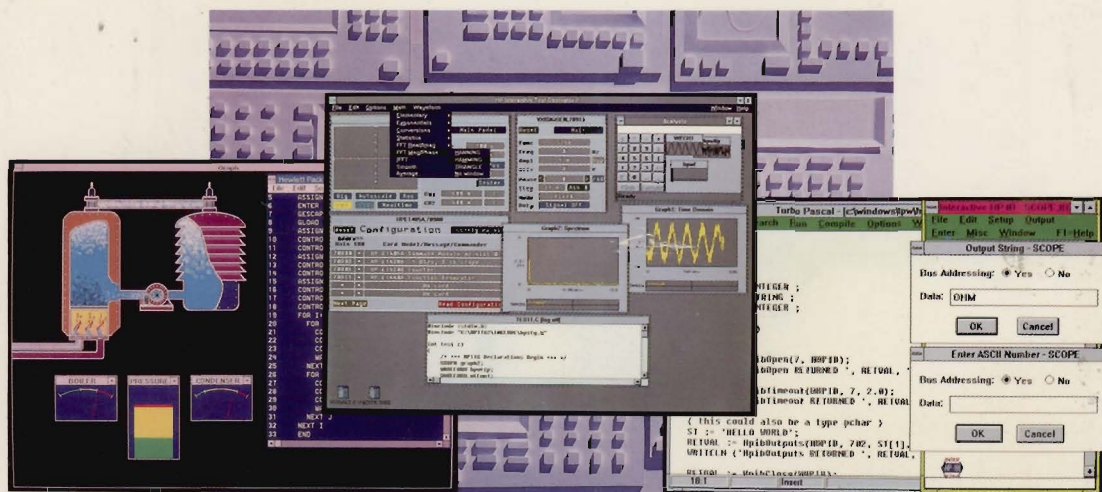


Installing and Using HP Instrument BASIC for Windows

Instrument Tools for
Windows.



Installing and Using HP Instrument BASIC for Windows



**HEWLETT
PACKARD**

**HP Part No. E2200-90000
Printed in USA February 1992**

Edition 1

Notice

The information contained in this document is subject to change without notice.

Hewlett-Packard Company (HP) shall not be liable for any errors contained in this document. HP MAKES NO WARRANTIES OF ANY KIND WITH REGARD TO THIS DOCUMENT, WHETHER EXPRESS OR IMPLIED. HP SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. HP shall not be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory, in connection with the furnishing of this document or the use of the information in this document.

Warranty Information

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Printing History

First Edition - February 1992

Preface

This manual is written for engineers and scientists who have some experience with programming and working with instruments.

About This Manual.

This guide explains HP Instrument BASIC installation, introduces the HP Instrument BASIC environment, and teaches common tasks that you can perform with HP Instrument BASIC.

To install HP Instrument BASIC, follow the instructions in chapter 1.

To get started with HP Instrument BASIC, perform the quick-start exercises in chapter 2.

The rest of this manual is designed to give you easy access to information about your current task, without requiring you to read the entire manual.

© Copyright 1992 Hewlett-Packard Company. All rights reserved.

This document contains proprietary information which is protected by copyright. All rights are reserved. Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws.

Microsoft® and MS-DOS® are U.S. registered trademarks of Microsoft Corporation.

Content of this Manual.

This manual is structured as follows:

Chapter	Summary
1: Installation at a Glance	Instructions for installing the HP Instrument BASIC software.
2: Quick Start	A quick introduction to the HP Instrument BASIC programming environment.
3: A Quick Tour of the HP Instrument BASIC Language	An overview of keywords used to perform common tasks.
4: Writing and Reading Files	An explanation of how to write and read files.
5: Communicating with Instruments	An explanation of how to communicate with HP-IB and RS-232 instruments.
6: Interacting with Users and Displaying Graphs	An introduction to using dialogs and stand-alone widgets.
7: Creating Control Panels	An explanation of tasks that are commonly used when creating an operator control panel.
8: Sharing Information with Other Applications	An explanation of techniques used to share information with other applications through the clipboard or Dynamic Data Exchange (DDE).
A: Language Overview	A short explanation of each HP Instrument BASIC keyword and widget.
Glossary	
Index	

Conventions Used in this Manual

This manual uses the following typographical conventions:

Example	Represents
File	Computer font represents text you will see on the screen, including menu names, features, or text you have to enter.
Refer to <i>Microsoft User's Guide</i>	Italicized words are used for book titles, emphasis, and arguments.
CREATE, 1 <i>"filename"</i>	In this example, the word in computer font represents text you type exactly as shown, and the italicized word represents an argument that you must replace with an actual value.
SUBEND RECOVER	Choices separated with a bar (), indicate that you should choose one of the options.
Return	Keycap font graphically represents a key on the keyboard or a button displayed on the screen.
Press Alt-F	Dash-separated keys represent a combination of keys to be pressed. In this example, hold down the Alt key and press the F key.
Dialog Box	Bold font indicates the first instance of a word defined in the glossary.

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Contents

1. Installation at a Glance	
To Install	1-2
To Verify the Installation	1-3
To Un-Install	1-4
2. Quick Start	
Writing a Five-Line Program	2-2
To Start HP Instrument BASIC	2-2
To Type a Program	2-3
To Run a Program	2-4
To Print a Program	2-5
To Save a Program	2-5
To Quit HP Instrument BASIC	2-6
A Typical Program Structure	2-6
To Work with HP Instrument BASIC	
Windows	2-7
To Display Windows	2-8
To Hide Windows	2-9
To Save the Window Layout	2-9
To Print Window Contents	2-9
To Clear Window Contents	2-10
Basic Skills	2-10
To Run HP Instrument BASIC	
Programs from the Program Manager	2-11
To Open Programs	2-12
To Edit Programs	2-13
To Debug Programs	2-14
To Get Help	2-14
To Use Example Programs	2-15
To Copy Statement Examples	2-15

To Copy Complete Examples	2-16
To Use the Command Line	2-18
To Save Programs	2-20
To Set HP Instrument BASIC Configuration Information	2-20
3. A Quick Tour of the HP Instrument BASIC Language	
To Use Variables	3-2
To Manipulate Strings	3-4
To Specify Numeric Arrays	3-5
To Write Text	3-6
To Format Output	3-8
To Read Text	3-10
To Iterate	3-11
To Use Conditional and Event-Initiated Branching	3-12
To Use Subroutines, Subprograms, and Functions	3-15
To Pass Variables	3-18
To Use Existing Subprograms	3-21
To Handle Errors	3-22
Summary	3-24
Other Information	3-27
4. Writing and Reading Files	
To Create Directories	4-2
To Create Files	4-3
To Assign I/O Paths	4-3
To Specify Text File Format	4-4
To Write or Read Data	4-4
To Format Output	4-6
To Close I/O Paths	4-6
Summary	4-7

5. Communicating with Instruments	
To Assign I/O Paths to Instruments . . .	5-1
To Determine HP-IB Device Selectors .	5-2
To Determine RS-232 Device Selectors	5-3
To Exchange Binary Data	5-3
To Prepare Instruments	5-4
To Initialize HP-IB Instruments . . .	5-4
To Initialize SCPI Instruments . . .	5-5
To Configure RS-232 Interfaces	5-5
To Send Commands	5-7
To Take Readings	5-7
To Initiate Measurements (HP-IB) . .	5-8
To Check Instrument Status (HP-IB) . .	5-10
To Service HP-IB Interrupts	5-12
To Close I/O Paths	5-14
Summary	5-15
 6. Interacting with Users and Displaying Graphs	
To Use Dialog Boxes	6-2
To Change Dialog Attributes	6-3
To Read Dialog Buttons	6-4
To Use Widgets	6-6
To Change Widget Attributes	6-7
To Read Current Widget Attributes . .	6-9
To Respond to Events	6-10
To Change Fonts	6-10
To Prompt for Keyboard Input	6-11
To Display Text	6-14
To Graph Data	6-15
To Create X-Y Plots	6-15
To Create Strip Charts	6-19
To Print Widgets	6-23
Summary	6-24

7. Creating Control Panels	
To Create Panels	7-2
To Create Menus	7-5
To Set Alarms	7-8
To Create Full-Screen Panels	7-10
To Secure Programs	7-11
To Run Programs	7-12
To Create Easy-To-Use Control Panels	7-13
Example	7-14
Summary	7-17
8. Sharing Information with Other Applications	
To Share Information Through the Clipboard	8-1
To Share Data Using Dynamic Data Exchange (DDE)	8-4
To Send HP Instrument BASIC Data to Other Applications	8-4
To Load DDE Subprograms	8-5
To Start Conversations	8-5
To Send Data	8-6
To Send Commands	8-6
To End Conversations	8-6
Example	8-7
To Send Information to HP Instrument BASIC	8-8
Example	8-9
Summary	8-11
A. Language Overview	
Clock and Calendar	A-1
Error Handling	A-1
Graphics	A-2
Display Widgets, Input Widgets, and Dialog Boxes	A-2
Widgets	A-2
Dialog Boxes	A-3
Keywords	A-3

Low-Level Graphics	A-4
Graphics Control	A-4
Plotting	A-5
Axes and Labels	A-5
Input and Output	A-6
Files	A-6
Graphics Input and Display	A-7
Instruments	A-7
Any Interface	A-7
HP-IB Only	A-7
Keyboard, Printer, and Alpha Window	A-8
Sound	A-9
Looping and Branching	A-9
Looping	A-9
Conditional Branching	A-9
Operators Used with Conditional	
Branching	A-10
Unconditional Branching	A-10
Event-Driven Branching (Interrupts) .	A-11
Math	A-13
General Math	A-13
Trigonometric Functions	A-14
Binary Math	A-14
Logical Operators	A-15
Matrix and Array Math	A-15
Program Editing and Development . . .	A-16
Strings	A-17
Subprograms and Functions	A-18
Variables	A-19
Other Keywords	A-19
Program Control	A-20

Glossary

Index

Figures

1-1. The HP Instrument BASIC Group Window	1-2
1-2. The HP Instrument BASIC Screen	1-3
2-1. The HP Instrument BASIC Environment	2-2
2-2. Example Program Output	2-4
2-3. Saving Programs	2-5
2-4. HP Instrument BASIC Program Structure	2-6
2-5. The HP Instrument BASIC Windows	2-8
2-6. To Run Programs when Starting HP Instrument BASIC	2-11
2-7. Opening Programs	2-12
2-8. Help Topics	2-14
2-9. Statement Example	2-15
2-10. Copying Program Examples	2-17
2-11. The Command Line	2-18
2-12. Saving Programs	2-20
3-1. Array Output	3-6
3-2. Printing to the Alpha Window	3-7
3-3. Formatted Output	3-8
3-4. INPUT Keyword Output	3-10
4-1. A New Directory	4-2
4-2. File Data	4-5
5-1. Example of Triggering Measurements	5-9
5-2. Unmasking the Status Byte	5-10
6-1. An INFORMATION Dialog	6-3
6-2. A Modified INFORMATION Dialog	6-4
6-3. A QUESTION Dialog	6-5
6-4. A METER Widget	6-6
6-5. A Modified METER Widget	6-8

6-6. A SLIDER Widget	6-9
6-7. A STRING Dialog	6-12
6-8. A STRING Widget	6-13
6-9. A LABEL Widget	6-14
6-10. An XY GRAPH Widget	6-17
6-11. An X-Y Graph with Two Traces	6-19
6-12. A STRIPCHART Widget	6-21
6-13. A Strip Chart with Two Traces	6-23
7-1. An Example Panel	7-2
7-2. A METER Widget on a Panel	7-4
7-3. A LABEL Widget on a Panel	7-5
7-4. A Pull-Down Menu	7-6
7-5. Multiple Menu Choices	7-8
7-6. A BARS Widget with Alarms	7-10
7-7. An Engine Monitor Example	7-16
8-1. A Modified Word for Windows Document	8-8

Tables

2-1. Command-Line Examples	2-19
3-1. Variable Names	3-2
3-2. Accessing Substrings	3-4
3-3. Accessing Array Elements	3-5
3-4. Formatting Examples	3-9
5-1. Character Format and Parity Settings (Register 4)	5-6



Installation at a Glance

HP Instrument BASIC for Windows is a programming language that allows you to easily communicate with instruments, create engineering graphics, develop control panels, and move information to other applications.

To install HP Instrument BASIC for Windows, you need the following:

- Personal computer using an 80286 microprocessor or higher.
- Microsoft® Windows version 3.0 or later (installed).
- One 5.25 inch high-density (1.2 Mbyte) disk drive.
-or-
One 3.5 inch (1.44 Mbyte) disk drive.
- 3 Mbytes of hard disk space.
- 4 Mbytes of extended memory.
- Microsoft mouse or compatible pointing device.
- For HP-IB instrument communications - an HP 82335 HP-IB card, or National Instruments GPIB-PCII/IIA or AT-GPIB card (installed).

Note



For National Instrument cards, the software version must be 2.0 or later. The files GPIB.DLL and GPIB.INI must be in the Windows directory.

The card and software must be installed and configured following National Instrument's instructions.

To Install

1. Start from the **Windows Program Manager**.
2. Insert **Disk 1** in your floppy disk drive.
3. Pull down the **File** menu from the **Program Manager** window and select **Run**.
4. Type the name of the floppy disk drive, **SETUP**, and press **(Enter)**. For example, **A:SETUP (Enter)**.
5. Answer the questions that the **SETUP** program asks.

The **SETUP** program checks your configuration, asks you where you want to install the software (default is **C:\IBASIC**), and copies the **HP Instrument BASIC** files to that directory. It also checks the cards installed and lists their locations.

SETUP also creates a group window called **HP Instrument BASIC** and creates two group window icons: **HP Instrument BASIC** and **HELP**.

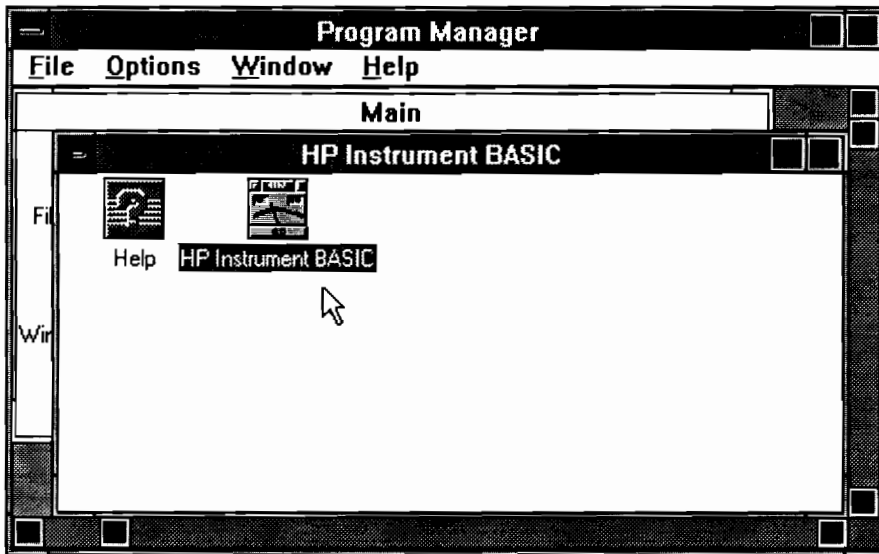


Figure 1-1. The **HP Instrument BASIC** Group Window

To Verify the Installation

Double-click on the **HP Instrument BASIC** icon. The screen should look like Figure 1-2.

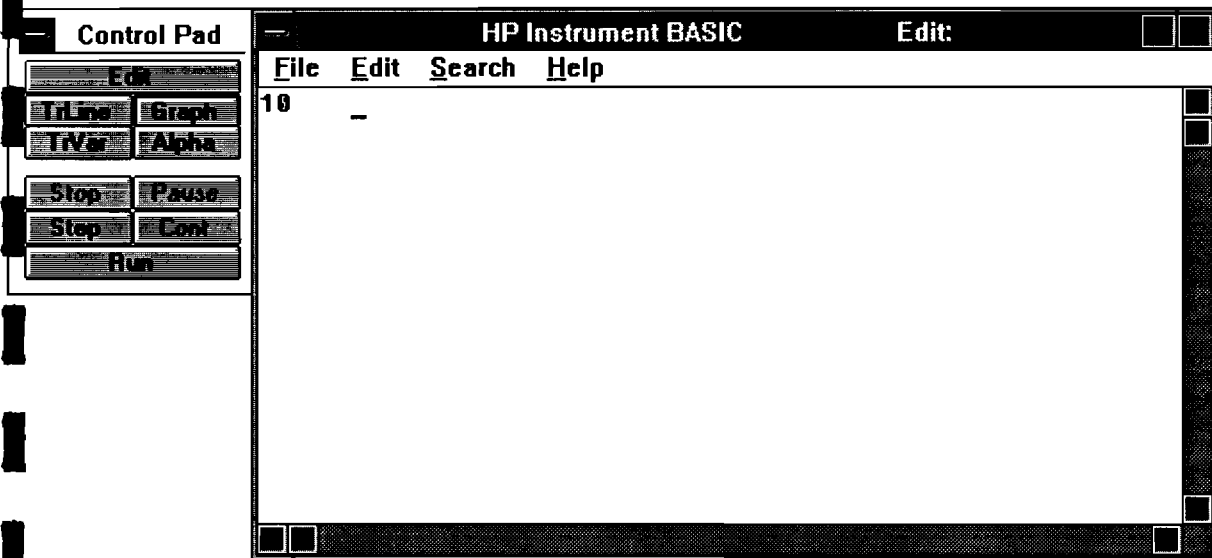


Figure 1-2. The HP Instrument BASIC Screen

For any late-breaking information not included in this manual, refer to the **Latest Information** selection under the **Help** menu.

If you get any error messages when you double-click on the **HP Instrument BASIC** icon, re-install by following the instructions in "To Install".

To Un-Install

Delete all the files in your installation directory (usually IBASIC) and then delete the directory. Also delete the Windows group window (HP Instrument BASIC) and the IBASIC.INI file from the \WINDOWS directory.

Quick Start

This chapter takes you through a simple exercise to help you become familiar with programming with HP Instrument BASIC. It also shows HP Instrument BASIC program structure and explains common program development tasks such as getting help and using HP Instrument BASIC debugging tools.

If you're not familiar with using a mouse or operating in the Windows environment, please refer to chapter 2 of the *Microsoft Windows User's Guide* for information such as:

- Selecting and canceling menus.
- Choosing menu commands.
- Using the control menus.
- Working with dialog boxes.
- Working with windows.

Writing a Five-Line Program

The easiest way to get familiar with HP Instrument BASIC is to write a program. Take about 10 minutes to complete this exercise which introduces you to HP Instrument BASIC programming:

- Typing in a program.
- Running a program.
- Printing a program.
- Saving a program.
- Quitting HP Instrument BASIC.

To Start HP Instrument BASIC

Double-click on the HP Instrument BASIC icon.

You'll see the control pad and the edit window:

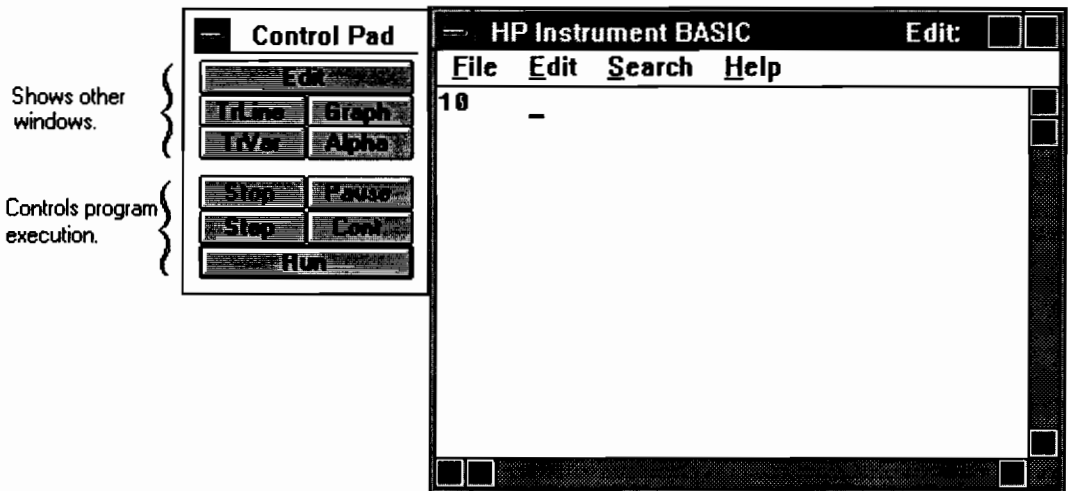


Figure 2-1. The HP Instrument BASIC Environment

To Type a Program

In the edit window, type in the following program; press **Enter** at the end of each line. This program prints the numbers from 1 to 10, their squares, and their cubes. The first line of the program is a comment.

(Don't type the line numbers, HP Instrument BASIC generates them automatically.)

```
10 ! Example Program
20 FOR I=1 TO 10
30 PRINT I,I^2,I^3
40 NEXT I
50 END
```

Some useful editing tips are:

- The default mode is overwrite; to insert a character, press **Insert**.
- To insert a line, press **Alt - I**.
- To delete a line, press **Alt - D**.

Note



- If you get a syntax error, correct the line so it appears exactly as shown. Press **Enter** after you change a line so the change is made and syntax-checked.
- You must enter keywords (such as FOR, PRINT, and NEXT) as all uppercase or all lowercase; HP Instrument BASIC converts them to uppercase.

To Run a Program

The program prints the values in the alpha window; the alpha window is displayed when it is written to. Another way to display the alpha window is to press the **Alpha** button on the control pad (to the left of the edit window).

When your program looks exactly like the program listed previously, press the **Run** button on the control pad.

The output looks like:

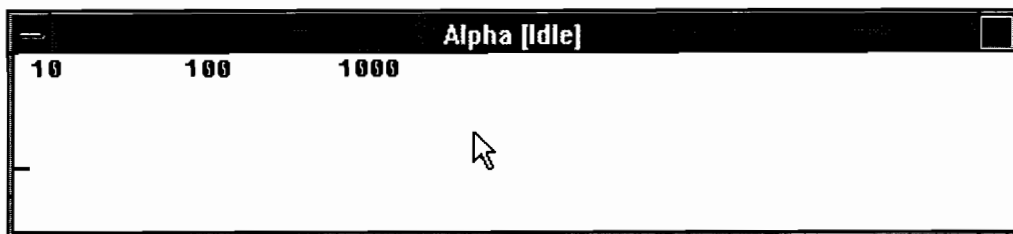


Figure 2-2. Example Program Output

The default size of the alpha window allows you to see the last line of the output. When the alpha window is selected (the title bar is highlighted), you scroll the output area with the up (▲) and down (▼) cursor keys on the keyboard. To select the alpha window, click on the window. To view more of the output, resize the alpha window and run the program again.

Note



If you get an error, press the **Help** button in the error dialog box to get additional information.

Now that you've created and run a program, you may want to print and save it.

To Print a Program

HP Instrument BASIC prints to your default Windows printer. To change the default printer, run the **Printer** program in the **Control Panel** from the **Program Manager** and select the new default printer.

To print, choose **Print** from the HP Instrument BASIC **File** menu in the **Edit** window.

To check on printing status, use the **Windows Print Manager**.

To Save a Program

To save a program for the first time, select **Save As** from the **File** menu in the **Edit** window. Type a file name for the program and press the **OK** button. Figure 2-3 shows a file that is saved to a text file. For a text file, the **.IBW** extension is automatically added to the file name.

For subsequent saves to the same file name, select **Save** from the **File** menu.

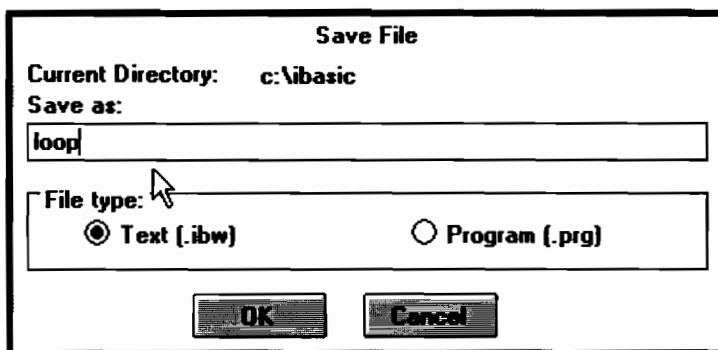


Figure 2-3. Saving Programs

2 To Quit HP Instrument BASIC

When you're ready to leave HP Instrument BASIC, select **Exit** from the **File** menu in the **Edit** window. If you have not saved your program changes, you will be prompted to do so.

A Typical Program Structure

Now that you've created a program, you've seen the general HP Instrument BASIC program structure. Figure 2-4 shows some more characteristics of an HP Instrument BASIC program.

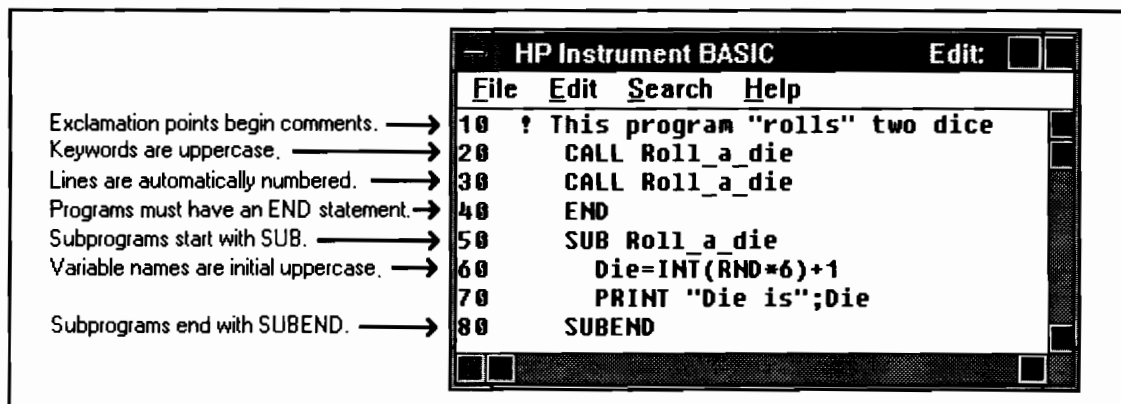


Figure 2-4. HP Instrument BASIC Program Structure

To Work with HP Instrument BASIC Windows

As you work with HP Instrument BASIC, you'll notice different windows that are used for different purposes. For example, you use the edit window while developing your program and the trace lines window while debugging. This section explains how to work with all these windows.

Figure 2-5 shows all the HP Instrument BASIC windows.

HP Instrument BASIC starts with two windows visible: the edit window and the control pad. The edit window is where you type in programs. The control pad controls program execution and allows you to access all other HP Instrument BASIC windows.

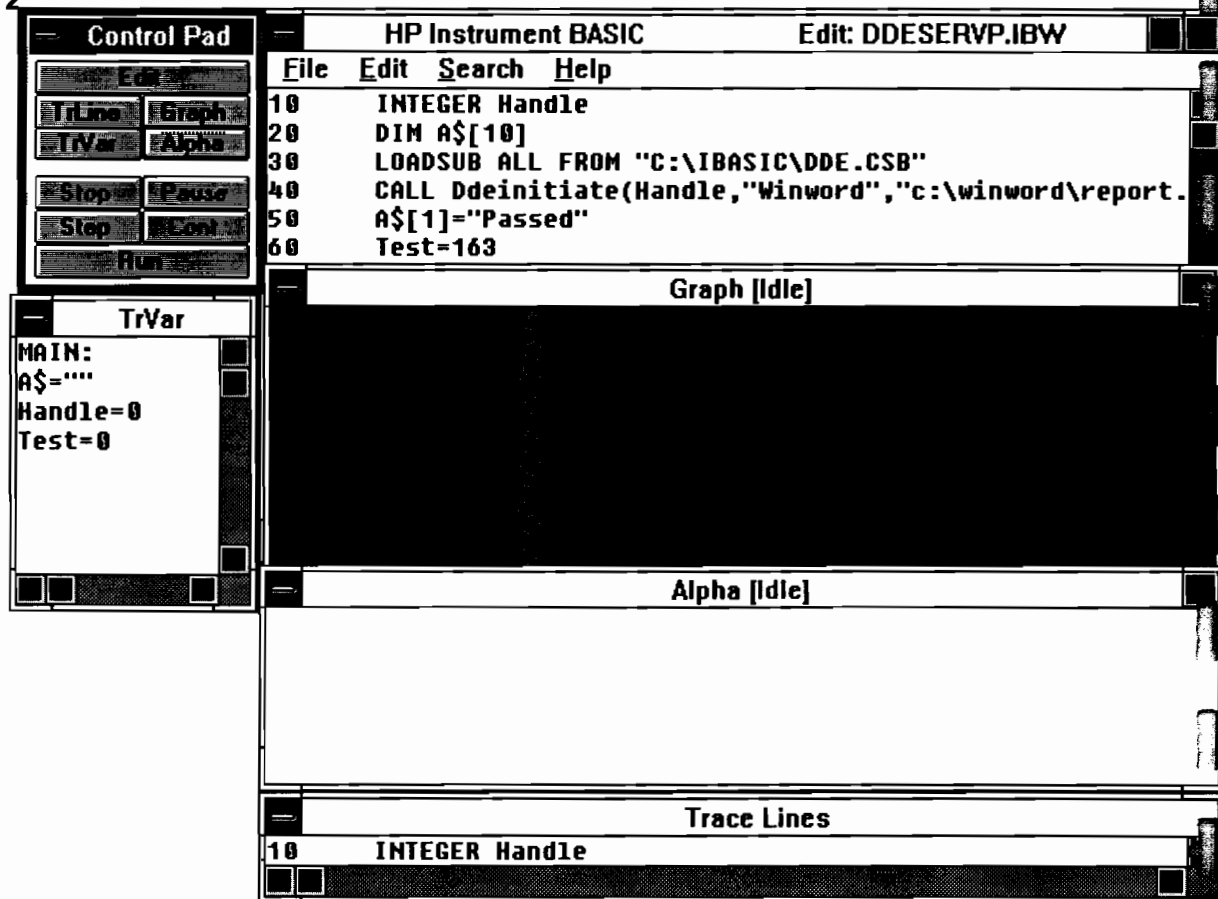


Figure 2-5. The HP Instrument BASIC Windows

To Display Windows

To display an HP Instrument BASIC window or to bring it to the top of a stack of windows, click the corresponding button on the control pad. For example, to view the alpha window, press the **Alpha** button on the control pad.

