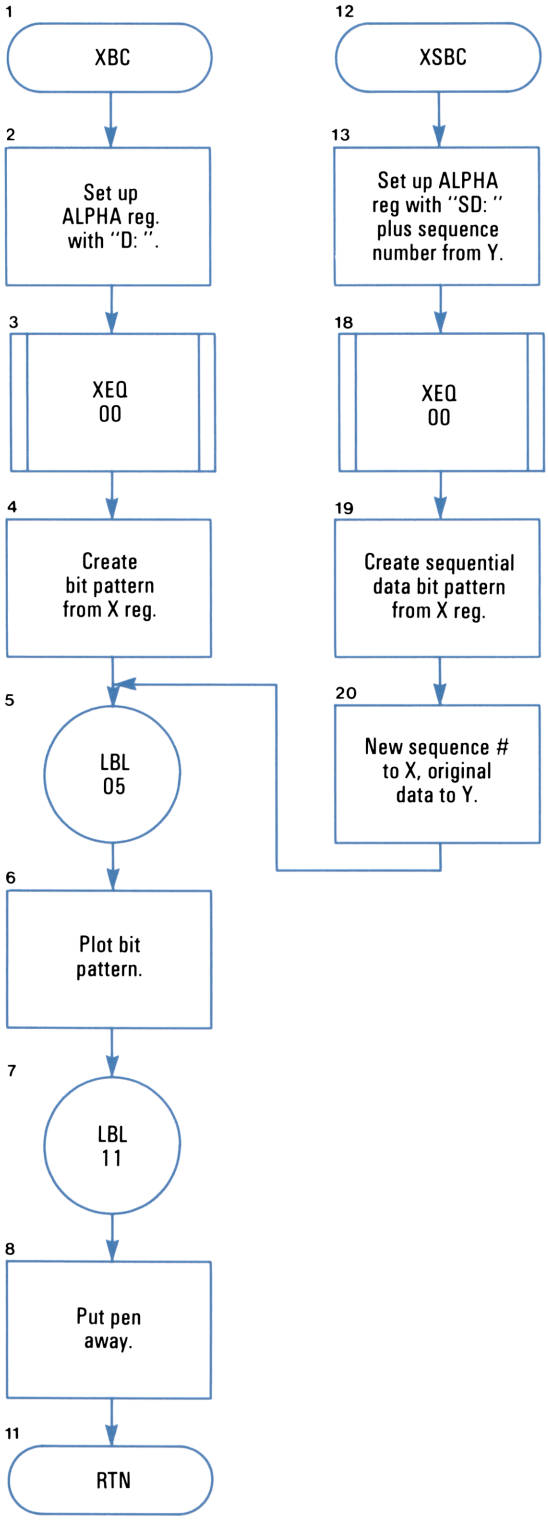
PLOTBC cont.


56
LBL
B
57
Set default seq #
to 0.
64
Prompt for seq #
with current seq
#.
66
Prompt for data.
163
Data
entered
?
Y
N
167
Flag 9 set
?
N
Y
67
XEQ XSBC:
label, plot seq
data.
68
XEQ c: move to
next row.
170
GTO b
(page 186)
70
LBL
C
71
Set ABC
flag (CF22).
72
Prompt for
"A DATA?"
154
Data
entered
?
N
Y
73
XEQ ABC: label,
plot alpha data.
74
XEQ c: move to
next row.
155
GTO b
(page 186)

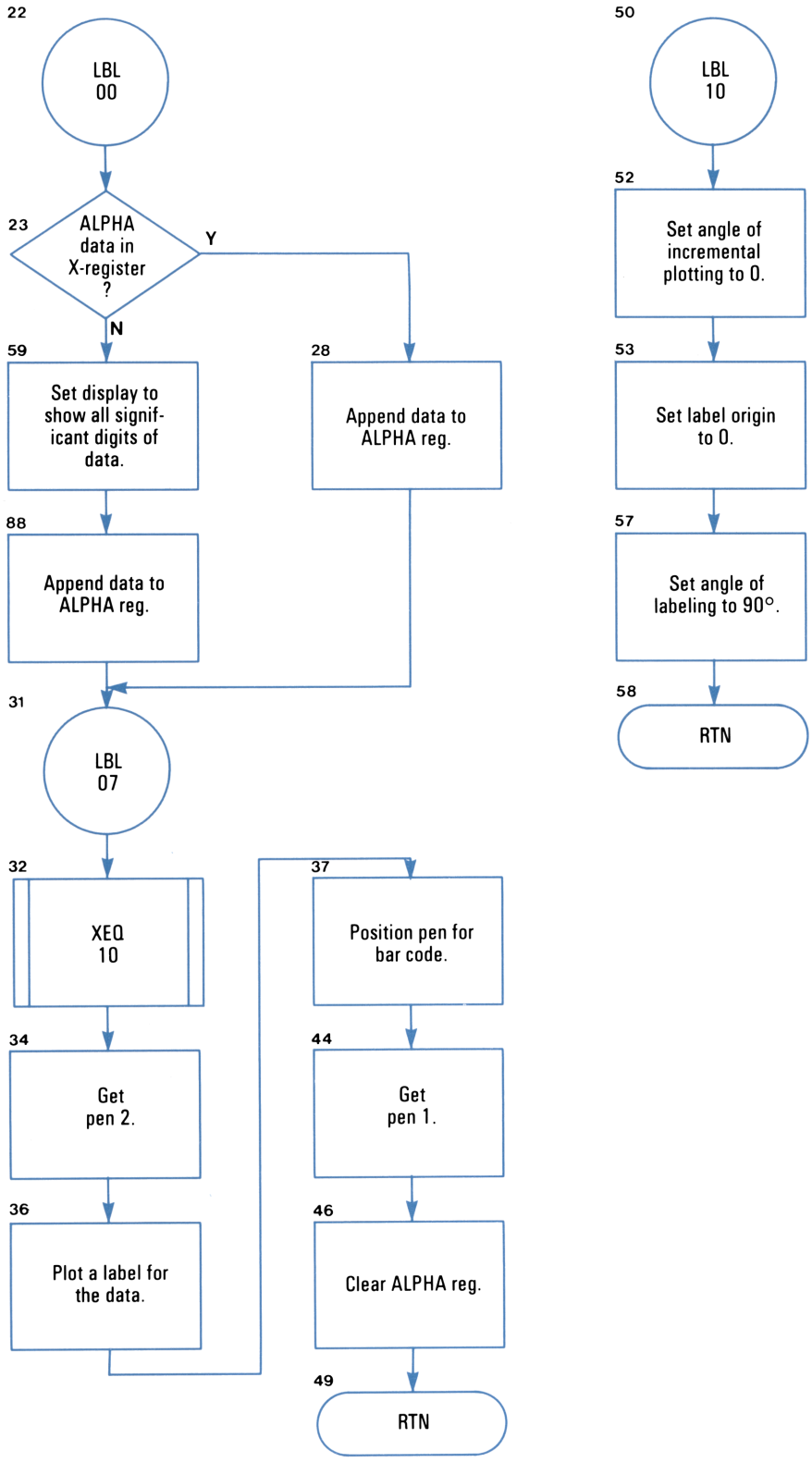
PLOTBC cont.


76
LBL D
77
Set AABC flag (SF22).
78
Prompt for "AA DATA?"
154
Data entered ?
N
Y
79
XEQ AABC: label, plot alpha append data.
80
XEQ c: move to next row:
GTO b
(page 186)
82
LBL E
85
Prompt for program name.
89
Alpha data entered ?
N
Y
95
Prompt for beginning row # and # of bytes/row.
101
Row # = 0 ?
N
Y
102
Set flag 09 to prevent program name from being plotted.
104
Flag 09 clear ?
N
Y
105
Label prog name.
106
LBL 10
107
XEQ PBC: plot, label 1 row of bar code.
108
XEQ c: move to next row.
109
Done? (row # in X = 0).
Y
N
114
Top of new page?
N
Y
122
Flag 09 set ?
Y
N
131
Label program name.
110
GTO b
(page 186)

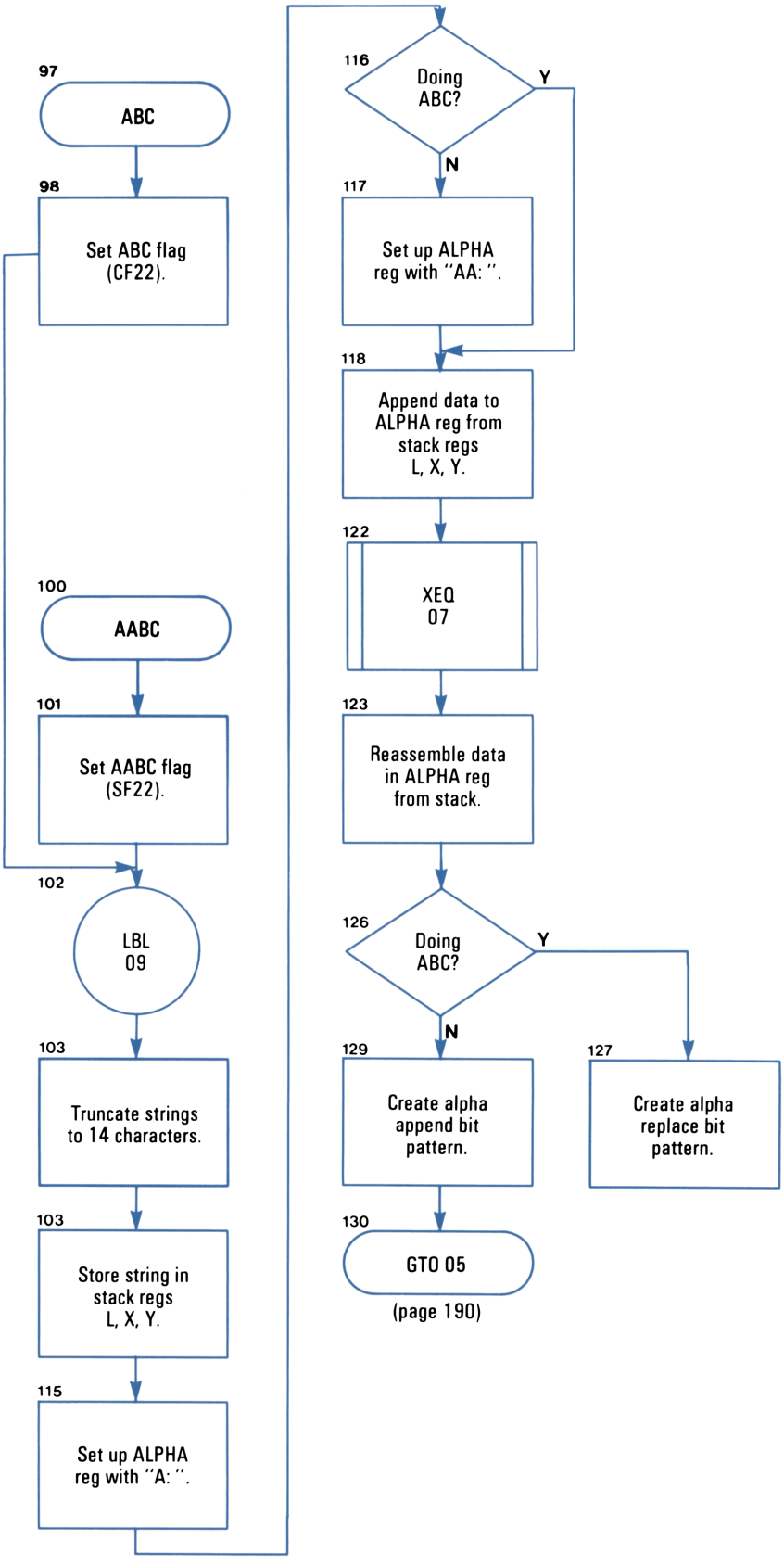
XBC


1
XBC
2
Set up ALPHA reg. with "D: ".
3
XEQ 00
4
Create bit pattern from X reg.
5
LBL 05
6
Plot bit pattern.
7
LBL 11
8
Put pen away.
11
RTN
12
XSBC
13
Set up ALPHA reg with "SD: " plus sequence number from Y.
18
XEQ 00
19
Create sequential data bit pattern from X reg.
20
New sequence # to X, original data to Y.

