Installing and Using Contents

Installing and Using the Windows Version

Other Manuals

{button www.htbasic.com,Inet("www.htbasic.com")}

Distributed with release 9.3

Installing HTBasic 9.3 for Windows

HTBasic is a technical programming language with extensive graphics, instrument control capabilities and interactive programming aids to speed program development. HTBasic is compatible with Hewlett Packard's "Rocky Mountain" BASIC for HP 9000 Series 200/300 computers. It is designed to offer powerful features and ease-of-use to engineers, scientists and programmers whose experience ranges from novice to expert.

Note: Rocky Mountain BASIC (RMB) is a dialect of BASIC. HTBasic and HP BASIC are particular implementations of RMB.

The HTBasic manual set consists of the Installing and Using Manual, the User's Guide, and the HTBasic Plus Programming Guide. Additional add-on products such as the HTBasic Workshop have their own manuals.

Using This Manual

The Installing and Using Manual provides guidelines to install, start, configure, and run HTBasic for Windows. This manual includes:

- Installing HTBasic for Windows
- Getting Started
- GUI DescriptionGetting_Started GUI_Description
- Using the Keyboard
- CRT & Graphic Drivers
- I/O Device Drivers
- Printer & Pixel Image Device Drivers
- Graphic Input Drivers
- Customizing the Environment
- Transfering Programs & Data from HP BASIC
- Changes from Earlier Releases

Using This Chapter

This chapter explains how to install and run the Windows Version of HTBasic. The distribution media contains a README.TXT file. The README file contains up-to-date information about HTBasic, manual corrections, and compatibility problems. Please pay close attention to the README file before proceeding with installation. Chapter one includes:

Installing HTBasic for Windows

HTBasic for Windows provides a Setup program on CD-ROM that makes it easy to install the application in Windows 95/98/Me or Windows NT 4.0/2000/XP. This section shows how to use the Setup program to install HTBasic for Windows, including:

- Pre-Installation Checks
- Installing on Windows
- Installing on Multiple Computers
- Installing on a Network

Pre-Installation Checks

Before you install HTBasic for Windows, take a few minutes to do the following pre-installation checks.

Step 1: Check Your HTBasic for Windows Package
To make sure your package contents are complete, check them against the following list. If any of the items are missing or damaged, contact your reseller or your local distributor. The HTBasic for Windows package should include:
CD-ROM entitled "HTBasic for Windows"
Installing and Using HTBasic Manual (HEA-0080)
User's Guide (HEA-0081)
HTBasic Plus Programming Guide (HEA-0082)

Step 2: Learn About Technical Support
When you install HTBasic for Windows, be sure to complete the online Registration Card. A U.S. and an International mail-in Registration Card with registration number labels and information on obtaining technical support are also included in the packaging.

When you complete the online registration, or return the registration card, we will be able to provide you with technical support, new product information, and HTBasic for Windows update information.
For North American technical support and product information, call 1-801-224-6550.
For international technical support please contact your local distributor.
Further information on HTBasic for Windows is available from the HTBasic for Windows Home Page on the World Wide Web. To open the HTBasic for Windows Home Page, use the following URL: http://www.htbasic.com

Step 3: Check System Requirements
The following table shows the minimum system requirements to install HTBasic for Windows. These requirements apply equally to Windows 95/98, and Windows NT 4.0.

Item	Minimum Requirements
Processor	Pentium class processor or better.
Memory	32 MB Ram.
Drive Space	30 MB of hard drive space.
Operating System	Windows 95/98/Me or Windows NT/2000/XP.
CD-ROM	CD-ROM drive or access to CD-ROM over a computer network.
Display Driver	HTBasic for Windows will run with a monochrome, 16-color, 256-color or 24-bit display driver.
Printers/Plotters	HTBasic for Windows can send screen dumps to any graphics printer supported by Windows. Plotters must be HPGL or PostScript compatible.

Installing on Windows

To install HTBasic for Windows on Windows 95/98/Me/NT/2000/XP, you will need to use the HTBasic for Windows CD.

NOTE: If you need to install the HP I/O Libraries or I/O cards, connect a printer or plotter, or connect an input device, see Chapter 6 - I/O Device Drivers in the Installing and Using Manual.

Insert the HTBasic for Windows CD. This CD has an autoplay feature which automatically launches the HTBasic Setup program. If it does not start automatically, click Start, Run, and type d:\cdsetup.exe (where "d" is your designated CD drive letter) in the text box and click OK.

Pay close attention to the License Agreement and the readme file as they appear during Setup. The readme file will give you the latest information on installation procedures and product developments. This file can also be referenced after Setup.

The initial Setup screen following the License Agreement file is the User Information screen. Please enter your name, company name (if applicable), and serial number in the corresponding lines. The eleven-digit serial number can be found on the back of the CD jewel case. Correct input of the serial number is required to complete the installation. Please save the original CD and serial number for future reference. The serial number will be stored on your system and will be returned by the command:

```
SYSTEM$ ("SERIAL NUMBER")
```

Please follow the instructions in the Setup program until installation is complete.

Installing on Multiple Computers

Since HTBasic for Windows is not copy protected, you may install HTBasic for Windows on more than one computer. However, HTBasic for Windows is copyrighted, protected by international treaty and provided to you only under license.

You have license to run HTBasic for Windows on only one computer at a time. To install HTBasic for Windows on more than one computer, repeat the installation instructions in "Installing on Windows".

Installing on a Network

HTBasic for Windows can also be installed on a network, but a license must be purchased for each user who will concurrently use it.

Before you can install HTBasic for Windows from or on a network, the system administrator must place the files on the network and configure them. Contact your system administrator for the network path name for your computer. To install HTBasic for Windows on the network, follow the installation instructions in "Installing on Windows".

Starting HTBasic for Windows

This section gives guidelines to start HTBasic for Windows, including:

- Creating a Shortcut Icon
- Setting Automatic Startup
- AUTOST Program
- Command line switches
- GESCAPE codes
- Starting the Application

NOTE: Creating a shortcut icon and/or setting automatic startup for HTBasic for Windows are optional steps. If you do not want to create either item, skip to "Starting the Application".

Creating a Shortcut Icon

The standard installation creates a shortcut icon on the desktop. You may create additional shortcuts to open HTBasic for Windows. Figure 1-1 shows a typical shortcut icon for HTBasic for Windows (identified as a shortcut by a jump arrow in the lower left-hand corner). Double-click the icon to open HTBasic for Windows from the shortcut. To create a HTBasic for Windows shortcut icon:

1. Right-click an empty area on the desktop to display a menu.
2. Point to New and click Shortcut to display the Create Shortcut dialog box.
3. Use Browse... to find and enter the path name on the Command line. By default HTBasic is installed at c:\program files\htbwin.
4. Click Next and then type a desired name for the shortcut icon. Then, click Finish to display the icon on the desktop.
5. Drag the icon to a desired location on the desktop or right-click the mouse in the desktop and click Arrange Icons or Line up Icons.

The HTBasic for Windows icon is identified as a **shortcut** by the jump arrow in the lower left-hand corner of the icon.

Figure 1-1. Typical HTBasic for Windows Startup Icon

Setting Automatic Startup

You can set up HTBasic for Windows to start automatically when Windows is started. After the startup information is created, the next time Windows is started, HTBasic for Windows is automatically started. HTBasic for Windows then executes an autostart (AUTOST) program. To set HTBasic for Windows for automatic startup:

1. Click Start and point to Settings and then click Taskbar... to display the Taskbar Properties dialog box.
2. Select Start Menu Programs, Start Menu Program Tab and click Advanced... to display the Exploring - Startup dialog box.
3. Open the Start Menu Programs folder by double clicking on the Programs folder and then double-click the StartUp folder. If you have not created a HTBasic for Windows shortcut icon, see "Creating a Shortcut" to create the icon.
4. Copy the shortcut icon to the StartUp folder by holding the **Ctrl** key down and dragging the icon to the right side of the dialog box. Then, release the mouse button to copy the icon.
5. Close the StartUp dialog box and restart Windows. After Windows has opened, HTBasic for Windows is automatically started.

AUTOST Program

Each time HTBasic starts, it checks the current directory for an AUTOST program file. If one is found, it is automatically loaded and executed. This allows you to load any necessary device drivers, customize HTBasic using any of the programmable statements and start any default application or menu program you desire.

To customize the AUTOST program, simply LOAD the "AUTOST" program, make the desired corrections and RE-STORE the "AUTOST" program. For example, if you plan on using an IEEE-488 card or a serial (RS-232) interface you should modify your AUTOST program to load the needed drivers. Chapters 5 to 8 contain instructions for loading and customizing drivers. If no driver is loaded, a statement such as

```
OUTPUT 719;A$
```

will produce error 163, Interface not present.

Also, if you plan on using plotter or printer drivers, the AUTOST file is a good place to put the PLOTTER IS or CONFIGURE DUMP statements needed to load the drivers.

Command Line Switches

Optionally, one or more command line switches can be specified when starting HTBasic. Command line switches affect the behavior of HTBasic. Usually, no switches are necessary. Read through the headings to identify any you may need to use.

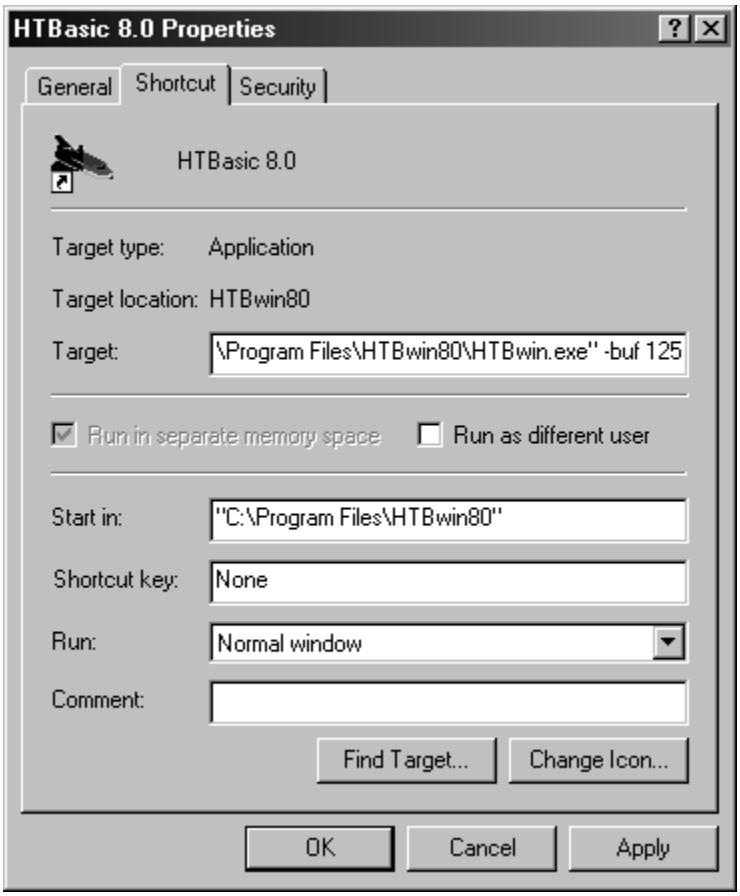
To set the command line switches and the initial MSI (directory) used by HTBasic, change the Properties associated with the icon used to start HTBasic.

Click on the HTBasic icon with the right mouse button and select Properties … to open the Properties Dialog Box.

Select the shortcut tab and locate the Target Input Box with the current path and .exe for HTBasic.

Modify the target .exe by adding the desired Command Line Switch to the end of the target statement. For example, the -buf 125 modifies the number of lines in the buffer for the Extended Output Area as shown in Figure 1-2 below.

Separate multiple switches with a space.



HTBasic command line switches can be abbreviated to the shortest unique abbreviation. The following paragraphs document the individual switches.

Alternate AUTOST Switch

The -ALT switch can be used to run an AUTOST program other than the one named "AUTOST" in the current directory. For example, the following statement will start HTBasic running and will use a file named STARTUP.BAS in the root directory of drive C, regardless of what the current directory is:

"C:\..\HTBwin\HTBwin.exe" -ALT c:\startup.bas

To start HTBasic without running any files, direct the -ALT switch to a non-existent file name for example:

"C:\Program Files\HTBWin\HTBwin.exe" -ALT junk

Extended Output Area Buffer Switch

The -buf switch specifies the number of lines to reserve for the Extended Output Area. The visible portion of the screen is called the Output Area. HTBasic saves lines that scroll off the top of the screen, effectively increasing the number of screen lines. The buffer containing the on- and off-screen lines of text is called the Extended Output Area buffer. The -buf switch determines the number of lines in the buffer. The default is 160. For example, if you have 25 lines on-screen but wish to scroll back and look at 100 previous lines, use this command:

"C:\Program Files\HTBWin\HTBwin.exe" -buf 125

Font Switch

The -fn switch specifies a Windows font to use in place of the default. To specify a point size, give the font name, a comma and the point size. If the font name contains a space, place quotes around it. If no font switch is used, the Windows Version uses the FixedSys font. The syntax is:

"C:\Program Files\HTBWin\HTBwin.exe" -fn "Font",size

Only fixed width fonts should be used. Although most Windows fonts represent the ISO 8859 (Latin 1) character set, not all do. If the keyboard is set to one character set and the screen font uses a different character set, it is possible for non-ASCII characters to display incorrectly or not at all.