Note



To display the control pad or bring it to the top of a stack of windows, double-click on the open area in any HP Instrument BASIC window.

2

To Hide Windows

When you don't want to view the contents of a window, select **Hide** from its control menu. The control menu is located at the upper-left corner of each window. The symbol on the control menu looks like a space bar; a control menu can be accessed by pressing **Alt-Space**.

A shortcut to hide windows, except for the edit window, is to double-click on the control menu symbol.

Caution



Double-clicking on the edit window's control menu, will *not* hide the window, it will close the HP Instrument BASIC application.

When you hide a window, its contents are not lost.

To Save the Window Layout

As with any Windows application, you can rearrange and resize the HP Instrument BASIC windows to suit your needs. To save this arrangement, select **Save Layout** from the control menu of the edit, alpha, or graph window. The layout information is stored in the **IBASIC.INI** file under the **\WINDOWS** directory and is read each time you start HP Instrument BASIC.

To Print Window Contents

Whenever you print from HP Instrument BASIC, the output is printed on the device specified by your default Windows printer.

To print the contents of a window, use one of the following steps:

- To print the contents of the edit window, choose **Print** from the **File** menu.
- To print the contents of the alpha or graph windows, choose **Print** from its control menu.
- To log information as it is output to the trace lines window, choose **Output** from its control menu and specify if you want the log sent to a printer and/or a file.

To Clear Window Contents

To clear the contents of the edit window, choose **New** from the **File** menu. You'll be prompted to save any changes.

To clear the contents of another window, choose **Clear** from its control menu.

Basic Skills

When you're creating programs, you perform common tasks no matter what language you use. This section explains the following programming development tasks:

- Running programs from the **Program Manager**.
- Opening programs.
- Using the editor.
- Using debugging tools.
- Getting help.
- Copying example programs to the edit window.
- Using the command line.
- Saving programs.
- Setting configuration information.

To Run HP Instrument BASIC Programs from the Program Manager

2

To open and run programs automatically when starting HP Instrument BASIC from the Program Manager, select Run from the File menu on the Program Manager. Type the path to HP Instrument BASIC and the program name.

For example:

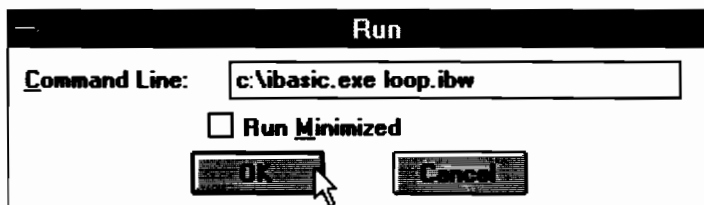


Figure 2-6. To Run Programs when Starting HP Instrument BASIC

Make sure you include the file name extension (such as .IBW or .PRG).

To Open Programs

There are two types of program file formats: text and program. Program-formatted files take more disk space and load faster, but are not readable in any editor except HP Instrument BASIC. Text-formatted files are human-readable.

Figure 2-7 shows the Open dialog box where you see all files of the file type selected.

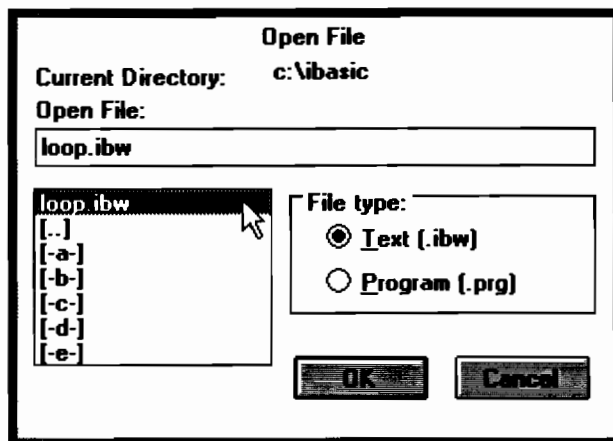


Figure 2-7. Opening Programs

To Edit Programs

The HP Instrument BASIC editor helps you enter programs by performing the following tasks:

- Automatically numbering the lines.
- Changing HP Instrument BASIC keywords to upper case.

Note



You must enter keywords in all upper or all lower case. If you enter a keyword with mixed case, HP Instrument BASIC interprets it as a variable. For example, `print` and `PRINT` are interpreted as keywords; `Print` is interpreted as a variable name.

-
- Changing variable names to begin with a capital letter. You can type variables using mixed case; HP Instrument BASIC capitalizes them consistently.
 - Automatically checking the syntax of the line when you press **Enter**.

The following editing operations are located under the **Edit** menu:

- **Cut, Copy, and Paste** - Allow you to move and copy selected lines. To select lines, position the cursor on a line and drag up or down.
- **Insert Line** - Inserts a line before the current line.
- **Delete Line** - Deletes the current line.
- **Clear Line** - Clears from the cursor to the end of the line.
- **Renumber** - Resequences the line numbers to increment by 10s.
- **Indent** - Indents the program so that structures such as `IF ... END IF` are easy to see.

To find or replace text in your program, use the selections under the **Search** menu.

To Debug Programs

HP Instrument BASIC has four tools to help you debug your program:

- **Trace Ln** (on the control pad) - Displays the trace lines window that shows program lines in the order they execute. The last line shown is the next line to execute.
- **Trace Var** (on the control pad) - Displays the trace variables window that shows the values of variables in the current context.
- **Step** (on the control pad) - Executes one line of the program. The last line shown in the alpha or trace lines window is the next line to execute.
- **Set Breakpoints** (on the Edit menu) - Allows you to set and clear execution breakpoints. Breakpoints pause execution before the line specified.

To Get Help

HP Instrument BASIC's online help system contains reference information. To start help, select a choice from under the **Help** menu or type **HELP** on the command line. Refer to "To Use the Command Line" later in this chapter, for information about using the command line.

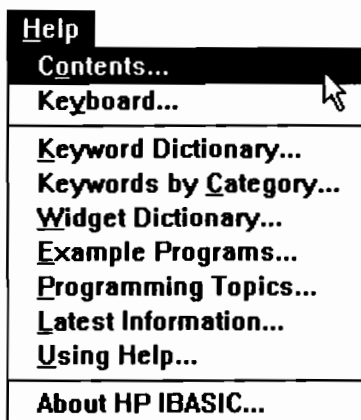


Figure 2-8. Help Topics

For information about using online help, refer to **Using Help** selection under the **Help** menu.

You can also get help when recovering from errors. Many error message dialog boxes have a **Help** button. When you press this button, additional information is displayed.

Note



You can also run **Help** by double-clicking on the **Help** icon in the **Instrument BASIC** group in the **Windows Program Manager**.

To Use Example Programs

Example programs are available from online help. These examples are easily copied into the edit window.

There are two types of examples:

- Statement examples.
- Complete examples.

To Copy Statement Examples

In online help, statement examples show you the basic use of a keyword or widget.



Figure 2-9. Statement Example

When you click on the **e>** icon, the line,

```
ASSIGN @File TO File_name$
```

is inserted above the current line in the edit window.

To Copy Complete Examples

There are two kinds of complete examples in online help:

- Task examples - General-Case or skeleton programs that allow you to fill in your program specifications. They are located under the **Example Programs** topic in online help.

Note



All examples in this manual are located under **Example Programs** as task examples.

- Keyword and widget examples - Complete program examples that demonstrate common usage. They are located under the **Program** menu from all widget and many keyword topics.

To copy either type of example to the editor, click on the **Copy Code** button. You'll be asked to save any program in the edit window before the example program replaces it.

2

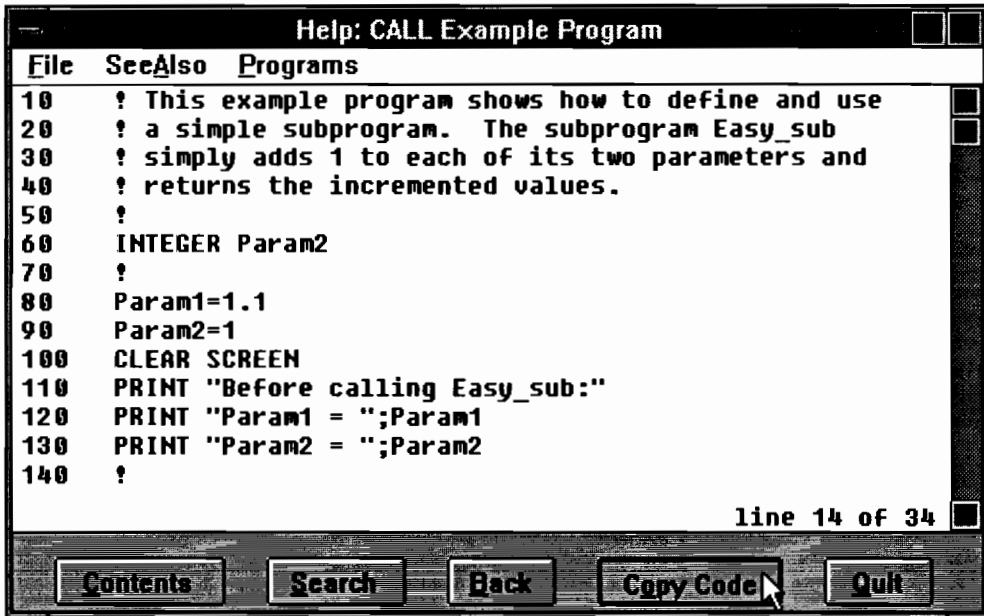


Figure 2-10. Copying Program Examples

2 To Use the Command Line

At the bottom of the alpha window, you can type HP Instrument BASIC keywords and immediately see the results. For example, if you ran the following program, the value 60 would be displayed.

```
10      A=10
20      B=20
30      C=30
40      D=A+B+C
50      PRINT D
60      END
```

But what if you're curious about the value of A? To find the value of A, type `PRINT A` **(Enter)** on the command line; 10 appears in the print area. Use the command line as a shortcut.

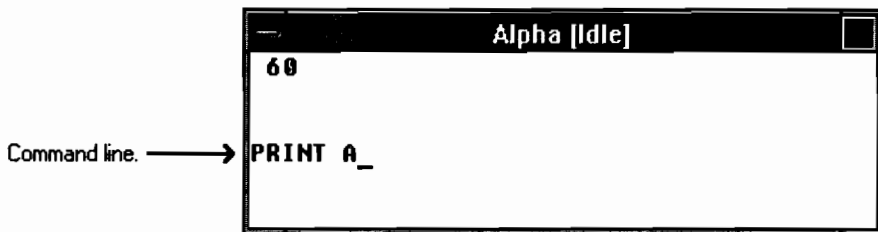


Figure 2-11. The Command Line

Table 2-1 shows some commonly-used command-line statements:

Table 2-1. Command-Line Examples

Statement	Action
HELP	Starts online help.
RUN	Runs the program.
OUTPUT 704; "BEEP"	Exercises your instrument.
800*12	Performs simple calculations.
CLEAR SCREEN	Clears the output from the alpha window.
PURGE "FILE"	Deletes the file.
CONTROL @Meter; SET ("X":50)	Changes the horizontal position of a widget.
CAT	Lists the contents of the current directory.

Note



You can type only HP Instrument BASIC statements on the command line; menu choices (such as **Open** or **Exit**) do not work.

To recall statements typed earlier on the command line, press **(Alt-▲)** until you see the statement you need. Edit the line and press **(Enter)**.

To Save Programs

When you save your programs, you have a choice of formats: text (file extension `.IBW`) or program (file extension `.PRG`). Programs saved in the program format take more disk space and load faster, but are only readable from HP Instrument BASIC. When you need a file in a human-readable form, choose text.

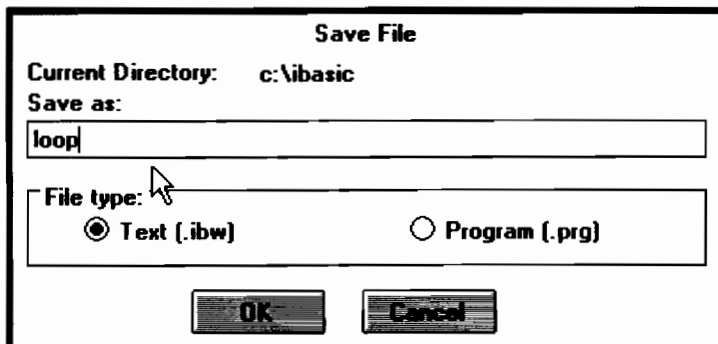


Figure 2-12. Saving Programs

To Set HP Instrument BASIC Configuration Information

The `IBASIC.INI` file keeps HP Instrument BASIC configuration information and is read each time HP Instrument BASIC starts. This file is located under the `\WINDOWS` directory.

An example IBASIC.INI file looks like:

```
[ibasic]
; pebmem=100000
; COMPATIBILITY_INSTR=1.0
PROGWIN=134,504,289,118,1
GRAPHWIN=124,516,42,328,0
EDITWIN=134,504,0,288,1
TOOLWIN=4,129,0,149,1
ZORDER=2305,2050,2052,3,
```

Some options saved in this file are:

- The layout of the HP Instrument BASIC windows.
This information is saved by the **Save Layout** choice on the HP Instrument BASIC control menu.
- The maximum amount of memory you'd like HP Instrument BASIC to use.

By default, HP Instrument BASIC allocates 1 Mbyte for your program. If your program needs more memory, edit the **IBASIC.INI** file and change the value of the **pebmem** parameter.

- The version of HP IBASIC on the instruments for which you create programs. When you set this level, HP Instrument BASIC performs syntax checking at that version's feature level. So if you have an instrument that runs IBASIC version 1.0, edit the **IBASIC.INI** file and change the **COMPATIBILITY_INSTR** parameter to 1.0.

Any line beginning with a semi-colon (;) is a comment and is ignored by HP Instrument BASIC.



A Quick Tour of the HP Instrument BASIC Language

This chapter summarizes key parts of the HP Instrument BASIC language. It discusses the following topics:

- Setting variables and using arrays.
- Writing text.
- Reading text.
- Iterating (Looping).
- Conditional and event-initiated branching.
- Subroutines, subprograms, and functions.
- Error handling.

Key Definitions



-
- **Keyword** - A reserved word that commands HP Instrument BASIC to perform some function. Some examples are PRINT and END. Keywords are shown all upper case.
 - **Statement** - A program line that contains at least one keyword. For example, PRINT "Hello". Statements are shown in computer font.
-

Note



This chapter presents common keywords with their most common usage. These keywords have additional parameters and can be used in more sophisticated ways. To see all the uses for keywords and their parameters, refer to the **Keyword Dictionary** in online help.

A complete list, and quick description, of the HP Instrument BASIC keywords is in Appendix A.

To Use Variables

The variable names used in HP Instrument BASIC must have the following characteristics:

- Must start with an upper-case letter (but can't be all upper case). Note that when you type a mixed-case variable name, HP Instrument BASIC changes it to initial capitalization only.
- Must be 15 characters or less.
- May contain underscores and numbers, but no symbols (such as % or #) or spaces.
- May end with a \$ (if a string variable).

In other words:

Table 3-1. Variable Names

Valid Variable Names	Invalid Variable Names
Print	print or PRINT
A1	1A
Final_results	Final.results
New_results	New results
Name\$	Name#

HP Instrument BASIC supports the following variable data types:

- **Real** - A double-precision (64-bit) floating-point number. Unless specified otherwise, all variables are considered real. For completeness, declare real variables before using them. For example:

```
10    REAL Cycles
20    Seconds=3
30    Khertz=6.2
40    Cycles=Seconds*Khertz
50    PRINT Cycles
60    END
```

The output is:

18.6

- **Integer** - A 16-bit whole number (-32768 to +32767). You must specify a variable as an integer before using it. When real numbers are assigned to integer variables, they are automatically rounded after the operation. For example:

```
10    INTEGER Cycles, Seconds
20    Seconds=3
30    Hertz=6.2
40    Cycles=Seconds*Hertz
50    PRINT Cycles
60    END
```

The output is:

19

- **String** - Strings are arrays of characters; you specify their size before using them. String variable names end with a \$ (such as `Name$`). Before you use string variables, you specify their size (in characters) with a `DIM` statement; for example `DIM Name$[6]`. If you do not dimension a string variable, the default length is 18 characters. When you assign a string constant to a string variable, it must be in double quotes. For example, `Name$="Arthur"`.

Refer to the next topic, "To Manipulate Strings", for details on string operations.

To Manipulate Strings

To concatenate strings, use the `&` symbol:

```
Name$ = First$ & Last$
```

To refer to an entire string, use the variable name; such as `Name$`. Table 3-2 shows how to access substrings using `PRINT A$[Start; Length]` and `PRINT A$[Start, Stop]`.

Table 3-2. Accessing Substrings

Statement	Output
<code>PRINT A\$</code>	DICTIONARY
<code>PRINT A\$[1]</code>	DICTIONARY
<code>PRINT A\$[1;4]</code>	DICT
<code>PRINT A\$[3,7]</code>	CTION
<code>PRINT A\$[3;1]</code>	C
<code>PRINT A\$[3,3]</code>	C



To Specify Numeric Arrays

Use the **REAL** or **INTEGER** keywords to declare numerical arrays before using them. For example, **INTEGER Data(4,4)**. If you don't dimension an array, the default upper-bound is 10.

Note



By default, the first element of an array is referenced by the index of zero. Therefore, an array with an upper bound of 4 (**DIM A(4)**) has 5 elements (**A(0)** through **A(4)**).

If you want arrays to start from index of one, refer to the **OPTION BASE** keyword in the **Keyword Dictionary** in online help.

Table 3-3 shows the output when parts of the one-dimensional array, **B**, are printed:

Table 3-3. Accessing Array Elements

Statement	Output
PRINT B(*)	1 3 5 7 11
PRINT B(1)	3
PRINT B(3,7)	ERROR 16 Improper Dimensions

For information about accessing subarrays, refer to the **MAT** keyword in online help.

The following example fills an array with information and prints the entire array:

```
10    INTEGER Data(2,3)
20    FOR I=0 TO 2
30      FOR J=0 TO 3
40        Data(I,J)=I*10+J
50      NEXT J
60    NEXT I
70    PRINT Data(*);
80    END
```

The results are:

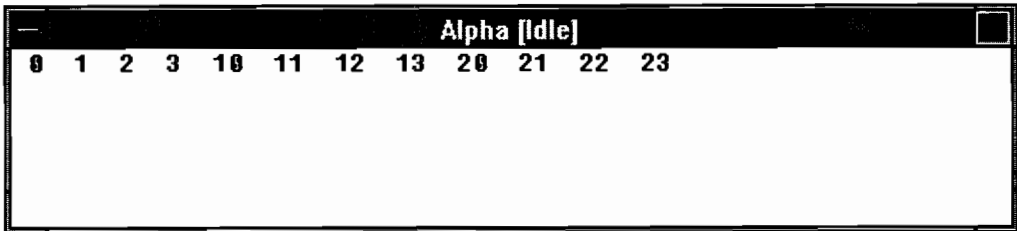


Figure 3-1. Array Output

The array is written in row-major order (the right-most subscript varies the fastest).

To Write Text

To write to the alpha window, use the PRINT and DISP keywords.

- PRINT - Writes to the print area of the alpha window. The print area holds up to 100 lines. Separate variable names with a comma to display output in columns.
- DISP - Writes to the display line in the alpha window. The display line is useful for presenting status messages. Each DISP statement overwrites the previous one; the line does not scroll.

The following example shows how lines printed to the print area scroll on the print area, but each line printed to the display line overwrites the previously displayed line.

```
10      FOR I=1 TO 20
20          PRINT "Print area line:",I
30          DISP "Display line:",I
40      NEXT I
50      END
```

The output from the example program is shown in Figure 3-2. To see the print area scroll, select the alpha window and press the up (▲) and down (▼) cursor keys on the keyboard.

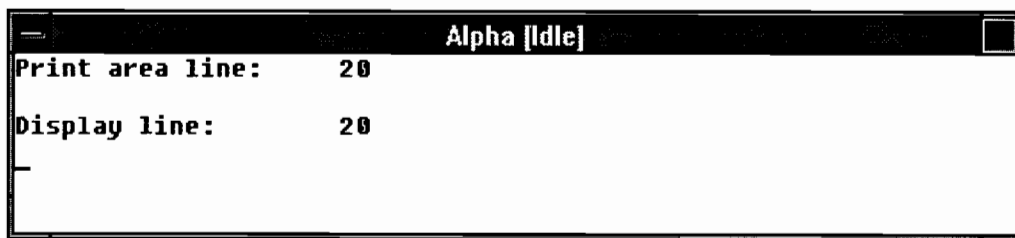


Figure 3-2. Printing to the Alpha Window

To Format Output

Use the comma to separate items in a PRINT or DISP statement to output in columns. Use the semi-colon to output in a more compact format or to suppress a line-feed.

```
10    A=10
20    B=20.6
30    C$="Alfred"
40    D=98.6
50    PRINT A;B;C$;D
60    PRINT A,B,C$,D
70    END
```

The alpha windows displays:

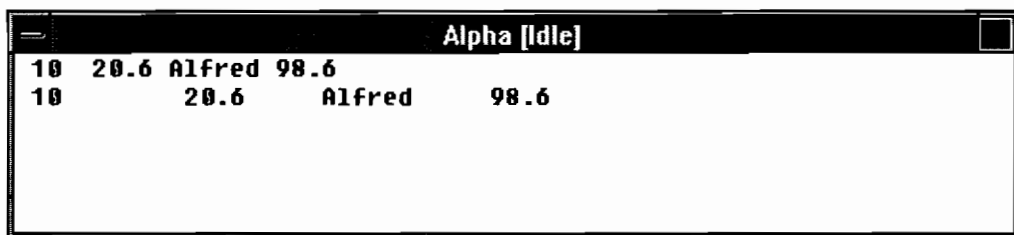


Figure 3-3. Formatted Output

Another way to format when writing is with an image specifier. An image specifier is a set of letters that specifies the format of each alphanumeric character to be printed; any letters in an image specifier must be upper case.

The following example shows the image specifier, DDDD.DD, rounds the number that is displayed.

```
10      Number=3333.99641
20      PRINT USING "DDDD.DD";Number
30      END
```

The display looks like:

3334.00

Table 3-4 shows examples of common image specifiers:

Table 3-4. Formatting Examples

Statement	Output	Explanation
PRINT A,B,C\$	14241 23.2655 BOB	Default format.
PRINT A;B;C\$	14241 23.2655 BOB	Compact format.
PRINT USING "DD.DD"; B	23.27	D prints a digit or a leading space. An image specifier rounds numbers; it does not truncate.
PRINT USING "DD"; A	ERROR 105 Numeric IMAGE field too small	An image specifier must include all of the integer part of a number.
PRINT USING "SDD.DD"; B	+23.27	S prints the sign of a number.
PRINT USING "DD.DD,X"; B,B	23.27 23.27	X prints a space (the image specifier is reused for the second value).
PRINT USING "ZZZ.DD"; B	023.27	Z prints a digit or a leading zero.

A complete set of image specifiers is listed under the IMAGE keyword in online help.

When specifying multiple letters in an image specifier, you can repeat the letters (DD.DDD) or specify the letter count (2D.3D).

To Read Text

The INPUT keyword displays a prompt in the alpha window and reads keyboard input.

INPUT displays a string to the display line in the text window and waits for you to type a response. Program execution pauses until the response is entered. If you don't specify a string to prompt for information, HP Instrument BASIC displays a "?".

The following example prompts for two numbers. You can enter both numbers on the same line (separated by a comma) or type a number, press **Enter**, type the other number, and press **Enter**.

```
10    INTEGER Number1,Number2
20    INPUT "Input 2 numbers:",Number1,Number2
30    PRINT Number1,Number2
40    END
```

When you press Run, you will see the INPUT prompt in the alpha window. Figure 3-4 shows the numbers 21 and 33 typed in response to the prompt.

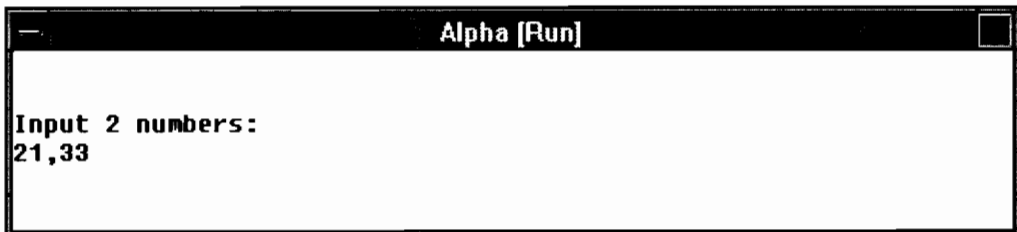


Figure 3-4. INPUT Keyword Output

To Iterate

The HP Instrument BASIC keywords that allow iteration are:

- **FOR ... NEXT** - Defines a loop which is repeated a specified number of times.

The following example prints every tenth number from 10 to 100 (10, 20, etc).

```
10   FOR I=10 TO 100 STEP 10
20     PRINT I
30   NEXT I
40   END
```

If the STEP 10 parameter is omitted, the step size is one.

- **REPEAT ... UNTIL** - Allows execution of a program segment until the specified condition is true. This loop is always executed at least once.

The following example prompts for a password until it is entered correctly.

```
10   DIM Password$(10),Current$(10)
20   Current$="Dirk"
30   REPEAT
40     INPUT "What is your password?",Password$
50   UNTIL Password$=Current$
60   PRINT "Password accepted."
70   END
```

- **WHILE ... END WHILE** - Allows execution of a program segment while the specified condition is true. This loop is executed zero times if the condition is false.

In the following example, the statement within the loop is not executed.

```
10   Pigs_fly$="False"
20   WHILE Pigs_fly$="True"
30     PRINT "You've just won a free lunch!"
40   END WHILE
50   PRINT "Sorry."
60   END
```

To Use Conditional and Event-Initiated Branching

The HP Instrument BASIC keywords that allow branching to other parts of the program are:

- IF ... THEN ... ELSE - Provides conditional branching to one of two alternative program segments.

The following example shows how the answer to a question determines which statement is executed.

```
10   DIM Response$[1]
20   INPUT "Do you like gum? (Y/N)",Response$
30   IF Response$="Y" OR Response$="y" THEN
40     PRINT "Have a stick of gum."
50   ELSE
60     PRINT "Have a mint."
70   END IF
80   END
```

- **SELECT ... CASE** - Provides conditional branching to one program segment when there are several alternatives.

The following example shows how to use **SELECT ... CASE** so that only one **PRINT** statement is executed. Note that the first branch that meets the **CASE** condition is taken.

```

10    INPUT "Enter the voltage:",Volts
20    SELECT Volts
30    CASE <10
40        PRINT "Voltage is too low."
50    CASE <20
60        PRINT "Voltage within design limits."
70    CASE ELSE
80        PRINT "Voltage is too high."
90    END SELECT
100   END

```

Note



Conditional looping and branching statements use one or more of the following operators: **<**, **>**, **<=**, **>=**, **=**, **<>** (not equal), **AND**, **OR**, **NOT**, and **EXOR**. Refer to the **Keyword Dictionary** in online help for more information.

- **ON event** - Allows an event-initiated branch to be taken each time *event* is detected. Some examples of event-initiated branching are:
 - **ON ERROR** - Executes a branch when the program encounters an error.
 - **ON INTR** - Executes a branch when an interface card generates an interrupt.
 - **ON TIMEOUT** - Executes a branch when an interface fails to respond.
 - **ON CYCLE** - Executes a branch each time the specified number of seconds has elapsed.

The following example interrupts you every five seconds while the FOR ... NEXT loop prints the value of I. If you respond y to cancel the alarm, the ON CYCLE keyword is canceled by the OFF CYCLE keyword.

```

10    DIM Cancel$[1]
20    ON CYCLE 5 GOSUB Alarm
30    FOR I=1 TO 500
40        PRINT I
50    NEXT I
60    STOP
70 Alarm: !
80    PRINT "5 seconds have gone by"
90    INPUT "Cancel alarm? (y/n)",Cancel$
100   IF Cancel$="y" THEN
110       OFF CYCLE
120       DISP "Alarm is off"
130   ELSE
140       END IF
150   RETURN
160   END

```

Generally, each event is enabled by the *ON statement* keyword and disabled by the *OFF statement* keyword. For another example of using event-initiated branching, refer to "To Service HP-IB Interrupts" in Chapter 5.

Each event has a priority assigned to it. For an event to be serviced, its priority must be greater than the current system priority. When no events are being serviced, the system priority is zero. The maximum priority is 17 (used for ON ERROR). An event can interrupt another event if the second event's priority is greater. If this happens, the first event waits until the second is done before completing.

For information about setting system priority, refer to SYSTEM PRIORITY in the **Keyword Dictionary** in online help.

To Use Subroutines, Subprograms, and Functions

There are three different ways to transfer program execution in HP Instrument BASIC: subroutines, subprograms, and user-defined functions. Subprograms and functions create different contexts than the main context.

Key Definition



Context - A block of program statements that is an independent unit. Each context has its own “private” set of variables, line labels, angle mode (DEG or RAD), and other attributes.

- **Subroutines** transfer program execution to a specific label or line in the same context (not crossing a subprogram or function boundary). Program execution resumes at the statement following the subroutine call after the RETURN keyword executes. Normal program execution continues through the subroutine (unless you prevent it).

Subroutine Summary:

- Called by GOSUB *label* | *line number*.
- Delineated by *label*: ! ... RETURN.
- Stays in the same context.
- Does not return values.

The following example shows the subroutine starting with label `Roll_a_die` called twice, to simulate rolling dice. The `STOP` keyword prevents normal execution from continuing through the subroutine.

```

10      GOSUB Roll_a_die
20      GOSUB Roll_a_die
30      STOP
40 Roll_a_die: !
50      Die=INT(RND*6)+1
60      PRINT "Die is";Die
70      RETURN
80      END

```

- **Subprograms** transfer program execution to a separate context; in other words, variables from where you call the subprogram are not defined in the subprogram's context. You must pass parameters into and out of subprograms; subprograms have their own set of local variables. Passing variables into and out of subprograms and using common blocks is explained in "To Pass Variables" later in this chapter.