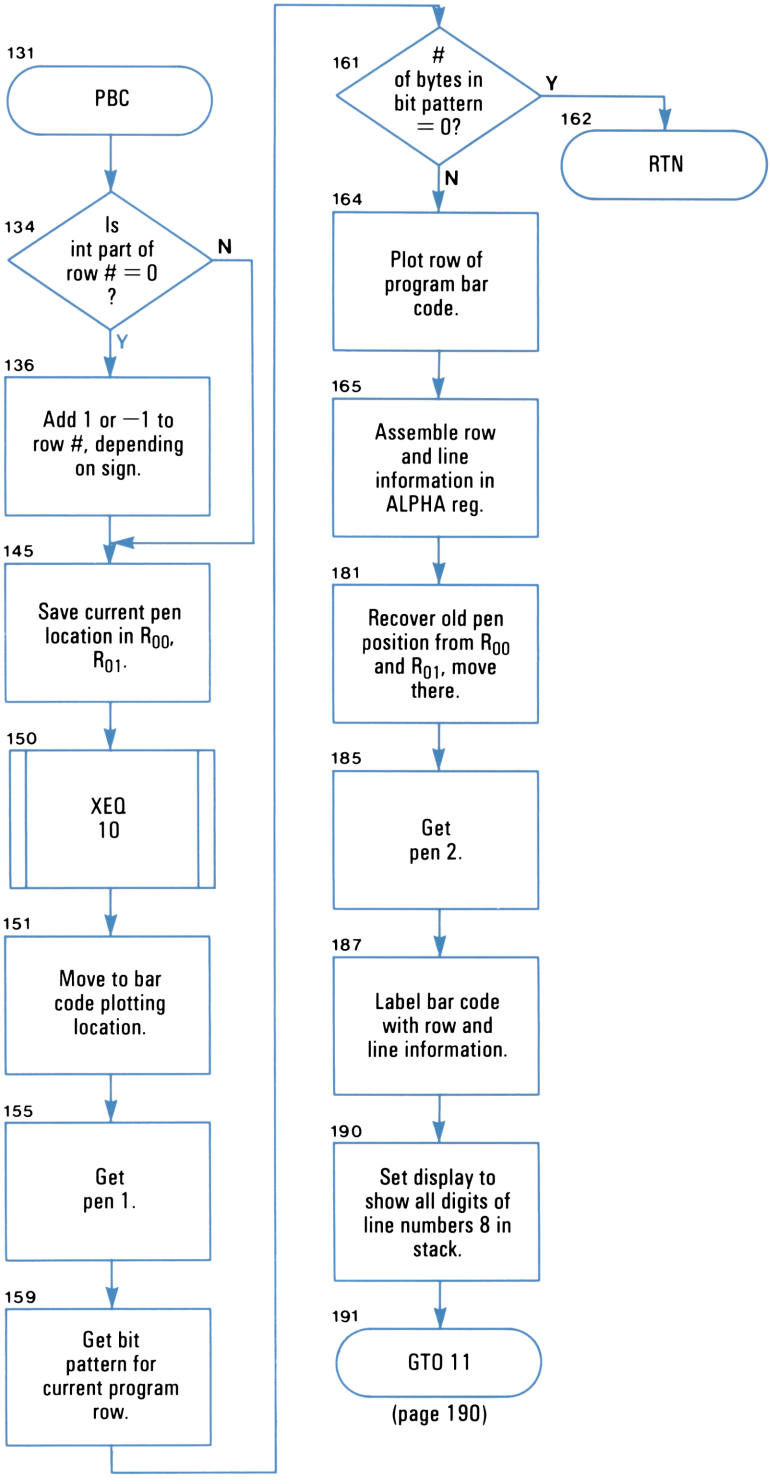
XBC cont.


22
LBL
00
23
ALPHA
data in
X-register
?
Y
N
59
Set display to
show all signif-
icant digits of
data.
28
Append data to
ALPHA reg.
88
Append data to
ALPHA reg.
31
LBL
07
32
XEQ
10
34
Get
pen 2.
36
Plot a label for
the data.
37
Position pen for
bar code.
44
Get
pen 1.
46
Clear ALPHA reg.
49
RTN
50
LBL
10
52
Set angle of
incremental
plotting to 0.
53
Set label origin
to 0.
57
Set angle of
labeling to 90°.
58
RTN

XBC cont.

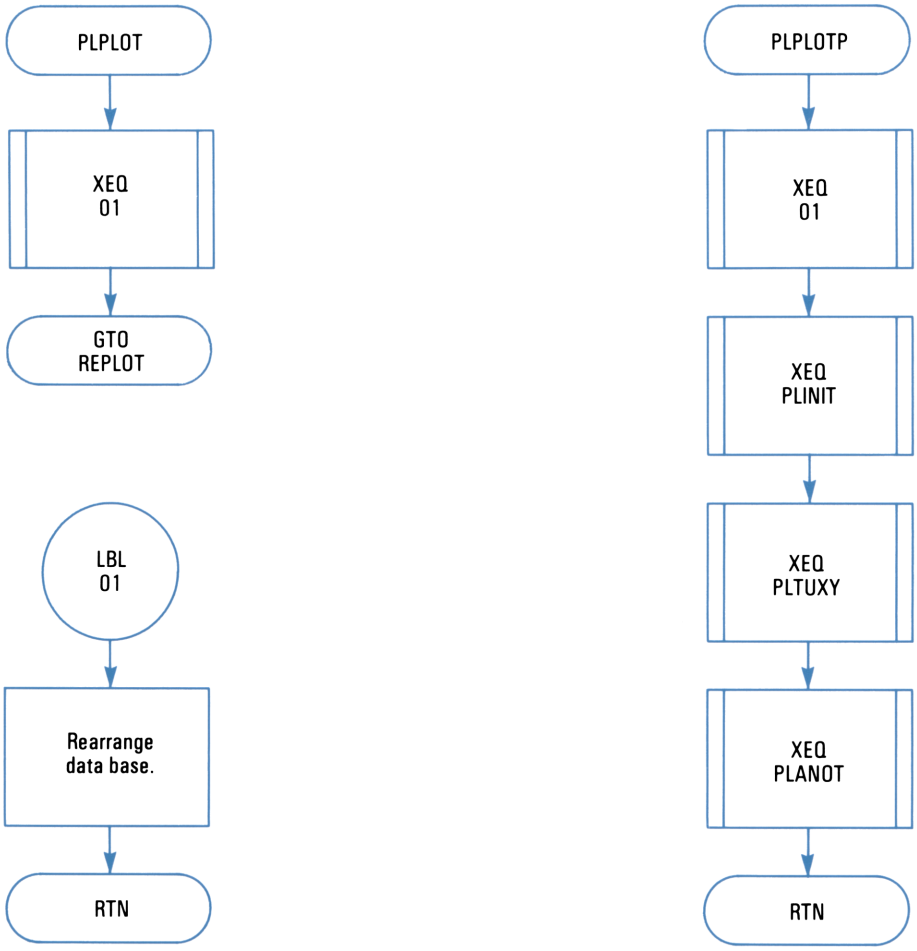

97
ABC
98
Set ABC flag (CF22).
100
AABC
101
Set AABC flag (SF22).
102
LBL 09
103
Truncate strings to 14 characters.
103
Store string in stack regs L, X, Y.
115
Set up ALPHA reg with "A: ".
116
Doing ABC?
Y
N
117
Set up ALPHA reg with "AA: ".
118
Append data to ALPHA reg from stack regs L, X, Y.
122
XEQ 07
123
Reassemble data in ALPHA reg from stack.
126
Doing ABC?
Y
N
129
Create alpha append bit pattern.
127
Create alpha replace bit pattern.
130
GTO 05
(page 190)

XBC cont.



(page 190)

PLPLOT/PLPLOTP



Appendix D

Bar Code

Contents

BAR 195
KWH 196
LABL 198
LDR 198
LINE 199
PLOTBC 199
PLPLOT/PLPLOTP 202
PRBC 202
PWIDTH 203
RAIN 203
SET 205
TERM 205
TITLES 206
XBC 207

BAR

PROGRAM REGISTERS NEEDED: 14

ROW 1: LINES 1-7

ROW 2: LINES 8-16

ROW 3: LINES 17-28

ROW 4: LINES 29-40

ROW 5: LINES 41-51

ROW 6: LINES 52-61

(continued)

ROW 7: LINES 62-72

ROW 8: LINES 73-75

KWH

PROGRAM REGISTERS NEEDED: 39

ROW 1: LINES 1-4

ROW 2: LINES 4-11

ROW 3: LINES 11-19

ROW 4: LINES 19-26

ROW 5: LINES 27-35

ROW 6: LINES 35-42

ROW 7: LINES 42-50

ROW 8: LINES 50-57

ROW 9: LINES 57-63

(continued)

ROW 10: LINES 64-70

ROW 11: LINES 70-72

ROW 12: LINES 72-73

ROW 13: LINES 73-77

ROW 14: LINES 77-86

ROW 15: LINES 86-93

ROW 16: LINES 93-100

ROW 17: LINES 101-107

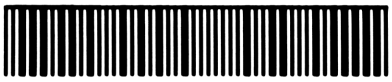
ROW 18: LINES 107-113

ROW 19: LINES 113-119

ROW 20: LINES 119-125

ROW 21: LINES 126-132

ROW 22: LINES 132-132



LABL

PROGRAM REGISTERS NEEDED: 12

ROW 1: LINES 1-2

ROW 2: LINES 3-5

ROW 3: LINES 5-9

ROW 4: LINES 10-16

ROW 5: LINES 16-20

ROW 6: LINES 21-28

ROW 7: LINES 28-31

LDR

PROGRAM REGISTERS NEEDED: 9

ROW 1: LINES 1-3

ROW 2: LINES 3-10

ROW 3: LINES 10-14

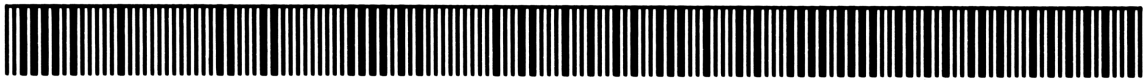
ROW 4: LINES 14-19

ROW 5: LINES 20-21

LINE

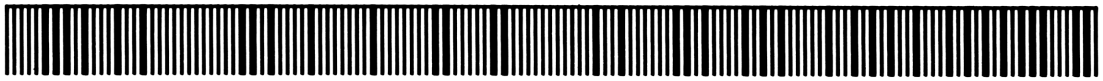
PROGRAM REGISTERS NEEDED: 10

ROW 1: LINES 1-2



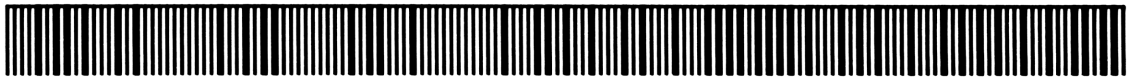
ROW 2: LINES 3-11

ROW 3: LINES 11-19

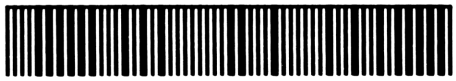


ROW 4: LINES 20-28

ROW 5: LINES 29-33



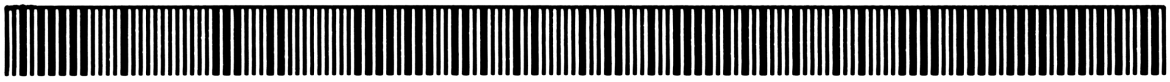
ROW 6: LINES 34-34



PLOTBC

PROGRAM REGISTERS NEEDED: 56

ROW 1: LINES 1-3

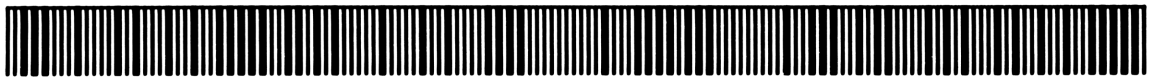


ROW 2: LINES 3-11

ROW 3: LINES 11-14

ROW 4: LINES 14-18

ROW 5: LINES 18-19



(continued)