If size is positive, the spacing is in pixels; if negative, the spacing is in points (1/72 inch increments). Windows 95 can remap character sets and can use TrueType fonts so any fixed-width font with the required characters may be used as a code page 437 font. For example:

"C:\...\HTBWin80\HTBwin.exe" -fn "Courier New",-12,255

selects a code page 437 mapping of the built-in Courier New font at 12-point size.

There are some fonts included with HTBasic, they may be found in the Lexical directory. These fonts are not hinted

This font information is passed to the HTBasic Windows editor. If the Edit Environment Dialog Box is updated this information will be saved for future instances of HTBasic. If the Edit Environment Dialog Box is not updated, this information will not stay persistent unless the –fn switch remains on the startup shortcut. This allows for multiple shortcuts to have multiple font information that will be retained in the Windows Editor.

SYSTEM$("FONT") returns the name of the Windows font in use.

Window Geometry Switch

The -geometry switch specifies the size and optionally the position of the HTBasic window. The syntax is:

"C:\Program Files\HTBWin\HTBwin.exe" -geometry *WIDTHxHEIGHT*[+*XOFF*+*YOFF*]

where, *WIDTH*, *HEIGHT*, *XOFF* and *YOFF* are numbers and only *WIDTHxHEIGHT* must be specified. *WIDTHxHEIGHT* specifies the size of the HTBasic window and can be specified in either pixels or characters, although the same units must be used for both. If the height is less than 100, the units are interpreted as characters (columns by rows), otherwise the units are interpreted as pixels. *XOFF* and *YOFF* specify the position of the HTBasic window on the display relative to the upper-left corner of the internal child window's display. *XOFF* and *YOFF* are always specified in pixels. By default, HTBasic creates a window that fills the screen.

The first example below creates a window that is 80 columns by 25 lines and the position is set by the window manager. The second example creates a window that is 1024 by 768 pixels, positioned 100 pixels from the top of the screen and 200 from the left edge.

"C:\Program Files\HTBWin\HTBwin.exe" -geometry 80x25
"C:\Program Files\HTBWin\HTBwin.exe" -ge 1024x768+200+100

Window Title Switch

The -title switch specifies the title displayed in the title bar of the HTBasic window and the minimized icon. If the title contains spaces or apostrophes, place quotes around it. The syntax is:
-title *NAME*
where *NAME* is the title string.

"C:\Program Files\HTBWin\HTBwin.exe" -title "Test System"

Workspace Memory Switch

The -w (workspace) switch specifies how much memory to set aside for your programs and data. The syntax is:

-w *amount*[k|m]

where *amount* should be replaced with a number specifying the amount of memory. *Amount* can optionally be followed by a "k" or an "m". If no "k" or "m" is given, the number specifies bytes. If "k" is given, the number specifies kilobytes and if "m" is given, the number specifies megabytes. *Amount* cannot include a period (i.e. 2.4m).

The default workspace size is 16 megabyte. Note that the amount of free memory reported can be somewhat less than that requested because device drivers or other memory users may allocate some of the memory during startup. The following example allocates thirty-two megabytes:

"C:\Program Files\HTBWin\HTBwin.exe" -w 32m

GESCAPE Codes

The GESCAPE statement exchanges device-specific information with a graphic device. It is specified as follows:
GESCAPE device-selector, code [,param(*)][;return(*)]

The device selector specifies the graphic device. The code value specifies the type of operation. The param array sends information to the device and the return array receives information from the device.

Several GESCAPE codes allow manipulation of the HTBasic window.

Code	Operation
30	Maximize the window
31	Hide the window
32	Restore the window
33	Set interior client of the app window position and size
34	Get interior client of the app window position and size
35	Bring the window to the top
36	Get the screen size
37	Returns the Title Bar enable flag
38	Hide / restore title bar
39	Set the DUMP size (% of paper width)
41	Minimize the window

The following GESCAPE CRT codes have been added for manipulation of the program window.

Code	Operation
46	Turn the Control Toolbar Off
47	Turn the Control Toolbar On
48	Turn the Status Bar Off
49	Turn the Status Bar On
50	Remove Main Menu
51	Restore Main Menu
52	Disable Borders on Parent Window
53	Enable Borders on Parent Window
54	Disable Minimize button on Parent Window
55	Enable Minimize button on Parent Window
56	Disable Maximize button on the Parent Window
57	Enable Maximize button on the Parent Window
58	Disable Close button on the Parent Window
59	Enable Close button on the Parent Window
60	Turn the Bookmark Toolbar Off
61	Turn the Bookmark Toolbar On
62	Turn the Debug Toolbar Off
63	Turn the Debug Toolbar On
64	Filename in titlebar off
65	Filename in titlebar on

The following GESCAPE CRT codes have been added for manipulation of the program child window.

Code	Operation
130	Maximize the window
131	Hide the window
132	Restore the window
135	Bring the window to the top
137	Returns the Title bar enable flag
138	Hide / Restore the Title bar (Toggle switch)
141	Minimize the window
152	Disable Borders on Child Window
153	Enable Borders on Child Window

The following example shows the syntax for several of the GESCAPES. Note that codes that set information have a comma before the array name while codes that get information have a semicolon.

```
10  INTEGER Get4(1:4),Set4(1:4),Get2(1:2),Set1(1:1)
20  DATA 90,100,500,300    ! Position of upper left corner:
30                         ! 90,100), Width = 500, Height = 300
40  READ Set4(*)
50  GESCAPE CRT,30          ! Maximize the window
60  GESCAPE CRT,31          ! Hide the window
```

```
70  GESCAPE CRT,32          ! Restore the window
80  GESCAPE CRT,33,Set4(*) ! Set position and size: X,Y,W,H
90  GESCAPE CRT,34;Get4(*) ! Get position and size: X,Y,W,H
100 GESCAPE CRT,35          ! Bring the window to the top
110 GESCAPE CRT,36;Get2(*) ! Get the screen size: W,H
120 GESCAPE CRT,37;Get3(*) ! Get the title bar enable flag
130 PRINT Get(2)            ! Print the Screen Size
140 PRINT Get(3)            ! Print the title bar enable flag
150 Set1(1)=50              ! Set the DUMP size to 50%
160 GESCAPE CRT,38          ! Hide window Title Bar
170 GESCAPE CRT,38          ! Restore window Title Bar
180 GESCAPE CRT,39,Set1(*) ! Set the DUMP size (default is 100%)
190 GESCAPE CRT,41          ! Minimize the window
200 GESCAPE CRT,32          ! Restore the window
210 END
```

Starting the Application

You can start HTBasic for Windows from the program group created by the Setup program or from the shortcut you created. To start HTBasic for Windows, click Start | Programs | TransEra HTBasic | HTBasic or double-click the shortcut icon. The initial display should be similar to that shown in Figure 1-3.

When HTBasic for Windows is started, an **application window** and an enclosed **program window** are displayed.

For the default setting, the status and tool bars are shown for the Application Window.

You can resize and/or reposition the application window, and can resize and/or reposition the program window within the application window boundaries.

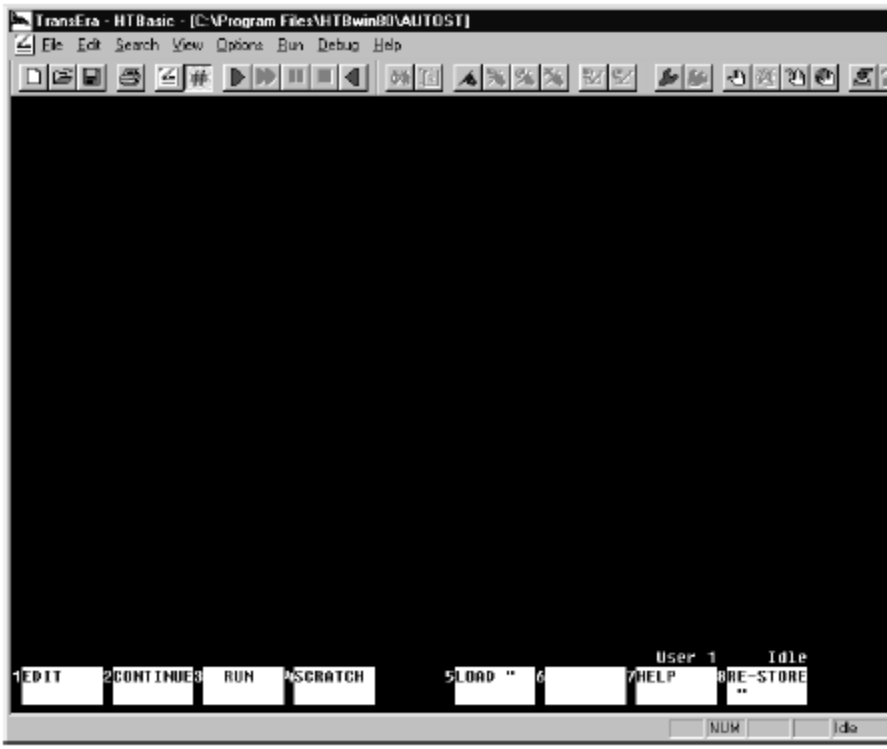


Figure 1-3. HTBasic Startup Display

Using Online Help

HTBasic for Windows includes a Help menu. The Help dialog box is the primary way to access online help topics. To display the Help dialog box, select the Help menu from the application window or the program window. Then:

Click Help | Contents & Index to display the online manual
Click Help | About... for information about HTBasic for Windows
Click Help | Using Help for Windows Help

Figure 1-4 shows a typical Help dialog box. In HTBasic for Windows, the Contents Tab, Index Tab, and Search Tab can be used from the Help dialog box.

This dialog box consists of several books and each book consists of a number of topics. For example, in Figure 1-4 Reference Manual is an open book with one open and three closed chapters. Note that a book can include other books as well as topics.

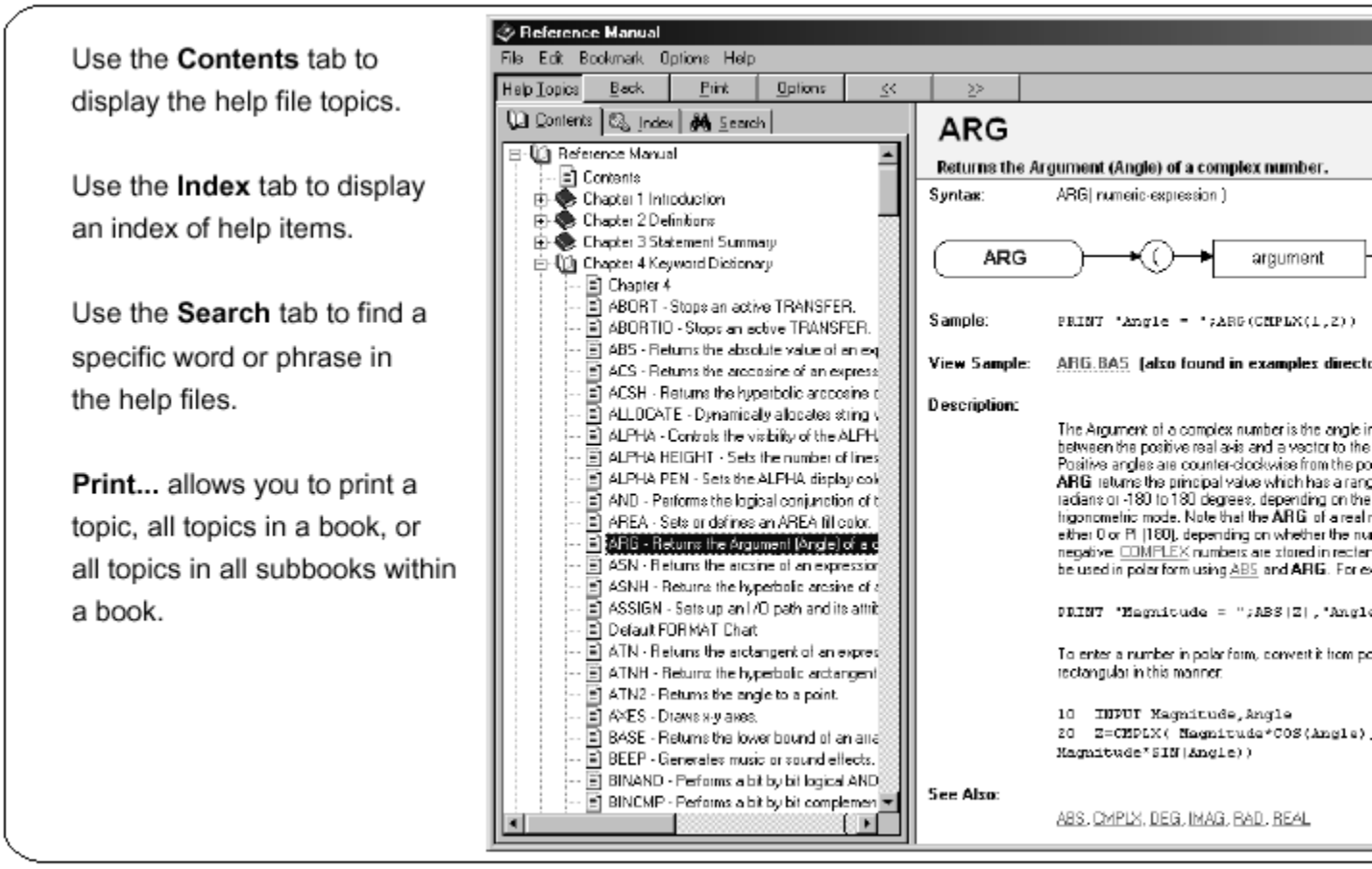


Figure 1-4. Typical Help Topics Dialog Box

To open a closed book, double-click the book title. To close an open book, double-click the book title. To display a specific help topic, double-click the topic name.