Subprograms are located at the end of the main program, and can be opened and saved separately; this topic is explained in "To Use Existing Subprograms". HP Instrument BASIC subprograms can be called recursively.

Subprogram Summary:

- Called by `CALL name`.
- Delineated by `SUB name ... SUBEND`.
- Starts a different context.
- Can return multiple values.

The following example shows a subprogram that is called twice to simulate a pair of dice being thrown. The subroutine is passed a value that specifies the number of sides on each die.

```
10    INPUT "Sides per die?",Sides
20    CALL Roll_a_die(Sides)
30    CALL Roll_a_die(Sides)
40    END
50 !
60    SUB Roll_a_die(Sides)
70        Die=INT(RND*Sides)+1
80        PRINT "Die is";Die
90    SUBEND
```

- **User-defined functions** transfer program execution to a different context and return a value to the calling context. User-defined functions are located at the end of the main program.

Function Summary:

- Called by *FNname*.
- Delineated by *DEF FNname ... FNEND*.
- Starts a different context.
- Returns a single value.

The following program uses a function to convert from decimal to hexadecimal. The function name ends with a \$ because the function returns a string value.

```
10      INPUT "Input a number 0-15:",Number
20      PRINT "Hex equiv. is: ";FNHex$(Number)
30      END
40      !
50      DEF FNHex$(Number)
60      IF Number<=9 THEN
70          RETURN CHR$(48+Number)
80      ELSE
90          RETURN CHR$(55+Number)
100     END IF
110     FNEND
```

To Pass Variables

Because subprograms are separate contexts, variables from other contexts must be passed into the subprogram before they are used. Use one of the following techniques to transfer variable values to subprograms (these techniques can also be used with user-defined functions):

- Passing variables by value only allows values to be passed *into* subprograms. These variables are essentially read-only; this status is accomplished by enclosing the set of variables to pass in double parentheses.

The following example shows how passing variables by value does not allow the new value of Alpha to be passed out of the subprogram.

```
10      Alpha=10
20      PRINT "Main: Alpha = ";Alpha
30      CALL Value((Alpha))
40      PRINT "Main: Alpha = ";Alpha
50      END
60 !
70      SUB Value(Alpha)
80          Alpha=50
90          PRINT "Subprogram: Alpha = ";Alpha
100     SUBEND
```

The results are:

```
Main: Alpha = 10
Subprogram: Alpha = 50
Main: Alpha = 10
```

- Passing variables by reference allows values to be passed both *into* and *out of* subprograms.

The following example shows how passing variables by reference shares variables between contexts. The value of Alpha is changed in the subprogram and therefore is changed in the main program.

```
10      Alpha=10
20      PRINT "Main: Alpha = ";Alpha
30      CALL Ref(Alpha)
40      PRINT "Main: Alpha = ";Alpha
50      END
60 !
70      SUB Ref(Alpha)
80          Alpha=50
90          PRINT "Subprogram: Alpha = ";Alpha
100     SUBEND
```

The results are:

```
Main: Alpha = 10
Subprogram: Alpha = 50
Main: Alpha = 50
```

- Using a common block allows a set of values to be available for reading or writing to subprograms (like passing by reference). The name of the common block is listed between the slashes. The common block statement in each context must match variable type, size, and position with the other common block statements of the same name.

The following example shows how a common block shares variables between contexts. The value of Alpha is changed in the subprogram and therefore is changed in the main program.

```
10      COM /Greek/ Alpha,Beta
20      Alpha=10
30      PRINT "Main: Alpha = ";Alpha
40      CALL Common
50      PRINT "Main: Alpha = ";Alpha
60      END
70 !
80      SUB Common
90          COM /Greek/ Alpha,Beta
100         Alpha=50
110         PRINT "Subprogram: Alpha = ";Alpha
120      SUBEND
```

The results are:

```
Main: Alpha = 10
Subprogram: Alpha = 50
Main: Alpha = 50
```

To Use Existing Subprograms

To use a subprogram with multiple programs, create it separately, save it in the program (.PRG) format, and then include it in other programs with the LOADSUB keyword. When a subprogram is included, it is placed at the end of the program.

To delete a subprogram, you must use the DELSUB keyword because a subprogram is a separate context,

The following example shows loading a password-checking subprogram called Check that was stored in the file, Sub_file.

```
10    LOADSUB ALL FROM "C:\IBASIC\SUBFILE.PRG"
20    Current$="Dirk"
30    CALL Check(Current$)
40 ! rest of program
50    DELSUB Check
60    END
70 ! subprogram is loaded here
```

The subprogram looks like:

```
10    SUB Check(Current$)
20    REPEAT
30        INPUT "Input your password:",Password$
40        IF Current$<>Password$ THEN PRINT "Password incorrect."
50    UNTIL Current$=Password$
60    PRINT "Password accepted."
70    SUBEND
```

To Handle Errors

To prevent errors from stopping execution, trap the errors and handle them in your program. HP Instrument BASIC provides the following keywords for error handling:

- **ON ERROR GOSUB | CALL** - Enables the error event interrupt. Any error that occurs branches to the specified location (subroutine or subprogram).
 - **ON ERROR GOSUB** has a local scope; it only traps errors that occur in the context where it appears.
 - **ON ERROR CALL** has a global scope; it traps errors that occur in any context.

Because the **ERROR** event has a software priority of 17 (the highest), an error will interrupt any other event.

- **RETURN | SUBEXIT** - Returns program execution to the line that caused the error. If the error has not been fixed, the error event will be triggered again and an infinite loop may result.
- **ERROR RETURN | ERROR SUBEXIT** - Returns program execution to the line following the line that caused the error.
- **OFF ERROR** - Disables the error event interrupt.
- **ERRN** - Returns the error number of the error that occurred. For a complete list of error numbers and their messages, refer to **Error Messages** under **Programming Topics** in online help.
- **ERRM\$** - Returns the error message of the error that occurred.

The following example shows an error handling subprogram that is called whenever any error occurs, but it only handles ERROR 31 Divide by zero and ERROR 80 Disk not in drive. With any other error, the error number and message is printed out and the program is stopped.

3

```
10    ON ERROR GOSUB Error
20    INPUT "Enter miles traveled and total hours:",Miles,Hours
30    Mph=Miles/Hours
40    PRINT "You traveled at";Mph;"Miles/hour"
50 ! Rest of main program
60    STOP
70 Error: !
80    SELECT ERR#
90        DIM Retry$(5)
100       Retry$="True"
110      CASE 31 ! Divide by zero error
120        INPUT "Hours must be greater than 0:",Hours
130      CASE 80 ! Disk not in drive
140        DISP "Insert the disk and press Continue"
150        PAUSE
160      CASE ELSE
170        DISP "Unexpected error:",ERR#,ERRM$
180        Retry$="False"
190      END SELECT
200      IF Retry$="False" THEN
210        STOP
220      ELSE
230        CLEAR ERROR
230        RETURN
240      END IF
250      END
```

Summary

Here are the keywords discussed in this chapter and other keywords that are closely related to the topics covered:

Related Keywords

Topic	Related Keywords	Description
Creating Arrays	DIM	Defines real or string arrays.
	INTEGER	Defines an integer array.
	REAL	Defines a real array.
	MAT	Performs various operations on arrays.
Writing	DISP	Prints to the overwriting display line of the alpha window.
	PRINT	Prints to the scrolling print area of the alpha window.
Formatting	IMAGE	Specifies an image format to be used by multiple statements.
	USING	Specifies an image format.
Printing Text	CLEAR SCREEN (CLS)	Clears the contents of the alpha window.
	DUMP ALPHA	Prints the alpha window to the printer.
	PRINTER IS 26	Sends output to printer.
Reading	INPUT	Prompts for input and accepts user input.
Iterating	FOR ... NEXT	Defines a loop that is repeated a specific number of times.
	LOOP ... END LOOP	Defines a loop that is repeated until the expression in an EXIT IF statement is evaluated as true.
	REPEAT ... UNTIL	Defines a loop that is executed until the specified condition is true.
	WHILE ... END WHILE	Defines a loop that is executed while the specified condition is true.

Related Keywords (continued)

Topic	Related Keywords	Description
Branching	GOTO	Allows branching to a line number or label.
Conditional Branching	IF ... THEN ... ELSE	Allows branching to one of two program segments.
	SELECT ... CASE ... END SELECT	Allows branching to one of multiple program segments.
Event-Initiated Branching	DISABLE	Disables all event-initiated branches currently defined.
	DISABLE INTR	Disables an HP-IB interrupt.
	ENABLE	Re-enables event-initiated branches suspended by DISABLE.
	ENABLE INTR	Enables an HP-IB interrupt.
	OFF <i>event</i>	Cancels specified event.
	ON CYCLE	Branches every specified number of seconds.
	ON ERROR CALL GOSUB GOTO RECOVER	Branches when an error occurs.
	ON INTR	Branches when an HP-IB card generates an interrupt.
	ON TIMEOUT	Branches when an interface fails to respond.
Subroutine	GOSUB	Branches to a specific line or label.
	RETURN	Returns from a subroutine call.
Subprogram	CALL	Branches to a subprogram (different context).
	COM	Passes variables in a common block.
	DELSUB	Deletes a subprogram.
	LOADSUB	Opens a subprogram saved in the program format and places it at the end of your program.
	SUB <i>name</i> ... SUBEND	Delineates a subprogram.

Related Keywords (continued)

Topic	Related Keywords	Description
Function	DEF FN <i>name</i> ... FNEND	Delineates a function.
	FN	Branches to a user-defined function.
Error Handling	ERRL	Indicates if an error occurred during execution of a specified line.
	ERRLN	Returns the program-line number of the most recent error.
	ERRM\$	Returns the text of the last error message.
	ERRN	Returns the error number of the last error message.
	ERROR RETURN	Returns program control to the line following the line which caused the most recent ON ERROR GOSUB.
	ERROR SUBEXIT	Returns program control to the line following the line which caused the most recent ON ERROR CALL.
	OFF ERROR	Cancels ON ERROR.
	ON ERROR	Defines a branch to take whenever an error occurs.

For complete information about any keyword, refer to the **Keyword Dictionary** in online help.

Other Information

This chapter quickly introduced you to common programming keywords and tasks in HP Instrument BASIC. The rest of the manual covers the following tasks:

- Chapter 4 - Reading and writing to files.
- Chapter 5 - Communicating with instruments.
- Chapter 6 - Interacting with users and graphing data.
- Chapter 7 - Creating an operator interface.
- Chapter 8 - Sharing information with other applications.





Writing and Reading Files

This chapter explains how to write data to files and subsequently read data from files to use in your program.

The following tasks are used when writing data to files:

- Creating a directory (if it doesn't exist).
- Creating the file (if it doesn't exist).
- Assigning an I/O path to the file.
- Writing the information.
- Closing the I/O path.

The following tasks are used when reading data from files:

- Assigning an I/O path to the file.
- Reading the information.
- Closing the I/O path.

To Create Directories

To make a new directory, use the CREATE DIR keyword.

The following statement creates a directory called NEWDIR:

```
CREATE DIR "NEWDIR"
```

Note that directory and file names are case insensitive; you can use any mixture of upper- and lower-case letters.

Figure 4-1 shows the newly-created directory as shown in the Windows File Manager.

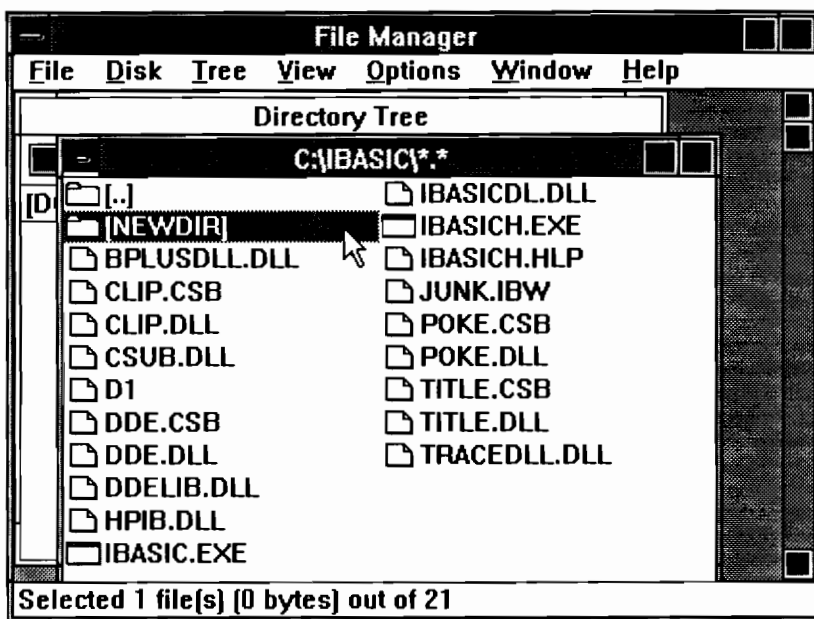


Figure 4-1. A New Directory

Note



To delete an empty directory (or a file), use the PURGE keyword. For example:

```
PURGE "NEWDIR"
```

To Create Files

Before you write to a file, it must exist. Use the CREATE keyword to create a file. If the file already exists, you'll get an error message: **ERROR 54 Duplicate file name.**

The following statement creates a file:

```
CREATE "NEWDIR\FILE",1
```

To Assign I/O Paths

To read and write files, assign an I/O path to each file with the ASSIGN keyword. You read from or write to the path. For example:

```
ASSIGN @File TO "NEWDIR\FILE"; FORMAT ON
```

The ASSIGN statement sets the file pointer to the beginning of the file. When you start writing, any existing data is overwritten.

I/O paths are automatically closed when the context in which they are created terminates. A subprogram or function context terminates when it ends and control returns to the main program. The main program context terminates when the program stops.

To Specify Text File Format

The default format mode is to read and write files in an HP Instrument BASIC-specific form. To specify a text file, use the FORMAT ON option. For example:

```
ASSIGN @File TO "FILE"; FORMAT ON
```

You must read data with the same format as it was written.

4

To Write or Read Data

Use the following keywords to write or read data:

- **OUTPUT** - Writes data to a file.

The following example writes an array of data to a file. Because no image specifier is used for the array, HP Instrument BASIC puts commas between values and a CR/LF at the end of the line.

```
10    DIM Data(5)
20    CREATE "FILE",1
30    ASSIGN @File TO "FILE";FORMAT ON
40    FOR I=0 TO 5
50        Data(I)=I
60    NEXT I
70    OUTPUT @File;Data(*)
80    END
```

The file contains:

```
0, 1, 2, 3, 4, 5
```

- ENTER - Reads data from a file.

The following example reads the array that was written to the file in the previous example and prints the values to the alpha window. Because the array specification in the PRINT statement ends with a semi-colon, the values output to the alpha window are separated by one space.

```
10    DIM Data(5)
20    ASSIGN @File TO "FILE";FORMAT ON
30    ENTER @File;Data(*)
40    PRINT Data(*);
50    END
```

The alpha window displays:

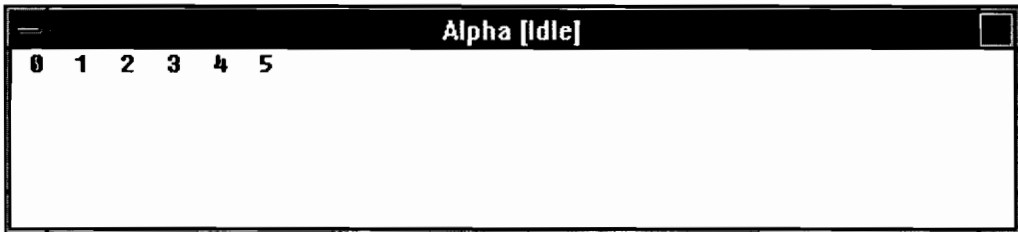


Figure 4-2. File Data

To Format Output

You format output to a file in the same way that you format output to the alpha window—by using an image specifier.

The following example uses the image specifier `SDD.DD` to print the sign of the number, two integer digits, and the fractional part (rounded to two digits).

```
10    DIM Data(5)
20    ASSIGN @File TO "FILE";FORMAT ON
30    FOR I=0 TO 5
40        Data(I)=2+I*.015
50    NEXT I
60    OUTPUT @File USING "SDD.DD";Data(*)
70    END
```

The file contains:

```
+2.00 +2.02 +2.03 +2.05 +2.06 +2.08
```

For more detailed information about formatting output, refer to “To Format Output” in Chapter 3.

To Close I/O Paths

After you have finished reading or writing, close the file by closing the I/O path. Use the `ASSIGN` keyword. For example:

```
ASSIGN @File TO *
```

The file closes when the I/O path is closed or reassigned.

I/O paths are automatically closed when the context in which they are created, terminates. A subprogram or function context terminates when it ends and control returns to the main program. The main program context terminates when the program stops.

Summary

Here are the keywords discussed in this chapter and other keywords that are closely related to the topics covered:

Related Keywords

Topic	Related Keywords	Description
Creating Directories	CREATE DIR	Creates a directory.
	MASS STORAGE IS (MSI)	Specifies the default directory.
	RENAME	Changes a directory's name.
Creating Files	CAT	Lists files.
	CREATE	Creates a MS-DOS®-type file.
	CREATE ASCII	Creates an ASCII-type file.
	CREATE BDAT	Creates a BDAT-type file.
	WILDCARDS DOS OFF	Enables or disables wildcard recognition in file names.
Deleting Directories or Files	PURGE	Deletes an empty directory or a file.
Assigning I/O Paths (Opening Files)	ASSIGN	Opens a file with the file pointer set to the beginning of the file.
Writing Text Data	ASSIGN ... FORMAT ON	Specifies text file format.
Writing Data	OUTPUT	Outputs data to a file.
Formatting Data	IMAGE	Specifies format for output.
	OUTPUT ... USING	Specifies the line number of the formatting statement to use.
Reading Data	ENTER	Inputs data from a file to variables.
Closing I/O Paths (Closing Files)	ASSIGN @name TO *	Closes the file.



Communicating with Instruments

This chapter discusses the tasks you perform when communicating with instruments:

- Assigning an I/O path to the instrument.
- Initializing the instrument.
- Sending commands to the instrument.
- Asking for a measurement and reading the values.
- Checking the instrument's status (HP-IB).
- Closing the I/O path.

Although you'll generally use instruments with an HP-IB interface, HP Instrument BASIC also supports instruments with an RS-232 interface. Unless specified, the information in this chapter applies to both interfaces.

To Assign I/O Paths to Instruments

HP Instrument BASIC has unified I/O; in other words, you communicate with instruments just as if they were files.

To simplify reading and writing, you assign an I/O path to each instrument's device selector with the ASSIGN keyword. You read from or write to the path. For example:

```
ASSIGN @Name to 709
```

Just as with file name assignments, instrument I/O paths are closed when the context in which they are created, terminates. A subprogram or function context terminates when it ends and control returns to the main program. The main program context terminates when the program stops.

To Determine HP-IB Device Selectors

An HP-IB instrument's device selector is in the format **xyyzz**.

- **xx** is the one- or two-digit interface select code. The HP-IB select code is 7 (unless you changed it). Valid HP-IB select codes are 3-8. If you are using a National Instrument card, you must use select code 14 or 15.

For details about using select codes and about using a 16-bit VGA card, refer to **Summary of Interface Select Codes** under **Programming Topics** in online help.

To find the select code for your cards, use the utility program under **Summary of Interface Select Codes** in **Programming Topics** in online help.

- **yy** is the bus address of the instrument. You usually press a button on the front panel of an instrument to see the bus address. Use a leading zero for a bus address that is less than 10. For example, use 09, instead of 9.
- **zz** is the secondary address (if it exists). In most cases, secondary addresses are not used and this part of the address is not included. Secondary addresses are typically used by cardcage-type instruments that incorporate multiple plug-in modules.

For example, if your HP-IB interface has a select code of 7 (the default) and the bus address is 9, the instrument device selector is 709:

ASSIGN $\odot$ Dmm TO 709

To Determine RS-232 Device Selectors

An RS-232 instrument's device selector is the select code of its interface: 9 (COM1), 10 (COM2), 11 (COM3), or 12 (COM4). You can have only one instrument connected to each port.

ASSIGN $\odot$ Scope TO 9

To Exchange Binary Data

To send or receive binary (unformatted) data, use the SWAP ON option in the ASSIGN keyword. Binary data (real or integer values) is transferred more quickly than text data (FORMAT ON). Note that binary and text are only methods of transferring data; the data transferred may be entered into variables of various data types.

Refer to your instrument's manual for details on transferring binary data. For example:

ASSIGN $\odot$ Dmm TO 709; FORMAT OFF, SWAP ON

The default mode (SWAP OFF) is the native personal computer format that transfers bytes in the opposite order than instruments do. The order of bytes only matters when binary data is exchanged (FORMAT OFF).

To Prepare Instruments

Before you communicate with an instrument, you need to complete the following tasks:

- If you're using an HP-IB instrument, initialize the interface and the instrument.
- If you're using an RS-232 instrument, configure the interface.

To Initialize HP-IB Instruments

Use the **CLEAR** keyword to initialize an HP-IB instrument. **CLEAR** prepares the instrument to listen and may set it to an instrument-dependent state; check the instrument's manual for details.

The following example, initializes a digital multimeter.

```
ASSIGN @Dmm TO 707  
CLEAR @Dmm
```

Note



Many instruments perform an initialization when powered-on. You generally use **CLEAR** to ensure an HP-IB instrument will respond to commands.



To Initialize SCPI Instruments

Standard Commands for Programmable Instruments (SCPI) is a standard set of commands used to communicate with instruments. Several of the examples in this chapter use SCPI commands.

To initialize a SCPI instrument, send commands to the instrument to reset it to a known state and reset the status reporting structure.

```
10  ASSIGN @Scope TO 707
20  OUTPUT @Scope;"*RST" ! Reset measurement state.
30  OUTPUT @Scope;"*CLS" ! Clear status event regs.
40  OUTPUT @Scope;"*SRE 0" ! Clear service request enable regs.
50  OUTPUT @Scope;"*ESE 0" ! Clear std. event enable regs.
60  OUTPUT @Scope;"STATUS:PRESET" ! Preset all other status regs.
70  END
```

5

To Configure RS-232 Interfaces

The RS-232 interface configuration on the computer must match the configuration settings of the instrument. Refer to the instrument's manual for information about the instrument's configuration.

The default RS-232 interface configuration is listed in the [ports] section of Windows configuration file, WIN.INI. To change the default configuration, use the Ports program from the Control Panel in the Program Manager, and restart Windows. If there are no defaults listed or you do not have a [ports] section, the default configuration is: 300 baud, odd parity, 2 stop bits, and 7 bits/character.

To change the configuration in your program, use the CONTROL keyword to set the values in CONTROL registers. These registers are:

- Baud rate - CONTROL Register 3
- Parity - CONTROL 4 (see Table 5-1)

- Enable parity - CONTROL Register 4 (see Table 5-1)
- Stop bits - CONTROL Register 4 (see Table 5-1)
- Character Length - CONTROL Register 4 (see Table 5-1)

Table 5-1. Character Format and Parity Settings (Register 4)

Parity Sense (Bits 5 & 4)	Parity Enable (Bit 3)	Stop Bits (Bit 2)	Character Length (Bits 1 & 0)
00 ODD parity 01 EVEN parity 10 Always one 11 Always zero	0 Disabled 1 Enabled	0 1 stop bit 1 1.5 stop bits (if 5 bits/char), or 2 stop bits (if 6, 7, or 8 bits/char)	00 5 bits/char 01 6 bits/char 10 7 bits/char 11 8 bits/char

The following example sets the attributes of the interface to 9600 baud (set register 3 to 9600) and 2 stop bit, 1 start bit, 8 data bits and no parity (set register 4 to binary(000111) = decimal(7)).

```

10    CONTROL 9,3;9600
20    CONTROL 9,4;7
30    END

```

Use the STATUS keyword to read current serial configuration values from the STATUS registers.

The following example uses the STATUS keyword to read the current baud setting and the CONTROL keyword to change it.

```

10    INTEGER Baud
20    STATUS 9,3;Baud
30    IF Baud<1200 THEN
40        Baud=1200
50        CONTROL 9,3;Baud
60    END IF
70    END

```

To Send Commands

Use the **OUTPUT** keyword to send commands to an instrument. Send commands as strings.

Different instruments accept different commands and command separators. Check the instrument's manual to find the correct commands and syntax.

The following example sets an HP 54601A oscilloscope to a default configuration, autoscale it, and set the probe attenuation factor to 10.

```
10    ASSIGN @Scope TO 707
20    CLEAR @Scope
30    OUTPUT @Scope;"*RST"
40    OUTPUT @Scope;":AUTOSCALE"
50    OUTPUT @Scope;":CHAN1:PROBE X10"
60    END
```

5

To Take Readings

Use the **OUTPUT** keyword to ask for an instrument reading; use the **ENTER** keyword to read the value from the instrument. For example:

```
OUTPUT @Dmm "READ?"
ENTER @Dmm; Voltage
```

The following example sets up the HP 34401A multimeter to a default state, asks for five voltage measurements, reads them, and then prints them.

```
10    DIM Volt(4)
20    ASSIGN @Dmm TO 722
30    CLEAR @Dmm
40    OUTPUT @Dmm;"*CLS; *RST"
50    OUTPUT @Dmm;"*SRE 0"
60    OUTPUT @Dmm;"*ESE 0"
70    OUTPUT @Dmm;"STATUS:PRESET"
80    OUTPUT @Dmm;"TRIGGER:COUNT 5"
90    OUTPUT @Dmm;"READ?"
100   ENTER @Dmm;Volt(*)
110   PRINT Volt(*);
120   END
```

5

To Initiate Measurements (HP-IB)

To take a reading at a specific time, use the TRIGGER keyword to initiate the measurement sequence. The TRIGGER keyword sends an IEEE-488 GET (Group Execute Trigger) command. First, set the instrument to accept triggering from the bus (refer to the instrument's manuals for the correct commands). Then send TRIGGER to the instrument and collect the readings.

In the following example, the function generator (HP 3314) is triggered by HP-IB and sweeps from 1 KHz to 10 KHz. The oscilloscope (HP 54501A) is triggered by the HP 3314 and displays the original signal as well as the signal directed through the DUT. Figure 5-1 shows the test layout.

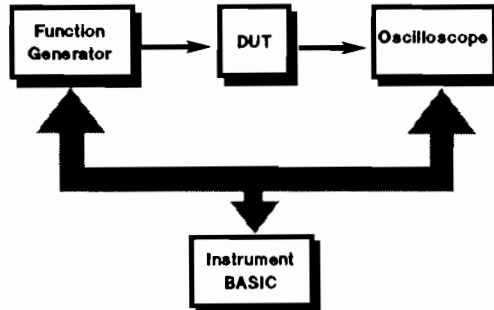


Figure 5-1. Example of Triggering Measurements

```

10  ASSIGN @Fgen TO 709
20  ASSIGN @Scope TO 722
30  CLEAR @Fgen
40  CLEAR @Scope
50  OUTPUT @Fgen;"SR2" ! Set bus trigger
60  OUTPUT @Fgen;"SW1" ! Set linear sweep
70  OUTPUT @Fgen;"ST 1KZ" ! Start Freq = 1 KHz
80  OUTPUT @Fgen;"SP 10KZ" ! Stop Freq = 10 KHz
90  OUTPUT @Fgen;"TI 1S" ! Trigger interval = 1 Sec.
100 OUTPUT @Scope;"TRIGGER:SOURCE CHANNEL1"
110 ! Send GET to the Function Generator
120 TRIGGER @Fgen
130 END
  
```

To Check Instrument Status (HP-IB)

Check instrument status to see if the measurement is completed or if any errors occurred. To read an instrument's status byte, use the SPOLL keyword. Although each instrument's status byte generally contains different information, bit 6 (RQS) is always set if the instrument has requested service.

When you use SPOLL to read the status byte, the status byte is cleared.

You generally must unmask (or enable) a status byte for it to contain status information. You may have to unmask other status groups so that they send status information to the status byte.

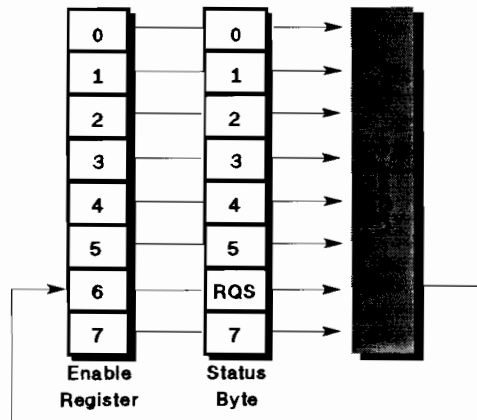


Figure 5-2. Unmasking the Status Byte

The following example takes a SPOLL of a digital multimeter (HP 33401A). The program enables bits 4, 5, and 6 of the status byte. When bit 6 (RQS) is set, the program checks each of the other bits and prints out the condition of the instrument. This example uses SCPI commands.

```

10    OPTION BASE 1
20    ASSIGN @Dmm TO 722
30    OUTPUT @Dmm;"*RST"
40    OUTPUT @Dmm;"*CLS"
50    OUTPUT @Dmm;"*SRE 0"
60    OUTPUT @Dmm;"*ERE 0"
70    OUTPUT @Dmm;"STATUS:PRESET"
80    ! Unmask status byte - bits 4, 5, and 6
90    OUTPUT @Dmm;"*SRE 112"
100   ! Unmask standard event byte for all events
110   OUTPUT @Dmm;"*ESE 255"
120   ! Ask for reading
130   OUTPUT @Dmm;"READ?"
140   REPEAT
150       Status=SPOLL(@Dmm)
160   UNTIL BIT(Status,6)=1 ! Until SRQ is set
170   !
180   IF BIT(Status,4)=1 THEN
190       PRINT "Messages Available"
200       ENTER @Dmm;Volts
210       PRINT Volts;
220   END IF
230   IF BIT(Status,5)=1 THEN PRINT "Standard Event"
240   END

```

To Service HP-IB Interrupts

HP-IB instruments can generate an interrupt when certain conditions, such as when errors are generated or a measurement is complete. You must configure the instrument to generate an SRQ when such a condition occurs; the conditions are specific to each instrument.