ROW 6: LINES 19-28

ROW 7: LINES 28-37

ROW 8: LINES 38-40

ROW 9: LINES 41-50

ROW 10: LINES 51-54

ROW 11: LINES 55-61

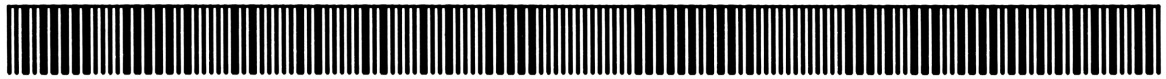
ROW 12: LINES 61-66

ROW 13: LINES 66-70

ROW 14: LINES 70-74

ROW 15: LINES 74-79

ROW 16: LINES 79-83



ROW 17: LINES 83-89

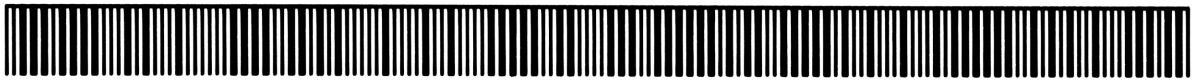
ROW 18: LINES 89-95

(continued)

ROW 19: LINES 95-99

ROW 20: LINES 100-107

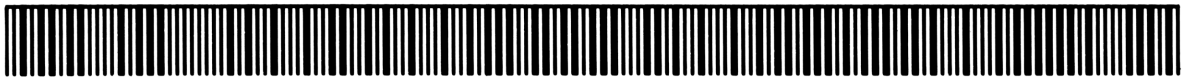
ROW 21: LINES 107-111



ROW 22: LINES 112-120

ROW 23: LINES 120-129

ROW 24: LINES 130-136

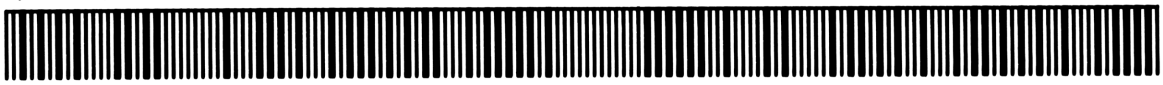


ROW 25: LINES 137-138

ROW 26: LINES 139-148

ROW 27: LINES 148-151

ROW 28: LINES 152-159



ROW 29: LINES 160-165

ROW 30: LINES 165-171

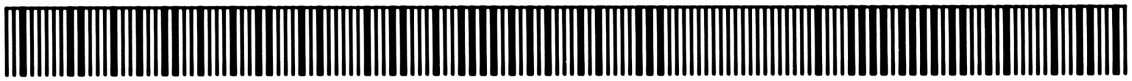
PLPLOT/PLPLOTP

PROGRAM REGISTERS NEEDED: 14

ROW 1: LINES 1-2

ROW 2: LINES 3-4

ROW 3: LINES 4-6



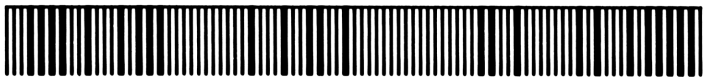
ROW 4: LINES 6-8

ROW 5: LINES 8-14

ROW 6: LINES 15-27

ROW 7: LINES 28-34

ROW 8: LINES 34-36



PRBC

PROGRAM REGISTERS NEEDED: 8

ROW 1: LINES 1-4

ROW 2: LINES 4-9

ROW 3: LINES 10-16

(continued)

ROW 4: LINES 16-22

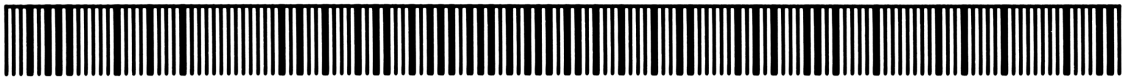
ROW 5: LINES 22-26

PWIDTH

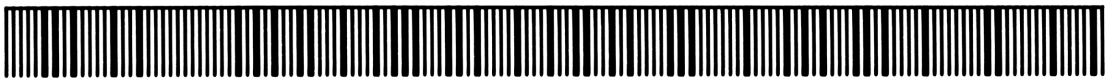
PROGRAM REGISTERS NEEDED: 12

ROW 1: LINES 1-3

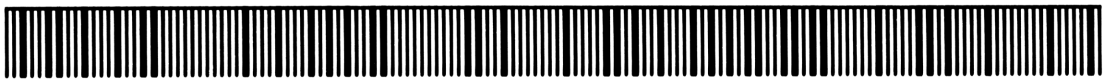
ROW 2: LINES 3-8



ROW 3: LINES 8-14



ROW 4: LINES 14-21



ROW 5: LINES 21-29

ROW 6: LINES 29-36

ROW 7: LINES 36-38

RAIN

PROGRAM REGISTERS NEEDED: 31

ROW 1: LINES 1-4

ROW 2: LINES 4-11

(continued)

ROW 3: LINES 11-17

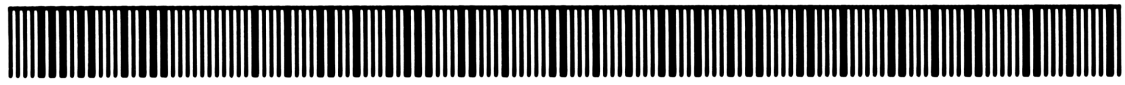
ROW 4: LINES 18-24

ROW 5: LINES 25-32

ROW 6: LINES 32-38

ROW 7: LINES 39-45

ROW 8: LINES 46-52

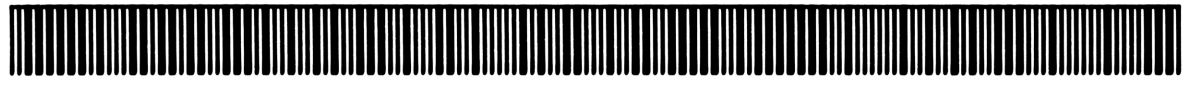


ROW 9: LINES 53-58

ROW 10: LINES 59-66

ROW 11: LINES 67-69

ROW 12: LINES 69-70



ROW 13: LINES 70-72

ROW 14: LINES 72-76

ROW 15: LINES 76-81

(continued)

ROW 16: LINES 81-85

ROW 17: LINES 86-91

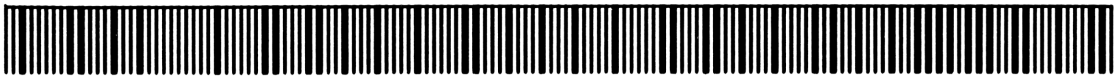
SET

PROGRAM REGISTERS NEEDED: 8

ROW 1: LINES 1-5

ROW 2: LINES 5-12

ROW 3: LINES 13-21



ROW 4: LINES 21-28

ROW 5: LINES 28-28

TERM

PROGRAM REGISTERS NEEDED: 3

ROW 1: LINES 1-5

ROW 2: LINES 5-8

TITLES

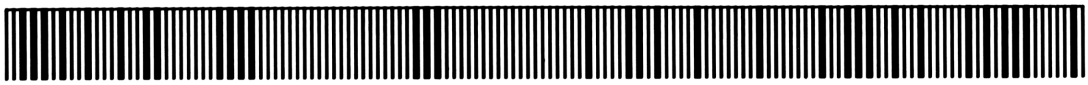
PROGRAM REGISTERS NEEDED: 33

ROW 1: LINES 1-3

ROW 2: LINES 3-8

ROW 3: LINES 9-16

ROW 4: LINES 17-21



ROW 5: LINES 21-25

ROW 6: LINES 25-31

ROW 7: LINES 31-36

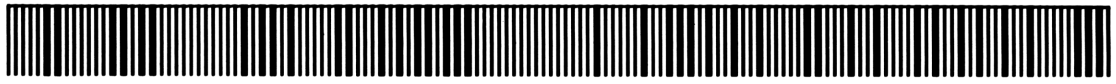
ROW 8: LINES 37-42

ROW 9: LINES 42-49

ROW 10: LINES 50-58

ROW 11: LINES 58-67

ROW 12: LINES 68-76



(continued)

ROW 13: LINES 77-86

ROW 14: LINES 87-96

ROW 15: LINES 96-107

ROW 16: LINES 107-115

ROW 17: LINES 115-124

ROW 18: LINES 124-129

XBC

PROGRAM REGISTERS NEEDED: 52

ROW 1: LINES 1-3

ROW 2: LINES 4-12

ROW 3: LINES 12-15

ROW 4: LINES 15-21

ROW 5: LINES 21-28

ROW 6: LINES 29-35

(continued)

ROW 7: LINES 36-45

ROW 8: LINES 45-55

ROW 9: LINES 56-64

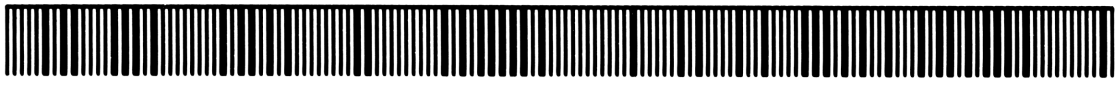
ROW 10: LINES 65-70

ROW 11: LINES 71-78

ROW 12: LINES 79-87

ROW 13: LINES 88-95

ROW 14: LINES 96-99



ROW 15: LINES 100-103

ROW 16: LINES 104-109

ROW 17: LINES 110-115

ROW 18: LINES 115-120

ROW 19: LINES 120-125

(continued)

ROW 20: LINES 126-131

ROW 21: LINES 131-139

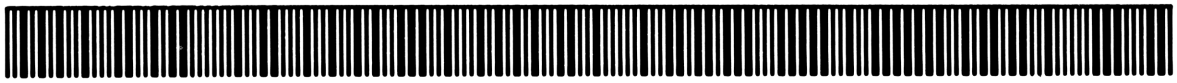
ROW 22: LINES 140-150

ROW 23: LINES 150-158

ROW 24: LINES 159-167

ROW 25: LINES 167-172

ROW 26: LINES 172-175



ROW 27: LINES 176-184

ROW 28: LINES 185-192

ROW 29: LINES 192-192

Appendix E

Reference Information

Contents

Introduction 210
Programmable Function XROM Numbers 211
HP-GL Commands and Series 80 Counterparts 212
Character Codes 212

Introduction

This appendix provides you with XROM information that is useful in some programming operations. Also included, for users familiar with plotters operated by desktop or larger computers, is a comparitive listing of plotting function used in the plotter module, the HP-GL (Hewlett-Packard Graphic's Language), and the Hewlett-Packard Series 80 computers.