Highlighting a topic name and clicking "Print..." allows you to print the topic. Highlighting a book name and then clicking "Print..." allows you to print all the topics in the book (whether the book is open or closed). For example, in Figure 1-4 since ARG is highlighted, clicking "Print..." allows you to print the topic.

To access Help from within the HTBasic Windows Editor, simply place the cursor on the keyword and press the F1 key. The online Help is instantly accessed for the specified keyword.

HELP Command

The HELP Command may also be used to look up keywords in the online manuals. To look up a keyword, type "HELP" on the command line followed by the keyword. For example, to look up the keyword GOTO, type:

```
HELP GOTO
```

and press ENTER. The first page of the keyword will be displayed and the entire manual can be accessed.

Using Windows Help System

Clicking the *Using Help* menu item displays the Windows Help dialog box (see Figure 1-4) that you can use to learn about the Windows help system.

Exiting HTBasic for Windows

The HTBasic user's interface provides several methods (see Figure 1-5) to close the program window, close all open files, shut down HTBasic for Windows, and return to Windows.

In addition to exiting with the user interface commands, HTBasic can be exited programmatically by using the QUIT and QUIT ALL statement. Using QUIT will close the program (child) window and return HTBasic to Idle Mode. Using QUIT ALL closes the program window, closes all open files, shuts down HTBasic for Windows and returns to Windows. These statements can be entered for immediate execution from the keyboard in a program that is not running or it can be executed from within a running program.

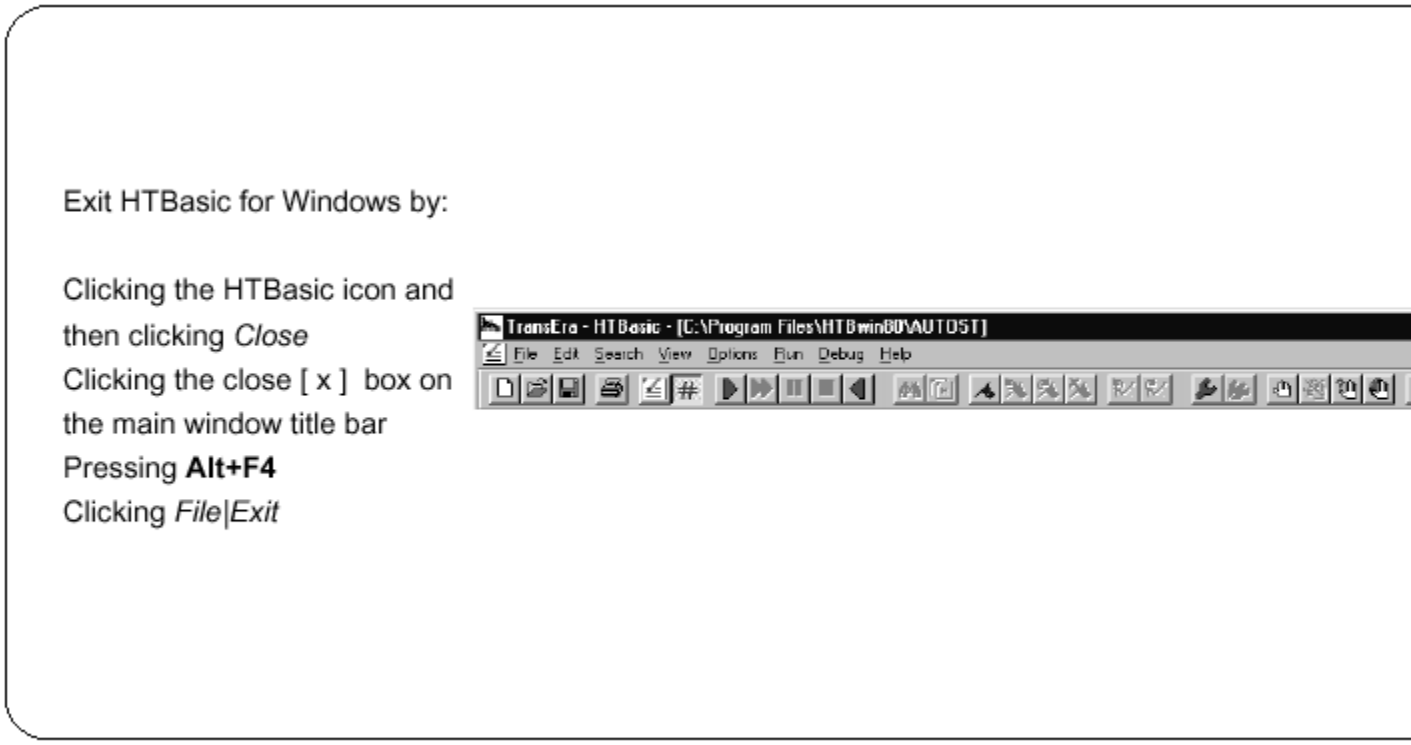


Figure 1-5. Exiting HTBasic for Windows

Getting Started
Using This Chapter

This chapter provides guidelines to get started using the Graphical User Interface (GUI) and either the HTBasic Legacy Editor or HTBasic Windows Editor and Debugger to create, run, and edit HTBasic for Windows programs, including:

- Program Development
- HTBasic Windows Editor
- HTBasic Legacy Editor
- Creating Programs
- Debugging Programs
- Running Programs
- Search and Replace Operations

Program Development

Program development under the HTBasic integrated environment is made easier by program documentation statements and live keyboard execution.

Program Documentation

Program documentation clarifies and explains the structure and function of a program. If its structure and function have been well documented, future program support is simplified.

In HTBasic, there are several ways to document a program: with the REM (remark) statement, comment tails "!", and with mnemonic variable names and program line labels.

The REM and "!" statements are used to insert comments into programs. A "!" comment may be appended to the end of any program statement. They may contain any text you wish and are ignored when the line is executed. When an INDENT command is given, the position of a REM statement is changed to match the current indentation level but the position of a "!" comment is left unchanged.

Line labels may be used in program statements in the place of line number references. The label name is independent of the line number and can be much more meaningful than a line number. It is more convenient to enter "EDIT Open_valve", for example, than to look up the currently assigned line number and enter "EDIT nnnn".

Live Keyboard Execution

At most any time you can enter from the keyboard a command, a list of expressions, or a program line that starts with a line number. Commands are executed, expressions are evaluated and their results displayed in the message line, and program lines are checked for correct syntax and stored into program memory. If results displayed on the message line are longer than 63 characters, the results are truncated to 63 characters. If a syntax error is detected in a program line, an error message is displayed in the system message line and the cursor is positioned over the error. You can use the HELP command to display the Reference Manual entry for the statement to determine how to correct the syntax error. For example:

```
INDENT             ! command
PI*2               ! expression
100 FOR J=1 TO 5   ! program line
```

Any statement you enter without a line number will be executed as an immediate command. Some commands, however, might have more than one meaning depending on how they are entered. For example, you can enter X<4 from the keyboard and HTBasic will display a 1 or 0 depending on its logical value. Suppose, however, that you enter X=4. This looks more like an assignment than a logical comparison, and this is just how HTBasic will treat it. If it was your objective to test whether X was equal to 4, you could do this by entering it in parenthesis: (X=4). In this case, HTBasic will display a 1 or 0 depending on the value of X.

HTBasic Windows Editor

In addition to the traditional HP BASIC-style editor (Legacy Editor), HTBasic 9.0 also includes a new Windows style editor. The HTBasic Windows Editor provides commands and user interface more similar in functionality to today's Windows programming applications.

By default, the Windows Editor is active. To select the Windows Editor, in the program window while in idle mode, select Options | HTB Editor and then check Windows. HTBasic remembers which Editor you were in last.

The Edit mode is started by pressing the Edit button in the control toolbar, pressing the EDIT key (or typing EDIT), or selecting the Edit | Edit Mode menu. The HTBasic Windows Editor features include:

- Line numbers toggled On and Off
- Cut, copy and paste text
- Undo and Redo
- Bookmarking
- Error Lines
- User-defined color coding of the source code

Line Numbers

A significant change from the Legacy Editor is the ability to toggle line numbers On and Off. When "On", line number behavior essentially remains consistent with the Legacy Editor except the Windows editor line numbers can be edited only through the RENumber, COPYLINES and MOVELINES commands. When in the "Off" position (default), the numbers are not removed, but the line numbers are no longer displayed. EDIT statements like MOVELINES will still reference the hidden line number for compatibility purposes.

To turn line numbers On and Off, select the View | menu in the program window. A check will appear by the Line Numbers menu option when the "On" position is selected. Line numbers can also be toggled On and Off by selecting the Line Number Button on the Control Toolbar.

Cut, Copy and Paste Text

A primary function of the HTBasic Windows Editor is the implementation of standard Windows mouse functions. Source code is more easily edited with cut, copy and paste functions.

The new edit functions can be selected in the program window in the Edit | menu. For example, to copy text, select the Edit | Copy menu option. You can also use the mouse to highlight (left-click and drag the cursor to the desired ending position) the desired text selection and either use the Ctrl+C shortcut key or right-click the mouse in the selection area to expose a "pop up" menu with the cut, copy and paste functions.

Undo and Redo

Another standard windows feature now implemented is the Undo and Redo function. For example, to Undo an immediately previous action, select the Edit | Undo menu option. Undo can be selected multiple times sequentially to return to a previous state. You can also use the shortcut keys Ctrl+Z (Undo) and Ctrl+Y (Redo).

Bookmarking

Bookmarking functions have been added to the new editor for improved code navigation. To insert a bookmark, move the cursor to the desired bookmark location and select the Toggle Bookmark Button on the Search Toolbar.

A rectangular mark will be placed next to the selected line. A bookmark can also be placed by selecting the Search | Bookmarks | Toggle Bookmark menu option or by selecting the shortcut key Ctrl+F6. Up to ten bookmarks per program are supported.

To move between bookmarks, select either the Previous Bookmark Button or the Next Bookmark Button from the Search Toolbar. Movement can also be achieved from the Search| Bookmarks | Previous Bookmark or Search | Bookmarks | Next Bookmark menu options or the Ctrl+F7 (Next) or Ctrl+Shft+F7 (Previous) shortcut keys.

To remove an individual bookmark, move the cursor to the line with the bookmark to be removed and either select the Search | Toggle Bookmark menu option (the Search | Toggle Bookmark menu option toggles when an active bookmark has been selected) or type Ctrl+F6 or select the Toggle Bookmark Button on the Search Toolbar. To remove all bookmarks, select the Search | Bookmarks | Remove All Bookmarks menu option or select the Remove All Bookmarks Button on the Search Toolbar.

ERROR Lines

For flexibility during program development, the new windows editor allows the user to leave errors within the code. These errors are, by default, color-coded red. To move between errors, select either the "Previous Error" button or the "Next Error" button from the Search Toolbar. Movement can also be achieved from the Search | Bookmarks | Previous Error or Search | Bookmarks | Next Error menu options.

User-defined Color

The HTBasic Windows Editor provides the user the ability to customize the color and font selection in the editor environment. Color allows individualized adaptation of the source code to fit each user's requirements. Color changes may be applied to keywords, comments, strings, errors, text, breakpoints, bookmarks and various backgrounds (See Figure 2-1).