To allow your program to respond to an interrupt, you must set up and enable the interrupt. Do this with the ON INTR and the ENABLE INTR keywords. You must unmask bit 2 of the HP-IB interface to allow it to accept interrupts. If the status byte of your instrument is masked, you must unmask it. Note that if you have multiple instruments on the HP-IB, you need to SPOLL each of them to find which instrument generated the interrupt.

Refer to the instrument's manual for details on unmasking the status byte.

The following program sets up and enables an interrupt. The HP 33401A digital multimeter is then asked to take a measurement. When the measurement is complete, an interrupt is generated. The program then branches to a subroutine that reads the data from the multimeter. You re-ENABLE the interrupt after you've serviced it so that additional interrupts can be serviced. This example uses SCPI commands.

```

10  OPTION BASE 1
20  DIM Volts(10)
30  ASSIGN @Dmm TO 722
40  CLEAR @Dmm
50  OUTPUT @Dmm;"*RST"
60  OUTPUT @Dmm;"*CLS"
70  OUTPUT @Dmm;"*SRE 0"
80  OUTPUT @Dmm;"*ESE 0"
90  OUTPUT @Dmm;"STATUS:PRESET"
100 Hpib=7
110 Mask=2
120 ON INTR Hpib GOSUB Take_read
130 ENABLE INTR Hpib;Mask
140 ! Unmask status byte
150 OUTPUT @Dmm;"*SRE 80"
160 ! Ask for measurements
170 OUTPUT @Dmm;"TRIGGER:COUNT 10"
180 OUTPUT @Dmm;"READ?"
190 LOOP
200 ! Execute other tasks while waiting for data
210 END LOOP
220 Take_read: ! Take the readings
230 ENTER @Dmm;Volts(*)
240 ! Re-enable interrupt using previous mask
250 ENABLE INTR Hpib
260 PRINT Volts(*);
270 END

```

For more information about event-initiated branching, refer to "To Use Conditional and Event-Initiated Branching" in Chapter 3.

To Close I/O Paths

When you have finished communicating with an instrument, you close the I/O path by using the **ASSIGN** keyword. For example:

```
ASSIGN @Scope T0 *
```

The I/O path closes when the I/O path name is closed or reassigned.

I/O paths are automatically closed when the context in which they are created, terminates. A subprogram or function context terminates when it ends and control returns to the main program. The main program context terminates when the program stops.

Summary

Here are the keywords that were discussed in this chapter and some keywords that are closely related to the topics covered:

Related Keywords

Topic	Related Keywords	Description
Assigning an I/O Path	ASSIGN	Opens a communication path to an instrument.
	ASSIGN ... SWAP ON OFF	Specifies if bit order is reversed.
	ASSIGN ... FORMAT ON OFF	Specifies binary or text transfer format.
Initializing HP-IB Instruments	CLEAR	Sets the instrument to a instrument-specific, pre-determined state.
Configuring RS-232 Interface	CONTROL	Sets the parity, baud rate, stop bits, and bits/character.
Writing to Interface Registers	CONTROL	Sends control information to interface registers.
Reading RS-232 Configuration	STATUS	Reads the current values of parity, baud rate, stop bits, and bit/character.
Reading Interface Registers	STATUS	Reads the contents of interface of I/O path name status registers.
Sending Commands	OUTPUT	Sends commands as strings.
Reading Measurements	ENTER	Inputs values into variables.
Initiating Measurements (HP-IB)	TRIGGER	Starts the measuring process.
Checking Instrument Status (HP-IB)	SPOLL	Checks and clears the status byte.



Related Keywords (continued)

Topic	Related Keywords	Description
Servicing Interrupts (HP-IB)	ENABLE INTR	Enables the specified interface to generate an interrupt which can cause event-initiated branches.
	ON INTR	Defines an event-initiated branch to be taken when an interface card generates an interrupt.
Closing I/O Paths	ASSIGN @name TO *	Closes the communication path to the instrument.

For information about STATUS and CONTROL registers for both RS-232 and HP-IB interfaces, refer to **STATUS and CONTROL Registers** under **Programming Topics** in online help.

For comprehensive instrument examples, refer to **Controlling Instruments** under **Example Programs** in online help.

Interacting with Users and Displaying Graphs

This chapter introduces you to the basics of using dialogs and widgets. You generally use dialogs and widgets to interact with users and to display graphs. To create a control panel with multiple widgets, refer to Chapter 7.

This chapter initially explains general guidelines for working with dialogs and widgets, and then shows how widgets and dialogs are used to perform specific tasks.

The topics explained in this chapter include:

- Using dialog boxes.
- Using widgets.
- Setting dialog and widget attributes.
- Getting input.
- Displaying text.
- Creating X-Y plots.
- Creating strip charts.
- Printing graphical data.

To Use Dialog Boxes

Use the **DIALOG** keyword to create dialog boxes that display messages, ask questions, or offer a choice of selections. There are seven different types of dialog boxes:

- **ERROR** - Displays an error message.
- **FILE** - Allows you to select a file from a list.
- **INFORMATION** - Displays an information string.
- **LIST** - Allows you to select an item from a list.
- **QUESTION** - Asks a yes/no question.
- **STRING** - Allows you to input a string.
- **WARNING** - Displays a warning message.

The following example creates a dialog that displays the status of a process.

```
10  DIALOG "INFORMATION","The process has completed."  
20  END
```

When this statement is executed, the following appears on your screen:

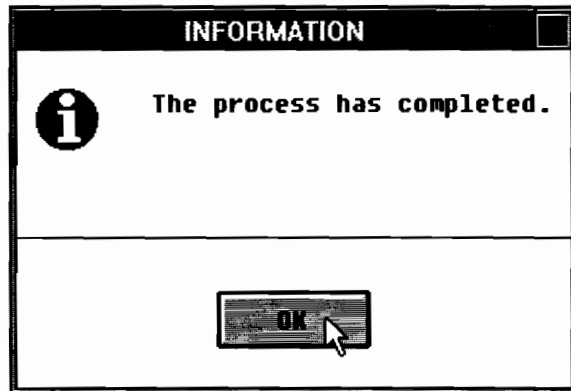


Figure 6-1. An INFORMATION Dialog

For details about dialog boxes, refer to the **DIALOG** keyword in the **Keyword Dictionary** or a dialog name under the **Widget Dictionary** in online help.

To Change Dialog Attributes

To customize the appearance of dialog boxes, use the **SET** option in the **DIALOG** statement to modify attributes such as button labels, title, position, and size. Position and size are specified in pixels.

Dialog boxes are positioned in relationship to the upper left corner of your screen. By default, dialog boxes appear in the center of the screen.

The following example displays an **INFORMATION** dialog box with the **DIALOG BUTTONS** and **TITLE** attributes changed:

```
10  OPTION BASE 1
20  DIM Str$(30)
30  DIM Button$(1)(30)
40  Str$="The process has completed."
50  Button$(1)="Continue"
60  DIALOG "INFORMATION",Str$;SET("DIALOG BUTTONS":Button$(*), "TITLE":"Status")
70  END
```

Note that dialogs and their associated attributes must fit on one line. The maximum line length in HP Instrument BASIC is 256 characters.

The dialog looks like:

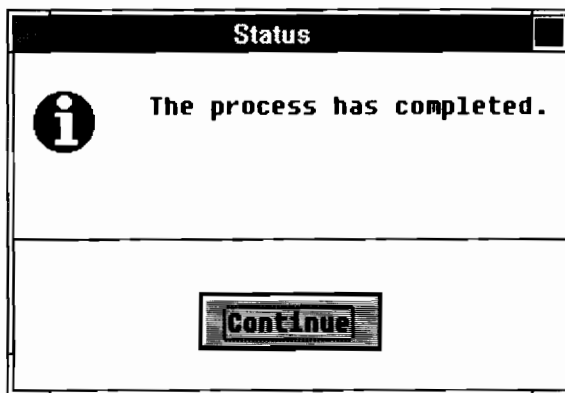


Figure 6-2. A Modified INFORMATION Dialog

To Read Dialog Buttons

The **DIALOG** keyword returns a number that indicates which button was pressed. For example:

```
DIALOG "QUESTION", "Yes or no", Button
```

The **Button** variable holds 0 if the **Yes** button is pressed and 1 if the **No** button is pressed.

The following example uses a QUESTION dialog that reads the button pressed. If no button is pressed in five seconds, Button equals -1 (timeout).

```
10 REAL Button
20 DIM String$(30)
30 String$="Do you want to stop?"
40 DIALOG "QUESTION",String$,Button;TIMEOUT 5
50 SELECT Button
60 CASE =0 ! "Yes" button pressed
70 STOP
80 CASE =1 ! "No" button pressed
90 String$="Program continuing"
100 DIALOG "INFORMATION",String$;TIMEOUT 2
101 GOTO 30
110 CASE =-1 ! Neither button pressed - timeout
120 String$="Timeout"
130 DIALOG "INFORMATION",String$;TIMEOUT 2
140 END SELECT
150 END
```

The dialog looks like:

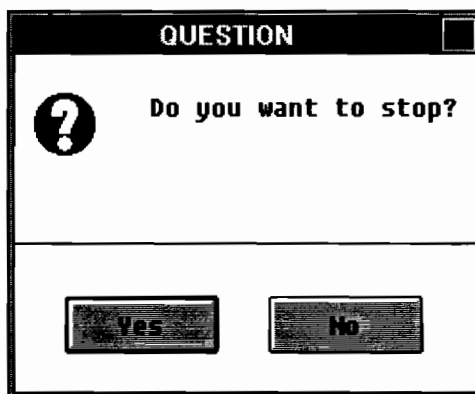


Figure 6-3. A QUESTION Dialog

To Use Widgets

Widgets are the graphical elements that accept input and display output. Widgets don't pop-up like dialog boxes; they remain on the screen until you delete them or the program ends.

To create a widget, use the ASSIGN keyword. For example:

```
ASSIGN OMeter TO WIDGET "METER"
```

The following example creates an analog meter:

```
10  ASSIGN OMeter TO WIDGET "METER"  
20  LOOP  
30  END LOOP  
40  END
```

Notice that widgets do not wait for input like the INPUT or DIALOG keywords do. You must create a loop or use the PAUSE keyword to keep the program running to display widgets.

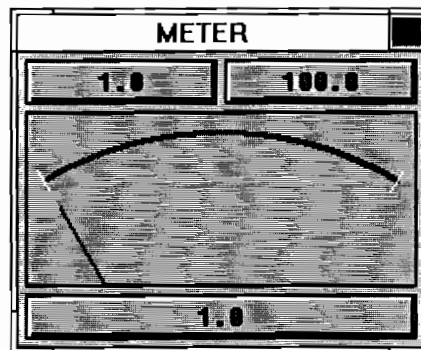


Figure 6-4. A METER Widget

Widgets are positioned within the graph window and are offset from the upper left corner of that window (dialogs are offset from the upper left corner of the screen). If widgets are contained on a panel, their position is

offset from the upper-left corner of the panel. For more information about panels, refer to Chapter 8.

For information about programmatically reading or changing the size, position, or other aspects of the graph window, refer to GESCAPE under the **Keyword Dictionary** on online help.

To delete a widget from the graph window, use the ASSIGN keyword:

```
ASSIGN @Meter TO *.
```

Note



Because widgets use the ASSIGN statement, they follow the same rules as other I/O paths. Therefore, widgets are deleted automatically when the context in which they are created, terminates. A subprogram or function context terminates when it ends and control returns to the main program. The main program context terminates when the program stops.

To Change Widget Attributes

To change the initial attributes of a widget, use the SET option in the ASSIGN statement.

The following example changes the title, and minimum and maximum values of the METER widget.

```
10  ASSIGN @Meter TO WIDGET "METER";SET("TITLE":"Volts","MINIMUM":25,"MAXIMUM":75)
20  LOOP
30  END LOOP
40  END
```

The modified meter looks like:

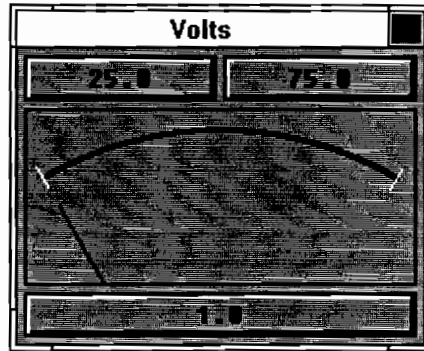


Figure 6-5. A Modified METER Widget

The following example shows how to use CONTROL statements to modify a METER widget to have the same appearance as in Figure 6-5. Unlike dialog attributes, you can set widget attributes on multiple lines. Notice the VISIBLE attribute that prevents the meter from appearing until the attributes are set:

```
10  ASSIGN @Meter TO WIDGET "METER";SET("VISIBLE":0)
20  CONTROL @Meter;SET("TITLE":"Volts")
30  CONTROL @Meter;SET("MINIMUM":25,"MAXIMUM":75)
40  CONTROL @Meter;SET("VISIBLE":1)
50  LOOP
60  END LOOP
70  END
```

Unless you specify otherwise, all widgets can be moved, sized, and maximized by the user when the widgets are displayed. To prevent the user from performing these actions, use the MOVABLE, MAXIMIZABLE, and RESIZABLE attributes.

Note



An easy way to experiment with different attributes is to use the command line in the alpha window. You can execute CONTROL statements while your program is running, and immediately see the effects.

To Read Current Widget Attributes

To query for the current attribute values, such as the current value displayed, use the STATUS keyword.

The following example reads the value of the slider and changes the slider background color if the value is between 40 and 60.

```
10    ASSIGN @Slider TO WIDGET "SLIDER"  
20    CONTROL @Slider;SET("HEIGHT":200)  
30    CONTROL @Slider;SET("BACKGROUND":6)  
40    LOOP  
50        STATUS @Slider;RETURN("VALUE":Value)  
60        IF Value>40 AND Value<60 THEN  
70            CONTROL @Slider;SET("BACKGROUND":7)  
80        ELSE  
90            CONTROL @Slider;SET("BACKGROUND":6)  
100       END IF  
110    END LOOP  
120    END
```

The slider looks like:

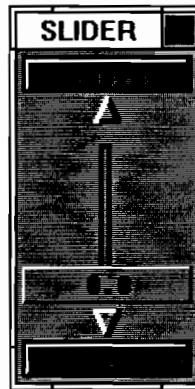


Figure 6-6. A SLIDER Widget

To Respond to Events

To respond to widget events, use the ON EVENT keyword. For example:

```
ON EVENT @Slider; "CHANGED" GOSUB Change
```

Each widget generates different events. For specific information, refer to individual widget entries in the Widget Dictionary in online help.

The following example performs the same actions as the previous example, but checks the value of the slider only when the value has changed:

```
10  ASSIGN @Slider TO WIDGET "SLIDER"
20  CONTROL @Slider;SET("HEIGHT":200)
30  CONTROL @Slider;SET("BACKGROUND":6)
40  ON EVENT @Slider,"CHANGED" GOSUB Change
50  LOOP
60  END LOOP
70  Change:
80  STATUS @Slider;RETURN("VALUE":Value)
90  IF Value>40 AND Value<60 THEN
100  CONTROL @Slider;SET("BACKGROUND":7)
110  ELSE
120  CONTROL @Slider;SET("BACKGROUND":6)
130  END IF
140  RETURN
150  END
```

To Change Fonts

You can adjust the font size on widgets that display text, such as LABEL, LIST, and PRINTER, and widgets that display numerical data, such as METER and SLIDER.

You change the fonts used on a widget by changing the FONTS attribute. The default font is a non-proportional font that depends on your display type. The default font is provided anytime you ask for a 9x15 font. If you ask for a specific font size, Windows creates a font as close as possible to the font requested. You should experiment with different font sizes for your display.

For a complete list of supported fonts, refer to **Available Fonts** under the **FONTS** attribute listing for widgets with changable fonts (such as **LABEL**, **LIST**, or **PRINTER**) in online help.

The size and the font of a text display widget are interdependent attributes. For example, if you set **WIDTH**, **HEIGHT**, and **FONT**, **ROWS** and **COLUMNS** are automatically calculated to fit. If you set **ROWS**, **COLUMNS**, and **FONT**, **WIDTH** and **HEIGHT** are automatically calculated to fit.

For more information on these interdependencies, refer to **Details about interdependent attributes** under certain widget topics (such as **LABEL**, **LIST**, or **PRINTER**) in online help.

To Prompt for Keyboard Input

An easy way to prompt for user input is with the **STRING** dialog. This dialog box pops up and prompts for a response.

6

The following example prompts for a password. If Cancel is pressed, the program stops; if OK is pressed, the password is checked. If no response is made in six seconds, the dialog times out.

```
10    Current$="Dirk"
20    DIM Str$(20)
30    Str$="Enter your password"
40    DIALOG "STRING",Str$,Button;TIMEOUT 6,RETURN("VALUE":Passwd$)
50    SELECT Button
60    CASE =-1
70        STOP
80    CASE =1
90        STOP
100   CASE =0
110       IF Passwd$=Current$ THEN
120           DIALOG "INFORMATION","Password OK";TIMEOUT 2
130       ELSE
140           DIALOG "INFORMATION","Password Invalid";TIMEOUT 2
150       END IF
160   END SELECT
170   END
```

The dialog looks like:

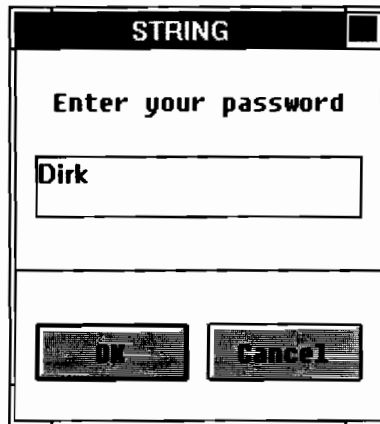


Figure 6-7. A STRING Dialog

Another way to prompt for user input is with the STRING widget.

The following example prompts for a value and stops the program when **Enter** is pressed.

```
10    DIM Title$[30]
20    Title$="Enter the minimum value"
30    ASSIGN @Input TO WIDGET "STRING";SET("TITLE":Title$)
40    ON EVENT @Input,"RETURN" GOTO End
50    LOOP
60    END LOOP
70 End:  !
80    STATUS @Input;RETURN("VALUE":Min$)
90    GRAPHICS OFF
100   PRINT "Minimum value is: ";Min$
110   END
```

The widget looks like:

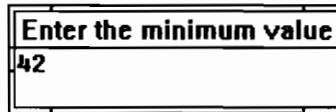


Figure 6-8. A STRING Widget

To Display Text

Use the LABEL widget to print one or more lines to a fixed-size area.

The following example uses a LABEL widget to display the test status and then display "Done".

```
10    DIM Status$[30]
20    ASSIGN @Label TO WIDGET "LABEL";SET("TITLE":"Test Status")
30    CONTROL @Label;SET("COLUMNS":30)
40    FOR I=0 TO 90 STEP 10
50        Status$="Running - "&VAL$(I)&"% Complete"
60        CONTROL @Label;SET("VALUE":Status$)
70        WAIT 1
80    NEXT I
90    Status$="Done"
100   CONTROL @Label;SET("VALUE":Status$)
110   WAIT 2
120   GRAPHICS OFF
130   END
```

The label looks like:

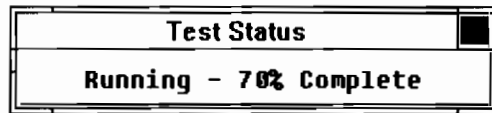


Figure 6-9. A LABEL Widget

Other widgets that display text are PRINTER and LIST. For details about their use, refer to the **Widget Dictionary** in online help.

To Graph Data

HP Instrument BASIC allows you to easily plot arrays of data. Graphing attributes let you specify such values as axis titles, trace labels, and scaling. There are three types of graphing attributes:

- Global (entire widget) attributes. For example, those that specify title, size, and placement.
- Axis-specific attributes. For example, those that specify axis titles, autoscaling, and scale (log or linear). These attributes apply to the axis specified in the last CURRENT AXIS attribute statement.
- Trace-specific attributes. For example, those that specify trace color, trace labels, point symbol. These attributes apply to the trace specified in the last CURRENT TRACE attribute statement.

To Create X-Y Plots

Use the XY GRAPH widget to plot an x versus y graph.

The following example creates a simple plot of a sine wave. The POINT CAPACITY attribute is set to the number of points plotted.

```
10    DIM Xdata(1:180),Ydata(1:180)
20    DEG
30    Num_points=180
40    ! Create Data
50    FOR I=1 TO Num_points
60        Xdata(I)=I*2
70        Ydata(I)=SIN(I*2)
80    NEXT I
90    ! Create XY GRAPH
100   ASSIGN @Plot TO WIDGET "XY GRAPH";SET("VISIBLE":0)
110   ! Set trace attributes
120   CONTROL @Plot;SET("POINT CAPACITY":Num_points)
130   CONTROL @Plot;SET("Y DATA":Ydata(*))
140   CONTROL @Plot;SET("X DATA":Xdata(*))
150   ! Set axis to autoscale
160   CONTROL @Plot;SET("CURRENT AXIS":"X")
170   CONTROL @Plot;SET("AUTOSCALE":1)
180   CONTROL @Plot;SET("CURRENT AXIS":"Y")
190   CONTROL @Plot;SET("AUTOSCALE":1)
200   ! Display graph
210   CONTROL @Plot;SET("VISIBLE":1)
220   LOOP
230   END LOOP
240   END
```

The graph looks like:

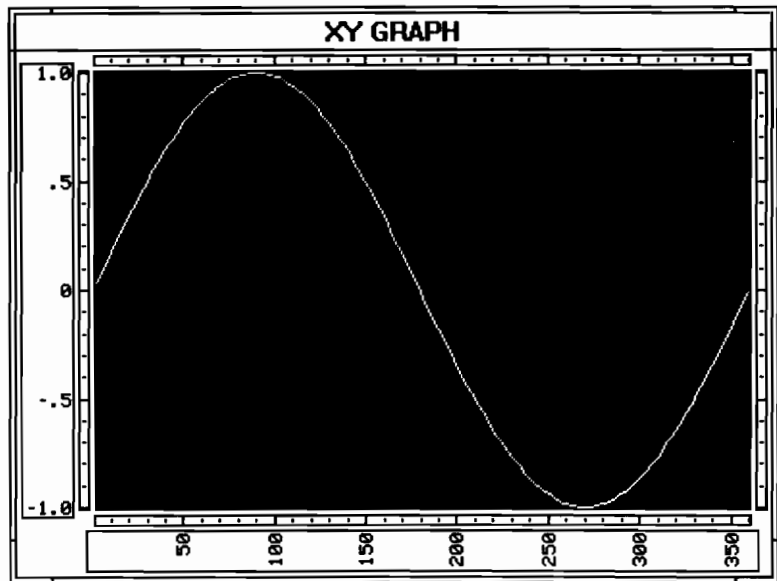


Figure 6-10. An XY GRAPH Widget

6

The following example plots two traces ($\sin(x)$ and $\cos(x)$). The TRACE COUNT attribute specifies the number of traces. The POINT CAPACITY attribute for each trace is set to 180. Note that the POINT CAPACITY must be set before values are assigned with the X DATA and Y DATA attributes.

Both the x- and y-axes are autoscaled. The SHARED X attribute makes both traces use the x data assigned to trace 1. The x and y data points are plotted with the X DATA and Y DATA attributes.

```

10    DIM Xdata(1:180),Y1data(1:180),Y2data(1:180)
20    DEG
30    Num_points=180
40    ! Create Data
50    FOR I=1 TO Num_points
60        Xdata(I)=I*2
70        Y1data(I)=SIN(I*2)
80        Y2data(I)=COS(I*2)
90    NEXT I
100   ! Create XY GRAPH
110   ASSIGN @Plot TO WIDGET "XY GRAPH";SET("VISIBLE":0)
120   ! Set X Axis Attributes
130   CONTROL @Plot;SET("CURRENT AXIS":"X","AXIS LABEL":"Degrees")
140   CONTROL @Plot;SET("AUTOSCALE":1)
150   ! Set Trace 1 Attributes
160   CONTROL @Plot;SET("CURRENT TRACE":1,"TRACE LABEL":"Sin(x)")
170   CONTROL @Plot;SET("POINT CAPACITY":Num_points)
180   CONTROL @Plot;SET("Y DATA":Y1data(*))
190   CONTROL @Plot;SET("SHARED X":1,"TRACE COUNT":2)
200   CONTROL @Plot;SET("X DATA":Xdata(*))
210   ! Set Trace 2 Attributes
220   CONTROL @Plot;SET("CURRENT TRACE":2,"TRACE LABEL":"Cos(x)")
230   CONTROL @Plot;SET("POINT CAPACITY":Num_points)
240   CONTROL @Plot;SET("Y DATA":Y2data(*))
250   ! Set Y AXIS Attributes
260   CONTROL @Plot;SET("CURRENT AXIS":"Y")
270   CONTROL @Plot;SET("ORIGIN":-1)
280   CONTROL @Plot;SET("RANGE":2)
290   ! Make graph visible
300   CONTROL @Plot;SET("VISIBLE":1)
310   LOOP
320   END LOOP
330   END

```

The following example shows a simple stripchart that plots a sine wave.

```
10  OPTION BASE 1
20  DIM Ydata(1)
30  DEG
40  Num_points=180
50  ! Create STRIP CHART
60  ASSIGN @Strip TO WIDGET "STRIPCHART";SET("VISIBLE":0)
70  ! Set X axis attributes
80  CONTROL @Strip;SET("CURRENT AXIS":"X")
90  CONTROL @Strip;SET("RANGE":360)
100 ! Set Y axis attributes
110 CONTROL @Strip;SET("CURRENT AXIS":"Y")
120 CONTROL @Strip;SET("AUTOSCALE":1)
130 ! Show the stripchart
140 CONTROL @Strip;SET("VISIBLE":1)
150 ! Set x and y values
160 I=0
170 LOOP
180   Ydata(1)=SIN(I)
190   CONTROL @Strip;SET("POINT LOCATION":I)
200   CONTROL @Strip;SET("VALUES":Ydata(*))
210   I=I+1
220 END LOOP
230 END
```

The graph looks like:

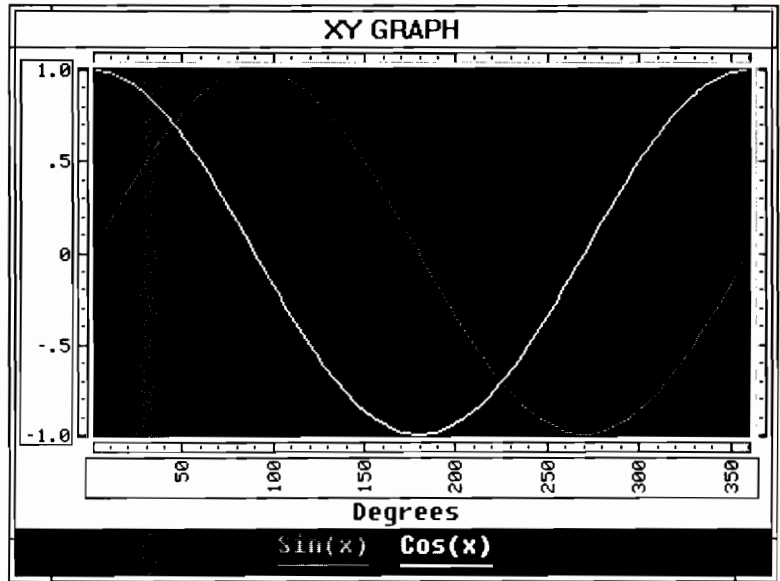


Figure 6-11. An X-Y Graph with Two Traces

To Create Strip Charts

Use the STRIPCHART widget to plot data on a “moving” graph. The STRIPCHART acts like an old-fashioned chart recorder that plots on an unwinding roll of paper.

You use the STRIPCHART essentially the same way you use the XY GRAPH, except you plot the point at each x position separately, instead of plotting all points at once.

You do this by putting the y position of each point in an array that is constantly updated. The array has as many dimensions as you have traces. Set the y position of the points with the VALUES attribute, set the x position with the POINT LOCATION attribute.