Programmable Function XROM Numbers

The HP 82184A Plotter Module's programmable functions can be entered in a program whenever the module is plugged into the HP-41, regardless of whether or not a plotter is also plugged in. While the module is plugged in, program lines containing plotter module functions are displayed and printed using the standard function names. If you later disconnect the module, these program lines are displayed and printed as XROM functions—with two identification numbers. These numbers indicate that the function belongs to a plug-in accessory. The first number identifies the accessory. (XROM accessory numbers 17 and 18 correspond to the plotter module.) The second number identifies the function for that accessory. When you remove the plotter module, its functions have the following XROM numbers.

Function	XROM Number	Function	XROM Number	Function	XROM Number
CLIPUU	XROM 17,01	MOVE	XROM 17,21	PDIR	XROM 18,02
CSIZE	XROM 17,02	PEN	XROM 17,22	PRCL	XROM 18,03
CSIZEO	XROM 17,03	PENDN	XROM 17,23	NEWPLOT	XROM 18,04
DGTIZE	XROM 17,04	PENUP	XROM 17,24	REPLOT	XROM 18,05
DRAW	XROM 17,05	PINIT	XROM 17,25	PLINIT	XROM 18,06
FRAME	XROM 17,06	PLOT	XROM 17,26	PLTUXY	XROM 18,07
GCLEAR	XROM 17,07	PLREGX	XROM 17,27	PLANOT	XROM 18,08
IDRAW	XROM 17,08	RATIO	XROM 17,28	Y?	XROM 18,09
IMOVE	XROM 17,09	RPLOT	XROM 17,29	X?	XROM 18,10
IPLOT	XROM 17,10	SCALE	XROM 17,30	BC	XROM 18, 11
LABEL	XROM 17,11	SETGU	XROM 17,31	BCA	XROM 18, 12
LDIR	XROM 17,12	SETUU	XROM 17,32	BCAA	XROM 18, 13
LIMIT	XROM 17,13	TICLEN	XROM 17,33	BCCKSM	XROM 18, 14
LOCATD	XROM 17,14	UNCLIP	XROM 17,34	BCO	XROM 18, 15
LOCATE	XROM 17,15	WHERE	XROM 17,35	BCP	XROM 18, 16
LORG	XROM 17,16	XAXIS	XROM 17,36	BCREGX	XROM 18, 17
LTYPE	XROM 17,17	XAXISO	XROM 17,37	BCSIZE	XROM 18, 18
LTYPEO	XROM 17,18	YAXIS	XROM 17,38	BCX	XROM 18, 19
LXAXIS	XROM 17,19	YAXISO	XROM 17,39	BCXS	XROM 18, 20
LYAXIS	XROM 17,20	PCLBUF	XROM 18,01		

If you use [XEQ] to enter a plotter module function into a program line for example, by setting the HP-41 to Program mode and pressing [XEQ] [ALPHA] MOVE [ALPHA] while the plotter module is not connected, the function is recorded, displayed, and printed as **XEQ**^{T} followed by the function name. Lines of this form slow program execution because the HP-41 searches for a matching Alpha label or function name—first in program memory, then in each module that is currently plugged in.

HP-GL Commands and Series 80 Counterparts

The following listings are provided for users who want to know the HP-GL (Hewlett-Packard Graphics Language) commands used internally in the HP 82184A Plotter Module's functions and/or the plotter module's function counterparts in the Hewlett-Packard Series 80 computers.

Plotter Module Function	HP-GL Functions Used	Series 80 Counterparts	Series 80/Plotter Module Functionality
CLIPUU	IW, OE	CLIP	Different
CSIZE, CSIZEO	OE, SR, SL	CSIZE	Similar
DGTIZE	DP, OD, OE, OS	DIGITIZE	Similar
DRAW	OC, OE, PA, PD, PU	DRAW	Same
FRAME	OC, OE, PA, PD, PU	FRAME	Same
GCLEAR	AF, OE	GCLEAR	Same
IDRAW	OC, OE, PA, PD, PU	IDRAW	Same
IMOVE	OC, OE, PA, PU	IMOVE	Same
IPLOT	OC, OE, PA, PD, PU	IPLOT	Similar
LABEL	CP, LB, IW, OA, OE, PA, PU	LABEL	Similar
LDIR	DI, OE	LDIR	Similar
LIMIT	DF, DI, IP, IW, OC, OE, OP, SP	LIMIT	Similar
LOCATD	DP, IW, OD, OE, OS	LOCATE	Same
LOCATE	IW, OE	LOCATE	Same
LORG	*-None-*	LORG	Same
LTYPE, LTYPEO	LT, OE	LINETYPE	Same
LXAXIS	CP, DI, IW, LB, PA, PD, PU, OA, OE, XT	LAXES	Similar
LYAXIS	CP, DI, IW, LB, PA, PD, PU, OA, OE, YT	LAXES	Similar
MOVE	OC, OE, PA, PU	MOVE	Same
PEN	OE, SP	PEN	Same
PENDN	OE, PD	*-None-*	
PENUP	OE, PU	PENUP	Same
PINIT	DF, DI, IW, OC, OE, OP, SP	PLOTTER IS	Similar
PLOT	OC, OE, PA, PD, PU	PLOT	Similar
PLREGX	OC, OE, PA, PD, PU	*-None-*	
RATIO	*-None-*	RATIO	Same
RPLOT	OC, OE, PA, PD, PU	RPLOT	Similar
SCALE	*-None-*	SCALE	Same
SETGU	IW, OE	SETGU	Same
SETUU	IW, OE	SETUU	Same
TICLEN	OE, EL	*-None-*	
UNCLIP	IW, OE	UNCLIP	Same

Plotter Module Function	HP-GL Functions Used	Series 80 Counterparts	Series 80/Plotter Module Functionality
WHERE	*-None-*	WHERE	Similar
XAXIS	OC, OE, PA, PD, PU	XAXIS	Similar
XAXISO	IW, OC, OE, PA, PD, PU, XT	XAXIS	Similar
YAXIS	OC, OE, PA, PD, PU	YAXIS	Similar
YAXISO	IW, OC, OE, PA, PD, PU, YT	YAXIS	Similar
PCLBUF	*-None-*	*-None-*	
PDIR	*-None-*	PDIR	Similar
PRCL	*-None-*		
BC	OC, OE, PD, PR, PU		
BCA	*-None-*		
BCAA			
BCCKSM			
BCO	OC, OE, PD, PR, PU	*-None-*	
BCP	*-None-*		
BCREGX			
BCSIZE			
BCX			
BCXS			

Plotter Character Code Numbers

CODE NUMBER	CHARACTER	CODE NUMBER	CHARACTER	CODE NUMBER	CHARACTER	CODE NUMBER	CHARACTER
32	SPACE	56	8	80	P	104	h
33	!	57	9	81	Q	105	i
34	"	58	:	82	R	106	j
35	#	59	;	83	S	107	k
36	$	60	<	84	T	108	l
37	%	61	=	85	U	109	m
38	&	62	>	86	V	110	n
39	'	63	?	87	W	111	o
40	(	64	@	88	X	112	p
41	)	65	A	89	Y	113	q
42	*	66	B	90	Z	114	r
43	+	67	C	91	[	115	s
44	,	68	D	92	\	116	t
45	-	69	E	93	]	117	u
46	.	70	F	94	^	118	v
47	/	71	G	95	_	119	w
48	0	72	H	96	`	120	x
49	1	73	I	97	a	121	y
50	2	74	J	98	b	122	z
51	3	75	K	99	c	123	{
52	4	76	L	100	d	124	|
53	5	77	M	101	e	125	}
54	6	78	N	102	f	126	~
55	7	79	O	103	g	127	⊢

Appendix F

Bar Code Specification Charts

Contents

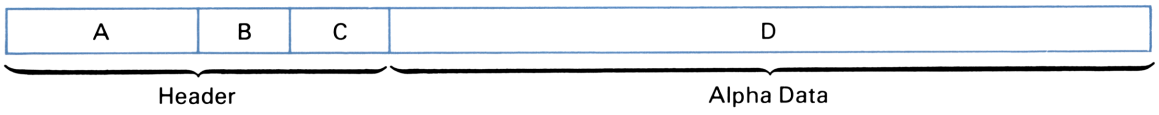
Introduction 213
Specification Charts for HP-41 Bar Code 213
Alpha Data Bar Code 213
Numeric Data and Sequenced Data Bar Code 213
Program Bar Code 215
One-Byte Paper Keyboard Bar Code 216
Direct Execution Bar Code 216
Two-Byte Paper Keyboard Bar Code 216
Programmable Function Derivation 218
Nonprogrammable Bar Code 219
HP-41 Function Table 219
Decimal Values for A through J, a through e, and the Stack 220

Introduction

The charts in this appendix provide you with a guide to constructing headers and determining function codes for the various types of HP-41 bar code described in section 7, Bar Code. This information is provided to aid advanced users whose applications require that they design the byte structure in individual bar code rows.

Specification Charts for HP-41 Bar Code

Alpha Data Bar Code



A	8-bit Checksum (End-around Carry)
B	4-bit Type Indicator (Set to 7 or 8), where 7 = Alpha Replace 8 = Alpha Append
C	4-bit Number of ASCII Characters in String
	16-bit (2-Byte) Header
D	Up to 14 Alpha Characters

Numeric Data and Sequenced Data Bar Code

A string of valid binary-coded digits in part C of numeric data bar code or part D of numeric sequenced data bar code consists of:

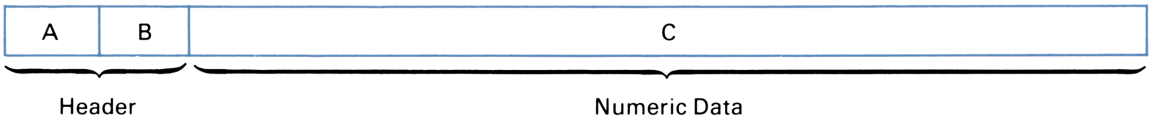
- One numeric digit (0 through 9) for each of one to ten mantissa digits, and, if needed, up to two exponent digits.
- When needed, a null (filler) digit, which is used as the first digit in the data string. (Refer to following table.)

- When needed, data specifying a decimal point, "+" sign, "−" sign, and "E" exponent symbol. (Refer to following table.)

Numeric Data Functions

Digit		Code
0		0000
1		0001
2		0010
3		0011
4		0100
5		0101
6		0110
7		0111
8		1000
9		1001
Null	(hex. "A")	1010
"."	(hex. "B")	1011
"+"	(hex. "C")	1100
"−"	(hex. "D")	1101
"E"	(hex. "E")	1110
Illegal	(hex. "F")	1111

Numeric Data Bar Code.



A 8-bit Checksum (End-around Carry)
B 4-bit Type Indicator (Set to 6)

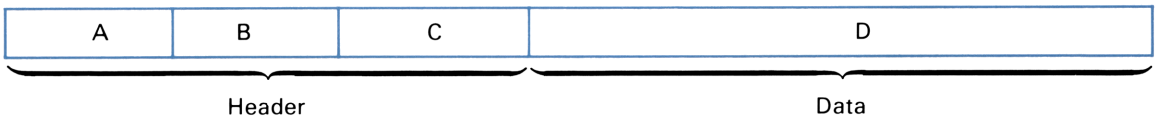
12-bit Header (1½ Bytes)

C Up to 10 mantissa digits and 2 exponent digits, plus (when needed) filler digits, the mantissa sign, the exponent sign, and the decimal point.

The row must be a minimum of three bytes long. If C contains only one digit, insert a pair of filler digits at the beginning of C. If C contains an even number of digits, insert one filler digit at the beginning of C. For example:

Number (Decimal)	Bit Pattern (Hexadecimal)			
	A	B	C	
	Checksum	Type	Filler(s)	Data
2	1 0	6	A A	2
24	8 E	6	A	2 4
2.4	1 8	6	none	2 B 4
2.4E4	F C	6	none	2 B 4 E 4
2.4E−4	6 9	6	A	2 B 4 E D 4

Sequenced Data Bar Code. This type of bar code is used with revision F and any subsequent revisions of the HP 82153A Wand. (Refer to the note on page 129.)