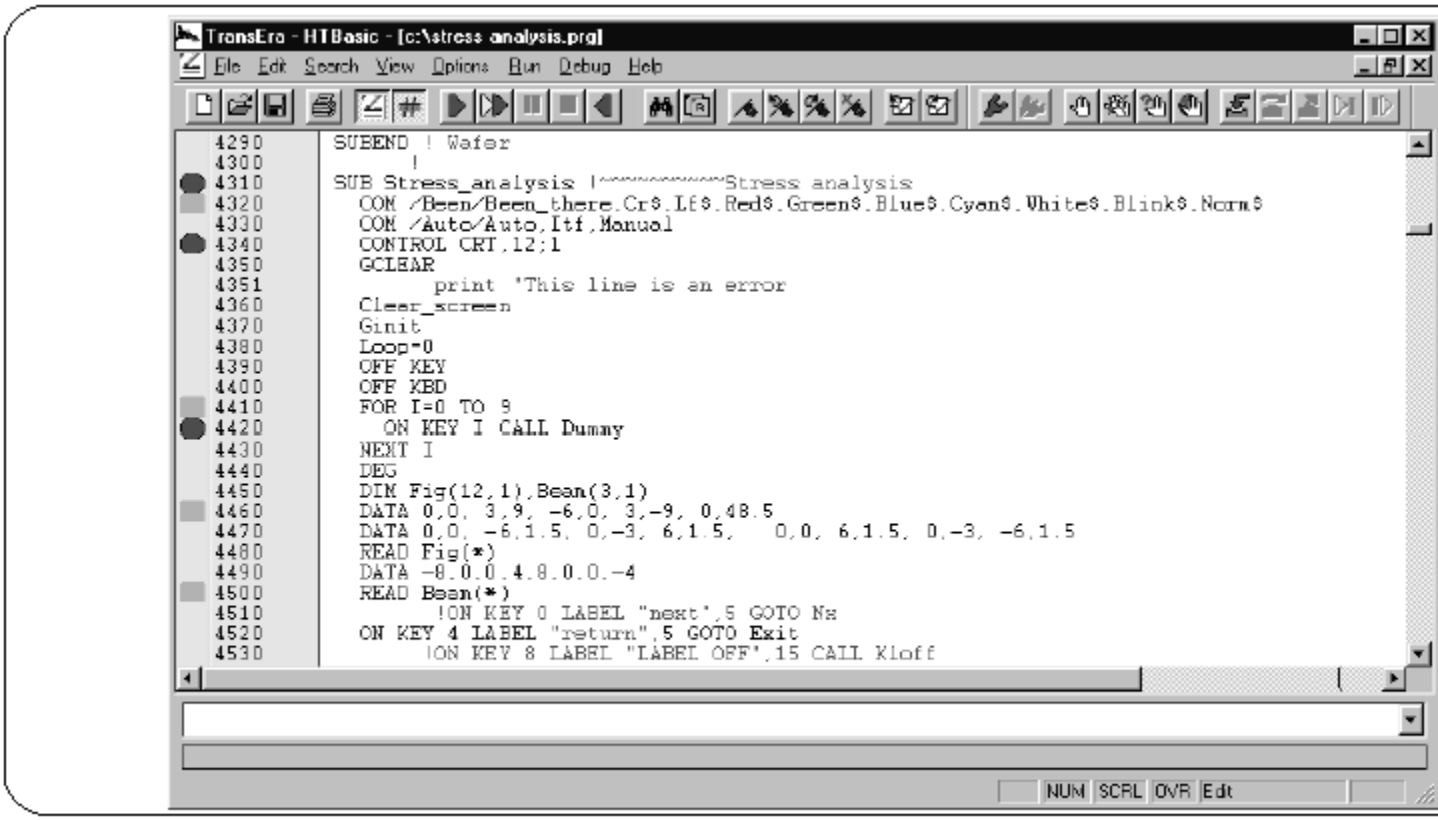


Figure 2-1. Program Window with Windows Editor

To make changes to the default settings of the editor, select the Options | Editor Environment menu option in the program window. Select an editor function, color and then select the "OK" button to save your choices. Custom color selections are also available. To return to the default editor choices, choose the "Default" button and then select the "OK" button. To change the font for all editor functions, select the "Font" button and a list of available fixed-width fonts will appear.

HTBasic Legacy Editor

The traditional way to develop an HTBasic program is with the built- in, full-screen, syntax-sensitive editor. The HTBasic Legacy Editor provides the look and functionality of the traditional RMB environment. Programmers trained in the RMB environment will face little, if any, training.

By default, the Windows Editor is active. To select the Legacy Editor, from the idle mode in the program window select Options | HTB Editor and then check Legacy.

While in Edit mode, a full screen of program lines is displayed. The Message Line displays the error messages, live keyboard calculator results, and status indicators. The softkey menu displays softkey labels corresponding to the function keys, the same as when the display is in normal mode.

This sections gives guidelines to using the HTBasic Legacy Editor, including the following topics:

- Starting Edit Mode
- Controlling the Cursor
- Syntax Checking
- Inserting and Deleting Lines
- COPYLINES and MOVELINES
- DEL Command
- INDENT Command
- REN Command
- SUB Mode
- Immediate Commands
- Associated Commands
- Terminating Edit Mode

Starting Edit Mode

The Edit mode is started by pressing the Edit button in the control toolbar, pressing the EDIT key (or typing EDIT), Ctrl+E, or selecting the Edit | Edit Mode menu.

The default increment for new line numbers automatically produced by the Legacy Editor is ten, but may be specified with an optional increment as follows:

```
EDIT 100,5
```

When editing an existing program, the current edit line will be set either to the last line edited, the last line with an error, or the line specified in the EDIT command. For example:

```
EDIT 100
```

starts the editor with line number 100 as the current edit line. You may specify a line number, line label, SUB program name or DEF FN function name. For example to edit the SUB program TEST enter:

```
EDIT SUB Test
```

Program lines up to the Rocky Mountain Basic limit of 255 characters can be edited, regardless of the screen width. Furthermore, if your screen or window size is wider than 80 characters, the full screen width is utilized. Lines longer than the screen width are scrolled as necessary to allow editing of any part of the line.

Controlling the Cursor

While in EDIT mode, the UP, DOWN, LEFT, and RIGHT arrow keys, NEXT WORD, LEFT WORD, PREV, NEXT, BOL, EOL, BEGIN, and END keys are used to move around the program. The mouse can also be used to move the cursor to a selected position or scroll through the program with the vertical scroll bar on the right side of the program window. If your mouse has a wheel, the wheel can be used to move the cursor as well. The INS CHR key toggles the overstrike mode to insert mode and back again. The mode remains in effect until the INS CHR key is pressed again. The DEL CHR key deletes the character under the cursor. The DEL LEFT key deletes the character left of the cursor.

Syntax Checking

No changes are made to the stored program until you press the ENTER key. The edit line is then checked for correct syntax and if there are no errors it is stored into memory. If a syntax error is detected, an error message is displayed on the message line and the cursor is positioned at the error. If you wish to abort the changes, press any key that moves to another program line.

Inserting and Deleting Lines

To insert a new program line between two existing program lines, or before the first line of the program, position the cursor on the following line and then press INS LN. If necessary, the program will be partially renumbered, and a new line number will be generated for you. After a line has been entered, a new line number is generated and displayed ready for you to enter the next program line. To exit insert line mode press the UP, DOWN, PREV, NEXT, BEGIN, END, INS LN, or DEL LN keys.

To delete a program line, position the cursor on the line you wish to delete, and press DEL LN. A line that has been accidentally deleted can be recovered by pressing the RECALL key followed by the ENTER key.

COPYLINES and MOVELINES

The COPYLINES command copies one or more program lines from one location to another while leaving the original lines in place. The MOVELINES command moves one or more program lines from one location to another. This differs from the COPYLINES command that leaves the original lines in place.

Appropriate renumbering occurs to insert the new program lines into the existing program. Line numbers are renumbered and updated if needed. Line number references in lines not being copied remain linked to the original lines rather than the newly created lines.

DEL Command

The DEL command removes program lines from memory. Once a DEL statement has been executed, the specified lines cannot be retrieved. SUB and DEF statements cannot be deleted unless the entire subprogram is included in the range.

INDENT Command

The INDENT command inserts spaces after the line numbers and before the leading keywords of all lines in a program to visually show the structure of the program. It does not move "!" comment statements. This command can only be executed from the keyboard. It cannot be included in a program.

REN Command

The REN command renumbers program lines, including the line references in all program statements such as GOSUB and GOTO to match the new line numbers. This command can only be executed from the keyboard. It cannot be included in a program.

SUB Mode

With the EDIT SUB Mode, the HTBasic Legacy Editor allows quick navigation of subprograms (SUB, DEF FN, CSUB). Press the SUB Mode key once to display and move through the subprogram list; press it again to switch back to regular editing mode.

Immediate Commands

Immediate commands can be entered in EDIT mode by first deleting the automatic line number and then entering the command. To delete the line number, backspace over it and then type over the top of it, or use CLR LN (not DEL LN) to clear the current line.

Associated Commands

The following program development commands are used in conjunction with the EDIT command. Please review the on-line Reference Manual for more complete details.

FIND	search for a string of characters
CHANGE	finds a string and replaces it with another string
SCRATCH	removes all program lines or COM blocks from memory
XREF	generates a program cross-reference listing
SECURE	makes a program section not listable

Terminating EDIT Mode

The Edit mode is ended by toggling the Edit button in the control toolbar, by unchecking the Edit | Edit Mode menu, or by pressing the CLR SCR, PAUSE, RUN, or STEP keys. It can also be terminated by entering a CAT or LIST command without a line number.

Creating Programs

This section gives guidelines to create HTBasic for Windows programs, including the following topics.

- Creating a Program
- Saving a Program
- Password Protection
- Opening a Program
- Printing a Program
- Closing a Program

Creating a Program

To create a new program you can enter text directly onto the program window from the keyboard. To create a new program:

If a program window (one with no text) is not displayed, click File | New to create a new untitled program.
Type the program lines from the keyboard to create a source program. Begin typing at the caret (blinking _ or |). Backspace and retype to correct errors.
Press the Enter key after each line to parse the line and move the cursor to the next line.

Example: Creating a Program

1. Exit and restart HTBasic for Windows to display an application window with an untitled program window.
2. Type EDIT, and the program lines shown in Figure 2-2 from the keyboard. Begin typing at the **caret** (the blinking _ icon) on the program window. Press the **Enter** key after entering each line of code. If you make a typing mistake, backspace and retype.

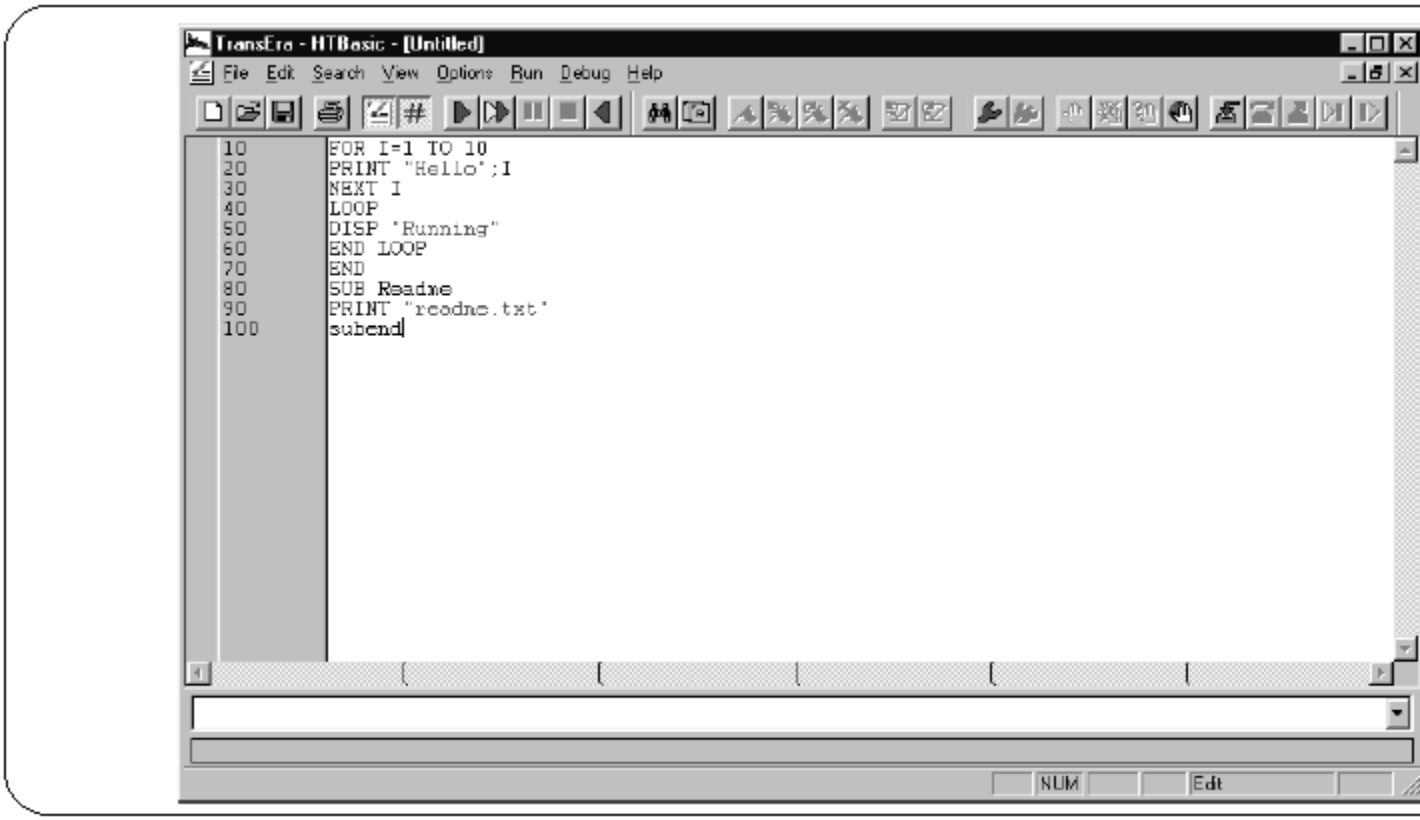


Figure 2-2. Example: Creating a Program

Saving/Storing Programs

Clicking File | Save As | Store As from the application window menu saves or stores the program under the currently specified file name with one of the following five extensions, the file type depends upon the menu choice (Save or Store):

PROG (*.prg)[stored]
BASIC Program (*.bas) [stored]
Text with line numbers (*.txt) [saved]
Text without linenumbers (*.txt) [saved]
Text with adjusted line references (*.txt) [saved] †
LIF ASCII (*.lif) [saved]
HP LIF ASCII (*.lhp) [saved]

To save or store a program:

Click File | Save | Store or File | Save As | Store As... to display the Save | Store As dialog box. If the file already exists, the file will be saved or stored to the same name and type as already exists without bringing up the Save | Store dialog box.
Type the file name in the "File name:" box. Select the file type from the "Save as type:" pulldown box. Select the directory to save/store the file from the "Save in:" box.
Click the "Save" or "Store" button to save/store the file.

Once modified, an asterisk "*"after the title in the program window title bar shows that the program has been modified from the original source.

Example: Storing a Program

This example shows one way to store the program created in the previous example. See Figure 2-3 for a typical Store As dialog box.