The strip chart looks like:

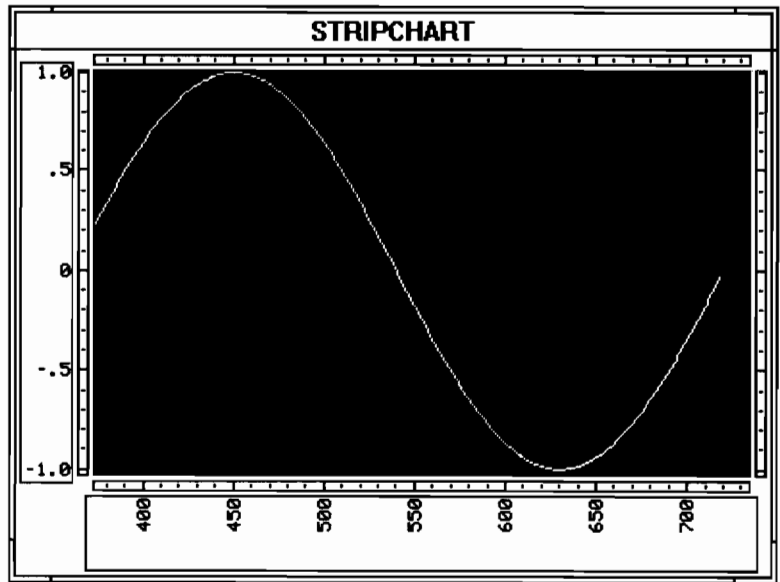


Figure 6-12. A STRIPCHART Widget

The following example is more complicated and shows a stripchart that plots two traces ($\sin(x)$ and $\cos(x)$). Both traces share the same x data.

```

10     OPTION BASE 1
20     DIM Ydata(2)
30     DEG
40     Num_points=180
50 ! Create STRIP CHART
60     ASSIGN @Strip TO WIDGET "STRIPCHART";SET("VISIBLE":0)
70     CONTROL @Strip;SET("SHARED X":1,"TRACE COUNT":2)
80     CONTROL @Strip;SET("MINIMUM SCROLL":10)
90 ! Set X Axis Attributes
100    CONTROL @Strip;SET("CURRENT AXIS":"X","AXIS LABEL":"Degrees")
110    CONTROL @Strip;SET("RANGE":360)
120 ! Set Y Axis Attributes
130    CONTROL @Strip;SET("CURRENT AXIS":"Y")
140    CONTROL @Strip;SET("RANGE":2)
150    CONTROL @Strip;SET("ORIGIN":-1)
160 ! Set Trace 1 Attributes
170    CONTROL @Strip;SET("CURRENT TRACE":1,"TRACE LABEL":"Sin(x)")
180 ! Set Trace 2 Attributes
190    CONTROL @Strip;SET("CURRENT TRACE":2,"TRACE LABEL":"Cos(x)")
200    CONTROL @Strip;SET("VISIBLE":1)
210 ! Set x and y values
220    I=0
230    LOOP
240        Ydata(1)=SIN(I)
250        Ydata(2)=COS(I)
260        CONTROL @Strip;SET("POINT LOCATION":I)
270        CONTROL @Strip;SET("VALUES":Ydata(*))
280        I=I+1
290    END LOOP
300    END

```

The strip chart looks like this:

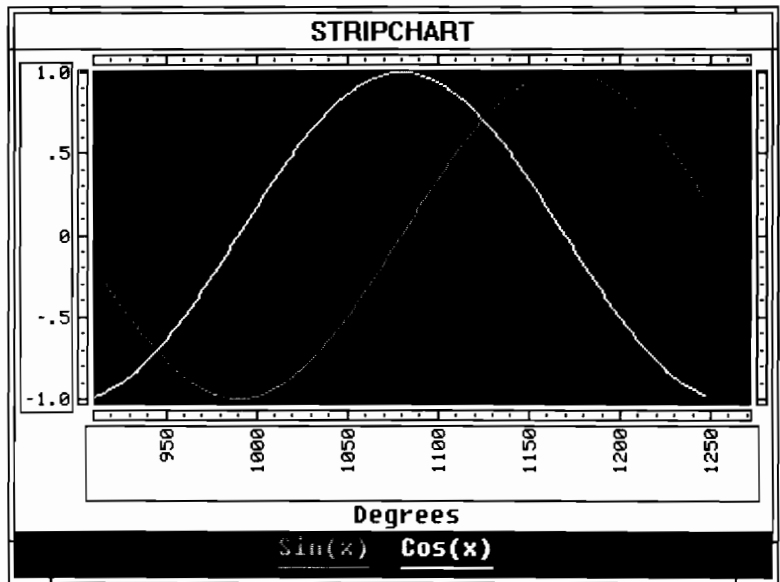


Figure 6-13. A Strip Chart with Two Traces

6

To Print Widgets

Because widgets are associated with the graph window, you print the graph window to get a printout of widgets on it. There are two ways to print the graph window:

- Use the DUMP GRAPHICS keyword in your program.
- Select the **Print** menu choice from control menu of the graph window.

By default, the width of the graph window fills 60% of the width of a sheet of paper. You change the percentage with GESCAPE 39.

The following example changes the graphic size to 80% of the paper:

```
10 INTEGER Per(0)
20 Per(0)=80
30 GESCPE CRT,39;Per(*)
40 END
```

Summary

Here are the keywords, dialogs, widgets, and attributes discussed in this chapter with other closely-related items.

Dialog Boxes

Dialog Name	Description
ERROR	Displays an error message.
FILE	Prompts the user to select a file from a list.
INFORMATION	Displays an information string.
LIST	Prompts the user to select an item from a list.
QUESTION	Asks a yes/no question.
STRING	Prompts the user to input a string.
WARNING	Displays a warning message.

Related Widgets

Topic	Widget Name	Description
Displaying Data	BAR	Displays a value as a bar.
	BARS	Displays a bar chart.
	METER	Displays a “VU meter”.
	STRIPCHART	Displays a moving two-axis plot.
	XY GRAPH	Displays a two-axis plot.
Displaying Text	LIST	Displays a scrollable list of strings.
	LABEL	Displays a string.
	PRINTER	Displays a block of text.
Getting Input	PUSHBUTTON	Displays a pushbutton.
	RADIOBUTTON	Displays a bank of buttons. Only one button in each bank can be pressed at any one time.
	SLIDER	Allows setting of a value by dragging a slider pad.
	STRING	Accepts a string.
	TOGGLEBUTTON	Allows input or display of data that has a binary state.

Common Widget and Dialog Attributes

Attribute	Of Widgets	Of Dialogs	Description
BACKGROUND	Yes	Yes	Sets the color.
BORDER	Yes	No	Specifies if the border is displayed.
DEFAULT BUTTON	No	Yes	Sets default button.
DIALOG BUTTONS	No	Yes	Sets button names.
HEIGHT	Yes	Yes	Sets the height in pixels.
INSIDE HEIGHT	Yes	Yes	Sets the height of the work area in pixels.
INSIDE WIDTH	Yes	Yes	Sets the width of the work area in pixels.
JUSTIFICATION	No	Yes	Sets the justification of the prompt string.
MAXIMIZABLE	Yes	Yes	Specifies if item can be maximized.
MOVABLE	Yes	Yes	Specifies if item can be moved.
RESIZABLE	Yes	Yes	Specifies if item can be resized.
RESTORE SCREEN	Yes	Yes	Specifies if contents of graphics window is redrawn.
TITLE	Yes	Yes	Sets title string.
STACKING ORDER	Yes	No	Sets the depth in the widget hierarchy being displayed (0 = top).
USER DATA	Yes	Yes	Sets program-specific data.
VISIBLE	Yes	No	Specifies if widget is visible.
WIDTH	Yes	Yes	Sets the width in pixels.
X	Yes	Yes	Sets the horizontal position (in pixels) of the left side of the item.
Y	Yes	Yes	Sets the vertical position (in pixels) of the top of the item.

Related Keywords

Topic	Keyword	Description
Setting a Time Out in a Dialog	TIMEOUT <i>n</i>	Sets a dialog box to time out in <i>n</i> seconds.
Creating Widgets	ASSIGN @ <i>name</i> TO WIDGET <i>widget</i>	Displays widget as it's created.
Deleting Widgets	ASSIGN @ <i>name</i> TO *	Destroys widgets.
Setting Values	CONTROL ... SET	Sets attributes.
Reading Values	STATUS ... RETURN	Reads current attributes.
Modifying Graph Window Options	GESCAPE	Returns or sets such information as size and position of the graph window.
Handling Events	DISABLE EVENT	Temporarily blocks an event.
	ENABLE EVENT	Reverses the effects of DISABLE EVENT.
	OFF EVENT	Deactivates an event-initiated branch.
	ON EVENT	Activates an event-initiated branch.

For complete information about the DIALOG keyword and dialog attributes, refer to the DIALOG keyword in the **Keyword Dictionary** in online help.

For complete information about widgets and their attributes, refer to the **Widget Dictionary** in online help.

For a complete list of GESCAPE functions, refer to the GESCAPE keyword in the **Keyword Dictionary** in online help.



2



Creating Control Panels

This chapter describes tasks used to create an operator interface or control panel. A control panel allows you to group multiple widgets and provides an easy way for others to use your program.

This chapter explains complex widget operations. For information about the basics of using widgets and setting attributes, refer to Chapter 6.

The topics explained in this chapter include:

- Creating panels.
- Creating menus.
- Setting alarms.
- Creating full-screen panels.
- Securing programs.
- Running programs.
- Creating easy-to-use panels.

To Create Panels

A panel is just a way to group a set of widgets. Use the PANEL widget as the base of your control panel and as the parent widget of the other widgets. Widgets can be placed in a hierarchy, therefore a panel can contain other panels or widgets.

Figure 7-1 shows a panel that contains four different widgets.

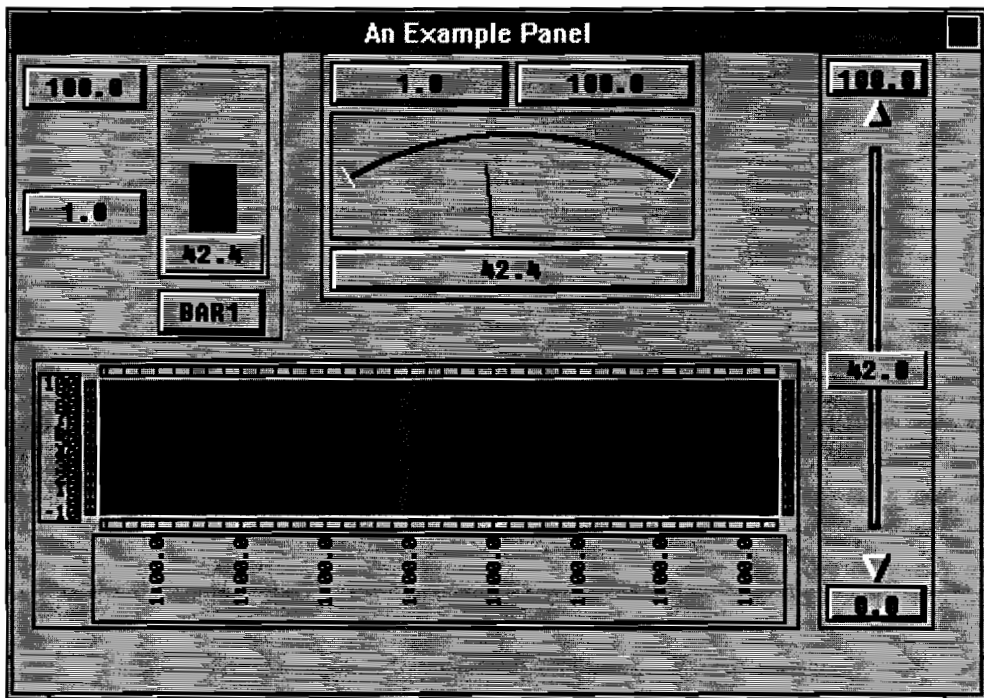


Figure 7-1. An Example Panel

Note that top-level panels are transient (like dialog boxes), they are placed relative to the upper left corner of the screen and their location is not limited to the graph window.

Key Definitions



- **Parent Widget** - A widget that contains other widgets. When the parent widget is moved, all its children move with it. For example, a PANEL widget is a parent widget when it contains other widgets such as stripcharts and menus.
- **Child Widget** - A widget that is contained on another widget. For example, a METER widget is a child when it is contained on a PANEL widget.
- **Level-0 Widget** - A top-level widget that is not a child widget, but may be a parent widget. By default, a level-0 widget has a title bar and resize border. For example, an XY GRAPH widget is a level-0 widget if it is not assigned to a parent widget.

The differences between child widgets and level-0 widgets are:

- A child widget's location is offset from the upper-left corner of the parent widget.
- Child widgets don't have titles.
- Child widgets are not resizable or movable (except from your program).

To create a child widget, use the PARENT option with the ASSIGN keyword:

```
10    ASSIGN @Panel TO WIDGET "PANEL"
20    ASSIGN @Meter TO WIDGET "METER";PARENT @Panel
30    LOOP
40    END LOOP
50    END
```

The child METER widget looks like this:

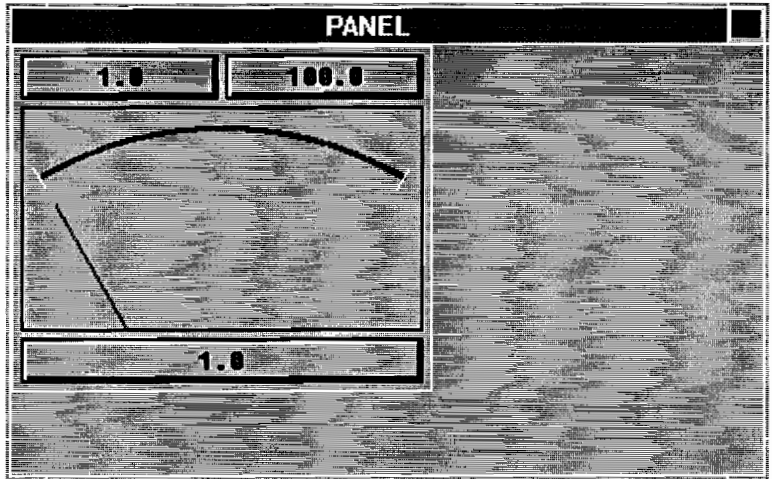


Figure 7-2. A METER Widget on a Panel

Because the meter widget does not have a title, you must use the LABEL widget to title widgets on a panel.

The following example places a title above a METER widget:

```
10    ASSIGN @Panel TO WIDGET "PANEL";SET("VISIBLE":0)
20    ASSIGN @Mlabel TO WIDGET "LABEL";PARENT @Panel,SET("VALUE":"Meter", "X":5)
30    ASSIGN @Meter TO WIDGET "METER";PARENT @Panel, SET("X":5)
40 ! Find the width and position of the meter
50    STATUS @Meter;RETURN("WIDTH":Meterw,"Y":Metry)
60    STATUS @Mlabel;RETURN("HEIGHT":Labelw)
70 ! Set the label to the right width and position
80    CONTROL @Mlabel;SET("WIDTH":Meterw,"Y":Metry-Labelw)
90    CONTROL @Panel;SET("VISIBLE":1)
100   LOOP
110   END LOOP
120   END
```

The panel looks like this:

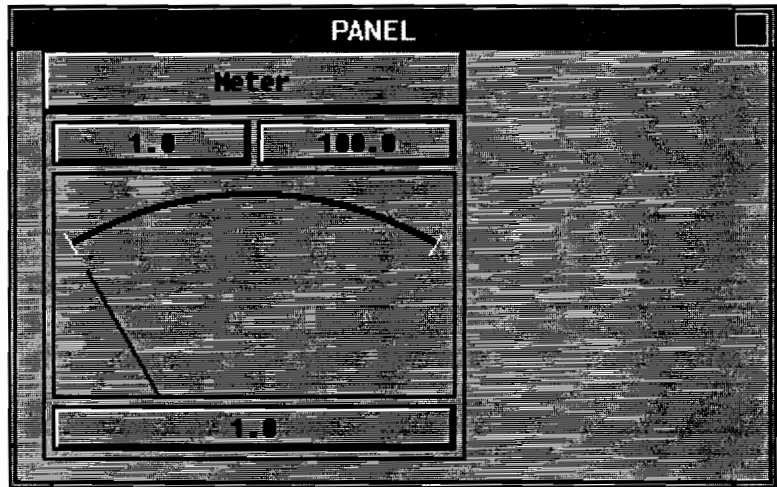


Figure 7-3. A LABEL Widget on a Panel

To Create Menus

To create menus on a panel, use the PULLDOWN MENU widget. Create menu items with the MENU BUTTON widget.

The following example creates a panel that has one menu (called File) with one menu item (Quit) that stops the program when selected.

```
10  ASSIGN @Panel TO WIDGET "PANEL"
20  ASSIGN @Menu TO WIDGET "PULLDOWN MENU";PARENT @Panel,SET("LABEL":"File")
30  ASSIGN @Choice TO WIDGET "MENU BUTTON";PARENT @Menu,SET("LABEL":"Quit")
40  ON EVENT @Choice,"ACTIVATED" GOSUB Done
50  LOOP
60  END LOOP
70 Done: !
80  GRAPHICS OFF
90  END
```

7

The menu looks like:

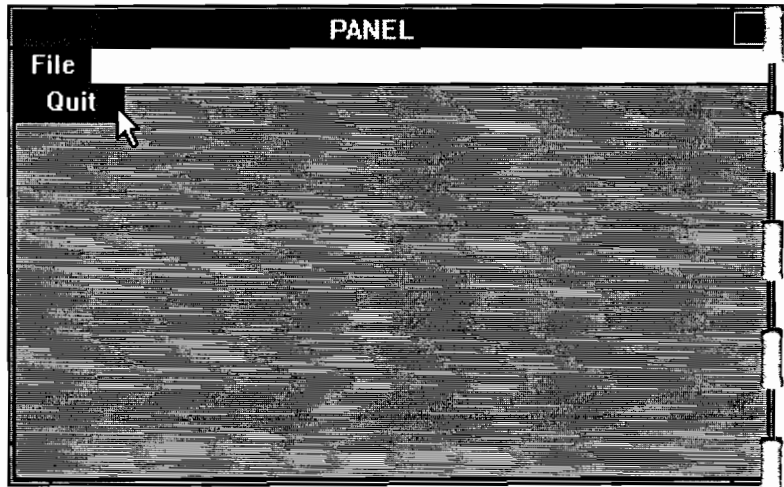


Figure 7-4. A Pull-Down Menu

There are five widgets that you use to make menus and menu items:

- CASCADE MENU - Creates a submenu that cascades off another menu.
- MENU BUTTON - Creates a menu item.
- MENU SEPARATOR - Creates a horizontal line to visually separate menu items.
- MENU TOGGLE - Creates a menu item with a check mark to indicate on or off.
- PULLDOWN MENU - Creates the menu.

Menu items are displayed in the order that they are created.

The following example creates a hierarchy of menu widgets. Note that the Auto Pilot choice is not available; the SENSITIVE attribute is set to 0.

```
10  ASSIGN @Panel TO WIDGET "PANEL";SET("VISIBLE":0)
20  ASSIGN @Menu TO WIDGET "PULLDOWN MENU";PARENT @Panel,SET("LABEL":"File")
30  ASSIGN @Item1 TO WIDGET "MENU BUTTON";PARENT @Menu,SET("LABEL":"About")
40  ASSIGN @Sep TO WIDGET "MENU SEPARATOR";PARENT @Menu
50  ASSIGN @Casc TO WIDGET "CASCADE MENU";PARENT @Menu,SET("LABEL":"Choose")
60  ASSIGN @Tog TO WIDGET "MENU TOGGLE";PARENT @Casc,SET("LABEL":"OK")
70  ASSIGN @Item2 TO WIDGET "MENU BUTTON";PARENT @Menu
80  CONTROL @Item2;SET("LABEL":"Auto Pilot","SENSITIVE":0)
90  ASSIGN @Item3 TO WIDGET "MENU BUTTON";PARENT @Menu,SET("LABEL":"Quit")
100  ON EVENT @Item1,"ACTIVATED" GOSUB About
110  ON EVENT @Item3,"ACTIVATED" GOSUB Done
120  CONTROL @Panel;SET("VISIBLE":1)
130  LOOP
140  END LOOP
150 About: !
160  DIALOG "INFORMATION","This program shows how menu widgets work."
170  RETURN
180 Done: !
190  GRAPHICS OFF
200  END
```

The panel looks like:

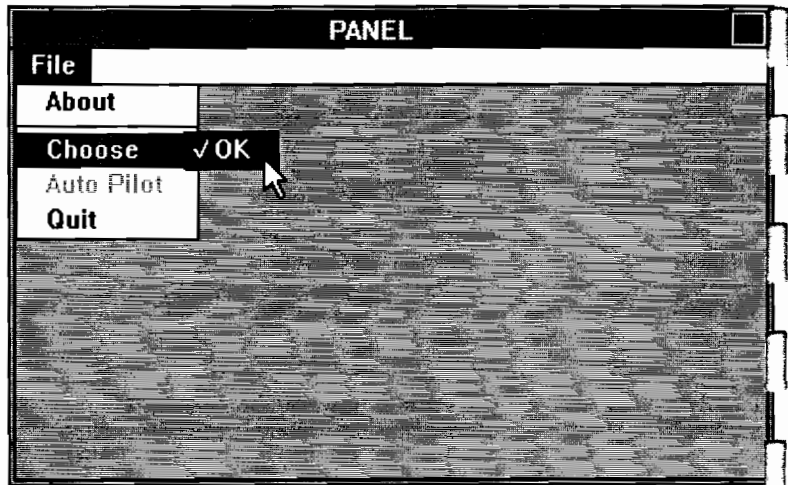


Figure 7-5. Multiple Menu Choices

To Set Alarms

It is often useful to set alarms on a control panel. To set alarms, use widgets that have ALARM events associated with them. These widgets are:

- METER
- BAR
- BARS

To set up alarms, perform the following steps:

1. Set low and/or high limits with the LOW LIMIT or HIGH LIMIT attributes. These limits define the boundaries between the alarm areas.
2. Decide if you want the alarm areas to be displayed in different colors than red (high), yellow (mid-range), and green (low). Use the LOW PEN and HIGH PEN attributes to set the colors.

3. Determine which alarm ranges will cause an alarm to be generated. Set the ALARM RANGES attribute to LOW, MIDDLE, and/or HIGH (you can have multiple alarm ranges).
4. Decide if you want an alarm to generate a BEEP or an EVENT. Set this with the ALARM TYPE attribute.
5. If an alarm generates an EVENT, set up an alarm handling routine.

The following example outputs random numbers to a BARS widget. A BEEP alarm is generated whenever the value is above 75 or below 25.

```
10    ASSIGN @Panel TO WIDGET "PANEL"
20    ASSIGN @Bar TO WIDGET "BARS";PARENT @Panel,SET("HEIGHT":200)
30    CONTROL @Bar;SET("LOW LIMIT":25)
40    CONTROL @Bar;SET("LOW PEN":4)
50    CONTROL @Bar;SET("HIGH LIMIT":75)
60    CONTROL @Bar;SET("HIGH PEN":5)
70    CONTROL @Bar;SET("ALARM RANGES":"LOW,HIGH")
80    CONTROL @Bar;SET("ALARM TYPE":"BEEP")
80    LOOP
90        CONTROL @Bar;SET("VALUE":RND*100)
100    WAIT 1
110    END LOOP
120    END
```

The panel looks like:

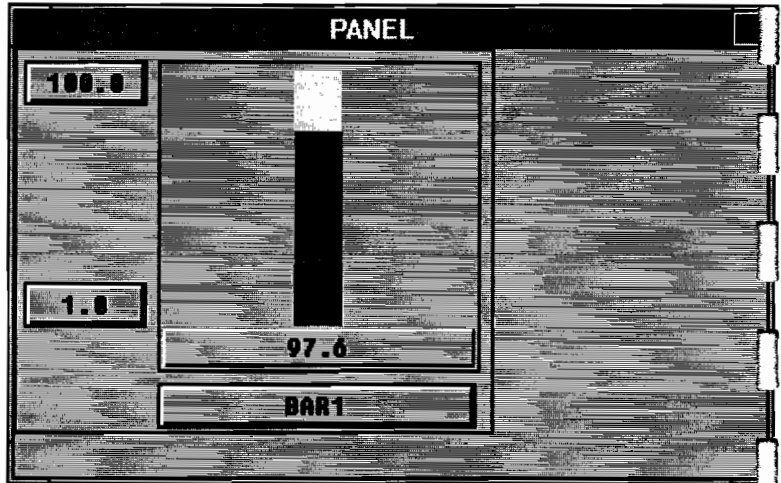


Figure 7-6. A BARS Widget with Alarms

To Create Full-Screen Panels

To create a full-screen panel, use GESCAPE 36 to find the size of your screen. Then create a panel with that size. Some other items to consider when creating a full-screen panel:

- To get a screen dump, maximize the graph window using GESCAPE 30 before printing. For example:

```
GESCAPE CRT, 30
```

Use the GRAPHICS OFF keyword to hide the graph window when the program ends.

- The control pad is displayed on top of the panel when you double-click on the panel (outside of child widgets). You can also use the keyboard to control program execution; press the **Scroll Lock** key to stop

your program and press the **Pause** key to pause. You can type keywords such as **STOP** or **CONT**; they are executed as if they were typed on the command line.

The following example sets up a full-screen panel, displays it for five seconds, and then stops the program:

```
10  INTEGER Screen(1)
20  GESCAPE CRT,36;Screen(*)
30  ASSIGN @Panel TO WIDGET "PANEL";SET("VISIBLE":0)
40  CONTROL @Panel;SET("WIDTH":Screen(0),"HEIGHT":Screen(1))
50 ! Maximize graph window
60  GESCAPE CRT,30
70  CONTROL @Panel;SET("VISIBLE":1)
80  WAIT 5
90  GRAPHICS OFF
100 ASSIGN @Panel TO *
110 END
```



To Secure Programs

Often you create a control panel to protect users who don't know about programming. You may also want to protect your program from user intervention; if you are concerned about users stopping the program and altering the code, you can secure the program.

Caution



Securing is not reversible. Make a copy of the non-secured program before using the **SECURE** keyword.

To secure a program, type **SECURE** on the command line. The lines of the program change to *****. Only the line numbers remain. You run secured programs the same way you run unsecured programs.

To Run Programs

There are several ways to allow users to run a program.

1. Secure the program, and let users **Open** and run it in the same way you did when developing it.
2. Start the program every time the user starts HP Instrument BASIC. Save your program to a file called **AUTOST** in the installation directory (usually **\IBASIC**). Note that there is no file extension on **AUTOST**.
3. Create a Windows icon that runs HP Instrument BASIC and the program. To do this, complete the following tasks:
 - a. Select the **HP Instrument BASIC** group so that the title is highlighted.
 - b. Select **New** from the **Windows Program Manger File** menu. Then choose **Program Item** and press **OK**.
 - c. From the **Program Item Properties** dialog box, type in the description of the program (its name) and fill in the command line.

For example, a command line might be
C:\IBASIC\IBASIC.EXE PROG.IBW.

4. Start Windows, HP Instrument BASIC, and a program all at once. At the MS-DOS prompt, while in the HP Instrument BASIC installation directory, type **WIN IBASIC** and the program name. For example:

C:\IBASIC> WIN IBASIC C:\LOOP.IBW

To Create Easy-To-Use Control Panels

Here are a few design guidelines to make control panels easier to use:

- Use generally accepted color associations (red denotes warning or danger, green denotes safe or normal).
- Don't rely on color alone; make sure you have other mechanisms to inform your users. For example, if a level is approaching a critical level, you may want to show the level, have the display turn red, and have the program beep. The redundancy makes cues easier to follow and considers the needs of the color-blind.
- Make sure there is enough color contrast so that information is clear when viewed by color-blind users or when printed.
- Make important or frequently-referenced objects bigger; for example, make a **Emergency Shutdown** button very large.
- Make sure that text is large enough to be easily seen under normal operating conditions.
- Position related objects together. Use a panel to graphically show that items are related. For example, position a graph and its numerical output close together. Or position a bank of meters together (if they are used as a group).
- Give the operator a summary area that contains a status line or some other quick indicator about the test status. Don't require the operator to check the status displayed on each widget.
- Avoid clutter. Only show information that is useful at each time. For example, use one part of the screen to show different information for each part of the test.

- Test your design. There is no perfect layout, but some are better than others. To make your control panel better, create a panel and have some of your users try it out for several days. They'll let you know how to make the layout better. The better the layout, the fewer mistakes and misunderstandings will occur.

Example

The following example shows an engine monitor control panel.

```

10  DIM Buf$(200)
20  ! Create the main panel widget
30  !
40  ASSIGN @Panel TO WIDGET "PANEL";SET("X":5,"Y":5,"WIDTH":450,"HEIGHT":340,
    "TITLE":"Engine Monitor","MAXIMIZABLE":1)
50  !
60  ASSIGN @M1lbl TO WIDGET "LABEL";SET("X":5,"Y":5,"COLUMNS":15,"HEIGHT":20,
    "VALUE":"Torque (ft/lbs)","BORDER":0),PARENT @Panel
70  ASSIGN @Meter1 TO WIDGET "METER";SET("X":20,"Y":25,"WIDTH":90,"HEIGHT":160,
    "ARC WIDTH":3,"ORIENTATION":"RIGHT","SHOW LIMITS":0),PARENT @Panel
80  CONTROL @Meter1;SET("MAXIMUM":300)
90  !
100 ASSIGN @B2lbl TO WIDGET "LABEL";SET("X":145,"Y":5,"COLUMNS":14,"HEIGHT":20,
    "VALUE":"Pressure (psi)","BORDER":0),PARENT @Panel
110 ASSIGN @Bar2 TO WIDGET "BAR";SET("X":165,"Y":25,"WIDTH":60,"HEIGHT":100),
    PARENT @Panel
120 CONTROL @Bar2;SET("MAXIMUM":200)
130 ASSIGN @B2dsp TO WIDGET "LABEL";SET("X":165,"Y":135,"WIDTH":60,"HEIGHT":20,
    "BORDER":0),PARENT @Panel
140 !
150 ! Create a child PANEL that contains a bank of engine measurements
160 ASSIGN @Subpanel TO WIDGET "PANEL";PARENT @Panel,SET("X":280,"Y":5,
    "WIDTH":140,"HEIGHT":145)
170 !
180 ASSIGN @Lb1 TO WIDGET "LABEL";PARENT @Subpanel,SET("X":10,"Y":0,"WIDTH":120,
    "HEIGHT":20,"BORDER":0,"VALUE":"Dwell (deg)")
190 ASSIGN @Disp1 TO WIDGET "LABEL";PARENT @Subpanel,SET("X":10,"Y":20,"WIDTH":120,
    "HEIGHT":20)
200 !
210 ASSIGN @Label2 TO WIDGET "LABEL";PARENT @Subpanel,SET("X":10,"Y":50,"WIDTH":120,
    "HEIGHT":20,"BORDER":0,"VALUE":"Timing (deg)")
220 ASSIGN @Disp2 TO WIDGET "LABEL";PARENT @Subpanel,SET("X":10,"Y":70,"WIDTH":120,
    "HEIGHT":20)

```

```

230 !
240 ASSIGN @Label3 TO WIDGET "LABEL";PARENT @Subpanel,SET("X":10,"Y":100,"WIDTH":120,
"HEIGHT":20,"BORDER":0,"VALUE":"RPM")
250 ASSIGN @Disp3 TO WIDGET "LABEL";PARENT @Subpanel,SET("X":10,"Y":120,"WIDTH":120,"HEIGHT":20)
260 !
270 ASSIGN @Lbl1 TO WIDGET "LABEL";SET("X":180,"Y":170,"WIDTH":240,"HEIGHT":20,
"VALUE":"Status Log","BORDER":0),PARENT @Panel
280 ASSIGN @Text TO WIDGET "PRINTER";SET("X":180,"Y":190,"WIDTH":240,"HEIGHT":100),
PARENT @Panel
290 !
300 Continue: ! Create values to display
310 FOR I=1 TO 10000
320 Torq=INT(150+150*RND)
330 Pres=INT(120+20*RND)
340 Dwell=INT(28+4*RND)
350 Timing=INT(15*RND)
360 Rpm=INT(500+6000*RND)
370 !
380 CONTROL @Meter1;SET("VALUE":Torq)
390 CONTROL @Bar2;SET("VALUE":Pres)
400 CONTROL @B2dsp;SET("VALUE":Pres)
410 CONTROL @Disp1;SET("VALUE":Dwell)
420 CONTROL @Disp2;SET("VALUE":Timing)
430 CONTROL @Disp3;SET("VALUE":Rpm)
440 !
450 OUTPUT Buf$ USING "#,K,DDDD,3X,DDD,DDD,DDD";"Update #";I,Torq,Pres,Dwell
460 CONTROL @Text;SET("APPEND TEXT":Buf$)
470 NEXT I
480 GOTO Continue
490 STOP
500 END

```

The engine monitor looks like:

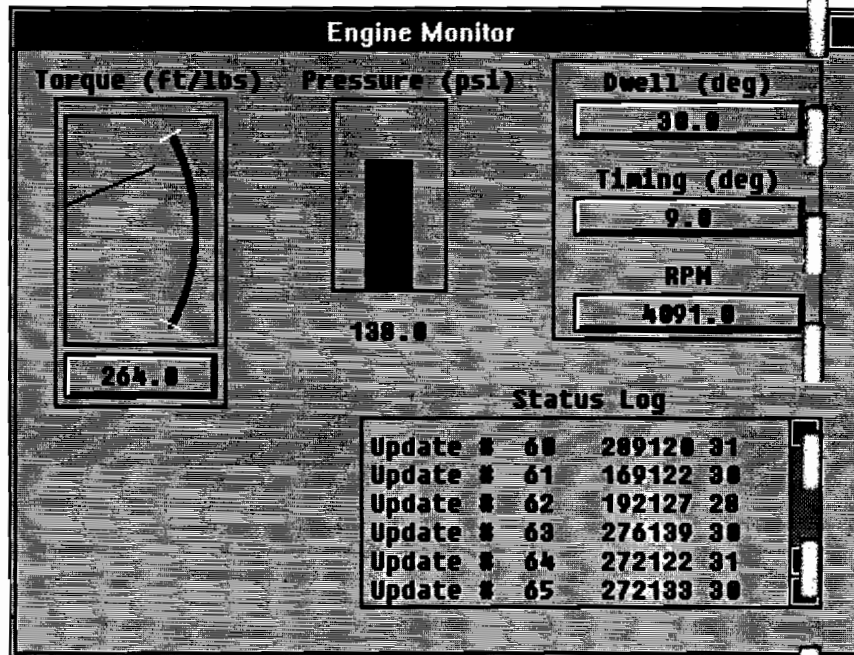


Figure 7-7. An Engine Monitor Example

Summary

Here are the widgets and keywords discussed in this chapter with other closely-related items.