A 8-bit Checksum (End-around Carry)

B 4-bit Type Indicator (Set to 9 or 10), where
9 = Numeric Data
10 = Alpha Replace

C 12-bit Sequence Number

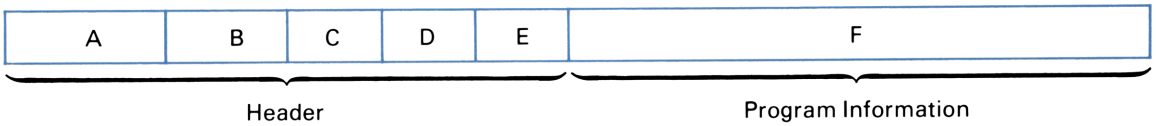
24-bit (3-Byte) Header

D Up to 10 mantissa digits and 2 exponent digits, plus (when needed) a filler digit, the mantissa sign, the exponent sign, and the decimal point; or up to 13 ALPHA characters.

The row must be a minimum of four bytes long. If D contains an odd number of digits, insert one filler digit at the beginning of D. For example:

Number (Decimal)	Bit Pattern (Hexadecimal)				
	A	B	C	D	
	Checksum	Type	Seq. No.	Filler	Data
123.6 Seq. = 1	0 F	9	0 0 1	A	1 2 3 B 6
1.236E23 Seq. = 2	6 3	9	0 0 2	none	1 B 2 3 6 E 2 3
1.236E–23 Seq. = 3	3 2	9	0 0 3	A	1 B 2 3 6 E D 2 3

Program Bar Code



A 8-bit Checksum (End-around Carry)

B 4-bit Type Indicator (Set to 1 or 2), where
1 = Nonprivate
2 = Private

C 4-bit Sequence Number (Mod 16)

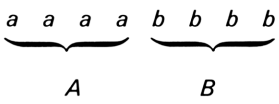
D 4-bit Number of Leading Broken Function Bytes

E 4-bit Number of Trailing Broken Function Bytes

24-bits = 3-Byte Header

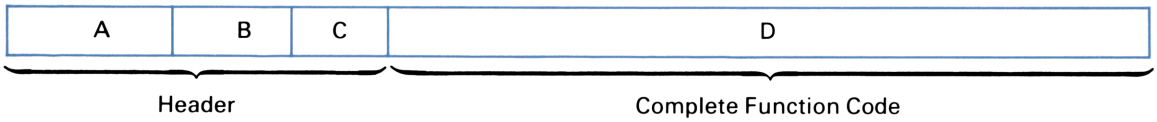
F Up to 13 Bytes of Program Information

One-Byte Paper Keyboard Bar Code



Function	Value of A	Value of B
0	0000	0000
1	1000	0001
⋮	⋮	⋮
9	1001	1001
[·]	0101	1010
[EEX]	1101	1011
[CHS]	0011	1100
[←]	1011	1101

Direct Execution Bar Code

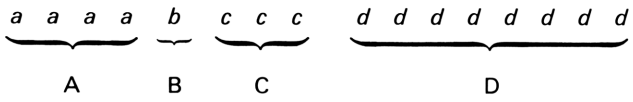


A 8-bit Checksum (End-around Carry)
B 4-bit Type Indicator (Set to Four)
C 4-bit Unused (Set to Zero)

16-bits = 2-Byte Header

D Up to 12 Bytes for Function Code

Two-Byte Paper Keyboard



1. Programmable Functions. These functions are found in the following locations in the HP-41 Function Table on the following page.

   Row 4, column 0, through Row 9, column 15
   Row 10, columns 8 through 14
   Row 12, columns 0, 14, and 15
   Row 13, column 0
   Row 14, column 0

   A 4-bit Checksum (End-around Carry)
   B 0
   C 000
   D 8-bit Function Code
   (Such as ABS, Row 6 Column 1 = 01100001)

2. Alpha Characters (Keys)

   A 4-bit Checksum (End-around Carry)
   B 0
   C 001
   D 8-bit ASCII with Most Significant 4 Bits Doubled
   (For example the ASCII for "A" is 01000001. With most significant nybble doubled, the bar code value for the character A becomes 10000001.)

3. Indirect

   A 10 (1010)
   B 0
   C 2 (010)
   D 128 (10000000)

4. Non-Programmables

   A 4-bit Checksum (End-around Carry)
   B 0
   C 4 (100)

	Function	Value	Function	Value
D	CAT	0	SST	8
	GTOL	1	STAYON	9
	DEL	2	PACK	10
	COPY	3	DELETE	11
	CLP	4	ALPHA	12
	R/S	5	PRGM	13
	SIZE	6	USER	14
	BST	7	ASN	15

5. XROM

   A 4-bit Checksum (End-around Carry)
   B 1
   C *f f f*
   D *g g h h h h h h*
   f f f g g = ROM I.D. Number
   h h h h h h = ROM Function Number

   For Example: WNDSCN has ROM I.D. = 27 and ROM Function = 5. Therefore *fffgg* = 11011, and *hhhhhh* = 000101.

Programmable Function Derivation

Position in HP-41 Function Table		Contents of:				
Row	**Column**	**Function**	**Byte 3**	**Byte 4**	**. . . .**	**Byte *N* + 4**
1	13, 14	[GTO], [XEQ] (Alpha)	(Row No. × 16) + Column No.	1st ASCII Character		*N*th (Last) ASCII Character
4–8	0–15	Refer to HP-41 Function Table.		N/A		N/A
9	0–15			Numeric Value of Argument		N/A
10	0–7	XROM	(Refer to Two-Byte Paper Keyboard Chart.)			N/A
10	8–13	Refer to HP-41 Function Table	(Row No. × 16) + Column No.	Numeric Value of Argument		N/A
10	14	[GTO] / [XEQ] ■ (IND)		Numeric Value of Argument (If GTO, msb = 0; if XEQ, msb = 1)		N/A
12	0–13	[LBL], [END]	(Row No. × 16) + 13 (LBL) or 0 (END)	1st ASCII Character		*N*th (Last) ASCII Character
12	14, 15	Refer to HP-41 Function Table.	(Row No. × 16) + Column No.	Numeric Value of Argument		N/A
13, 14	0–15		(Row No. × 16) + 0	Numeric Value of Argument		N/A

Nonprogrammable Bar Code

Non-programmable Function	Contents of: (N = The Number of ASCII Characters)					
	Byte 3	Byte Four	Byte Five	. . .	Byte *N* + 3	Byte *N* + 4
CATALOG	0	Catalog Number	N/A		N/A	N/A
GTO ·	1	Argument (right justified)			N/A	N/A
GTO · ·	1	15	255		N/A	N/A
DEL	2	Argument (right justified)			N/A	N/A
COPY	3	1st ASCII Character	2nd ASCII Character		*N*th ASCII Character	N/A
CLP *	4	1st ASCII Character	2nd ASCII Character		*N*th ASCII Character	N/A
R/S	5	N/A	N/A		N/A	N/A
SIZE	6	Argument (right justified)			N/A	N/A
BST	7	N/A	N/A		N/A	N/A
SST	8	N/A	N/A		N/A	N/A
ON	9	N/A	N/A		N/A	N/A
PACK	10	N/A	N/A		N/A	N/A
DELETE	11	N/A	N/A		N/A	N/A
ASN	15	$A \times 16 + B$ †	1st ASCII Character		$N - 1$ ASCII Character	*N*th ASCII Character ($N < 7$)

*This three-byte bar code by itself will clear the program where the PC (program counter) is located. If there is an argument it is an Alpha string.

† A = ABS (keycode) DIV 10, B = ABS (keycode) MOD 10. If keycode < 0 then *A* = [ABS (keycode) Div 1048] MOD 16.

HP-41 Function Table

Low Order Bits

High Order Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	NULL	LBL 00	LBL 01	LBL 02	LBL 03	LBL 04	LBL 05	LBL 06	LBL 07	LBL 08	LBL 09	LBL 10	LBL 11	LBL 12	LBL 13	LBL 14	ONE BYTE
1	digit 0	1	2	3	4	5	6	7	8	9	·	EEX	(digit entry) CHS	GTO α	XEQ α		
2	RCL 00	RCL 01	RCL 02	RCL 03	RCL 04	RCL 05	RCL 06	RCL 07	RCL 08	RCL 09	RCL 10	RCL 11	RCL 12	RCL 13	RCL 14	RCL 15	
3	STO 00	STO 01	STO 02	STO 03	STO 04	STO 05	STO 06	STO 07	STO 08	STO 09	STO 10	STO 11	STO 12	STO 13	STO 14	STO 15	
4	+	−	*	/	X<Y?	X>Y?	X≤Y?	Σ+	Σ−	HMS+	HMS−	MOD	%	%CH	P−R	R−P	
5	LN	X^2	SQRT	Y^X	CHS	e^X	LOG	10^X	e^X-1	SIN	COS	TAN	ASIN	ACOS	ATAN	DEC	
6	1/X	ABS	FACT	X=0?	X>0?	LN(1+X)	X<0?	X=0?	INT	FRAC	D−R	R−D	HMS	HR	RND	OCT	
7	CL	X<>Y	PI	CLST	R↓	RDN	LASTX	CLX	X=Y?	X≠Y?	SIGN	X≤0?	MEAN	SDEV	AVIEW	CLD	
8	DEG	RAD	GRAD	ENTER↑	STOP	RTN	BEEP	CLA	ASHF	PSE	CLRG	AOFF	AON	OFF	PROMPT	ADV	
9	RCL *nn*	STO *nn*	ST+ *nn*	ST− *nn*	ST* *nn*	ST/ *nn*	ISG *nn*	DSE *nn*	VIEW *nn*	ΣREG *nn*	ASTO *nn*	ARCL *nn*	FIX *n*	SCI *n*	ENG *n*	TONE *n*	TWO BYTE
10	XROM	XROM	XROM	XROM	XROM	XROM	XROM	XROM	SF *nn*	CF *nn*	F?C *nn*	FC?C *nn*	FS? *nn*	FC? *nn*	GTO/XEQ IND		
11		GTO 00	GTO 01	GTO 02	GTO 03	GTO 04	GTO 05	GTO 06	GTO 07	GTO 08	GTO 09	GTO 10	GTO 11	GTO 12	GTO 13	GTO 14	
12						ALPHA LABEL AND END INSTRUCTIONS									X<>*nn*	LBL *nn*	
13								GTO *nn*									THREE BYTE
14								XEQ *nn*									
15		TEXT 1	TEXT 2	TEXT 3	TEXT 4	TEXT 5	TEXT 6	TEXT 7	TEXT 8	TEXT 9	TEXT 10	TEXT 11	TEXT 12	TEXT 13	TEXT 14	TEXT 15	UP TO 16 BYTE

Decimal Values for A Through J, a Through e, and the Stack

A	B	C	D	E	F	G	H	I	J
102	103	104	105	106	107	108	109	110	111

a	b	c	d	e	X	Y	Z	T	L
123	124	125	126	127	115	114	113	112	116

Function Index

General Plotting Functions:

CLIPUU	Specifies plot bounds in user units.	**Page 74**
CSIZE	Sets character space height.	**Page 97**
CSIZEO	Sets character space height, aspect ratio, and slant.	**Page 97**
DGTIZE	Identifies coordinates of current pen position.	**Page 104**
DRAW	Draws line to point (*x*,*y*).	**Page 78**
FRAME	Frames active plotting area.	**Page 78**
GCLEAR	Advances page on plotters that have a page feed mechanism.	**Page 75**
IDRAW	Draws line to a point *x* and *y* units from current point.	**Page 79**
IMOVE	Moves pen to a point *x* and *y* units from current point.	**Page 79**
LABEL	Prints contents of the ALPHA register.	**Page 92**
LDIR	Sets angle of rotation for printing labels.	**Page 95**
LIMIT	Sets graphic limits in millimeters.	**Page 69**
LOCATD	Sets plot bounds by digitizing two opposite corners.	**Page 106**
LOCATE	Sets plot bounds in graphic units.	**Page 73**
LORG	Sets label origin position.	**Page 94**
LTYPE	Selects line type.	**Page 89**
LTYPEO	Selects line type and length of repeat pattern.	**Page 90**
LXAXIS	Draws and labels *x*-axis.	**Page 102**
LYAXIS	Draws and labels *y*-axis.	**Page 103**
MOVE	Moves pen to point (*x*,*y*).	**Page 78**
PCLBUF	Clears I/O buffer.	**Page 69**
PDIR	Rotates axes for incremental and relative plotting.	**Page 86**
PEN	Selects pen.	**Page 88**
PENDN	Lowers pen.	**Page 88**
PENUP	Lifts pen.	**Page 88**
PINIT	Creates or initializes I/O buffer.	**Page 68**
PRCL	Recalls an I/O buffer register.	**Page 149**
RATIO	Calculates *x*-to-*y* ratio of current graphic limits.	**Page 71**
SCALE	Sets user scale.	**Page 72**
SETGU	Switches module to GU mode.	**Page 68**
SETUU	Switches module to UU mode.	**Page 68**
TICLEN	Sets tic lengths.	**Page 100**
UNCLIP	Resets plot bounds to graphic limits.	**Page 74**
WHERE	Enters coordinates of last point and current pen status.	**Page 106**
XAXIS	Draws *x*-axis.	**Page 99**
XAXISO	Draws *x*-axis with tics.	**Page 100**
YAXIS	Draws *y*-axis.	**Page 99**
YAXISO	Draws *y*-axis with tics.	**Page 100**

Plot-Option Functions:

IPLOT	Moves or draws to a point *x* and *y* units from current point.	**Page 81**
PLOT	Moves or draws to point (*x*,*y*).	**Page 81**
PLREGX	Moves or draws to series of coordinate points stored in data registers.	**Page 83**
RPLOT	Moves or draws to a point (*x*,*y*) relative to an assumed origin.	**Page 82**

Utility Plotting Program:

NEWPLOT	Initializes module for generating a plot.	**Page 22**
PLANOT	Annotates plot according to NEWPLOT and REPLOT parameters.	**Page 61**
PLINIT	Initializes module for plotting from NEWPLOT and REPLOT parameters.	**Page 38**
PLTUXY	Generates a function or data plot according to NEWPLOT and REPLOT parameters.	**Page 39**
REPLOT	Prompts for plot generation or parameter editing.	**Page 25**
X?	Prompts for next *x*-coordinate.	**Page 41**
Y?	Prompts for next *y*-coordinate.	**Page 41**

Bar Code Functions:

BC	Plots a row of HP-41 bar code.	**Page 127**
BCA	Creates bit pattern for Alpha-Replace bar code.	**Page 130**
BCAA	Creates bit pattern for Alpha-Append bar code.	**Page 131**
BCCKSM	Computes checksum of bit pattern in ALPHA register.	**Page 137**
BCO	**Plotter Option:** Plots bar code row having user-specified leading and trailing bars.	**Page 143**
	Printer Option: Prints a row of HP-41 bar code on HP 82162A Thermal Printer.	**Page 134**
BCP	Generates bit pattern for program row.	**Page 131**
BCREGX	Generates bit pattern from data in a series of storage registers.	**Page 136**
BCSIZE	Calibrates module to pen width and sets HP or non-HP bar code type.	**Page 111**
BCX	Creates bit pattern for nonsequenced bar code.	**Page 128**
BCXS	Creates a bit pattern for sequenced bar code.	**Page 129**

Subject Index

Page numbers in **bold** type indicate primary references; page numbers in standard type indicate secondary references.

A

AABC program, **124**
ABC program, **123-124**
Absolute plotting units, **109**, 148
Accessing a buffer, 49
Acquiring coordinates, **40**
Advancing a page, **75**
Alpha-Append bar code, **124**, **131**
Alpha bar code, **121-122**
Alpha bit pattern, **127**
Alpha data, **26**
Alpha data bar code, **121-122**, **124**, **213**
Alpha keyboard, 23, 25, 122, 124
Alpha labels, 92
Alpha label searching, **211**
ALPHA register, 127, 131, 135, 137, 142
Alpha-Replace bar code, **128**, **130**
Alpha string in coordinate pair, **84**
Alternative directional bits, **144**
Alternating bars and spaces, **146**
Alternative bar code, **109**, 143
Angular mode, **86**, 98
Angular rotation of axes, **151**
Anisotropic scaling, **72**
ANNOT parameter, **28**, 34, 56
  changing, 28, **35**, 59
  default, **28**, 43
ANNOT sign convention, **43**
Annotated program listings
  BAR subroutine, **167-168**
  KWH, **161-162**
  Pen width calibration, **169**
  PLOTBC program, **169-170**
  PLPLOT/PLPLOTP subroutine, **173**
  Printing on HP 82162A Thermal Printer, **172-173**
  RAIN, **160-161**
  TITLES subroutine, **168-169**
  Utility plotting program, **162-167**
  XBC program, **171-172**
Annotation parameter, see *ANNOT parameter*
APU'S, 109, 113, 121, 146, 148
ASCII character set, **212**, **220**
Aspect ratio, **97**
Assumed origin, **82**
Automatic labeling, **102-103**
Automatic plotting, **32**
Automatic plotting control, **25**
Automatic scaling, 29, **55**
Autoscale, 29, 31, 49
  example, **56**
  how to use, **56**
  option, **55**
  parameter 30
  simultaneous buffer filling, **60**
  specifying, **55**
Axis
  limits, 24, 30
  drawing, **99-101**
  horizontal, **99**
  label formats, **102**
  plotting without tics, **100**
  rotation, **86-88**
  vertical, **99**

B

Bar and space proportions, **109-111**
Bar chart, **16**
Bar code,
  Alpha 121-122, **124**
  Alpha-Append, **124**, **131**
  Alpha data specifications, **213**
  Alpha-Replace, **128**, **130**
  alternating bars and spaces, **146**
  alternative, 136, 143
  alternative directional bits, **144**
  BAR, **195-196**
  bar and space proportions, **109-111**
  beginning row **120**
  bit pattern, **129**, 130, 131, 135, **136**, 137, 138, 143, 144, 145
  bit pattern, generating **136**
  byte length, **126**
  byte parameters, 120, **126**
  changing rows, **119**
  checksum, **126**
  checksum computing, **137**
  checksum generating, **136**
  computation time, **126**
  data, **121-122**
  default format, **109**
  dimensions, **119**
  direct execution, **136-138**, **142**
  direct execution specifications, **216**
  directional bars, **144**
  directional bits, **143**
  extra bits, **144**
  filler bits, **129**
  final row, 132
  first byte, **136**
  format, default, **109**
  functional bits, **138**
  general parameters, **121**
  generation, halting, **120**
  HP-41 directional bars, **127**, 144
  HP-41 paper keyboard, **128**
  KWH, **196-197**
  label direction, **121**
  labeling, **114**
  labels, **119**, 124
  LABL, **198**
  last byte, **136**
  LDR, **198**
  leading bits, 145
  left byte, 140
  leftmost byte, **136**
  LINE, **199**
  maximum row length, **128**
  non HP-41, 136, 144

nonprogrammable, **219**
nonsequential, 125, **128**, 133
null bytes, 130, 131
number of bars, **128**
number of bytes, 129, 134, **137**
number of bytes per row, **126**
number of characters, **130**
numeric, **121-122**
numeric data, **128**, 129
numeric data labels, **121**
numeric data specifications, **213-214**
paper keyboard, **136-140**
paper keyboard specifications, **216-217**
parameters, **69**
PLOTBC, **199-201**
plot direction, **121**
plotter option, **134**
PLPLOT/PLPLOTP, **202**
PRBC, **202-203**
printer option, **134**
private, 120, **126**
program, **131**
program, byte parameter, **125**
program, label, 135
program, parameters, **134-135**
program, specifications, **215**
proportional, **146**
PWIDTH, **203**
replotting rows, **133**
RAIN, **203-205**
right byte, **140**
rightmost byte, **136**
row length, **120**
row number, **126**
row parameter, **126**
rows per page, **119**
scanning, **130**
sequenced data, **122-127**, **129**
sequenced data specifications, **213-215**
SET, **205**
single program rows, **124-127**
single rows, **133**
size, **111**
size parameter, **109**, **112**, **145**
size parameter calibration, **114**
size parameter default values, **112-113**
special HP-41, **136**
standard HP-41, **127**
storage registers used, **121**, **136**,144
systems, other, **108**
terminology, **109**
TERN, **205**
TITLES, **206-207**
trailing bits, **145**
type 0, 144, 146
type 1, **145**
type 2, **146**, 147
type 3, **146**, 147
types, **108**
using subroutines, **120-121**
width ratio, **145**
XBC, **207-209**
XROM functions, **141**
BC, 113, 114, **127-128**, 132, 141, 143
BC and null bytes, **142**
BCA, 127, 130
BCAA, 127, **131**
BCCKSM, 136, **137**, 142
BCO, plotter option, **143-147**
BCO, printer option, 133, **134**
BCP, 127, **131**, 132, 133, 135
BCREGX, **136**
BCSIZE, **110-112**, 146
example, **113**
parameters, **112**, **146**
parameters default, 151
BCX, 113, 114, 127, **128**
BCXS, 127, **129**
BEEP, **140**
Beginning data storage register, **30**
Bit, **109**
Bit pattern, **109**, 135, **137-146**
Blue keys, **11**
Boundaries, **14**
Buffer accessing, **49**
Buffer filling, 29, 30, 56, **59**
plotting, **60**
simultaneous autoscaling, **60**
NAME parameter, **59**
PLTPRM parameter, **59**
pointer, **59-60**
pointer, how PLTUXY uses, **60**
x,y, **60**
y only, **60**
Buffer, I/O, see *I/O Buffer*
Buffer pointer, **60**
NAME, **49**
type 0, **48**
Buffer specifiers,
NAME parameter, **60**
PLTPRM parameter, **60**
Buffer types, **30**, 49
type 0, **48**, 49
type 1, 50, **52-54**, 55
type 2, 50, 55
Buffer type specifiers, **60**
Buffer, *x,y-*, **60**
Buffers, plotting from, **48**
Building a buffer without plotting, **39**
Byte, **109**

C

Calibrating the plotter module, **111**
Calibrating the bar code size parameter, **114**
Changing graphic limits, 64, **69**, 70
Changing limit parameters, **105**
Changing parameters, **37**
Changing plot bounds, **72-73**
Changing tic length, **100**
Characters, 30
aspect ratio, **97**
changing shape, **96-99**
character set, **97**
choice, 29
codes, **212**, **220**
size and shape, **96-99**
slant, **97**
slant angle range, **98**
space height, **151**
special plot, **31**
width **97**
CHART HOLD/CHART LOAD Lever, **11**
Checksum, **109**
bar code, **126**
eight bit summation, **138**, **142**
four bit mirror, **138-139**
four bit summation, **138**, **139-142**
running, **109**, 126
Clearing the I/O buffer, **68-69**
Clip area, hard, **64**
Clip area, soft, **66**
CLIPUU, 73, **74**
Computation time bar code, **126**
Configuring memory, 11, 13, 15, 16
Controlling line type, **88-91**
Controlling pen conditions, **88-91**