Click File | Store or File | Store As... to display the Store | Store As dialog box.
Type *example* in the "File name:" box and select PROG (*.prg) as the file type. Select a directory in which to store the program.
Click "Store" to store the program as a PROG file.

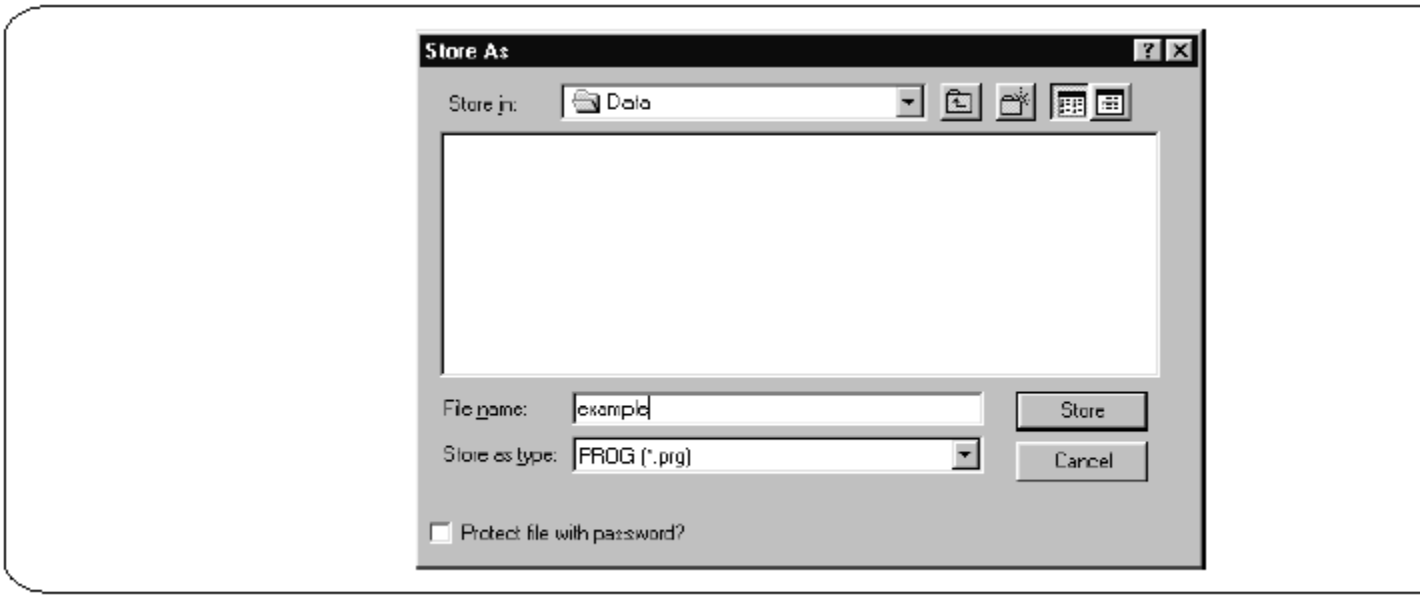


Figure 2-3. Example: Storing a Program

† NOTE: When using the text with adjusted line references option, it will be necessary to verify lines that refer to USING statements using their line numbers. These line numbers are converted to Line Labels that are inserted above the USING statement.

Password

It is possible to protect a stored program with a password by checking the Password check box. This will bring up the Password Dialog box. It will not be possible to edit or view the program code until the password is resupplied.

Reverting to Last Saved/ Stored Version

Clicking File | Revert to Last... from the application window menu opens the last saved/stored version of the file and cancels all changes that occurred since the file was last saved/stored.

Saving/Storing Programs As

If you want to save or store the program under a different file name or file type, clicking File | Save As | Store As... displays a standard Save / Store As dialog box from which you can rename and/or retype the file to be saved/stored.

Opening a Program

You can use the *File* menu from the application window to open an existing program by using File | Open... or File | Recent File 1-10... menu items. Clicking File | Open... displays a standard Open dialog box (see Figure 2-4) from which you can select the file to open. You can display the files for any of the file types shown in the dropdown box.

When the Change MSI option on the File menu is checked, opening a file from the MRU will change the MSI to the folder where the file is located. Opening a file without this option checked will keep the MSI at its current setting when opening files from the MRU.

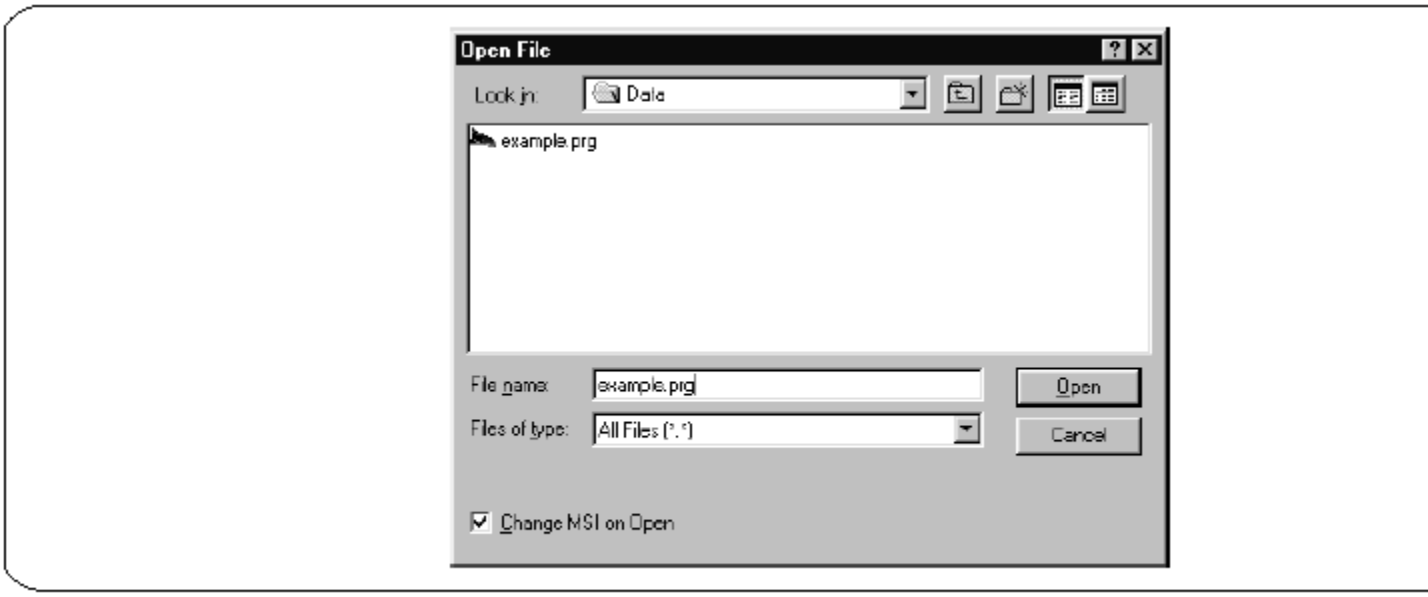


Figure 2-4. Example: Opening a Program

Dragging either an ASCII or PROG file on to either the HTBwin.exe or a shortcut to HTBwin.exe will also open the file. Also, dropping a file into an open edit window will open the file directly into the editor. Dropping a PROG file onto the output window, will open and run the prog file.

Opening Existing Programs

After a program has been created and saved, you can open the program. To open a program:

In the HTBasic application window, click File | Open... or click the control toolbar Open icon to display the Open dialog box.
or..
If the file to be opened has been recently saved/stored, click File | Recent File1...10 (and click on the appropriate file from the recent file list) to open the desired program.

Example: Opening a Program

For this example, we will use File | Recent File1...10 to open the example.prg program we stored in the previous example. To do this:

1. Exit and restart HTBasic for Windows to display an application window and a new program window.
2. Click the x icon in the program window to close the window.
3. Click **File** | **1 example.prg** to open the program.

Printing a Program

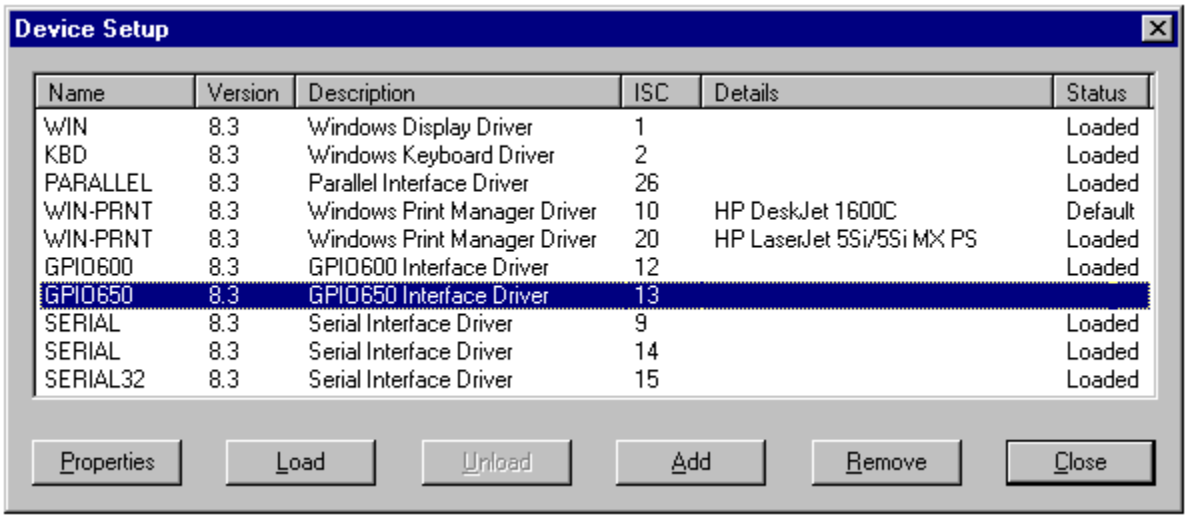
For both new and existing programs, you can use File | Print Program to print the program to a file or to a selected printer.

To print a program to a printer:

1. Click **File** | **Print Program** to display the Print Program dialog box.
2. Select the Output device and the print range desired.
3. Click OK to print the program.

Device Setup

To load a Window's print driver, select File | Device Setup. This will cause the "Device Setup" dialog box to appear. Loading drivers using LOAD BIN or equivalent statements, will cause them to appear in the list of drivers listed in the "Device Setup" dialog box. The "Device Setup" dialog box displays the Name, Version, Description, Status, and the associated ISC (if applicable) of the driver. The Status of the drivers will be left blank to indicate that they are loaded. (See Figure 2-5)



Click on the "Add" button of the dialog box. This will cause the "Device Driver Selection" dialog box to appear with a highlighted selection of "WIN-PRNT: Windows Print Manager Driver". Currently, only Window's print drivers can be added, deleted or modified through this graphical user interface. Once "Add" has been selected, the "Configure Print Driver Properties" dialog box will appear (see Figure 2-6).

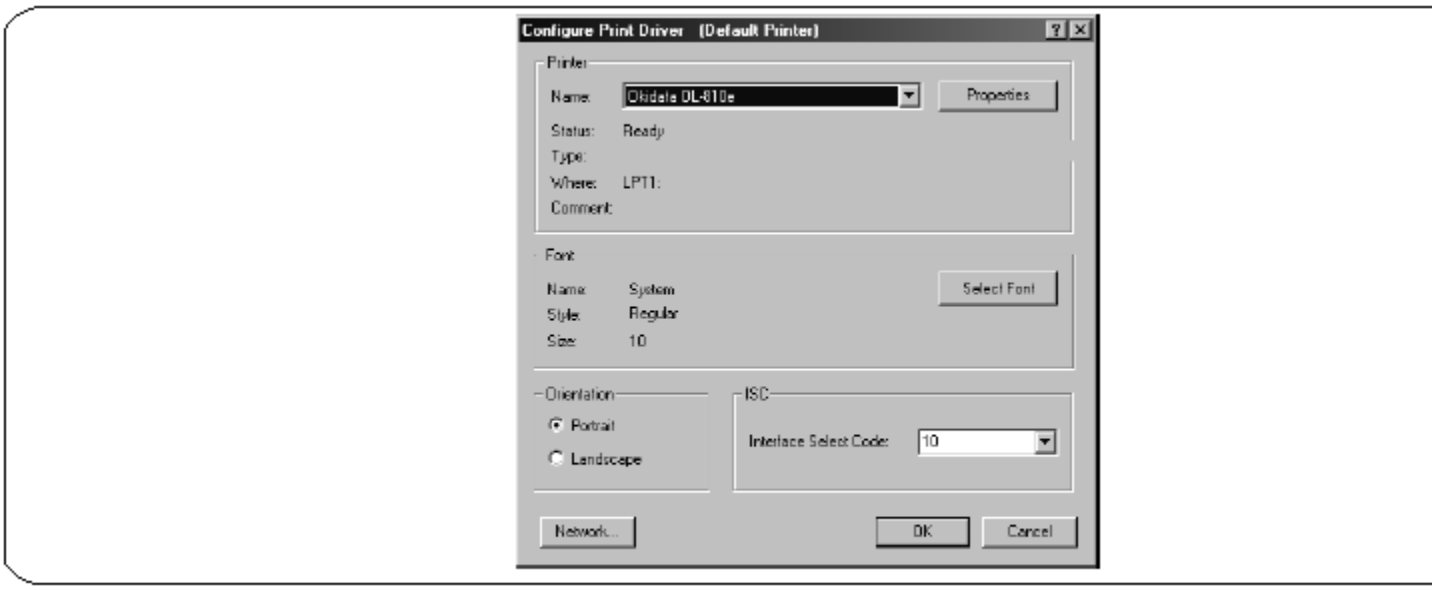


Figure 2-6. Configure Print Driver Dialog Box

In this dialog box, you can select the print driver configuration you desire. For the print driver, you can choose associations for a printer, a font, a page orientation and an ISC, as well as set properties for the printer. Once you have made the desired selections, click on the "OK" button of the dialog to confirm your choices. The new print driver will now appear among the list of drivers.