Widgets

Topic	Related Widgets	Description
Parent Widgets	CASCADE MENU	Creates a menu within a menu.
	PANEL	Groups widgets together.
	PULLDOWN MENU	Creates a menu.
Creating Menus	CASCADE MENU	Creates a menu within a menu.
	MENU BUTTON	Creates a menu item.
	MENU SEPARATOR	Creates a visual separator between menu items.
	MENU TOGGLE	Creates a menu item that toggles between two choices.
Setting Alarms	PULLDOWN MENU	Creates a menu.
	BAR	Displays a value as a bar.
	BARS	Displays a bar chart.
	METER	Displays a "VU meter".

Keywords

Topic	Related Keywords	Description
Getting Screen Size	GESCAPE 36	Returns the size (in pixels) of your screen.
Maximizing the Graph Window	GESCAPE 30	Maximizes graph window so that widgets can be printed.
Securing Programs	SECURE	Prevents program lines from being seen. This keyword is not reversible.

Sharing Information with Other Applications

This chapter introduces some common ways to share information between HP Instrument BASIC and other applications. For example, you may want a report to contain test results that are produced by your HP Instrument BASIC program. This chapter explains ways to share the test data with the report.

You can share information in two ways:

- Through the Windows clipboard.
- Using Dynamic Data Exchange.

To Share Information Through the Clipboard

An easy way to share information with other applications is through the Windows clipboard. You can share information in the following ways:

- To send program lines, Copy or Cut lines from your program into the clipboard.
- To send a bitmap of the screen, press the **Print Screen** key. This is useful for widgets and full-screen panels. To send a bitmap of only the active window to the clipboard, press **Alt-Print Screen**.

Note



To see the contents of the clipboard without pasting them, open the **Clipboard** icon from the Windows Program Manager.

- To send a bitmap of the graph window, use the **Graphtoclipbrd** subprogram from the **CLIP.CSB** file.

The following program copies a widget from the graph window to the clipboard. The program loads subprograms from the **CLIP.CSB** file from the HP Instrument BASIC installation directory (usually **\IBASIC**).

```
10  ASSIGN @Meter TO WIDGET "METER"  
20  LOADSUB ALL FROM "C:\IBASIC\CLIP.CSB"  
30  CONTROL @Meter;SET("VALUE":42)  
40  CALL Graphtoclipbrd  
50  DELSUB Toclipboard TO END  
60  END
```

Note this subprogram copies the non-transient contents of the graph window only, such as level-0 widgets or plots that are drawn on the graph window (not panels or dialogs).

- To send numerical data, put it in a real array and use the `Toclipboard` subprogram from the `CLIP.CSB` file.

The following program copies values from an array to the clipboard. The program loads subprograms from the `CLIP.CSB` file from the HP Instrument BASIC installation directory (usually `\IBASIC`).

```
10    DIM Volt(10)
20    LOADSUB ALL FROM "C:\IBASIC\CLIP.CSB"
30    FOR I=0 TO 10
40      Volt(I)=I
50    NEXT I
60    CALL Toclipboard(Volt(*))
70    DELSUB Toclipboard TO END
80    END
```

You can send only one array at a time.

Once the information is on the clipboard, you can paste it into a word processor, a spreadsheet, or another Windows application. From the destination application, choose the **Paste** selection from the **Edit** menu.

To read numerical data from the clipboard into your program, use the `Fromclipboard` subprogram from the `CLIP.CSB` file.

The following program reads the values for an array from the clipboard. The program loads subprograms from the `CLIP.CSB` file from the HP Instrument BASIC installation directory (usually `\IBASIC`).

```
10    DIM Volts(10)
20    LOADSUB ALL FROM "C:\IBASIC\CLIP.CSB"
30    CALL Fromclipboard(Volts(*))
40    PRINT Volts(*);
50    DELSUB Toclipboard TO END
60    END
```

To Share Data Using Dynamic Data Exchange (DDE)

Dynamic Data Exchange (DDE) is a mechanism that allows Windows applications to communicate programmatically with each other; this communication is called a **conversation**. DDE contains a set of protocols that allow you to write HP Instrument BASIC programs that share data and commands with other applications.

Key Definitions



- **Application Name** - The DDE name for the application. For example, the application name for HP Instrument BASIC is **Ibasic**. The application must be in your MS-DOS search path; the application name does not include a drive specifier or path.
 - **Data Topic** - An application-specific identifier of the kind of data. For example, a word processor's topic would be the document name. HP Instrument BASIC has two possible topics: **Main** and **System**. You'll use **Main** most of the time and use **System** only for requesting generic information from the application (such as topics or data types supported by the server).
 - **Data Item** - An application-specific identifier for each piece of data. For example, a spreadsheet data item might be a cell location; a word processor data item might be a bookmark name. An HP Instrument BASIC data item is a variable name.
-

To Send HP Instrument BASIC Data to Other Applications

To programmatically share HP Instrument BASIC data with other applications, perform the following tasks:

1. Load the DDE routines into HP Instrument BASIC.
2. Start the DDE conversation, specifying the application name and the data topic.

3. Send the data or commands to the data item location.
4. End the DDE conversation.

When HP Instrument BASIC starts the DDE conversation, it is called the **client application**. The application that accepts data and commands from HP Instrument BASIC, the **server application**, does not need to be running when the HP Instrument BASIC program runs. If the server application is not running, HP Instrument BASIC automatically starts it (after querying you) and loads the specified data topic file. If the server application is running, the correct data topic file must be loaded.

To Load DDE Subprograms

The DDE subprograms are in the DDE.CSB file in the installation directory (usually \IBASIC). To use them in your program, use the LOADSUB keyword. For example, to load all the DDE subprograms:

```
LOADSUB ALL FROM "DDE.CSB"
```

To Start Conversations

Use the **Ddeinitiate** subprogram to start the conversation. You specify both the application name and the data topic. For example:

```
CALL Ddeinitiate (Handle, "Winword", "\winword\test.doc")
```

Handle is the identifier for this conversation and must be an integer variable. **Handle** is returned by **Ddeinitiate** and must be used whenever communicating with the other application. **Winword** is the application name for Word for Windows. Note that the application name does not contain a path; the application must be in your MS-DOS search path. **\winword\test.doc** is the data topic. Note that the data topic must contain the path to the file (if it's not in the current directory). If

HP Instrument BASIC cannot find the data topic, it returns an error.

To Send Data

Use the **Ddepoke** subprogram to send information to other applications. For example, to send a string to a bookmark in Word for Windows:

```
CALL Ddepoke (Handle, "bkmk1", A$)
```

To Send Commands

Use the **Ddeexecute** subprogram to send a string to other applications. For example, to print a document in Word for Windows:

```
CALL Ddeexecute (Handle, "%fp~").
```

The example sends the keystrokes that select **Print** from the **File** menu; they are:

- % - The **Alt** key
- fp - The **F** and **P** keys
- ~ - The **Enter** key

The keystrokes must be valid for the server application.

To End Conversations

Use the **Ddeterminate** subprogram when you've finished exchanging information. For example:

```
CALL Ddeterminate (Handle)
```

Example

The following HP Instrument BASIC program sends two strings to the Word for Windows document called REPORT.DOC. The document, REPORT.DOC, contains two bookmarks, **bkmk1** and **bkmk2**, that are the data items to which the strings are written.

The strings contain the test number and status (**Passed** or **Failed**). After the information is sent, the report is printed.

```
10  INTEGER Handle
20  DIM A$(10)
30  LOADSUB ALL FROM "C:\IBASIC\DDE.CSB"
40  CALL Ddeinitiate(Handle,"Winword","c:\ibasic\report.doc")
50  A$[1]="Passed"
60  Test=163
70 ! Run test here
80 ! If the test fails, set A$ to "Failed"
90  CALL Ddepoke(Handle,"bkmk1",VAL$(Test))
100 CALL Ddepoke(Handle,"bkmk2",A$)
110 CALL Ddeexecute(Handle,"%fp~")
120 CALL Ddeterminate(Handle)
130 DELSUB Ddeinitiate TO END
140 END
```

The report looks like:

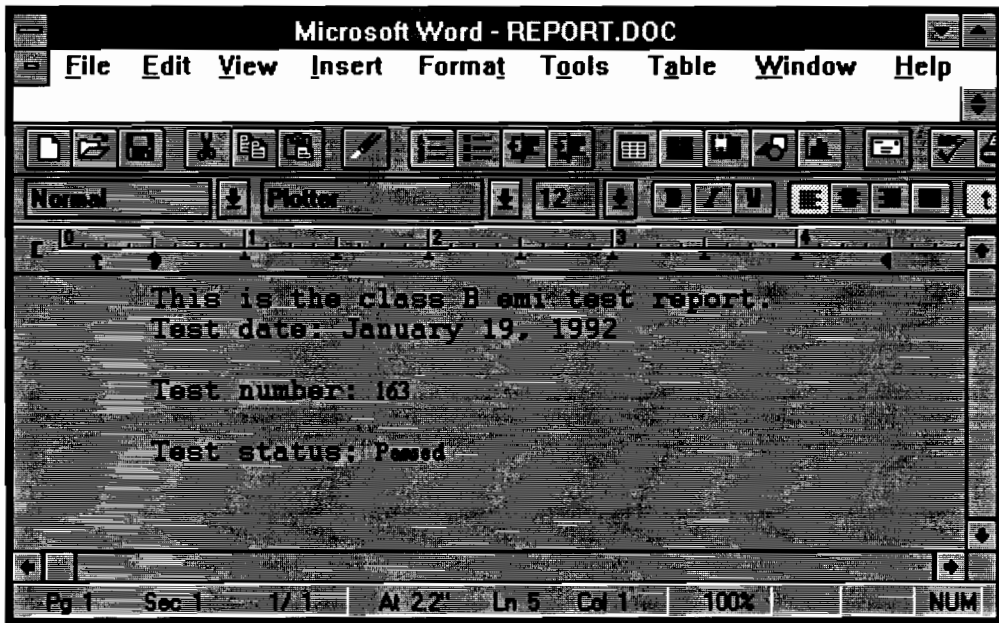


Figure 8-1. A Modified Word for Windows Document

To Send Information to HP Instrument BASIC

To send information to HP Instrument BASIC, follow these steps from the application that is talking to HP Instrument BASIC:

1. Start the DDE conversation, specifying the application name (Ibasic) and the data topic (Main).
2. Send commands to HP Instrument BASIC, such as GET or RUN.
3. Retrieve information from HP Instrument BASIC, such as the values of HP Instrument BASIC variables.
4. End the DDE conversation.

These are essentially the same steps as when HP Instrument BASIC sends commands to other applications; but the commands used are from the other application and are different than HP Instrument BASIC commands. You must refer to that application's DDE documentation for information on the commands and how to use them.

Example

The following example shows an Excel macro called `Ibasic()` that starts HP Instrument BASIC, opens the `VOLT.IBW` program, and runs it. Then `Ibasic()` modifies the program by changing the lines that set the `Instr$` and `Dev` variables, and runs the program again.

The HP Instrument BASIC program is:

```
10      INTEGER Dev
20      REAL Voltage
30      DIM Instr$[80]
40      Dev=722
50      Instr$="HP3457"
60      ASSIGN @Path TO Dev
70      CLEAR @Path
80      OUTPUT @Path;"Read?"
90      ENTER @Path;Voltage
100     PRINT Instr$,"Voltage: ";Voltage
110     END
```

The Excel macro is:

Excel Macro

	A	B
1	Ibasic()	
2	=INITIATE("Ibasic", "Main")	
3	=EXECUTE(A2, "GET ""\IBASIC\VOLT.IBW""")	
4	=EXECUTE(A2, "RUN")	
5	=REQUEST(A2, "Voltage")	
6	=SET.VALUE(B5, A5)	
7	=EXECUTE(A2, B7)	50 Instr\$="HP3458"
8	=EXECUTE(A2, B8)	40 Dev=702
9	=EXECUTE(A2, "RUN")	
10	=REQUEST(A2, "Voltage")	
11	=SET.VALUE(B10, A10)	
12	=TERMINATE(A2)	
13	=RETURN()	

The voltages requested by the spreadsheet are put in cells B5 and B10.

Summary

Here are the menu items, files, and subprograms discussed in this chapter and other items that are closely related to the topics covered:

Topic	Related Items	Description
Sharing Through Clipboard	Edit - Cut	Moves information to the clipboard.
	Edit - Copy	Duplicates information and places it on the clipboard.
	Edit - Paste	Copies information from the clipboard.
	CLIP.CSB	Contains subprograms to communicate with the clipboard.
	Toclipboard	A subprogram that copies numeric data to the clipboard.
	Fromclipboard	A subprogram that copies numeric data from the clipboard.
Copying Graphics	Print Screen	Copies screen bitmap to the clipboard.
	Alt-Print Screen	Copies a bitmap of the active window to the clipboard.
	Graphtoclipbrd	A subprogram that copies graph window bitmap to the clipboard.



Topic	Related Items	Description
Using DDE	DDE.CSB	Contains subprograms to communicate through DDE mechanisms.
	Ddeinitiate	A subprogram that starts a DDE conversation.
	Ddeexecute	A subprogram that sends a command to an application.
	Ddepoke	A subprogram that sends a string to an application.
	Dderequest	A subprogram that retrieves data from an application.
	Ddeterminate	A subprogram that ends a specific DDE conversation.
	Ddeterminateall	A subprogram that ends all DDE conversations.

To get more information about the subprograms in the *.CSB files, refer to **Dynamic Data Exchange (DDE) and Libraries (.CSB files)** under **Programming Topics** in online help.

Language Overview

Clock and Calendar

DATE	Converts a formatted date into a number of seconds.
DATE\$	Converts a number of seconds into a formatted date.
TIME	Converts a formatted time of day into the number of seconds past midnight.
TIME\$	Converts a number of seconds past midnight into a formatted time of day.
TIMEDATE	Returns the value of the real-time clock.

Error Handling

ERRL	Indicates whether an error occurred during execution of a specified line.
ERRLN	Returns the program-line number of the most recent error.
ERRM\$	Returns the text of the last error message.
ERRN	Returns the most recent program execution error.
ERROR RETURN	Returns program control to the line following the line which caused the most recent GOSUB. Used with ON ERROR GOSUB to avoid retrying the line that caused the error (use RETURN to return control to the line which caused the error).
ERROR SUBEXIT	Returns program control to the line following the line which caused the most recent CALL. Used with ON ERROR CALL to avoid retrying the line that caused the error (use SUBEXIT to return control to the line which caused the error).

Graphics

Display Widgets, Input Widgets, and Dialog Boxes

Widgets

BAR	Displays a value as a bar.
BARS	Displays a bar chart.
CASCADE MENU	Creates a menu within a menu.
HPGL VIEW	Displays HPGL graphics files.
LABEL	Displays a string.
LIST	Allows selection from a scrollable list of strings.
MENU BUTTON	Creates a menu item.
MENU SEPARATOR	Creates a visual separator between menu items.
MENU TOGGLE	Creates a menu item that toggles between two choices.
METER	Displays a "VU meter".
PANEL	Groups widgets together.
PRINTER	Displays scrollable lines of text.
PULLDOWN MENU	Creates a menu.
PUSHBUTTON	Displays a pushbutton.
RADIOBUTTON	Displays a bank of buttons. Only one button in each bank can be pressed at any one time.
SCROLLBAR	Creates a scrollbar for a panel so that you can view the parts of the panel that are off screen.
SEPARATOR	Creates a visual separator on a panel.
SLIDER	Allows setting of a value by dragging a slider pad.

STRING	Accepts a string.
STRIPCHART	Displays a moving two-axis plot.
TOGGLEBUTTON	Allows input or display of data that has a binary state.
XY GRAPH	Displays a two-axis plot.

Dialog Boxes

ERROR	Displays an error message.
FILE	Prompts the user to select a file from a list.
INFORMATION	Displays an information string.
LIST	Prompts the user to select an item from a list.
QUESTION	Asks a yes/no question.
STRING	Prompts the user to input a string.
WARNING	Displays a warning message.

Keywords

ASSIGN	Creates a widget and assigns a “widget handle” (equivalent to an I/O path) to the newly created widget. It is also used to destroy existing widgets.
CONTROL	Sets the values of the specified attributes for the specified widget.
DIALOG	Generates a dialog box.
DISABLE EVENT	Temporarily disables an event.
ENABLE EVENT	Reverses the effects of the DISABLE EVENT keyword.
STATUS	Returns the value of each attribute listed for the specified widget.
ON OFF EVENT	Activates/deactivates the event-initiated branch that is taken after a widget generates an event.

Low-Level Graphics

Graphics Control

AREA	Selects an area fill color.
CLIP	Redefines a soft-clip area.
DUMP GRAPHICS	Copies the contents of the graph window to a printing device.
GCLEAR	Clears the graph window.
GESCAPE	Sends and returns device-dependent graphics information.
GINIT	Resets graphics parameters to power-on values.
GLOAD	Loads the graph window from an INTEGER array of Windows bitmap-formatted data.
GRAPHICS ON OFF	Shows or hides the graphic window.
GSTORE	Copies the contents of the graph window to an INTEGER array of Windows bitmap-formatted data.
PLOTTER IS	Specifies the default plotting device or file.
RATIO	Returns the physical aspect ratio of the plotter's hard-clip limits.
SET PEN	Defines the color of entries in the color map.
SHOW	Defines the isotropic plotting units that appear in the VIEWPORT area.
VIEWPORT	Specifies an area in which WINDOW and SHOW statements are mapped.
WHERE	Returns the current logical position of the pen.
WINDOW	Defines the anisotropic plotting units that appear in the VIEWPORT area.

Plotting

DRAW	Draws a line to a specified point.
IDRAW	Draws a line incrementally to a specified point.
IMOVE	Moves the pen incrementally to a specified point.
IPLOT	Draws a line incrementally to the specified point with optional pen control.
LINE TYPE	Selects a plotting line type.
MOVE	Moves the pen to a specified point.
PDIR	Specifies rotation for IPLOT, RPLOT, RECTANGLE, POLYGON, and POLYLINE.
PEN	Selects a plotter pen.
PENUP	Lifts the pen from the plotting surface.
PIVOT	Specifies rotation for lines made with moves, draws, plots, polygons, or rectangles.
PLOT	Draws a line to the specified point with optional pen control.
POLYGON	Draws all or part of a closed polygon.
POLYLINE	Draws all or part of an open polygon.
RECTANGLE	Draws a rectangle that can be filled and edged.
RPLOT	Draws a line relative to a movable origin with optional pen control.

Axes and Labels

AXES	Draws axes with optional tick marks.
CSize	Sets the size and aspect ratio for labeled characters.
FRAME	Draws a frame around the current clipping area.
GRID	Draws a full grid pattern for axes.
LABEL	Draws alphanumeric labels.
LDIR	Defines the angle for drawing labels.
LORG	Specifies a labeling location relative to the pen location.

Input and Output

Files

ASSIGN	Assigns an I/O path name and attributes to a file.
CAT	Lists the contents of the mass storage media's directory.
CONTROL	Sends control information to an interface or a table associated with an I/O path name.
COPY	Copies files and entire directories.
CREATE	Creates an DOS-type file on a mass storage media.
CREATE ASCII	Creates an ASCII-type file on a mass storage media.
CREATE BDAT	Creates a BDAT-type file on a mass storage media.
CREATE DIR	Creates a directory on a mass storage media.
ENTER	Inputs data from a device, file, string, or buffer to a list of variables.
GET	Reads a text file into memory as a program.
LOAD	Loads a program-type file into memory.
LOADSUB	Loads subprograms from a program-type file into memory.
MASS STORAGE IS or MSI	Specifies the default mass storage device.
OUTPUT	Outputs items to a specified device, file, string variable, or buffer.
PURGE	Deletes a file or empty directory.
RENAME	Changes a directory's name or file's name and/or path.
SAVE and RE-SAVE	Creates a text file and writes program lines as strings into the file. RE-SAVE writes to an existing file.
STATUS	Returns the value from a specified interface status register.
STORE and RE-STORE	Creates a program-type file and writes a program from memory into the file in an internal format.
WILDCARDS	Enables and disables wildcard recognition within certain file related commands.

Graphics Input and Display

Refer to “Graphics”

Instruments

Any Interface

ASSIGN	Associates an I/O path name and attributes with a device, group of devices, mass storage file, or buffer.
CONTROL	Sends control information to an interface or a table associated with an I/O path name.
ENTER	Inputs data from a device, file, string, or buffer to a list of variables.
IMAGE	Provides formats for use with ENTER, OUTPUT, DISP, LABEL, and PRINT operations.
OUTPUT	Outputs items to a specified device, file, string variable, or buffer.

HP-IB Only

ABORT	Terminates bus activity and asserts IFC.
CLEAR	Places specified devices in a device-dependent state.
LOCAL	Returns specified devices to their local state.
LOCAL LOCKOUT	Sends the LLO message, disabling all device's front-panel controls.
PASS CONTROL	Passes Active Controller capability to another device.
REMOTE	Sets specified devices to their remote state.
SPOLL	Returns a serial poll byte from a specified device.
STATUS	Returns the value from a specified interface status register.
TRIGGER	Sends the trigger message to specified devices.

Keyboard, Printer, and Alpha Window

ALPHA ON OFF	Shows/hides the alpha window.
CLEAR SCREEN (CLS)	Clears the alpha window.
CONTROL	Sends control information to an interface or a table associated with an I/O path name.
CRT	Returns 1, which is the select code of the CRT display.
DISP	Outputs items to the CRT display line.
DUMP ALPHA	Transfers the contents of the alpha window to a specified device.
ENTER	Inputs data from a device, file, string, or buffer to a list of variables.
IMAGE	Provides formats for use with ENTER, OUTPUT, DISP, LABEL, and PRINT operations.
INPUT	Inputs data from the keyboard to a list of variables.
KBD	Returns 2, which is the select code of the keyboard.
ON KEY ... LABEL	Defines and enables an event-initiated branch to be taken when a softkey is pressed.
OFF KEY	Cancels any event-initiated branches previously defined and enabled by an ON KEY statement.
OUTPUT	Outputs items to a specified device, file, string variable, or buffer.
PRINT	Outputs items to the current PRINTER IS device.
PRINTER IS	Specifies a device for PRINT, CAT, and LIST statements.
PRT	Returns 26, usually the device selector of an external printer.
STATUS	Returns the value from a specified interface status register.
TAB	Moves the print position ahead to a specified point; used within PRINT and DISP statements.
TABXY	Specifies the print position on the internal CRT; used with PRINT statements.

Sound

BEEP Produces an audible tone.

Looping and Branching

Looping

FOR ... NEXT	Defines a loop which is repeated a specified number of times.
LOOP ... END LOOP	Defines a loop which is repeated until the expression in an EXIT IF statement is evaluated as true.
EXIT IF	Provides a conditional exit.
REPEAT ... UNTIL	Allows execution of a program segment until the specified condition is true.
WHILE ... END WHILE	Allows execution of a program segment while the specified condition is true.

Conditional Branching

END IF	Ends an IF ... THEN conditional branching structure.
END SELECT	Ends a SELECT ... CASE conditional branching structure.
IF ... THEN	Provides conditional branching.
ELSE	Provides a conditional execution of a program segment.
ON <i>expression</i>	Transfers program execution to one of several locations based on the value of an expression.
RETURN	Transfers program execution from a subroutine to the line following the invoking GOSUB .
SELECT ... CASE	Allows execution of one program segment of several.
SUB	Defines the beginning of a SUB subprogram and specifies its formal parameters.

Operators Used with Conditional Branching

AND	Returns 1 or 0 based on the logical AND of two arguments.
EXOR	Returns 1 or 0 based on the logical exclusive-or of two arguments.
NOT	Returns 1 or 0 based on the logical complement of an argument.
OR	Returns 1 or 0 based on the logical inclusive-or of two arguments.
=	Equality.
<>	Inequality.
<	Less than.
<=	Less than or equal to.
>	Greater than.
>=	Greater than or equal to.

Unconditional Branching

CALL	Transfers program execution to a specified subprogram and passes parameters.
ERROR RETURN	Returns program control to the line following the line which caused the most recent GOSUB. Used with ON ERROR GOSUB to avoid retrying the line that caused the error (use RETURN to return control to the line which caused the error).
ERROR SUBEXIT	Returns program control to the line following the line which caused the most recent CALL. Used with ON ERROR CALL to avoid retrying the line that caused the error (use SUBEXIT to return control to the line which caused the error).
GOTO	Transfers program execution to a specified line.
GOSUB	Transfers program execution to a specified subroutine.
RETURN <i>expression</i>	Transfers program execution from a user-defined function by returning a value to the calling context.
SUBEXIT	Transfers control from within a subprogram to the calling context.

Event-Driven Branching (Interrupts)

DISABLE	Disables event-initiated branching (except for ON END, ON ERROR, and ON TIMEOUT).
DISABLE INTR	Disables interrupts defined by the ON INTR statement.
ENABLE	Re-enables all event-initiated branches previously suspended by DISABLE.
ENABLE INTR	Enables the specified interface to generate an interrupt which can cause event-initiated branches.
OFF CYCLE	Cancels any event-initiated branches previously defined and enabled by an ON CYCLE statement.
OFF ERROR	Cancels any event-initiated branches previously defined and enabled by an ON ERROR statement. Further errors are reported to the user in the usual fashion.
OFF EVENT	Cancels any event-initiated branch previously defined and enabled by an ON EVENT statement.
OFF INTR	Cancels any event-initiated branch previously defined and enabled by an ON INTR statement.
OFF KEY	Cancels any event-initiated branch previously defined and enabled by an ON KEY statement.
OFF TIMEOUT	Cancels any event-initiated branch previously defined and enabled by an ON TIMEOUT statement.
ON CYCLE	Defines and enables an event-initiated branch to be taken each time the specified number of seconds has elapsed.
ON ERROR	Defines and enables an event-initiated branch which results from a trappable error.
ON EVENT	Defines and enables an event-initiated branch to be taken when an event is generated.

ON INTR	Defines an event-initiated branch to be taken when an interface card generates an interrupt.
ON KEY ... LABEL	Defines and enables an event-initiated branch to be taken when a softkey is pressed.
ON TIMEOUT	Defines an event-initiated branch to be taken when an I/O timeout occurs on the specified interface.
SYSTEM PRIORITY	Sets a minimum level of system priority for event-initiated branches.
WAIT FOR EVENT	Suspends program execution until an event occurs for an active widget.

Math

General Math

+	Addition operator.
/	Division operator.
^	Exponentiation operator.
*	Multiplication operator.
-	Subtraction operator.
ABS	Returns an argument's absolute value.
DIV	Divides one argument by another and returns the integer portion of the quotient.
DROUND	Returns the value of an expression, rounded to a specified number of digits.
EXP	Raises the base e to a specified power.
FRACT	Returns the fractional portion of an expression.
INT	Returns the integer portion of an expression.
LET	Assigns values to variables.
LGT	Returns the log (base 10) of an argument.
LOG	Returns the natural logarithm (base e) of an argument.
MAX	Returns the largest value in a list of arguments.
MAXREAL	Returns the largest number available.
MIN	Returns the smallest value in a list of arguments.
MINREAL	Returns the smallest number available.
MOD	Returns the remainder of integer division.
MODULO	Return the modulo of division.
PI	Returns an approximation of π .
PROUND	Returns the value of an expression, rounded to the specified power of ten.

RANDOMIZE	Modifies the seed used by the RND function.
RND	Returns a pseudo-random number.
SGN	Returns the sign of an argument.
SQRT	Returns the square root of an argument (same as SQR).
SQR	Returns the square root of an argument (same as SQRT).