Coordinates, *x*,*y*, 22
acquiring, **40**
prompting for, **41**
terminating keyboard entry, **41**
Corner points, **106-107**
COS plot, **34-35**
Cosine of PDIR angle, **149-150**
Creating the I/O buffer, **68**
Creating new graphic limits, **73**
CSIZE, **97**, 121
CSIZEO, **97-98**, 121

D

Data base
initialization, **42**
parameters, **26**
plotting, 59, 61
registers, 25, 26
table, 26
Data, erroneous, **31**
Dealer information, **159**
Decimal value of bits, **144**
Default ANNOT parameter, 28, 43
Default bar code format, 109
Default bar code size parameters, **112-113**
Default BCSIZE parameters, **151**
Default conditions, HP 7470A plotter, **150**
Default conditions, PINIT, **69**
Default graphic limits, **38**, 64, 68, 121
Default hard clip, **64**
Default parameters, 13, 14, 23
Default parameters, PLTPRM, **30**
Default plotting conditions, **150-151**
Default plotting parameters, 124
Default soft clip, **66**
Default tic length value, **101**
Defining new user scale, **72**
Defining plot bounds, **106**
Defining user units, **66**
Determining parameter elements, **39**
Determining pen position, **106**
DGTIZE, **104-106**
DGTIZE, in a program, **105**
Diamond example, **82-85**
Differing *x* and *y* unit scales, **72**
DIG program, **105-106**
Digitizing, **104-107**
Direct execution bar code, **136-138**, **142**
Direct execution bar code specifications, **216**
Directional bars, **144**
Directional bits, 138, 143, 145
Directional bits, alternative, **144**
Display, 12
Display setting, **125**, 127
DRAW, **78**, 81
Drawing
a box at graphic limits, **78**
a box at plot bounds, **78**
a line, **78**
an axis, **99-101**
by increments, **79**
lines in graphic limits, **78**
lines in plot bounds, **78**

E

Editing,
ANNOT parameter, 28, 59
buffer registers, **25**
plotting data base, **25**
programs, **127**
register procedure, 26, 27
Eight-bit binary code, 136
Eight-bit summation checksum, **138**, **142**
Enter key, **104-107**
Equal *x* increment, **58**
Erroneous data, **31**
Erroneous values, **47**
Error conditions, **151**
Error messages, **154-155**
Examples, Autoscale, **56**
Examples, PRCL, **150**
Examples, PLREGX, **85**
Examples, beginning, **12**
Examples, DIAMOND, **82-85**
Examples, setting up for, 127

F

Filler bits, bar code, **129**
Filling buffers, 56, 59
Final register not specified, **59**
Finding length of long side of graphic limits, **71**
First byte, **136**, 144
FIX, 12, 15, 29, 59, 150
Flag 9, **120**
Flag 17, **93**, 120
Flag 22, **120**
Flag 23, **120**
Flag 27, **115**, **120**
Flag 29, **117**, **120**, 121
Flag 32, **154**
Flowcharts,
BAR, **181**
NEWPLOT, **174**
PLANOT, **179-180**
PLINIT, **175**
PLOTBC, **186-189**
PLPLOT/PLPLOTP, **194**
PLTUXY, **176-178**
PWIDTH, **185**
REPLOT, **175**
TITLES, **182-184**
XBC, **190-193**
Formats, axis label, 102
Four-bit mirror checksum, **138-139**
Four-bit summation checksum, **138**, **139-142**
FRAME, 70, 71, 73, 74, **78**
Framing the plotting area, **78**
Function
code, **142**
descriptions, 12
index, **221-222**
name searching, **211**
plot, sine, **33-34**
plot, cosine, **34-35**
type indicator, **142**
Functions that set pen status to down, **81**

G

GCLEAR, **75**
General plotting options, **39**
Global label, **29**
Graph, **15**
Graphic
area dimensions, **65**
area ratio, **65**
area *x*-dimension, **65**
area *y*-dimension, **65**
scale, **64**
units, **65**
Graphic limits, 64, 72, 92, 102, 103, 115, 128
bar code, **119**
changing, 64, **69**, 70

default, 38, 64, 68, 121
drawing a box at, **78**
drawing lines in, **78**
new, **73**
revising, **64**
specifying, **69**
Grids, plotting, **101-102**
Grids, partial, **102**
GU mode, **67**, 78, 81, 99, 106
GU scale factors, **150**
GU's, **65**, 73, 83, 97, 99, 106, 121

I

IDRAW, **79**, 80, 81
IMOVE, **79**, 80
Increasing plotting skills, **61**
Incremental move, **79**
Increments, tic, **28**
Individual plotting control, **25**
Initialization, **58**
Initializing, **58**, 64
HP-41, **71**, **76**, **86-87**
data base, 42
I/O buffer, **68**, 73
plotter, **58**
plotter module, **10**
Integer only tic labels, **53**
Internal pen status, **85**
International service information, **158**
I/O Buffer, **10**, 38, 110, 112, 133, 134, **148-149**
configuring memory, 11, 13, 15, 16
permanence, **11**
pointer, **24**
returning registers to memory, **18**
clearing, **68-69**
contents, **148-149**
creating, **68**
initializing, **68**
reinitializing, **73**
IPLOT, **81**, 82

H

Hard clip area, **64**
Hard clip limits default, **64**
Hewlett-Packard Graphics Language, 7, 210, **211-212**
Hewlett-Packard Interface Loop, 7, **11**
Horizontal axis, **99**
Horizontal labels, **103**
HP 7470A Plotter, **70**
HP 7470A Plotter default conditions, **150**
HP 82106A Memory Module, 10, 11
HP 82153A Optical Wand, 7, 108, 115, 137
HP 82153A Optical Wand revisions E and F, **129**
HP 82160A HP-IL Module, 7, 133, 155
HP 82162A Thermal Printer, 108, 127, 133, 137, 138, 140, 144, 145
HP 82170A Quad Memory Module, **11**
HP 82175A Black Print Thermal Paper, **133**
HP-41 bar code, **108**
HP-41 character set, **94**
HP-41 function table, **219**
HP-GL, 7, 210, **211-212**
HP-IL, 7, **11**

K

Key assignments, 11, 21
Key assignments, PLOTBC program, **115**, **119**
Keyboard entered coordinates, **44**
Keyboard overlay, 21, 26
Keys, top row **21**

L

LABEL, **92-94**
Labels,
Alpha, **92**
automatic, **102-103**
bar code, **114**
direction, **151**
direction, bar code, **121**
global, **29**
horizontal, **103**
location, **94**
number of characters, **93-94**
numeric data bar code, **121**
orientation, **94**
origin, **95**, 150
plot, **28**
position, 92, **94**
printing, **92-94**
spacing, **93**
tic, **28-29**
vertical, **103**
XBC program, **121**
LABL program, **98-99**
Last byte, **136**, 144
Last point in bounds, **150**
LDIR, **102**
LDR program, **95-96**
Leading and trailing bits, **144-145**
Leftmost byte, **136**, 140
Length of shorter side of graphic limits, **71**
Lifting pen between points, **84**
LIMIT, **69-70**, 71, 72, 106
parameters, **69**
parameters, changing, **105**
setting default conditions, **150**
setting label origin, **95**
Limited one year warranty, **156**
Limits,
default graphic, **64**
graphic, **64**
physical, 69, 70
Line graph, **15**
LINE program, **89**
Line type, 29, 30, 78, 80, **89**
Line type 1, 61, 150
Line type repeat patterns, **90**
LOCATD, **106-107**
LOCATE, **73-74**
Long dimension, **65**
Long side of graphic limits, **71**
LORG, **94-95**
position, **102**
value, **149-150**
LSTROW, **119**
LTYPE, **89**
LTYPEO, **90**
LXAXIS, **102-103**
LYAXIS, **103**

M

Major tic labels, default, **51**
Major tics, default, **51**
Major *x*-axis tic increments, **28-29**
Major *y*-axis tic increments, **28-29**
Manual plotting, **38**
Maximum slant angle, **98**
Mechanical limits, **70**
Memory requirements, 10, **11**, 42, 45, 48, 50, 53, 59, 68, 69, 148
Minor tic increments, **28-29**
Minor tics, **28**
Minor *x*-axis tic increments, **28-29**

Minor *y*-axis tic increments, **28-29**
Mode, GU, **67**
Mode, plotting, **67**
Mode, UU, **67**
MOVE, 12, **78**
Moving the pen, **78-79**, **106**

N

NAME buffer pointer, 49
NAME parameter, 24, 27, 28, 30, 32, **39-40**, 56, 59
  buffer filling, **59**
  buffer specifiers, **60**
  using Y?, **42**
Narrow bars, **145-146**
Narrow spaces, **146**
Negative parameter value, 43
NEWPLOT 14, 15, 21, **22**, 27, **38**, 39, 41, 46, 56, 154
  example, **31-34**
  flowchart, **20**
  plotting data base, **22**
  prompts, **23**
Non HP-41 bar code, 136, 137, 144
Nondefault annotation parameters, 29, 35, 44, 46
Nonprogrammable bar code, **219**
Nonsequential bar code, 125, **128**
Nonsequential program bar code, **133**
Null bytes, **142**
  bar code **130-131**
  characters, **126**
Number of bytes, bar code, **137**
Numeric bar code, **121-122**
Numeric data, 26
Numeric data bar code, **128-129**
Numeric data bar code specifications, **213-214**
Numeric data functions table, **214**
Numeric data labels, bar code, **121**
Numeric XINC parameter, **44**
NXTROW, **119**

O

OCTA subroutine, **36**
Origin, **81**
  assumed, **82**
  point, **69**
Original axes, **87**

P

P1 and P2 coordinates, 150
Packing memory, **120**, **125**, 126, 127, 131
Page feeding, **75**
Paper keyboard bar code, **136-140**
Paper keyboard bar code specifications, **216-217**
Parameter elements, **28**
Parameter values, negative, **43**
Parameters, **34**
  limit, **69**
  XINC, **30**
  ANNOT, 28, 56
  ANNOT, changing, **35**
  ANNOT, editing, **28**, 59
  autoscale, **30**
  bar code, **69**
  bar code byte, **126**
  bar code row, **126**
  bar code size, **109**, **112**, 145
  bar code size, calibration, **114**
  bar code size, default values, **112**
  BCSIZE, **112**
  BCSIZE, default, **151**
  changing, **37**
  data base, **26**
  default, 13, 14, 23
  default ANNOT, 28, 43
  default bar code size, **113**
  default plotting, **124**
  default PLTPRM, **30**
  determining elements, **39**
  editing, 24, **25-27**
  editing ANNOT, **28**
  F_x, 28, 29
  F_y, 28, 29
  general bar code, **121**
  limit changing, **105**
  NAME, 24, 28, 30, 32, **39-40**, 56, 59,
  NAME buffer filling, **59**
  NAME buffer specifiers, **60**
  nondefault annotation, 29, 35, 44, 46
  plotting, 22, **69**
  PLTPRM, 29-31, 34, **39-40**, 56, 60
  PLTPRM, buffer filling, **59**
  PLTPRM, buffer specifiers, **60**
  PLTPRM, resetting, **56**
  program bar code, **134-135**
  program bar code byte, **125**
  program name, **126**
  user accessible, **25**
  x and *y*, reversing, **73**
  XAXAT, 27, 34, 56,
  XINC, 24, 27, 33, **39-40**, 55, 56, 59
  XINC, numeric, 44
  XMAX, 24, 27, 30, 55
  XMAX, resetting, **61**
  XMIN, 24, **27**, 30, 55
  XMIN, resetting, **61**
  YAXAT, 27, 34, 56
  YMAX, 24, 27, 30, 55
  YMAX, resetting, **61**
  YMIN, 24, 27, 30, 55
  YMIN, resetting, **61**
Partial grids, **102**
PBC program, **124-127**, 133
PCLBUF, 11, 31, 42, 57, **69**
PDIR, 79, 82, **86-88**
PDIR angle indicator, **149**
PDIR quadrant, **150**
PEN, **88**
Pen, **30**
  pen 1, 150
  choice, **29**
  down **80-81**, 84, 85
  movement restrictions, **64**
  movement, rotating, **82**
  number, setting, **88**
  position, determining, **106**
  position, range, **148**
  stall, **30**
  status, **80-81**, 104, 106
  status, parameter, **105**
  travel outside the plotting area, **80**
  travel, rotating direction, **79**
  up **80-81**, 84, 85, 150
  width **109-110**, 146
PENDN, 81, **88**, 104
PENUP, **88**, 104
Physical limits, 69, 70, 106
PINIT, 10, 11, 21, 38, **68-69**, 71, 72, 73, 112, 121, 127, 150, 151
  default conditions **69**
  setting default conditions, **150**
  setting label origin, **95**
PLANOT, 20-22, 25, 27, 28, 31, 33, 36, 59, **61**, 154
  automatic execution, **61**
  operation, **61**
  routine **61**