To remove a window's print driver, select File | Device Setup. This will cause the "Device Setup" dialog box to appear. Select the print driver you wish to remove from the list of print drivers and click on the "Remove" button of the dialog box (the default print driver cannot be removed). The selected print driver will now be removed from the list of drivers.

To modify the configuration for a windows print driver, select File | Device Setup. This will cause the "Device Setup" dialog box to

appear. Select the print driver you wish to modify from the list of print drivers and click on the "Properties" button of the dialog box. Once "Properties" has been selected, the "Configure Print Driver Properties" dialog box will appear. In this dialog box, you can modify the selected print driver. For the print driver, you can choose to modify any of the settings associated with the print driver, including the printer, font, page orientation, ISC and the properties associated with the printer itself. Once you have made the desired modifications, click on the "OK" button of the dialog to confirm your choices. The modified print driver will now appear among the list of drivers.

Closing a Program

Once a program has been saved/stored, you can close the program. If you have not made any changes to the program to be saved, the file is saved in the same format and with the same file name. If you attempt to close the program or exit the application without first saving, a File Changed dialog box (see Figure 2-7) is displayed.

Figure 2-7. File Changes Dialog Box

When the program has been closed, the application window reverts to the inactive mode (no program window is displayed). To close a program:

Click File | Close to display a File Changed dialog box (does not appear if no changes have been made to the program).
Click "OK" to discard the changes in the program and to delete the program window or click "Cancel" to return to the program window.

Debugging Programs

The HTBasic Debugger was designed to promote optimal programming effectiveness and flexibility using HTBasic within the Windows operating system. The Debugger tools allow the user to view the program in specific detail. The Debugger can be run by selecting the Debug | Run Debugger menu option or clicking on the "Run Debug" button on the Debug Toolbar. The key features of the HTBasic Debugger are:

- Breakpoints
- Step functions (step in, step over, step out)
- Run to/from cursor
- Independent debug windows

Breakpoints

Breakpoints provide a "pause" in program execution so that variable values and other parameter changes may be observed. The HTBasic Debugger supports line, conditional, and global breakpoints. Line breakpoints pause execution in a specified line. Conditional breakpoints pause execution at a specific line if a specified condition is met. Global breakpoints pause program execution when a specified variable reaches a specific value and condition regardless of where the program is.

To insert a line breakpoint, position the cursor in the editor on the desired line and either select the Debug | Toggle Breakpoint menu option, click the mouse arrow on the Toggle Breakpoint button in the debug toolbar, or right-click the mouse in either the Edit Window, Program Window, or the Code Window to reveal a context-sensitive menu and select the Breakpoints | Toggle Breakpoint option.

To access the Set Conditional Breakpoint Dialog Box for a conditional breakpoint, position the cursor in the editor on the desired line and either select the Debug | Conditional Breakpoint menu option, click the mouse arrow on the Conditional Breakpoint button in the debug toolbar, or right-click the mouse in the Edit, Program, or Code Window to reveal a context-sensitive menu and select the Breakpoints | Conditional Breakpoint option. In the Set Conditional Breakpoint Dialog box, enter appropriate variable, subprogram, condition, and value information and select "OK".

To open the Set Global Breakpoint Dialog Box for a global breakpoint, simply select the Debug | Global Breakpoint menu option, click the mouse arrow on the Global Breakpoint button in the debug toolbar, or right-click the mouse in the Edit, Program, or Code Window to reveal a context-sensitive menu and select the Breakpoints | Global Breakpoint option. In the Set Global Breakpoint Dialog box, enter appropriate variable, subprogram, condition, and value information and select "OK".

Step Functions

Three step functions (step in, step over and step out) provide a quick and easy way to methodically debug code. Step functions allow the user to move through and observe program execution. "Step in" steps one line of code at a time, following all program branches. "Step out" continues to end of context, stopping when entering the calling context. "Step over" runs the entire sub context and stops at the next executable line of the current context. "Step out" and "Step Over" will stop at all breakpoints in sub contexts.

The step functions can be run by selecting the Debug | Step Over, Step Into or Step Out menu option or clicking on the desired step function button on the Debug Toolbar or by right clicking in the Program or Code Window for the context-sensitive pop up menu options.

Similar in functionality to Step in, programmatically the STEP function key (Alt-F1) executes one line of the program, displays the next program line in the message area, and then pauses. If the program is not currently paused, the first press of this key causes the program to be prerun. If no prerun errors are detected the first program line is displayed in the message area. The next press of the STEP key executes the first program line, displays the next program line in the message area, and then pauses. The STEP function key steps into subprograms one line at a time.

Run to / from Cursor

Running to the cursor and continuing from the cursor provide a fast and flexible way to move through code. Run to Cursor and Continue from Cursor are similar in functionality to a breakpoint with the cursor acting as a breakpoint. Run to / from Cursor is governed by the same rules as the CONTINUE command.

Run to Cursor can be run by selecting the Debug | Run To Cursor menu option clicking on the Run To Cursor button on the Debug Toolbar or by right clicking the mouse in the Program or Code Window for the context-sensitive menu options. Continue from Cursor can be run by selecting the Debug | Continue From Cursor menu option, clicking on the Run from Cursor button on the Debug Toolbar or by right clicking the mouse in the Program or Code Window for the context-sensitive menu options.

Debug Windows

One of the most powerful tools in the Debugger is the new Debug Windows (See Figure 2-8). Six new windows are provided to assist in debugging programs by permitting the user to monitor variables, subroutines, breakpoints and the BASIC source code as it executes.

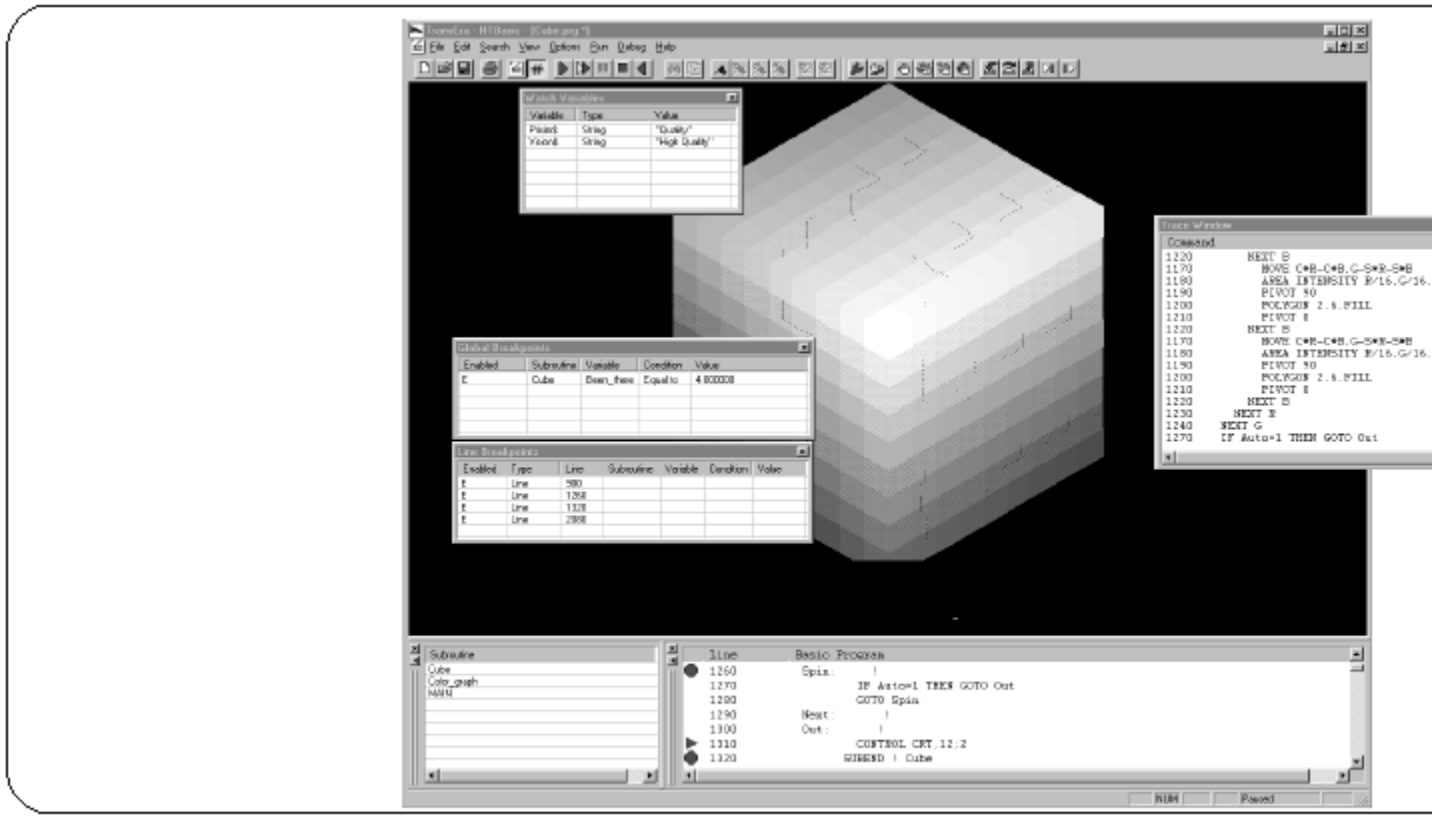


Figure 2-8. Debug Windows

The Debug Windows Dialog Box is activated from the View | Debug Windows menu option. Once opened, the user may choose which window or windows to activate by checking the appropriate box. When first opened, the debug windows are docked at the bottom of the program window.

All Debug Windows may be moved, resized and rearranged to suit the working style of any programmer. To move, simply double-click the two vertical bars (horizontal top bars if vertically docked) to create a floating window, then click on the title bar and drag to desired location. Please note the change in window outline that signifies a new docking position. To return to a previous docked position, double-click on the title bar. To resize, move the mouse cursor to the edge or corner of the window until a double-arrow cursor appears, then click and drag the cursor to the desired window size. If two or more windows are docked together, a docked window control button is made available. This enables both expanding and contracting the docked window.

The Watch Window permits the programmer to watch the values change during the program run for the list of user-defined variables at each step of the program. There are "Variable" (variable name), "Type" and "Value" columns in the Watch Window. Type can be Array, Integer, Real, Complex, String, Long or I/O Path. Watch variable names are case sensitive.

To add variables to the Watch Window, select the Debug | Add Watch Variable menu option. Watch variable names can be changed by double-clicking on the name. Watch variable values may be changed by clicking on the value. To remove a variable, select the variable in the Watch Window, then select the Debug | Remove Watch Variable menu option or select the Debug | Remove All Watch Variables menu option to remove all watch variables. Adding variables may also be performed from a pop-up, context-sensitive menu accessed with the right mouse button in the Edit and Watch Windows. Watch variables may also be removed by right clicking the mouse in the Watch Window for context-sensitive menu options.

The Line Breakpoints Window allows the user to observe the line breakpoints as the program is run. Breakpoint parameters monitored are "Enabled", "Type", "Line", "Subroutine", "Variable", "Condition" and "Value". Line breakpoints may be individually enabled and disabled from this window by highlighting the breakpoint and right-clicking the mouse to access the context-sensitive menu. Now select Enable-Disable Breakpoint. To remove a breakpoint, highlight the breakpoint to be removed, right-click and select Remove Breakpoint. To remove all breakpoints, select Remove All Breakpoints.

The Global Breakpoints Window permits the user to observe the global breakpoints as the program is running. The Global Breakpoint Window monitors "Enabled", "Subroutine", "Variable", "Condition" and "Value" status. Global breakpoints may be added from this window. Simply right click in the window area to access the pop-up menu, select Add Global Breakpoint to open the Set Global Breakpoint Dialog box, which may also be accessed from the Global Breakpoint button on the toolbar or from the Debug | Global Breakpoint menu option. Global breakpoints can be individually enabled and disabled from this same context-sensitive menu. Highlight a breakpoint, right-click the mouse and select Enable-Disable Breakpoint. To remove a breakpoint, highlight the breakpoint to be removed, right-click and select Remove Global Breakpoint. To remove all breakpoints, select Remove All Breakpoints.

The Trace Window permits the user to observe which commands are being executed in a running program. There is nothing to "set" in this window. It automatically monitors and notes each HTBasic command line as it is executed. Only a "Command" column exists in the Trace Window.

The Trace Window differs fundamentally from HTBasic's TRACE Statement in that the Trace Window provides a running trail of commands executed. The TRACE Statement is limited to only what appears on the message line before it scrolls away. Please note that the TRACE Statement tracks only the commands executed: it tracks neither breakpoints nor subroutines.

To clear the Trace Window, in the window right-click the mouse and select Clear Window. Restarting a program will also clear the Trace Window.