Trigonometric Functions

ACS	Returns the arccosine of an argument.
ASN	Returns the arcsine of an argument.
ATN	Returns the arctangent of an argument.
COS	Returns the cosine of an angle.
DEG	Sets the degrees mode.
RAD	Sets the radians mode.
SIN	Returns the sine of an angle.
TAN	Returns the tangent of an angle.

Binary Math

BINAND	Returns the bit-by-bit logical-and of two arguments.
BINCMP	Returns the bit-by-bit complement of an argument.
BINEOR	Returns the bit-by-bit exclusive-or of two arguments.
BINIOR	Returns the bit-by-bit inclusive-or of two arguments.
BIT	Returns the state of a specified bit of an argument.
ROTATE	Returns a value obtained by shifting an argument's binary representation a number of bit positions, with wrap-around.
SHIFT	Returns a value obtained by shifting an argument's binary representation a number of bit positions, without wrap-around.

Logical Operators

AND	Returns 1 or 0 based on the logical AND of two arguments.
EXOR	Returns 1 or 0 based on the logical exclusive-or of two arguments.
NOT	Returns 1 or 0 based on the logical complement of an argument.
OR	Returns 1 or 0 based on the logical inclusive-or of two arguments.

Matrix and Array Math

BASE	Returns the lower bound of a dimension of an array.
DET	Returns the determinant of a matrix.
DOT	Returns the dot product of two vectors.
MAT	Performs various operations on numeric and string arrays.
MAT REORDER	Reorders the elements in an array according to the subscript list in a vector.
RANK	Returns the number of dimensions in an array.
SIZE	Returns the number of elements in a dimension of an array.
SUM	Returns the sum of all the elements in a numeric array.

Program Editing and Development

COPYLINES	Copies program lines from one position to another.
DEL	Deletes specified program lines from memory.
DELSUB	Deletes specified subprograms from memory.
EDIT	Displays the edit window.
FIND	Searches for a character sequence in a program.
GET	Reads a text file into memory as a program.
HELP	Starts the help facility.
INDENT	Indents a program to reflect its structure.
LIST	Lists program lines in the alpha window.
LOAD	Loads a program-type file into memory.
LOADSUB	Loads subprograms from a program-type file into memory.
MOVELINES	Moves program lines from one position to another.
REM	Allows comments on program lines.
REN	Renumbers program lines.
SAVE and RE-SAVE	Creates a text file and writes program lines as strings into the file. RE-SAVE writes to an existing file.
SCRATCH	Erases selected portions of memory.
SECURE	Makes program lines unlistable.
STORE and RE-STORE	Creates a program-type file and writes a program from memory into the file in an internal format.

Strings

&	Concatenates two string expressions.
CHR\$	Converts a numeric value into one character byte (one ASCII character).
DVAL	Converts an alternate-base representation into a numeric value.
DVAL\$	Converts a numeric value into an alternate-base representation.
IVAL	Converts an alternate-base representation into an INTEGER number.
IVAL\$	Converts an INTEGER into an alternate-base representation.
LEN	Returns the number of bytes (ASCII characters) in a string expression.
LWC\$	Converts all the ASCII characters in a string to lower-case characters.
NUM	Returns the decimal value of the first byte (the first ASCII character) in a string.
POS	Returns the position of a string within a string expression.
REV\$	Reverses the order of the characters in a string expression.
RPT\$	Repeats the characters in a string expression a specified number of times.
TRIM\$	Removes the leading and trailing ASCII blanks from a string expression.
UPC\$	Converts all the ASCII characters in a string to upper-case characters.
VAL	Converts a string of ASCII digits into a numeric value.
VAL\$	Returns an ASCII string expression representing a specified numeric value.

Subprograms and Functions

CALL	Transfers program execution to a specified subprogram and passes parameters.
DEF FN	Defines the beginning of a function subprogram.
DELSUB	Deletes specified subprograms from memory.
ERROR RETURN	Returns program control to the line following the line which caused the most recent GOSUB. Used with ON ERROR GOSUB to avoid retrying the line that caused the error (use RETURN to return control to the line which caused the error).
ERROR SUBEXIT	Returns program control to the line following the line which caused the most recent CALL. Used with ON ERROR CALL to avoid retrying the line that caused the error (use SUBEXIT to return control to the line which caused the error).
FN	Invokes a user-defined function.
FNEND	Defines the bounds of a user-defined function subprogram.
LOADSUB	Loads subprograms from a program-type file into memory.
RETURN	Transfers program execution from a subroutine to the line following the invoking GOSUB.
SUB	Defines the beginning of a SUB subprogram and specifies its formal parameters.
SUBEND	Defines the bounds of a subprogram.
SUBEXIT	Transfers control from within a subprogram to the calling context.

Variables

ALLOCATE	Dimensions and allocates memory for arrays or string variables during program execution.
BASE	Returns the lower bound of a dimension of an array.
COM	Dimensions and reserves memory for variables in a common area for access by more than one context.
DATA	Specifies data accessible via READ statements.
DEALLOCATE	Reclaims memory previously allocated.
DIM	Dimensions and reserves memory for REAL numeric arrays and strings.
INTEGER	Dimensions and reserves memory for INTEGER variables and arrays.
OPTION BASE	Specifies the default lower bound for arrays.
RANK	Returns the number of dimensions in an array.
READ	Inputs data from DATA lists to variables.
REAL	Dimensions and reserves memory for full-precision (REAL) variables and arrays.
REDIM	Changes the subscript range of an array.
RESTORE	Causes a READ statement to access the specified DATA statement.
SIZE	Returns the number of elements in a dimension of an array.

Other Keywords

EXECUTE	Runs the specified DOS batch file, DOS executable file, or Windows application.
SYSTEM\$	Returns selected system status and configuration information.

Program Control

CONT	Resumes execution of a paused program.
END	Terminates program execution and marks the end of the main program segment.
PAUSE	Suspends program execution.
QUIT	Exits HP Instrument BASIC.
RUN	Starts program execution.
STOP	Terminates execution of the program.
WAIT	Causes program execution to wait a specified number of seconds.

Glossary

alpha window

The window that contains output from the PRINT and DISP keywords, and allows you to type statements to be immediately executed.

angle mode

The current units used for expressing angles. Either degrees or radians may be specified, using the DEG or RAD keywords, respectively. The default at power-on and SCRATCH A is radians.

A subprogram “inherits” the angle mode of the calling context. If the angle mode is changed in a subprogram, the mode of the calling context is restored when execution returns to the calling context.

application name

The DDE name of the application that the DDE conversation is with. The application name for HP Instrument BASIC is Ibasic.

array

A structured data type that contains an arrangement of data in one or more dimensions. The data is accessed via indexes.

An array can be of type REAL, INTEGER or string. Arrays are created with the DIM, REAL, INTEGER, ALLOCATE, or COM statements. Arrays have one

to six dimensions. The default **OPTION BASE** in every environment is zero.

If an array is not explicitly dimensioned, it is implicitly given the number of dimensions used in its first occurrence, with an upper bound of 10. Undeclared strings have a default length of 18 bytes (18 ASCII characters).

ASCII

The acronym for "American Standard Code for Information Interchange". It is a commonly used code for representing letters, numerals, punctuation, special characters, and control characters.

attribute

A quality that determines the function and appearance of a widget or dialog. Each attribute has a value that is set or retrieved with the **CONTROL** or **STATUS** keywords, respectively. Most attributes have default values.

bitmap

A bit pattern or picture stored in a Windows-specific format.

cascading menu

A submenu on a pull-down menu that provides additional selections to a menu selection.

child widget

Any widget that exists within another widget.

Child widgets are created within parent widgets when **ASSIGN** is used with the secondary keyword **PARENT**.

A parent widget may contain multiple child widgets; child widgets may also be parents.

click

To press and immediately release the mouse button.

client application

The application that initiates a Dynamic Data Exchange (DDE) conversation.

clipboard

A Windows "holding area" that allows you to exchange information between applications. Information is generally sent to the clipboard with the **Cut** or **Copy** selection under the **Edit** menu. Information is generally retrieved from the clipboard with the **Paste** selection under the **Edit** menu.

command line

The line in the alpha window in which you can type statements that are immediately executed.

common block

A set of variables that are passed between contexts and are not cleared at pre-run. A common block is created with the **COM** keyword. *See also* **context** and **pre-run**.

common dialog attribute

An attribute of all dialogs, such as **DEFAULT** **BUTTON** or **JUSTIFICATION**.

common widget attribute

An attribute of all widgets, such as **TITLE** or **BORDER**.

context

A specific block of program statements that is treated as an independent unit. Each context has its own "private" set of variables, line labels, angle mode (**DEG** or **RAD**), and other attributes.

Every program has a main context, which includes all statements before the END statement. A program may have any number of additional subprogram (SUB) contexts and function (DEF FN) contexts.

control menu

The menu at the upper-left corner of a window. The symbol on the menu looks like a space bar. The control menu can be accessed by **Alt-Space**.

control pad

The set of buttons that control program execution (for example, **Run** and **Stop**) and windows display (for example, **Alpha** and **Tr Var**). The control pad is generally located at the upper-left corner of the screen.

control panel

A user interface that allows others to easily use your program.

conversation

When two application are communicating or exchanging information through Dynamic Data Exchange (DDE) mechanisms.

data topic

An application-specific identifier used in Dynamic Data Exchange (DDE) that describes the kind of data. For example, a word processor's topic would be the document name. HP Instrument BASIC has two possible topics: **Main** and **System**. You use **Main** most of the time and **System** only for requesting generic information from the application (such as topics or data types supported by the server).

data item

An application-specific location used in Dynamic Data Exchange (DDE). For example, a spreadsheet data item might be a cell location; a word processor data item might be a bookmark name. An HP Instrument BASIC data item is a variable name.

DDE

See **Dynamic Data Exchange**.

device selector

A numeric expression that specifies the source or destination of an I/O operation.

A device selector can be either an interface select code or a combination of an interface select code and an HP-IB primary address. To construct a device selector with a primary address, multiply the interface select code by 100 and add the primary address. For example, a device selector that specifies the device at address 1 on interface select code 7 is 701.

A secondary address may be appended after a primary address by multiplying the device selector by 100 and adding the address. This may be repeated up to 6 times, adding a new secondary address each time. For example, 70502 selects interface 7, primary address 05, and secondary address 02.

DFS

This is the acronym for "DOS File System."

directory name

A directory name specifies a directory of files on a hierarchically structured mass storage volume. Directory names follow the same rules as file names.
Also see **file name**.

dialog

A transient window that presents information or queries for a response. A dialog suspends program execution until the operator takes some action or the dialog times out. A dialog is created with the DIALOG keyword.

double-click

To press and release a mouse button twice in rapid succession. Double-clicking is usually a short-cut to selecting a choice and performing an action. For example, in the **Open** dialog box, double-clicking on a file name selects the file and opens it.

Refer to the Windows manual for more information.

drag

To press *and continue to hold down* a mouse button while moving the mouse. A drag ends when you release the mouse button.

Dynamic Data Exchange (DDE)

A mechanism that allow Windows applications to communicate programmatically with each other.

edit window

The window in which you type program lines.

event

An action, such as an error or user interaction, that generates an interrupt. *See also* **interrupt** and **event-initiated branching**.

event-initiated branching

A programming technique that uses interrupts to redirect program flow. *See also* **event** and **interrupt**.

file name

A name used to identify a file. The name consists of from 1 to 8 characters, *optionally* followed by a



period and an extension of from 1 to 3 characters. All alphanumeric characters (A through Z, a through z, and 0 through 9) are valid in DOS file names and extensions. However, lower-case alpha characters are “case-folded” into upper-case characters. In addition, the following characters may be used: \$, &, #, %, ', !, (,), -, _, @, ^, {, }, and ~.

Spaces may not be used in file names or extensions. Also, the period is not valid, except for the period that separates the extension. Note that DOS reserves certain file name extensions (.COM, .SYS, and .EXE) which should not be used for file names.

Note

When WILDCARDS DOS is enabled, the wildcard characters ? and * cannot be used explicitly in a file name.

focus

The indicator of the widget dialog or HP Instrument BASIC window that is selected and therefore responds to keyboard or mouse actions.

When you move the pointer within the border of a widget or dialog, it gains the focus. Any subsequent keyboard or mouse actions are input to the widget or dialog.

When you click in an HP Instrument BASIC window, it gains the focus and is displayed on top of the other windows. Any subsequent keyboard or mouse actions are input to that window.

function

A procedural call that returns a value. The call can be to a user-defined-function (such as FNInvert) or a machine-resident function (such as COS or EXP). The value returned by the function is used in place of the function call when evaluating the expression containing the function call.

GDU

See graphic display unit.

graph window

The window in which widgets are placed and plots are drawn.

graphic display unit (GDU)

1/100 of the shortest axis on the plotting device. Graphic display units are the same size on both the X and Y axes.

graphics font

The type of characters displayed by LABEL. These are also called "stroked fonts" or "vector fonts". The characters are drawn by a graphics pen.

hard clip limits

The physical limits of the plotting device.

hierarchy

A precedence system to evaluate an expression. Operations with the highest precedence are performed first. Multiple operations with the same precedence are performed left to right. The following tables show the hierarchy for numeric and string operations.

Math Hierarchy

Precedence	Operator
Highest	Parentheses may be used to force any order of operations. Functions: user-defined and machine-resident. Exponentiation: $\wedge$. Multiplication and division: $*$, $/$, MOD , DIV , and MODULO . Addition, subtraction, monadic plus, and minus: $+$ and $-$. Relational operators: $=$, $<>$, $<$, $>$, $<=$, and $>=$ NOT . AND .
Lowest	OR and EXOR .

String Hierarchy

Precedence	Operator
Highest	Parentheses. Functions (user-defined and machine-resident) and substring operations.
Lowest	Concatenation: & .

HPGL

Hewlett Packard Graphics Language. Used to communicate between HP Instrument BASIC and plotters.

HP-IB

Hewlett-Packard's implementation of the IEEE-488 interface.

I/O path

An identifier that allows you to access files, instruments, and widgets. An I/O path is closed when the context in which it is created, terminates (unless the I/O path is passed as a variable or placed in a common block). An I/O path is not the same as an MS-DOS path or pathname.

image specifier

A special code that can be used with certain I/O keywords to format data. For example, the image specifier, `SDD.D`, formats a number as a sign (plus or minus) followed by two digits, a decimal point, and a fractional digit.

input focus

See **focus**.

INTEGER

A numeric data type stored internally in two bytes. Two's-complement representation is used, giving a range of -32 768 thru +32 767. If a numeric variable is not explicitly declared as an **INTEGER**, it is a **REAL**.

integer

A number with no fractional part; a whole number.

interface

A hardware or software mechanism that allow different systems to communicate. For example, the HP-IB interface allows your computer to communicate with instruments.

interface select code

A numeric expression that selects an interface for an I/O operation. For example, the interface select code 1, is used for the CRT. Interface select codes 3-8, are used for HP-IB cards. *Also see device selector.*

interrupt

An condition that indicates that a particular event has occurred. An interrupt stops the current operation and proceeds to the interrupt-handling routine. *See also event-initiated branching.*

keyword

A group of upper-case ASCII letters that has a predefined meaning to HP Instrument BASIC. Keywords can be typed using all lower-case or all upper-case letters.

level-0 widget

A widget that is created without first creating a parent widget to contain it.

A level-0 widget may exist independently, or it may be a parent to child widgets.

A level-0 widget has a title bar on top of it and a resize border around it. You can assign most of the widgets to be level-0 widgets. All dialogs are level-0.

literal

A string constant. When double-quotes are used to delimit a literal, those quotes are not part of the literal. To include quotes in a literal, type two consecutive double-quote marks.

mask

A “filter” that allows you to see only the bits specified.

menu bar

The area just below the title bar in a level-0 widget.
The menu bar contains menu widgets.

operator

The person who interacts (using the mouse or keyboard) with the widgets and dialogs on the screen as the program is running.

operator panel

See control panel.

parent widget

A widget that contains other widgets.

Child widgets are created within parents when ASSIGN is used with the secondary keyword PARENT.

A parent widget may contain multiple child widgets; child widgets may also be parents.

pen

The virtual pen that draws graphics. You can specify the color and the location of the pen.

pixel

A single dot of a computer monitor's display that varies in color and intensity.

pointer

The graphical image that maps to the movement of the mouse. A pointer allows you to make selections. The HP Instrument BASIC pointer is usually an arrow shape, but changes to an I-beam when inserting text in the edit window.

You move the focus from one widget to the next by moving the pointer from one widget to the next.

pre-run

The set of actions that resets the program and checks for errors before the program starts to run. Important initialization activities take place during pre-run, such as reserving memory for certain variables and closing I/O paths. The amount of time required for pre-run varies depending on the length of the program.

program line

A statement that is preceded by a line number (and an optional line label) and stored with the Enter key into a program.

pterodactyl

Any of various extinct flying reptiles of the order Pterosauria of the Jurassic and Cretaceous periods. Pterodactyl are characterized by wings consisting of a flap of skin supported by the very long fourth digit on each front leg.

REAL

A numeric data type that is stored internally in eight bytes using sign-and-magnitude binary representation. One bit is used for the number's sign, 11 bits for a biased exponent (bias = 1023), and 52 bits for a mantissa. On all values except 0, there is an implied "1." preceding the mantissa (this can be thought of as the 53rd bit). The range of REAL numbers is approximately:

−1.797 693 134 862 32 E+308 thru
 −2.225 073 858 507 2 E−308 , 0, and
 +2.225 073 858 507 2 E−308 thru
 +1.797 693 134 862 32 E+308

If a numeric variable is not explicitly declared as INTEGER, it is REAL.

recursive

The ability of a subprogram or function to call itself.

resize border

The border that appears around level-0 widgets and dialogs that is used to resize them. To use the resize border:

- Use the mouse to move the pointer to the resize border.
- When the pointer changes shape, press-and-hold the left mouse button and drag the border to the new position.
- Release the left mouse button and the widget redraws at the new size.

row-major order

The order of accessing an array in which the right-most subscript varies the fastest.

SCPI

See **Standard Commands for Programmable Instruments.**

screen origin

The upper-left corner of the screen, which has the graphics coordinates (0,0).

secondary address

A device-dependent command sent on HP-IB that sets another level of address specification. A secondary address allows you to send commands to a specific portion (such as a plug in card) of a complex device (such as a card cage) that only has one bus address.

Also see **device selector.**

server application

The application that responds to a Dynamic Data Exchange (DDE) conversation.

soft clip limits

The restriction of the graphics output to a specific rectangular region of the available plotting area.

Output that extends beyond this region is “clipped”, meaning that it is not drawn.

Standard Commands for Programmable Instruments (SCPI)

A programming language designed specifically for electronic test and measurement instruments. It is implemented in the firmware of instruments.

statement

A keyword combined with any additional items that are allowed or required with that keyword. If a statement is placed after a line number and stored, it becomes a program line. If a statement is typed without a line number and executed, it is called a command.

status byte

An HP-IB instrument-specific byte that contains information about the instrument.

string

A data type comprised of a contiguous series of characters. The first character in a string is in position 1. The maximum length of a string is 32 767 characters.

If a string is not explicitly dimensioned, it is implicitly dimensioned to 18 bytes (18 ASCII characters). Each element in an implicitly dimensioned string array is dimensioned to 18 bytes (18 ASCII characters).

When a string is empty, it has a current length of zero and is called a “null string”.

subprogram

A program segment that begins a new context that remains in existence until the subprogram is properly exited or program execution is stopped.

A subprogram can be a CSUB, a SUB subprogram or a user-defined-function subprogram (DEF FN). Subprograms must be located after the END statement of the main program.

Subprograms can be recursive.

subroutine

A program segment accessed by a GOSUB statement and ended with a RETURN statement.

substring

A substring is a contiguous series of characters that compose all or part of a string. Substrings may be accessed by specifying a beginning position, or a beginning position and an ending position, or a beginning position and a maximum substring length.

For example, substrings of A\$ may be specified by the beginning position (A\$[3]), beginning and ending positions (A\$[3,9]), or beginning position and length (A\$[3;6]).

tab group

A group of child widgets, that are all the same, that are created within ASSIGN statements, one after the other in the program. You create a tab group so that the operator can move the focus from one widget in the group to the next (and wrap around again), without having to traverse every widget in the PANEL. Only the widgets that have a TAB STOP attribute can be members of a tab group and can accept the system focus. Level-0 widgets *cannot* be members of a tab group.

One of the most common examples of a tab group is a row of buttons lined up across the bottom of a PANEL. The buttons present the operator with a group of related choices, such as and .

title bar

The area at the top of a level-0 widget that contains the title text for the widget, as well as the window menu button, minimize button, and maximize button.

trace lines window

The window that lists program lines as they execute.

trace variables window

The window that maintains a list of the variables in the current context and their values.

transient widget

A widget that is not constrained to the graphics window. A transient widget is created using the TRANSIENT option to the ASSIGN keyword.

user-defined function

A procedural call that returns a value. The value returned by the function is used in place of the function call when evaluating the expression containing the function call.

VU meter

An analog meter used in audio equipment to display volume levels.

widget

A graphical object that allows users to interact with the program or view data. A widget is created using the ASSIGN statement.

work area

The area of a widget in which the widget performs its essential function. For example, in the PRINTER widget, the area that contains the text is the work area. In the METER widget, it is the area that contains the meter arc, needle, tick marks, limits boxes, and value box.

A level-0 widget surrounds its work area with a title bar, resize border, and (sometimes) a menu bar.

Index

Special characters

&, 3-13, A-17
 +, A-13
 -, A-13
 /, A-13
 <, A-10
 <=, 3-13, A-10
 <>, 3-13, A-10
 =, 3-13, A-10
 >, 3-13, A-10
 >=, 3-13, A-10
 ^, A-13

A

ABORT, A-7
 ABS, A-13
 accessing array elements, 3-5
 accessing substrings, 3-4
 ACS, A-14
 adding strings. *See* concatenating strings
 addition, A-13
 addresses
 bus, 5-2
 secondary, 5-2
 ALARM event, 7-8
 ALARM RANGES attribute, 7-9
 ALARM TYPE attribute, 7-9
 example of, 7-9
 ALLOCATE, A-19
 ALPHA OFF, A-8
 ALPHA ON, A-8
 alpha window. *See* windows

command line, 2-18
 formatting output, 3-8-9. *See also*
 formatting output
 opening, 2-4
 printing contents of, 2-9
 reading input, 3-10
 scrolling, 2-4, 3-7
 using to change widget attributes,
 6-8
 writing to, 3-7
 alpha window keywords, A-8
 altering programs. *See* editing programs
 Alt-Print Screen, 8-1, 8-11
 AND, 3-13, A-10, A-15
 appearance
 saving window layout, 2-9
 application name
 definition of, 8-4
 HP Instrument BASIC, 8-4
 Word for Windows, 8-5
 applications
 sharing clipboard information, 8-1-3
 sharing information with DDE, 8-4-10
 AREA, A-4
 array indexes, 3-5
 array keywords, A-15
 arrays. *See* numeric arrays, strings
 graphing, 6-15
 numeric, 3-5
 outputting, 3-6
 string, 3-3
 ASN, A-14

ASSIGN, 4-7, 5-15, 5-16, A-3, A-6, A-7

 closing files, 4-6

 closing instrument paths, 5-14

 creating child widgets, 7-3

 creating widgets, 6-27

 deleting widgets, 6-7, 6-27

 exchanging binary data, 5-3

 to files, 4-3

 to instruments, 5-1

assigning I/O paths

 related keywords, 4-7

ATN, A-14

attributes

 axis-specific, 6-15

 changing dialog, 6-3

 changing widgets, 6-7-9

 common, 6-26

 reading current widget, 6-9

 trace-specific, 6-15

automatically closing files, 4-6

automatic line numbering, 2-13

AUTOSCALE attribute

 examples of, 6-19

 examples of , 6-17

AUTOST, 7-12

AXES, A-5

axes keywords, A-5

axis-specific attributes, 6-15

B

BACKGROUND attribute, 6-26

BARS widget, 6-25, 7-8, 7-17, A-2

 example of, 7-9

BAR widget, 6-25, 7-8, 7-17, A-2

BASE, A-15, A-19

baud rate

 changing, 5-5

BEEP, A-9

BINAND, A-14

binary data

 exchanging, 5-3

binary math keywords, A-14

BINCMP, A-14

BINEOR, A-14

BINIOR, A-14

BIT, A-14

bitmaps

 sharing, 8-1, 8-2

bits/character

 setting, 5-5

bookmarks

 as data items, 8-4

BORDER attribute, 6-26

branching

 conditional, 3-12-15

 event-initiated, 3-12-15

 IF ... THEN, 3-12

 related keywords, 3-24

 SELECT ... CASE, 3-12

branching keywords, A-9-12

breakpoints

 setting, 2-14

bus addresses, 5-2

buttons

 dialog box, 6-4

C

calendar keywords, A-1

CALL, 3-16, 3-24, A-10, A-18

capitalization

 of keywords, 2-13

 of variables, 2-13

cards

 select codes, 5-2

CASCADE MENU widget, 7-6, 7-17,

 A-2

 example of , 7-6

CASE, 3-12, 3-24, A-9

CAT, 4-7, A-6

cell locations

 as data items, 8-4

Change, 2-13

- changing
 - dialog attributes, 6-3
 - graph window appearance, 6-7
 - strings, 2-13
 - widget attributes, 6-7-9
 - widget fonts, 6-10
 - width of graphic printouts, 6-23
- changing programs. *See* editing programs
- chart recorder, 6-19. *See also* STRIP
 - CHART widget
- checking
 - HP-IB instrument status, 5-10
 - installation, 1-3
 - printing status, 2-5
 - syntax, 2-13
- child widgets
 - creating, 7-3
 - definition of, 7-3
 - differences from level-0 widgets, 7-3
- CHR\$, A-17
- Clear, 2-10
- CLEAR, 5-15, A-7
- clearing
 - breakpoints, 2-14
 - lines, 2-13
 - status byte, 5-10
 - window contents, 2-10
- Clear Line, 2-13
- CLEAR SCREEN, 3-24, A-8
- client application
 - definition of, 8-5
- CLIP, A-4
- clipboard
 - reading numerical data from, 8-3
 - sharing information, 8-1-3
- CLIP.CSB, 8-2, 8-3, 8-11
- clock keywords, A-1
- closing I/O paths
 - files, 4-6
 - instruments, 5-14
 - widgets, 6-7
- closing the edit window, 2-9
- closing vs hiding windows, 2-9
- CLS, 3-24, A-8
- column output format, 3-8
- COM, 3-20, 3-24, A-19
- comma
 - using to format output, 3-8
- command history, 2-19
- command line, 2-18
 - command history, 2-19
 - using to change widget attributes, 6-8
- commands
 - sending to instruments, 5-7
- comments
 - entering, 2-3
 - in IBASIC.INI, 2-21
- common attributes
 - widget and dialog, 6-26
- compact output format, 3-8
- COMPATIBILITY_INSTR, 2-21
- compatibility with HP IBASIC, 2-21
- complete examples
 - copying, 2-16
- COM port
 - interface select code, 5-3
- concatenation, A-17
- concatenation string, 3-4
- conditional branching, 3-12-15
 - keywords, A-9
 - related keywords, 3-24
- conditional operators, 3-13
 - keywords, A-10
- configuration information, 2-20
- configuring
 - RS-232 instruments, 5-5
- CONT, A-20
- contexts
 - closing, 4-6, 5-14
 - definition of, 3-15
- CONTROL, 5-15, A-3, A-6, A-7

- changing RS-232 configuration, 5-5
 - changing widget attributes, 6-8
 - setting attributes, 6-27
 - using on command line, 6-8
 - control menu
 - Clear**, 2-10
 - hiding and closing windows, 2-9
 - Output, 2-9
 - Print, 6-23
 - control pad, 2-2
 - Alpha** button, 2-4
 - displaying, 2-8
 - displaying windows, 2-8
 - Run** button, 2-4
 - Step** button, 2-14
 - trace lines window, 2-14
 - trace variables window, 2-14
 - control panels. *See* panels
 - creating, 7-2-5
 - examples of, 7-2, 7-14
 - conversations. *See* DDE
 - ending, 8-6
 - sending commands, 8-6
 - sending data, 8-6
 - starting, 8-5
 - converting
 - keyword capitalization, 2-13
 - variable capitalization, 2-13
 - Copy**, 2-13, 8-1, 8-11
 - COPY**, A-6
 - Copy Code**, 2-16
 - copying
 - complete examples, 2-16
 - statement examples, 2-15
 - COPYLINES**, A-16
 - COS**, A-14
 - CREATE**, 4-3, 4-7, A-6
 - CREATE ASCII**, 4-7, A-6
 - CREATE BDAT**, 4-7, A-6
 - CREATE DIR**, 4-2, 4-7, A-6
 - creating
 - child widgets, 7-3
 - control panels, 7-2-5
 - dialog boxes, 6-2-5
 - directories, 4-2
 - directories, related keywords, 4-7
 - easy-to-use panels, 7-13
 - files, 4-3
 - files, related keywords, 4-7
 - full-screen panel, 7-10
 - menus, 7-5-8
 - menus, related widgets, 7-17
 - strip chart, 6-19
 - widgets, 6-6-11
 - x-y plots, 6-15
 - CSIZE**, A-5
 - CURRENT AXIS** attributes, 6-15
 - CURRENT TRACE** attributes, 6-15
 - Cut**, 2-13, 8-1, 8-11
- ## D
- DATA**, A-19
 - data items
 - definition of, 8-4
 - HP Instrument BASIC, 8-4
 - data topics
 - definition of, 8-4
 - HP Instrument BASIC, 8-4
 - Word for Windows, 8-5
 - data types
 - integer, 3-3
 - real, 3-3
 - string, 3-3
 - DATE**, A-1
 - DATE\$**, A-1
 - DDE**
 - ending conversations, 8-6
 - Excel example, 8-9
 - handles, 8-5
 - loading subprograms, 8-5
 - sending commands, 8-6
 - sending data, 8-6