when to use, **61**
PLINIT, 20-22, 25, 31, 33, **38**, 56, **58**, 61, 154,
executing before autoscaling, **58**
executing manually, **38**
routine, **38**
PLOT, **81**, 82
Plot bounds, 66, 71, 72, 92, 102, 103, 106
graphic limits, 67, **74**
changing, 67, **72**
drawing a box, **78**
drawing lines, **78**
Plot direction, bar code, **121**
Plot labeling, **28**
Plot option functions, **81-86**
PLOT? prompt, 20, 21, 24, **25**, 26, 61
PLOTBC program, 109, **114-120**, 143
example, **115-119**
flowchart, **118**
key assignments, **115**, **119**
main prompt, **115**, **118-119**
pens used, **115**
Plotter,
keyboard, 64
keys, **73**
mechanical limits of, 70
option bar code, **134**
scaling, **148**
supplies, **159**
Plotter module,
care, **156**
operating temperatures, **156**
removing, **156**
storage temperatures for, **156**
Plotting
a function, **40**
a star, **50-53**
an axis without tics, **100**
and buffer filling, 60
area, 128, 130
area framing, 61, **78**
area, pen travel outside of, **80**
area, physical size of, **38**
automatic, 25, 32
automatic control of, **25**
bar charts, **16**
boundaries, **14**
buffer types, **30**
conditions, default, **150-151**
control, individual, **25**
data base, 22, 59, 61
data base, editing, **25**
data base, setting up, **51**
example, cubic equation, **57**
from buffers, **48**
functions, **12**
grids, **101-102**
increments, **14**
line graphs, **15**
manually, **38**
mode, **67**
mode, functions affecting, **67-68**
options, general, **39**
parameters, 22, 26, **69**
parameters, default, **124**
point, **29**
points, increasing number, **33**
precision, 44-45, 58
registers, according to *x*, **83-86**
shrinking, **72**
skills, increasing, **61**
special shapes, **31**, **60**
status, **150**
stretching, **72**
two or more functions, **21**
PLREGX, 81, **83**, 105, 106
PLREGX, example, **85**
PLTPRM,
beginning register, **29**
default, **58**
numeric form, **29**
parameter 29, 30, 31, 34, 49, 56, 60
parameter buffer filling, **59**
parameter buffer specifiers, **60**
parameter resetting, **56**
parameters, **39**, **40**
subroutines, **31**
PLTUXY, 20-22, **24**, 25, 29-34, 36, **39**, 44, 55, 56, 58-61
PLTUXY, using buffer filling pointer, **60**
Pointer buffer, **60**
Pointer buffer, filling, **59-60**
PRBC program, **136**
PRCL, 148, **149-150**
PRCL, example, **150**
Precision plotting, **44-45**, **58**
Printer option bar code, **134**
Printing bar code, **133**
Printing labels, **92-94**
Private bar code, **126**
Product information, **159**
Program bar code, **131**
Program bar code, nonsequential, **133**
Program bar code parameters, **134-135**
Program bar code specifications, **215**
Program label, bar code, **135**
Program listings, annotated,
BAR subroutine, **167-168**
KWH, **161-162**
Pen width calibration, **169**
PLOTBC program, **169-170**
PLPLOT/PLPLOTP subroutine, **173**
Printing on HP 82162A Thermal Printer, **172-173**
RAIN, **160-161**
TITLES subroutine, **168-169**
Utility plotting program, **162-167**
XBC program, **171-172**
Program name parameter, **126**
Programmable function derivation, **218**
Programs, **123-124**
DIG, **105-106**
PRBC, **136**
XBC, **114**
LABL, **98-99**
LDR, **95-96**
LINE, **89**
PBC, **124-127**, 133
PLOTBC, 109, **114-120**, 143
PWIDTH, **110**, 146
SET, **76**, 103, 106, 115, 125
TERM, **77**, 131, 134
XBC, 109, **122-123**
Proportional bar code, **146**
PWIDTH program, **110**, 146

R

RATIO, **71**, 73
Ratio, aspect, **97**
Ratio, graphic area, **65**
Recalling I/O buffer contents, **149-150**
Register availability, 11, 13
Register editing, 30, 58
Register editing procedure, **25-27**
Reinitializing the I/O buffer, **73**
Reflected plots, 71, 73
Removing the plotter module, **10**, **156**
Repeat patterns, line types, **90**
REPLOT 19, 20, **25-27**, 36, **38**, 39, 41, 56, 58, 59, 61, 154
REPLOT, bypassing, **58**

Resetting PLTPRM parameter, 56
Resetting the plot bounds **74**
Resetting XMAX parameter, **61**
Resetting XMIN parameter, **61**
Resetting YMAX parameter, **61**
Resetting YMIN parameter, **61**
Reversing the direction of user scales, **71**
Reversing *x* and *y* parameters, **73**
Revising the plot bounds, **73**
R_f, **48**
R_{final}, **48**
R_i, **48**
Rightmost bits, **144**
Rightmost byte, **136**, 140
$R_{initial}$, **48**
Rotating axes, **86-88**
Rotating direction of pen travel, **79**
Rotating pen movement, **82**
Rotation of plotting axes, **151**
RPLOT, 81, **82**
RPLOT indicator, **149**
RPLOT, not last function, 150
Running checksum **109**, 126, 127

S

Saving steps, **76**
Scale, **14**
SCALE, **72-73**
Scale factors, **148**
Scaling, automatic, **29**
SCI, **29**
Sequential data bar code, **122-123**
Sequential data bar code specifications, **213-215**
SET program, **76**, 103, 106, 115, 125
SETGU, **68**, 106
Setting pen number, **88**
Setting the user scale, **71-72**
Setting up for examples, **127**
SETUU, **68**
Short dimension, **65**
Shrinking plots, **72**
Sign convention, ANNOT, **43**
Sine function plot, **33-34**
SINE subroutine **13**, 32
Single bar code rows, program, **124-127**
SIZE, 11, 31
Slant, **97**
Slant angle, maximum, **98**
Slant angle range, **98**
Soft clip area, **66**
Spacing tics, **100**
Special plot shapes, **31**
Special shapes, plotting, **60**
Specifiers, buffer type, **60**
Specifying autoscaling, **55**
Specifying graphic limits, **69**
Stack registers, 127, 130, 131, 144
Standard HP-41 bar code, **127**
Storage registers, 129, 145
Storage registers used for bar code, **121**, **136**, 144
Storing digitized coordinates, **105**
Stretching plots, **72**
Switching between UU mode and GU mode, **67**

T

Technical assistance, **159**
TERM program, **77**, 131, 134
Terminating examples, **77**
Terminating keyboard entry of coordinates, **41**
Terminology bar code, **109**
Tic increments, **28**, 61
  major *x*-axis, **28-29**
  major *y*-axis, **28-29**
  minor *x*-axis, **28-29**
  minor *y*-axis, **28-29**
Tic labels, **28-29**
Tic labels, integer only, **53**
Tic length, **100**, 151
Tic length value, default, **101**
Tic spacing, **100**
Tic spacing parameter, **100**
TICLEN, **100**, 102
Tics, integer only, **53**
Tics, plotting an axis without, **100**
Top row keys, PLOTBC program, **115**, **119**
Tracing an illustration, 104
Trailing and leading bits, 144, 145
Two-byte paper keyboard chart, **140**
Two-function plots, **34**
Type 0 bar code, **144-146**
Type 0 buffer, 48, 49, 51,
Type 0 buffer pointer, 48
Type 1 bar code, **145**
Type 1 buffer, 50, 55
Type 2 bar code, **146-147**
Type 2 buffer, 50, 55
Type 3 bar code, **146-147**
Type specifiers, buffer, **60**

U

UNCLIP, **74-75**
Universal product code, **146**
Unrotated axes, **87**
User-accessible parameters, **25**
User annunciator, **33**
User keyboard, **21**
User mode, 11, **21**
User scale, 66, 73
  new, **72**
  setting, **71-72**
  reverse direction, **71**
User units, **66**
User units, defining, **66**
Using Y? for NAME parameter, **42**
Using plot option functions, **80-81**
Utility plotting program 7, 12, **19-61**
Utility plotting program, example, **13-15**
UU mode, **67**, 72, 76, 78, 81, 92, 99, 106, 150
UU scale factors, 150
UU's, **66**, 73, 83, 99

V

Vertical axis, **99**
Vertical labels, 103

W

Wand scanning errors, **135**
Warranty, **156-157**
Warranty and service, who to contact, **157-158**
Warranty, United Kingdom, **157**
WHERE, **106**
Wide bars, **145-146**
Width ratio, bar code, **145**
Window, **66**
WNDDTX, **122**, 130
WNDSCN, 137, 139, 140
WNDTST, 137, 140

X

x and *y* parameters, reversing, **73**
x-axis, **27-29**, 43
x-axis tic increments, 28, 29

x-coordinate, varied interval, **46**
x increments, equal, **58**
x intercept, 99, 100
X? subroutine, **41**, 45
XAXAT parameter, 27, 34, 56
XAXIS, **99**
XAXISO, **100**, 102
XBC program, 109, **114**, **121-122**
XBC program, example, **123**
XBC program, labels, **121**
XINC parameter, 24, 27, 30, 33, **39-40**, 55, 56, 59
XINC parameter, numeric, **44**
XMAX parameter, 27, 30, 55
XMAX parameter, resetting, **61**
X minimum, **27**
XMIN parameter, 27, 30, 55
XMIN parameter, resetting, **61**
XROM function bar code, **141**
XROM functions, **141**, **210**
XROM numbers, **141**, **210**
XSBC program, **122-123**
x,*y* coordinates, 22, 40-41

Y

y-axis, 28, 29, 43
y-axis, coincident with maximum and minimum y values, **56**
y-axis tic increments, **28-29**
y intercept, **99-100**
Y? subroutine, **41**, 44
YAXAT parameter, 27, 34, 56
YAXIS, **99**
YAXISO, **100**, 102
YMAX parameter, 27, 30, 55
YMAX parameter, resetting, **61**
YMIN parameter, 27, 30, 55
YMIN parameter, resetting, **61**
y-axis, **27**
Y minimum, **27**

Corvallis Division
1000 N.E. Circle Blvd., Corvallis, OR 97330, U.S.A.

82184-90001

Printed in U.S.A. 9/82