The CALL Stack Window was designed to open a view into the CALL Stack (the CALL Stack is used by BASIC to track subroutines accessed by CALL statements) so that one can see it operate at each step of the running program. There is nothing to "set" in this window. It automatically monitors and notes what is going on in the program defined CALL Stack as the program is running. Only a "Subroutine" column exists in the CALL Stack Window.

The Code Window displays the program source code as the program is running. There are "Line" (Number) and "BASIC Code Lines" columns in the Code Window. These columns show the Line Number within the BASIC source code program and the actual text of the code lines. Breakpoints, Bookmarks and the program pointer (showing exactly from where the computer is executing code) are seen in this window as well.

The Code Window differs from the Trace Window in that the Trace Window is simply a stepwise display of program lines as they run, showing only those lines of code, which have already executed. The Code Window may be used to browse forward or backward through the source code, including those lines of code, which have not yet executed. The Code Window can also be used to view and modify breakpoints within the BASIC source code program at the time of the debug run.

TRACE Statement

The TRACE statement controls the display of trace information from a running program and can pause program execution before executing a specific program line. The trace output is sent to the system message line. The trace output is also sent to the PRINTALL IS device if PRINTALL is enabled. Tracing slows program execution.

TRACE ALL enables program tracing. Either the entire program or just a range of program lines may be traced. For example:

```
TRACE ALL 1000,1200
```

enables tracing during the execution of program lines 1000 through 1200. The trace output displays the program line before it is executed and any modified simple variables or array elements and their new values. If a full array is modified then only the array name is displayed. TRACE OFF form turns off all tracing. TRACE PAUSE will pause program execution before the specified program line is executed. If no line number or label is specified, the program pauses before the next program line is executed and the current TRACE PAUSE line is deactivated. For example:

```
TRACE PAUSE 250
```

will pause the program before line 250 is executed. Used in conjunction with the HTBasic Windows Debugger, these statements will enable you to find errors in your programs quickly and efficiently, cut debugging time, and increase productivity.

Running Programs

Once a program has been created, you can run the program to produce the program output. This section gives guidelines to get started running programs, including these topics.

- Running a Program
- Pausing a Program
- Stopping a Program

Running a Program

When a new program has been created or an existing program has been opened, you can use Run | Run from the program window menu, use the Run (right-facing green arrow) icon on the control toolbar or right click in the Edit, Program, or Code Window for a context-sensitive Run menu option.

Example: Running/Pausing a Program

This example shows a way to run and pause the example.prg program.

Open the example.prg program.
Click Run | Run or the Run icon to run the program.
Click Run | Pause or the pause icon (yellow double bar) to pause the program.

Running a Program From a File

From the Applications Window, you can use File | Run Program... to run a program directly from a file without opening the file. You can run the program before you open any other files, or you run the program when the application is in edit mode. When File | Run Program... is executed, a standard Open File dialog box appears from which you can select a program to be run directly from the file, without having to first open the file. Also, dropping a PROG file onto the output window will open and run the prog file.

Pausing Programs

You can pause a running program by using the Run | Pause menu, pushing the pause (yellow double bar) icon, right click in the Program or Code Window for a context-sensitive menu Pause option, or using the System F4 menu key.

Stopping Programs

You can stop a running program by using the Run | Stop menu, right click in the Program or Code Window for a context-sensitive menu Stop option, or pushing the stop (red square) icon in the Control Toolbar.

When the program stops (or is paused), you can rerun the program by using the Run | Run menu, or clicking the Start (right-facing green arrow) icon in the control toolbar or by clicking the Run (F3) softkey from the System Menu softkeys.

Search and Replace Operations

This section gives guidelines to edit HTBasic for Windows programs, including these topics.

- Moving in the Program Window
- Finding Items
- Replacing Items
- Going to Items

Moving in the Program Window

In the previous section we used the program window primarily to display the program code, but did not make any edit changes to the program.

The GUI uses Windows actions for using the mouse. Clicking the mouse on text that you wish to edit highlights the letter you selected. The mouse can also be used to move the cursor in the program by dragging the vertical scroll bar on the right side of the program window. If your mouse has a wheel, the wheel can be used to move the cursor as well.

Finding Items

When the application is in the program window, you can use Search | Find... menu or press the Find button in the Search Toolbar to find a specified item in the open program. Clicking this menu item displays a standard Find dialog box (see Figure 2-9). When a string is typed into the "Find What" box, the search will look for the next or previous use of the string.

Figure 2-9. Find Dialog Box

All searches begin at the point where the search is initiated, by default, and search forward through the code. Selecting a search direction in the Find Dialog box allows the user to search forward (down) or backward (up). To find the first occurrence of an item, the search must start at the beginning of the program.

To narrow the search, in the Find Dialog box select "Match the whole word only", or select "Match Case" (case sensitive), or select "Wrap at beginning/end" to continue your search from the beginning/end of the program back to the start of the search.

To find multiple instances of a specified item, simply push the "F3" key or select the Search | Find Next menu or push the "Shift + F3" key or select the Search | Find Previous menu after finding the first occurrence of the search. The search function automatically takes the cursor to the next/previous occurrence of the string.

The FIND command also searches for a specified character sequence (case sensitive) in a program. Consult the on-line Reference Manual for details regarding the FIND command.

Replacing Items

Use Search | Replace... to replace one or more occurrences of an item. Selecting the Search | Replace... menu or pressing the Replace button in the Search Toolbar displays the Replace dialog box as shown in Figure 2-10.

Figure 2-10. Replace Dialog Box

In the Replace Dialog box, clicking either "Find Next" or "Replace" highlights the next instance of the searched text. Pressing the "Find Next" key again searches for the next instance without making a replacement. Pressing the "Replace" key again makes the replacement and moves the cursor to the next instance of the searched string.

To replace a selected item in the program using the Replace Dialog box:

Type in the text to be changed in the "Find What:" box and type in the desired text in the "Replace With:" box.
Choose to "Find Next", to locate the text to replace. Choose "Replace" to have each change confirmed, or "Replace All" to make the change throughout the program without querying the user.
If desired, click "Cancel" to quit.

If a change is required in a specific range, first highlight the range with the mouse and check the Replace in | Selection option in the Replace Dialog Box before executing the search and replace operation. Selecting the "Whole File" option (default) performs the Replace function selected to the entire file.

Editing the line found by the replace operation is not allowed while the Replace Dialog box remains open. However, once closed, the Replace Dialog box will save and re-open to the previous replace option.
Programmatically, the CHANGE command allows you to search and replace character sequences. The CHANGE command is described in detail in the on-line Reference Manual.

Going to Items

In the edit mode, clicking Search | Goto... displays the Goto dialog box (see Figure 2-11). From the Goto dialog box, you can select a line number (check Line), a line label (check Label), a function (check FN) or a Subprogram (MAIN is considered to be a subprogram) (check Sub) as the Goto Item.

Example: Going to a Subprogram

For the example program illustrated in Figure 2-11, clicking Search | Goto... or pressing CTRL+G, selecting "Sub" in the Goto Item box (see Figure 2-11), entering SUB Stress_analysis as the Sub request, and clicking OK moves the cursor to the "Stress_analysis" Subroutine.

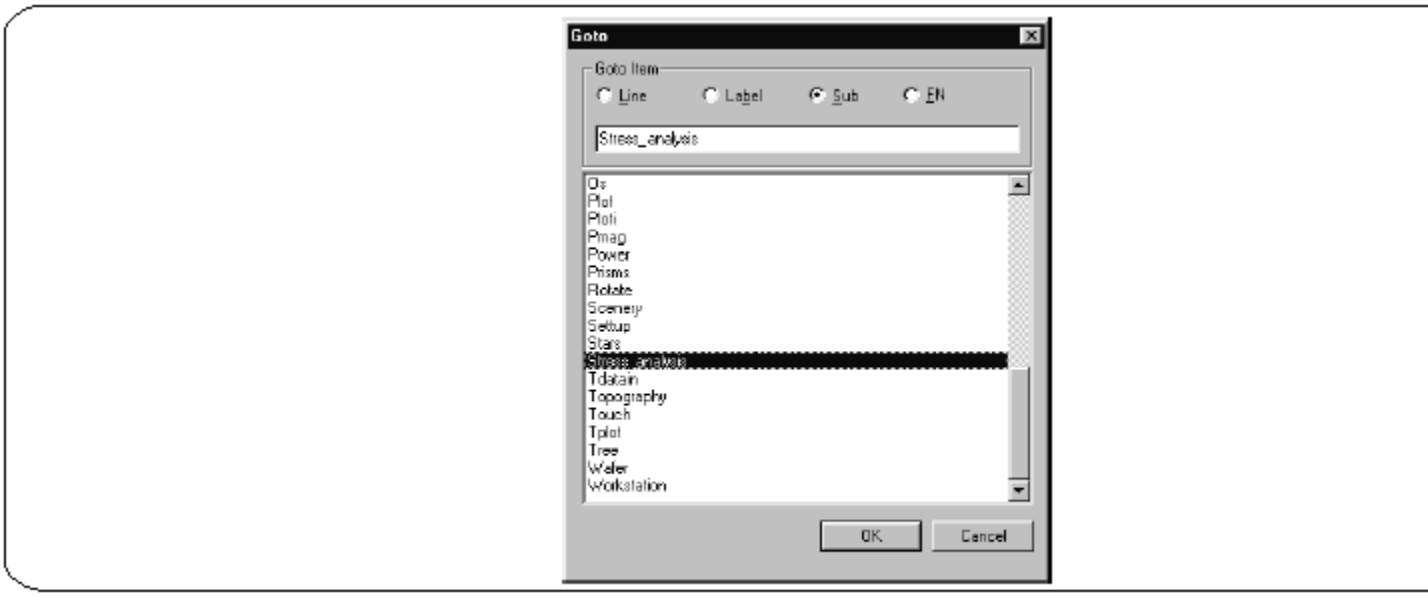


Figure 2-11. GOTO Dialog Box

GUI Description

This chapter describes the HTBasic for Windows Display and Graphical User Interface (GUI), including:

- Introducing the GUI
- Application Window Description
- Program Window Description
- GUI Menus
- Control Characters
- Setting the Environment

Introducing the GUI

In addition to the traditional RMB environment, all program development, editing, and debugging in HTBasic for Windows can now be done via the Graphical User Interface (GUI). The GUI provides an easy, flexible way to develop and edit HTBasic for Windows programs using standard Windows techniques.

The GUI is fully compatible with previous versions of HTBasic for Windows and with programs developed in earlier versions of the HP BASIC language.

GUI Windows

There are two windows for HTBasic for Windows, the application window (see Figure 3-1) and the program window (see Figure 3-2). Depending on the operating mode of the program, some window elements change.

The application window provides the menus, toolbar, and status bar that can be used for program development. The program window is enclosed in the application window and provides the environment in which you can create, edit, and run programs.

Application Window Description

When HTBasic for Windows first starts, the application window (and an enclosed Program Window) appears. The application window provides a framework window from which you can develop HTBasic for Windows programs. This section give guidelines to use the GUI application window.

Application Window Display

Figure 3-1 shows a typical application window display when no programs are open (inactive state). Figure 3-2 shows a typical application window display when a program is open (active state).

For inactive state, only four of the eight menus are displayed on the application window.

Only the “New” and “Open” icons on the control toolbar are operational.

In inactive state, the application window does not have an enclosed Program Window.

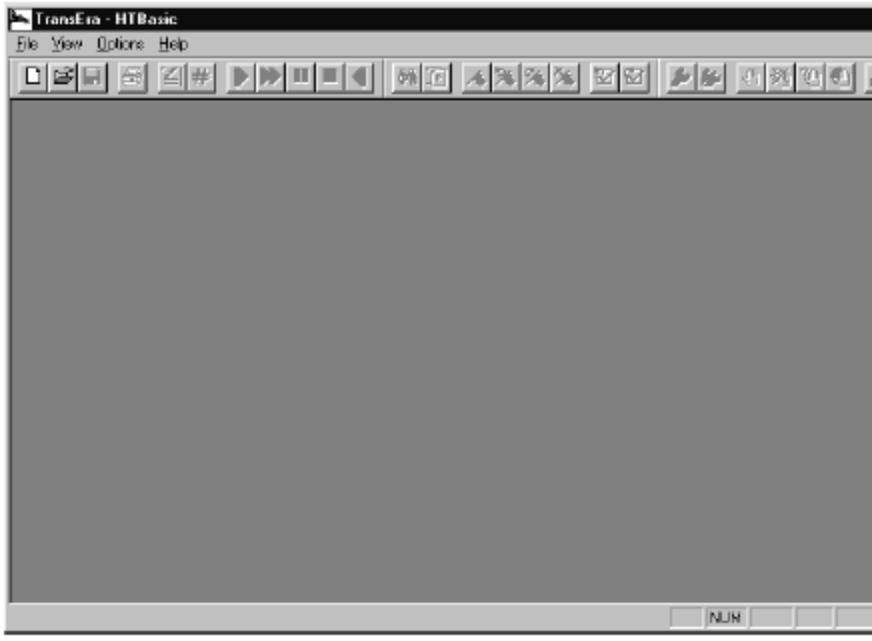


Figure 3-1. Typical Application Window - Inactive State

There are two main differences in application window displays for active state and inactive state. In the inactive state, the menu bar has only four menus and no program window is displayed. The inactive state occurs when no program is open. In the active state, the full menu bar of eight menus is displayed and the Program Window is displayed.