- sending information to
 - HP Instrument BASIC, 8-8
- sending keystrokes, 8-6
- sharing information, 8-4-10
- starting conversations, 8-5
- Word for Windows example, 8-7
- DDE.CSB, 8-5, 8-12
- Ddeexecute, 8-6, 8-12
- Ddeinitiate, 8-5, 8-12
- Ddepoke, 8-6, 8-12
- Dderequest, 8-12
- Ddeterminate, 8-6, 8-12
- Ddeterminateall, 8-12
- DEALLOCATE, A-19
- debugging programs, 2-14
 - trace lines window, 2-14
 - trace variables window, 2-14
- DEFAULT BUTTON attribute, 6-26
- default configuration
 - RS-232, 5-5
- DEF FN, 3-17, 3-26, A-18
- definitions. *See* glossary
 - application names, 8-4
 - child widgets, 7-3
 - client applications, 8-5
 - contexts, 3-15
 - data items, 8-4
 - data topics, 8-4
 - keywords, 3-1
 - level-0 widgets, 7-3
 - parent widgets, 7-3
 - server applications, 8-5
 - statements, 3-1
- DEG, A-14
- DEL, A-16
- Delete Line, 2-13
- deleting
 - directories, 4-3
 - directories, related keywords, 4-7
 - files, 4-3
 - files, related keywords, 4-7
 - HP Instrument BASIC installation, 1-4
 - lines, 2-3, 2-13
 - deleting widgets, 6-7
 - DELSUB, 3-21, 3-24, A-16, A-18
 - DET, A-15
 - device selector
 - HP-IB, 5-2-3
 - RS-232, 5-3
 - DIALOG, A-3
 - creating dialog boxes, 6-2-5
 - dialog boxes, A-3
 - changing attributes, 6-3
 - common attributes, 6-26
 - creating, 6-2-5
 - list of, 6-2, 6-24
 - position of, 6-3
 - dialog buttons
 - reading, 6-4
 - DIALOG BUTTONS attribute, 6-3, 6-26
 - DIM, 3-3, 3-24, A-19
 - dimensioning
 - related keyword, 3-24
 - directories
 - creating, 4-2
 - deleting, 4-3
 - DISABLE, 3-24, A-11
 - DISABLE EVENT, 6-27, A-3
 - DISABLE INTR, 3-24, A-11
 - disabling
 - events, 3-14
 - DISP, 3-7, 3-24, A-8
 - displaying
 - data, related widgets, 6-25
 - graphs, 6-15-23
 - text, 3-7, 6-14
 - text, related widgets, 6-25
 - windows, 2-8
 - DIV, A-13
 - division, A-13

DOS

- running programs from, 7-12

DOT, A-15

double-clicking

- closing edit window, 2-9

- hiding windows, 2-9

- starting **Help**, 2-15

- starting HP Instrument BASIC, 2-2

- viewing control pad, 2-8

double-quotes

- using in string variables, 3-3

DRAW, A-5

DROUND, A-13

DUMP ALPHA, 3-24, A-8

DUMP GRAPHICS, 6-23, A-4

dumping. *See* printing

DVAL, A-17

DVAL\$, A-17

Dynamic Data Exchange. *See* DDE**E**

easy program, 2-2

EDIT, A-16

editing programs, 2-3, 2-13

edit window, 2-2. *See also* windows

- closing, 2-9

- printing contents of, 2-9

ELSE, **3-12**, A-9

ENABLE, 3-24, A-11

ENABLE EVENT, 6-27, A-3

ENABLE INTR, 3-24, 5-16, A-11

enabling error event interrupt, 3-22

enabling events, 3-14

enabling interrupts

- HP-IB, 5-12

END, A-20

END IF, 3-24, A-9

ending DDE conversations, 8-6

END LOOP, 3-24, A-9

END SELECT, **3-12**, 3-24, A-9

END WHILE, 3-24

ENTER, **4-5**, 4-7, 5-15, A-6, A-7, A-8

- reading instrument values, **5-7**

entering comments, 2-3

entering keywords, 2-3

entering programs, 2-3

ERRL, 3-26, A-1

ERRLN, A-1

ERRM\$, **3-22**, 3-26, A-1ERRN, **3-22**, 3-26, A-1

ERRNL, 3-26

ERROR dialog, 6-2, 6-24, A-3

error handling

- related keywords, 3-26

error handling keywords, A-1

error messages, 2-4

- getting help about, 2-15

ERROR RETURN, **3-22**, 3-26, A-1,
A-10, A-18

errors

- handling, 3-22

- priority, 3-22

ERROR SUBEXIT, **3-22**, 3-26, A-1,
A-10, A-18

event-driven branching keywords, A-11

event-initiated branching, 3-12-15

- related keywords, 3-24

events

- disabling, 3-14

- enabling, 3-14

- priority, 3-14

- widget, 6-10

example programs, 2-15-17

Example Programs, 2-16

examples

- copying complete, 2-16

- copying statement, 2-15

- keyword, 2-16

- manual, 2-16

- task, 2-16

- widget, 2-16

Excel

DDE example, 8-9
 exchanging binary data, 5-3
 EXECUTE, A-19
 Exit, 2-6
 EXIT IF, A-9
 exiting HP Instrument BASIC, 2-6
 EXOR, 3-13, A-10, A-15
 EXP, A-13
 exponentiation, A-13

F
 faster opening programs, 2-20
 FILE dialog, 6-2, 6-24, A-3
 files
 assigning I/O paths to, 4-3
 creating, 4-3
 deleting, 4-3
 formatting output, 4-6
 keywords, A-6
 reading from, 4-5
 related keywords, 4-7
 writing text to, 4-4
 writing to, 4-4
 Find, 2-13
 FIND, A-16
 finding strings, 2-13
 finding the control pad. *See* displaying
 windows
 five-line program, 2-2
 FN, 3-17, 3-26, A-18
 FNEND, 3-17, A-18
 fonts
 changing widget, 6-10
 FOR ... NEXT, 3-11, 3-24, A-9
 FORMAT OFF binary data, 5-3
 FORMAT ON, 4-7
 writing text files, 4-4
 formatting data
 related keywords, 4-7
 formatting output, 3-8-9
 comma, 3-8

 image specifier, 3-8-9
 related keywords, 3-24
 semi-colon, 3-8
 to files, 4-6
 FRACT, A-13
 FRAME, A-5
 Fromclipboard, 8-3, 8-11
 full-screen panel
 creating, 7-10
 function keywords, A-18
 functions, 3-17. *See also* user-defined
 functions

G

GCLEAR, A-4
 GESCAPE, 6-27, A-4
 changing graph window, 6-7
 GESCAPE 30, 7-17
 maximizing the graph window, 7-10
 GESCAPE 36, 7-17
 to find size of screen, 7-10
 GESCAPE 39
 changing size of graphic printed, 6-23
 GET, A-6, A-16
 GET (Group Execute Trigger), 5-8. *See also* TRIGGER
 getting help, 2-14
 getting input
 related widgets, 6-25
 getting programs. *See* opening programs
 GINIT, A-4
 GLOAD, A-4
 glossary, Glossary-1-18
 GOSUB, 3-15, 3-24, A-10
 GOTO, 3-24, A-10
 graphics keywords, A-3
 low-level, A-4
 graphics keywords and widgets, A-2-5
 GRAPHICS OFF, 7-10, A-4
 GRAPHICS ON, A-4
 graphing data, 6-15-23

Graphoclipbrd, 8-11
 sharing bitmaps, 8-2
graph window. *See* windows
 changing appearance of, 6-7
 deleting widgets from, 6-7
 printing contents of, 2-9
 widget position in, 6-6
GRID, A-5
Group Execute Trigger (GET). *See*
 TRIGGER
group window
 HP Instrument BASIC, 1-2
GSTORE, A-4
H
handles, 8-5
handling errors, 3-22
HEIGHT attribute, 6-26
help
 accessing, 2-14
 copying longer examples, 2-16
 copying statement examples, 2-15
 error messages, 2-4, 2-15
 manual examples, 2-16
 topics, 2-14
 using, 2-15
 using example programs, 2-15-17
Help, 2-14
HELP, A-16
hiding vs closing windows, 2-9
HIGH LIMIT attribute, 7-8
HIGH PEN attribute, 7-8
history. *See* command line, command
 history
HPGL VIEW widget, A-2
HP-IB
 checking instrument status, 5-10
 determining device selector, 5-2-3
 initializing instruments, 5-4
 initiating measurements, 5-8
 interrupts, 5-12

 keywords, A-7
HP IBASIC
 compatibility with, 2-21
HP Instrument BASIC
 application name, 8-4
 data item, 8-4
 data topic, 8-4
 general language summary, 3-24
 group window, 1-2
 icon, 1-2
 installing, 1-1, 1-2, 1-3
 quitting, 2-6
 starting, 2-2
 windows, 2-7-10

I

Ibasic
 application name, 8-4
IBASIC.INI
 comments, 2-21
 configuration information, 2-20
 saving window layout, 2-9
 specifying compatibility, 2-21
 specifying memory usage, 2-21
.IBW
 program format, 2-20
icon
 HP Instrument BASIC, 1-2
IDRAW, A-5
IF ... THEN, 3-12, 3-24, A-9
IMAGE, 3-8-9, 3-24, 4-7, A-7, A-8
 formatting output, 4-6
image specifier, 3-8-9
 writing to files, 4-6
IMOVE, A-5
Indent, 2-13
INDENT, A-16
indexes
 of arrays, 3-5
INFORMATION dialog, 6-2, 6-24, A-3
 examples of, 6-2, 6-4

initializing

HP-IB instruments, 5-4

RS-232 instruments, 5-5

SCPI instruments, 5-5

initiating measurements, 5-8

INPUT, 3-10, 3-24, A-8

input keywords, A-6-9

inputting text, 3-10, 6-11-13

inserting characters, 2-3

inserting lines, 2-3, 2-13

Insert Line, 2-13

INSIDE HEIGHT attribute, 6-26

INSIDE WIDTH attribute, 6-26

installation

deleting, 1-4

requirements, 1-1

verification, 1-3

installing HP Instrument BASIC, 1-1,

1-2, 1-3

instrument interrupts

related keywords, 5-16

instruments

assigning I/O paths to, 5-1

bus address, 5-2

checking status (HP-IB), 5-10

closing paths to, 5-14

configuring RS-232, 5-5

initializing HP-IB, 5-4

initializing SCPI, 5-5

keywords, A-7

preparing for communication, 5-4-6

related keywords, 5-15

RS-232 default configuration, 5-5

secondary address, 5-2

sending commands to, 5-7

servicing HP-IB interrupts, 5-12

taking readings from, 5-7

INT, A-13

integer, 3-3

INTEGER, 3-24, A-19

arrays, 3-5

interface cards

supported, 1-1

interface select codes

HP-IB, 5-2

RS-232, 5-3

interrupt keywords, A-11

interrupts

enabling, 3-22, 5-12

servicing HP-IB, 5-12

I/O paths

assigning to files, 4-3

assigning to instruments, 5-1

closing to files, 4-6

closing to instruments, 5-14

creating widgets, 6-6

deleting widgets, 6-7

IPLOT, A-5

iterating, 3-11-12

FOR ... NEXT loop, 3-11

keywords, A-9-12

related keywords, 3-24

REPEAT ... UNTIL, 3-11

STEP, 3-11

WHILE ... END WHILE, 3-11

IVAL, A-17

J

JUSTIFICATION attribute, 6-26

K

KBD, A-8

keyboard keywords, A-8

keystrokes

sending with DDE, 8-6

keyword examples. *See* examples

keywords

capitalization, 2-13

definition of, 3-1

entering, 2-3

using online help examples, 2-16

L

LABEL, A-5
 label keywords, A-5
LABEL widget, 6-10, 6-25, A-2
 example of, 6-14, 7-4
Latest Information, 1-3
LDIR, A-5
LEN, A-17
LET, A-13
 level-0 widgets
 definition of, 7-3
 differences from child widgets, 7-3
LGT, A-13
 line length
 maximum, 6-4
 line numbering, 2-3, 2-13
 lines
 clearing to end, 2-13
 deleting, 2-13
 indenting, 2-13
 inserting, 2-13
 renumbering, 2-13
 selecting, 2-13
 line syntax checking, 2-13
LINE TYPE, A-5
LIST, A-16
LIST dialog, 6-2, 6-24, A-3
 listing
 printing a, 2-5
LIST widget, 6-10, 6-25, A-2
 live keyboard, 2-18. *See also* command line
LOAD, A-6, A-16
 loading DDE subprograms, 8-5
 loading programs. *See* opening programs
LOADSUB, 3-21, 3-24, 8-5, A-6, A-16, A-18
LOCAL, A-7
LOCAL LOCKOUT, A-7
LOG, A-13
 logging trace lines

 to a file, 2-9
 to a printer, 2-9
 logical operator keywords, A-15
LOOP ... END LOOP, 3-24, A-9
 using with widgets, 6-6
 looping, 3-11-12. *See also* iterating
 looping keywords, A-9-12
LORG, A-5
 losing window contents, 2-9
 low-level graphics keywords, A-4
LOW LIMIT attribute, 7-8
LOW PEN attribute, 7-8
LWC\$, A-17

M

macro
 Excel, 8-9
Main
 data topic, 8-4
 making directories. *See* creating directories
 making files. *See* making files
 manipulating
 strings, 3-4
 manual examples
 in online help, 2-16
 masking
 status byte (HP-IB), 5-10
MASS STORAGE IS (MSI), 4-7, A-6
MAT, 3-24, A-15
 math keywords, A-13-15
MAT REORDER, A-15
 matrix keywords, A-15
MAX, A-13
MAXIMIZABLE attribute, 6-8, 6-26
 maximizing the graph window, 7-10
 maximum line length, 6-4
MAXREAL, A-13
 memory usage
 specifying, 2-21

MENU BUTTON widget, 7-5, 7-6, 7-17,
A-2

example of , 7-6

menus

control, 2-9

creating, 7-5-8

MENU SEPARATOR widget, 7-6, 7-17,
A-2

example of, 7-6

MENU TOGGLE widget, 7-6, 7-17,
A-2

example of , 7-6

menu widgets

list of, 7-6

METER widget, 6-10, 6-25, 7-8, 7-17,
A-2

examples of, 6-6, 6-7, 6-8, 7-3

MIN, A-13

MINREAL, A-13

MOD, A-13

modifying programs. *See* editing
programs

MODULO, A-13

MOVABLE attribute, 6-8, 6-26

MOVE, A-5

MOVELINES, A-16

MS-DOS

running programs from, 7-12

N

names

of variables, 3-2

New, 2-10

NEXT, 3-11, 3-24, A-9

NOT, 3-13, A-10, A-15

NUM, A-17

numbering lines, 2-13

numeric arrays, 3-5

accessing elements, 3-5

default length, 3-5

index to elements, 3-5

sharing with other applications, 8-2

O

OFF, 3-24

OFF CYCLE, A-11

OFF ERROR, 3-22, 3-26, A-11

OFF EVENT, 6-27, A-3, A-11

OFF INTR, A-11

OFF KEY, A-8, A-11

OFF TIMEOUT, A-11

ON, 3-13, A-9

ON CYCLE, 3-13, 3-24, A-11

ON ERROR, 3-13, 3-22, 3-24, 3-26,
A-11

ON EVENT, 6-10, 6-27, A-3, A-11

ON INTR, 3-13, 3-24, 5-16, A-12

ON KEY, A-8, A-12

online help, 2-14. *See also* help

ON TIMEOUT, 3-13, 3-24, A-12

opening files, 4-3

opening programs, 2-12

operator panels. *See* panels

operators

conditional, 3-13

OPTION BASE, 3-5, A-19

OR, 3-13, A-10, A-15

order of array output, 3-6

other keywords, A-19

output

formatting. *See* formatting output

formatting files, 4-6

formatting text, 3-8-9

Output

logging trace lines, 2-9

OUTPUT, 4-4, 4-7, 5-7, 5-15, A-6, A-7,
A-8

output keywords, A-6-9

outputting. *See* printing

overwriting characters, 2-3

P**panels**

- creating control, 7-2-5
- creating easy-to-use, 7-13
- full-screen, 7-10
- position, 7-2
- related keywords, 7-17
- using color on, 7-13

PANEL widget, 7-2, 7-17, A-2**PARENT**

- creating child widgets, 7-3

parent widgets

- definition of, 7-3
- related widgets, 7-17

parity

- setting, 5-5

PASS CONTROL, A-7**passing by reference**

- subprograms, 3-19

passing by value

- subprograms, 3-18

passing variables

- subprograms, 3-18-20

Paste, 2-13, 8-3, 8-11**paths**

- I/O. *See* I/O paths

PAUSE, A-20

- using with widgets, 6-6

PDIR, A-5**pebmem, 2-21****PEN, A-5****PENUP, A-5****PI, A-13****PIVOT, A-5****PLOT, A-5****PLOTTER IS, A-4****plotting data, 6-15-23****plotting keywords, A-5****POINT CAPACITY attribute**

- examples of, 6-15, 6-17

POINT LOCATION attribute, 6-19**POLYGON, A-5****[ports]**

- RS-232 default configuration, 5-5

POS, A-17**preparing instruments, 5-4-6****prerequisites, 2-1****pressing **Enter****

- syntax checking, 2-13

.PRG

- loading subprograms, 3-21

- program format, 2-20

PRINT, 3-7, 3-24, A-8**PRINTER IS, 3-24, A-8****printer keywords, A-8****PRINTER widget, 6-10, 6-25, A-2****printing**

- arrays, 3-6
- programs, 2-5
- selecting the printer, 2-5
- status of, 2-5
- text, 3-7, 6-14
- text, related keywords, 3-24
- widgets, 6-23
- window contents, 2-9

****Print Screen**, 8-1, 8-11****priority**

- errors, 3-22
- system, 3-14

program control keywords, A-20**program editing and development**

- keywords, A-16

program execution

- transferring, 3-15-21

program format

- loading subprograms, 3-21
- programs, 2-20

program lines

- clearing, 2-13
- copying, 2-13
- cutting, 2-13
- deleting, 2-13

- indenting, 2-13
- inserting, 2-13
- pasting, 2-13
- renumbering, 2-13
- sharing, 8-1
- stepping through, 2-14

Program Manager

- running programs from, 2-11, 7-12

Program menu, 2-16

programs

- debugging, 2-14
- editing, 2-13
- example, 2-15-17. *See also* examples
- faster-loading, 2-20
- formats, 2-20
- introductory, 2-2
- opening, 2-12
- printing, 2-5
- running, 2-4, 7-12
- running from **Program Manager**, 2-11
- saving, 2-5, 2-20
- securing, 7-11
- tracing lines, 2-14
- tracing variables, 2-14
- typical structure, 2-6
- typing in, 2-3

- prompting for keyboard input, 3-10, 6-11-13

PROUND, A-13

PRT, A-8

- PULLDOWN MENU** widget, 7-5, 7-6, 7-17, A-2

- examples of, 7-5, 7-6

PURGE, 4-7, A-6

- deleting directories and file, 4-3

- PUSHBUTTON** widget, 6-25, A-2

Q

- QUESTION** dialog, 6-2, 6-24, A-3

- example of, 6-5

- QUIT**, A-20

- quitting HP Instrument BASIC, 2-6

R

- RAD**, A-14

- RADIOBUTTON** widget, 6-25, A-2

- RANDOMIZE**, A-14

- RANK**, A-15, A-19

- RATIO**, A-4

- READ**, A-19

reading

- current widget attributes, 6-9
- dialog buttons, 6-4
- from files, 4-5
- from files, related keywords, 4-7
- numerical data from clipboard, 8-3
- status byte, 5-10
- text, 3-10, 6-11-13
- text, related keywords, 3-24
- values from instruments, 5-7

- readme file, 1-3

- real, 3-3

- REAL**, 3-24, A-19

- arrays, 3-5

- recall, 2-19

- RECTANGLE**, A-5

- REDIM**, A-19

reference

- passing by, 3-19

registers

- RS-232 configuration, 5-5

related keywords

- assigning I/O paths, 4-7
- branching, 3-24
- creating files and directories, 4-7
- deleting files and directories, 4-7
- dimensioning, 3-24
- error handling, 3-26
- formatting data to files, 4-7
- formatting text output, 3-24
- instrument interrupts, 5-16
- instruments, 5-15

- iterating, 3-24
- panels, 7-17
- printing text, 3-24
- reading from files, 4-7
- reading text, 3-24
- subprograms, 3-24
- subroutines, 3-24
- widgets, 6-27
- writing to files, 4-7
- related widgets
 - creating menus, 7-17
 - displaying data, 6-25
 - displaying text, 6-25
 - getting input, 6-25
 - parent widgets, 7-17
 - setting alarms, 7-17
- REM, A-16
- REMOTE, A-7
- REN, A-16
- RENAME, 4-7
- Renumber, 2-13
- renumbering lines, 2-13
- REPEAT ... UNTIL, 3-24, A-9
 - iterating, **3-11**
- requirements
 - installation, 1-1
- RE-SAVE, A-6, A-16
- resequencing lines, 2-13
- RESIZABLE attribute, **6-8**, 6-26
- RE-STORE, A-6, A-16
- RESTORE, A-19
- RESTORE SCREEN attribute, 6-26
- RETURN, **3-15**, 3-22, 3-24, A-9, A-10, A-18
 - reading attributes, 6-27
- REV\$, A-17
- RND, A-14
- ROTATE, A-14
- row-major order, 3-6
- RPLOT, A-5
- RPT\$, A-17
- RQS, 5-10
- RS-232
 - configuring, 5-5
 - default configuration, 5-5
 - device selector, 5-3
- Run
 - from **Program Manager**, 2-11
- RUN, A-20
- running programs, 2-4, 7-12
 - from DOS, 7-12
 - from **Program Manager**, 2-11, 7-12
- S**
 - Save, 2-5, 2-20
 - SAVE, A-6, A-16
 - Save As, 2-5, 2-20
 - saving programs, 2-5, 2-20
 - saving window layout, 2-9
 - SCPI
 - examples using, 5-7, 5-10, 5-12
 - initializing instruments, 5-5
 - SCRATCH, A-16
 - SCROLLBAR widget, A-2
 - scrolling the alpha window, 2-4, 3-7
 - Search, 2-13
 - secondary address, 5-2
 - SECURE, **7-11**, 7-17, A-16
 - securing programs, 7-11
 - select codes
 - HP-IB, 5-2
 - RS-232, 5-3
 - SELECT ... CASE, **3-12**, 3-24, A-9
 - selecting a printer, 2-5
 - selecting lines, 2-13
 - semi-colon
 - using to format output, 3-8
 - sending
 - commands to instruments, 5-7
 - DDE commands, 8-6
 - DDE data, 8-6

- DDE information to
 - HP Instrument BASIC, 8-8
- SENSITIVE attribute
 - example of , 7-6
- SEPARATOR widget, A-2
- serial. *See* RS-232
- server application
 - definition of, 8-5
- servicing HP-IB interrupts, 5-12
- SET
 - setting attributes, 6-27
 - setting dialog attributes, 6-3
 - setting widget attributes, 6-7
- Set Breakpoints, 2-14
- SET PEN, A-4
- setting
 - alarms, 7-8
 - alarms, related widgets, 7-17
 - breakpoints, 2-14
 - interrupts, HP-IB, 5-12
 - printers, 2-5
- SGN, A-14
- SHARED X attribute
 - examples of , 6-17
- sharing
 - bitmaps, 8-1, 8-2
 - information through clipboard, 8-1-3
 - information through DDE, 8-4-10
 - numerical data, 8-2
 - program lines, 8-1
- SHIFT, A-14
- SHOW, A-4
- simple program, 2-2
- SIN, A-14
- SIZE, A-15, A-19
- size of screen, 7-10
- skills needed, 2-1
- SLIDER widget, 6-10, 6-25, A-2
 - example of, 6-9
- software priority. *See* priority
- sound keyword, A-9
- specifying
 - compatibility, 2-21
 - memory usage, 2-21
 - numeric arrays, 3-5
 - string arrays, 3-3
- SPOLL, 5-15, A-7
 - checking HP-IB instrument status, 5-10
- SQR, A-14
- SQRT, A-14
- SRQ, 5-10
- STACKING ORDER attribute, 6-26
- starting DDE conversations, 8-5
- starting HP Instrument BASIC, 2-2
- startup information, 2-20
- statement examples
 - copying, 2-15
- statements
 - definition of, 3-1
- statement syntax checking, 2-13
- STATUS, 5-15, A-3, A-6, A-7, A-8
 - reading attributes, 6-27
 - reading current widget attributes, 6-9
 - reading RS-232 configuration, 5-6
- status byte
 - masking, 5-10
 - reading, 5-10
- Step**, 2-14
- STEP, 3-11
- stepping through program lines, 2-14
- STOP, A-20
- stop bits
 - setting, 5-5
- STORE, A-6, A-16
- storing programs. *See* saving programs
- STRING dialog, 6-2, 6-11, 6-24, A-3
 - example of, 6-12
- strings, 3-3
 - accessing, 3-4
 - changing, 2-13

- concatenating, 3-4
- default length, 3-3
- dimensioning, 3-3
- finding, 2-13
- keywords, A-17
- manipulating, 3-4
- STRING widget, 6-25, A-3
 - example of, 6-12
- STRIPCHART widget, 6-19, 6-25, A-3
 - examples of, 6-19
 - examples of , 6-21
- structure
 - of typical programs, 2-6
- SUB, 3-16, 3-24, A-9, A-18
- SUBEND, 3-16, A-18
- SUBEXIT, 3-22, A-10, A-18
- subprograms, 3-16
 - keywords, A-18
 - passing by reference, 3-19
 - passing by value, 3-18
 - passing variables, 3-18-20
 - related keywords, 3-24
 - using common blocks, 3-20
 - using existing, 3-21
- subroutines, 3-15
 - related keywords, 3-24
- substrings
 - accessing, 3-4
- subtraction, A-13
- SUM, A-15
- SWAP ON
 - binary data, 5-3
- syntax checking, 2-13
- System**
 - data topic, 8-4
- SYSTEM\$, A-19
- SYSTEM PRIORITY, 3-14, A-12

T

- TAB, A-8
- TABXY, A-8

- taking readings from instruments, 5-7
- TAN, A-14
- task examples
 - in online help, 2-16
- text
 - changing widget fonts, 6-10
 - reading, 3-10, 6-11-13
 - writing, 3-7, 6-14
 - writing to files, 4-4
- text format
 - programs, 2-20
- THEN, 3-12, 3-24, A-9
- TIME, A-1
- TIME\$, A-1
- TIMEDATE, A-1
- TIMEOUT, 6-27
 - examples of, 6-5, 6-12
- TITLE attribute, 6-3, 6-26
- Toclipboard, 8-2, 8-11
- TOGGLEBUTTON widget, 6-25, A-3
- topics. *See* data topics
- topping windows, 2-8
- TRACE COUNT attribute
 - examples of, 6-17
- trace lines window, 2-14. *See also* windows
 - printing contents of, 2-9
- trace-specific attributes, 6-15
- trace variables window, 2-14. *See also* windows
 - printing contents of, 2-9
- transferring program execution, 3-15-21
- transient
 - panels, 7-2
- TRIGGER, 5-8, 5-15, A-7
- triggering measurements, 5-8
- trigonometric keywords, A-14
- TRIM\$, A-17
- typical program structure, 2-6
- typing keywords, 2-3
- typing programs, 2-3

U

unconditional branching keywords, A-10
 unformatted data
 exchanging, 5-3
 unified I/O, 5-1. *See also* I/O paths
 un-installing HP Instrument BASIC,
 1-4
 unmasking
 status byte (HP-IB), 5-10
 UNTIL, 3-24, A-9
 iterating, 3-11
 UPC\$, A-17
 USER DATA attribute, 6-26
 user-defined functions, **3-17**
 using
 color on panels, 7-13
 command line, 2-18
 common blocks, subprograms, 3-20
 dialog boxes, 6-2-5
 example programs, 2-15-17
 existing subprograms, 3-21
 widgets, 6-6-11
 USING, 3-24, 4-7
 Using Help, 2-15

V

VAL, A-17
 VAL\$, A-17
 value
 passing by, 3-18
 values
 passing, 3-18-20
 VALUES attribute, 6-19
 variable keywords, A-19
 variable names, 3-2
 as data items, 8-4
 variables. *See* data types
 capitalization, 2-13
 verifying the installation, 1-3
 VGA card, 5-2
 viewing windows. *See* displaying windows

VIEWPORT, A-4
 VISIBLE attribute, 6-26
 example of, 6-8

W

WAIT, A-20
 WAIT FOR EVENT, A-12
 WARNING dialog, 6-2, 6-24, A-3
 WHERE, A-4
 WHILE ... END WHILE, **3-11**, 3-24,
 A-9
 WIDGET
 creating widgets, **6-6**
 widget examples, 2-16. *See also* examples
 widgets
 changing attributes, 6-7-9
 changing fonts, 6-10
 common attributes, 6-26
 deleting, 6-7
 events, 6-10
 list of, 6-25, A-2-3
 online examples of, 2-16
 position of, 6-6
 printing, 6-23
 reading current attributes, 6-9
 related keywords, 6-27
 setting alarms on, 7-8
 using, 6-6-11
 using PAUSE with, 6-6
 WIDTH attribute, 6-26
 WILDCARDS, 4-7, A-6
 WINDOW, A-4
 windows
 clearing contents of, 2-10
 displaying, 2-8
 hiding, 2-9
 losing contents, 2-9
 printing contents of, 2-9
 saving layout, 2-9
 trace lines, 2-14
 trace variables, **2-14**

- working with, 2-7-10
- Windows skills needed, 2-1
- WIN.INI**
 - RS-232 default configuration, 5-5
- Word for Windows
 - application name, 8-5
 - data topic, 8-5
 - DDE example, 8-7
- working with HP Instrument BASIC
 - windows, 2-7-10
- writing
 - text, 3-7, 6-14
 - to files, 4-4

- to files, related keywords, 4-7

X

- X attribute, 6-26
- X DATA attribute
 - examples of , 6-15, 6-17
- XY GRAPH widget, 6-15, 6-25, A-3
 - examples of, 6-15, 6-17

Y

- Y attribute, 6-26
- Y DATA attribute
 - examples of, 6-15, 6-17



Customer Reorder Number
E2200-90000



Printed in U.S.A. E0292

E2200-90600 HP Manufacturing Number