Although the application window contains the title bar, menu bar, toolbars, status bar and status indicators, they are more functional with an active program window. These features are explained in detail in the following section.

For the active state operation, the full menu bar and the enclosed Program Window are displayed.

The control toolbar is operational.

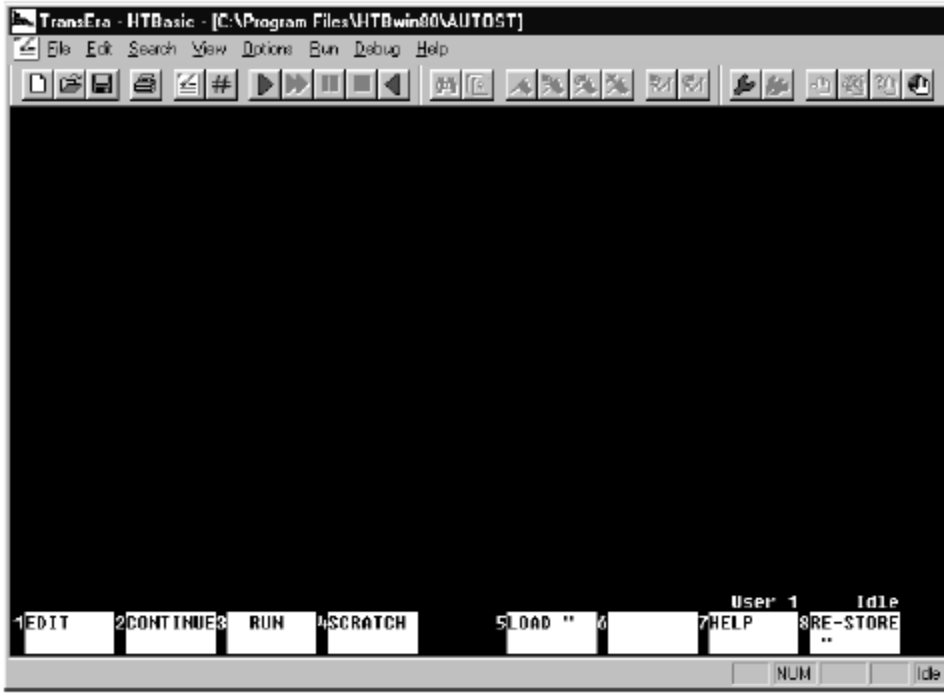


Figure 3-2. Typical Application Window - Active State

Program Window Description

The Program Window provides the area used to develop and edit programs. The Program Window has no associated menus other than a context menu for the title bar. However, the menus are fully functional with an active program window. This section describes the program window, including:

- Program Window Elements
- Positioning the Window
- Using the Status Bar
- Using the Toolbars
- Moving the Toolbars

Program Window Elements

The HTBasic display operates in one of two modes, Normal, and Edit. The Normal display mode is used while HTBasic is idle, waiting for a command, or while a program is running. The EDIT display mode is used by the HTBasic Windows Editor, the HTBasic Legacy Editor, and the Debugger. Figure 3-3 shows the elements of a typical HTBasic Program Window and the following table contains a description of each of the features.

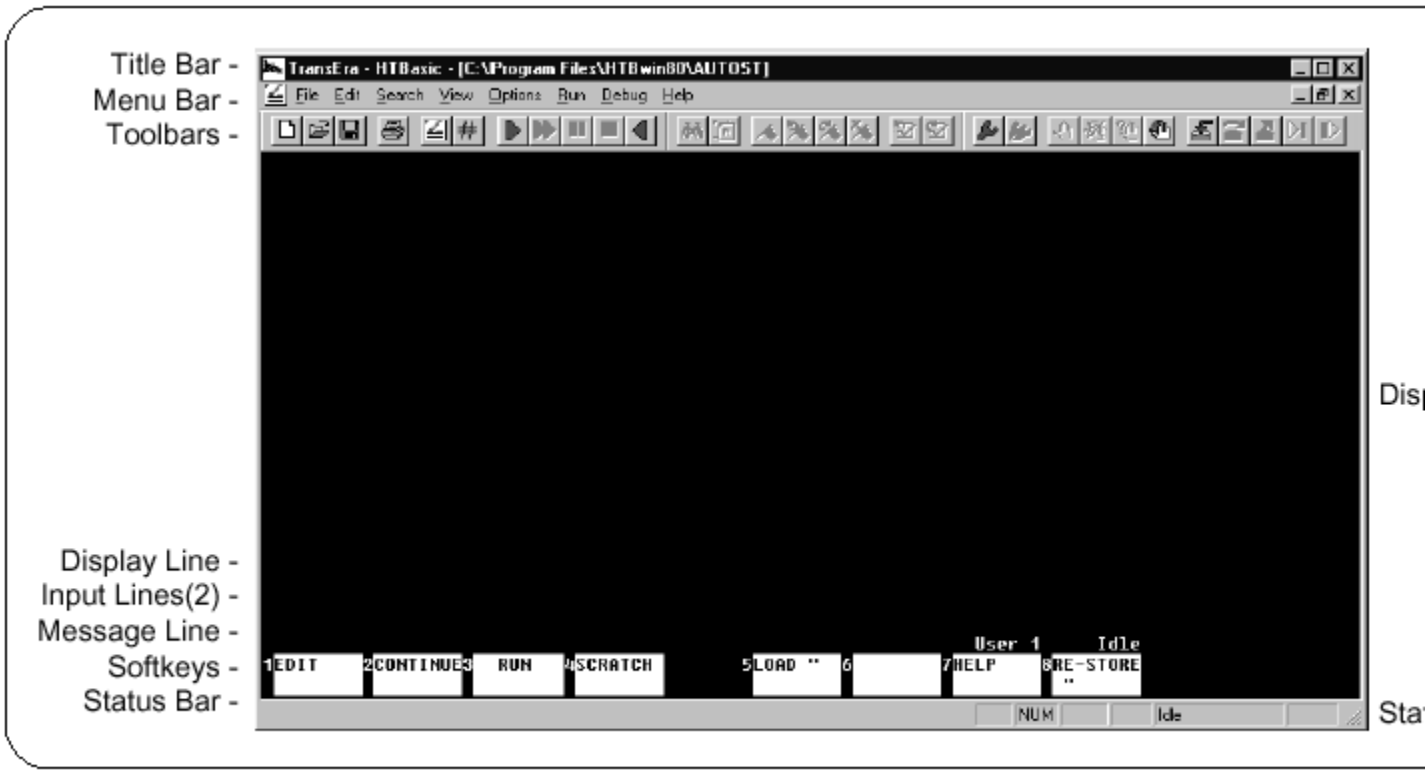


Figure 3-3. Application Window Elements

Program Window Elements

Title Bar The title bar consists of the application icon, the name of the application, the program name, and the minimize, maximize, and close buttons. An asterisk (*) is appended to the program title if the source or program has been modified since the last save. The program window title bar is hidden when the program window is maximized (default), but the program name and location is shown in the application window's title bar.

Menu Bar The menu bar contains the pulldown menus you can use for program development. Some items are not always available. See "GUI Menus," page 3-12, for details on the menus.

Toolbar(s) A toolbar contains images that represent shortcuts for certain menu items. By clicking a toolbar button, the action performed is the same action as if selected from the menu. A toolbar is shown when the desired toolbar is checked in the View menu. A toolbar is hidden when it is unchecked in the View menu. See "Using the Toolbar" for details.

Input Line The input line is used to enter commands, program lines, calculator expressions, and other keyboard input. When ENTER is pressed the Input line buffer is sent to either the INPUT, LINPUT, or ENTER KBD statement or to the command and statement parser.

Vertical Scrollbar The vertical scrollbar allows scrolling up or down in a program using standard Windows operation. Scrolling can also be done automatically when moving through text. When the caret moves beyond the window boundaries, the window scrolls accordingly. If the mouse has a wheel, the wheel can also be used to move the cursor.

Display Line The Display Line is used for displaying prompts for the INPUT and the LINPUT statements and for

displaying text with the DISP statement. The display line maintains a current print location and may be set by the CONTROL CRT,8 statement. The DISPLAY ALL function has no effect on the Display Line and other control characters are not stored in the display line buffer. The display line buffer content may be read by a program using SYSTEM$("DISP LINE").

Message Line The message line displays error messages, live keyboard calculator results, and the current program state. The current state identifies whether the HTBasic program is running, paused, stopped, or waiting for input. Messages are limited to 63 characters in length.

Softkey Menu A softkey is a function key (F1 — F 10) with programmable output that displays labels corresponding to each softkey. The label indicates the action that is performed when the corresponding function key is pressed. Softkeys can be hidden or displayed. There are several sets of softkeys that can be displayed including the system softkeys and three sets of user softkeys. There are two softkey layouts available that correspond to either the HP Nimitz or ITF keyboard.

Display Area The display (output) area is always present during the entire operation of the application. The Program Window created by the application remains within the display area. This area starts at the top line of the screen below the title bar (minimized) or at the top line of the screen below the control toolbar when maximized, and extends down to one line above the message line.

Status Bar The status bar provides keyboard and current program state information. The status bar is displayed when the View | Status Bar menu item is checked or is hidden when this menu item is unchecked. See "Using the Status Bar" for details on using the status bar.

Positioning the Window

When HTBasic for Windows is saved or closed and then restarted, the application window is returned in the size and position it had when the application closed. Procedures to position and resize the application window are illustrated in Figure 3-4.

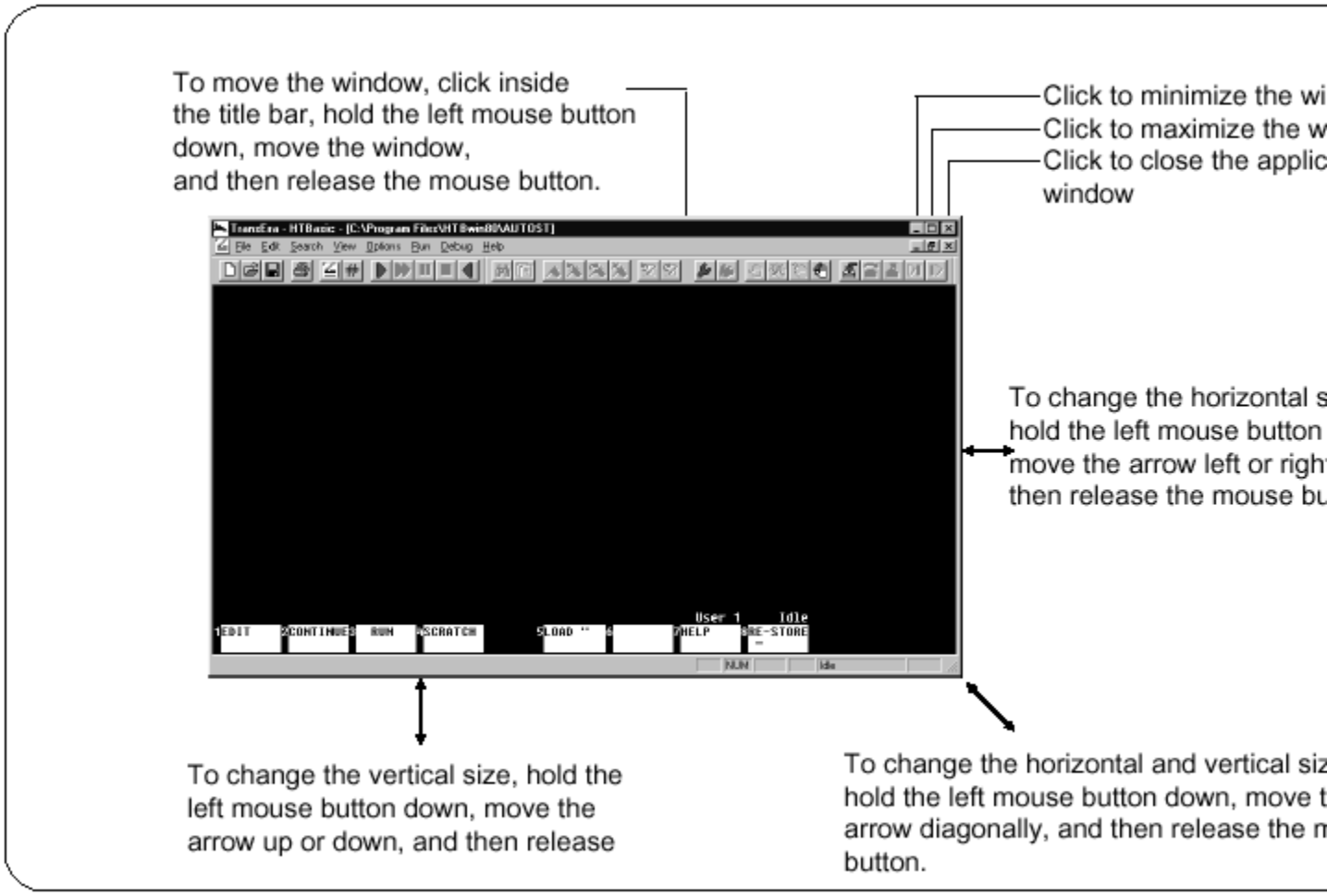


Figure 3-4. Positioning the Application Window

Using the Status Bar

The status bar (see Figure 3-5) is located at the bottom of the application window and provides information about the keyboard status and current state of HTBasic for Windows. You can show the status bar by checking the View | Status *Bar* menu item in the application window. You can hide the status bar by unchecking View | Status Bar.

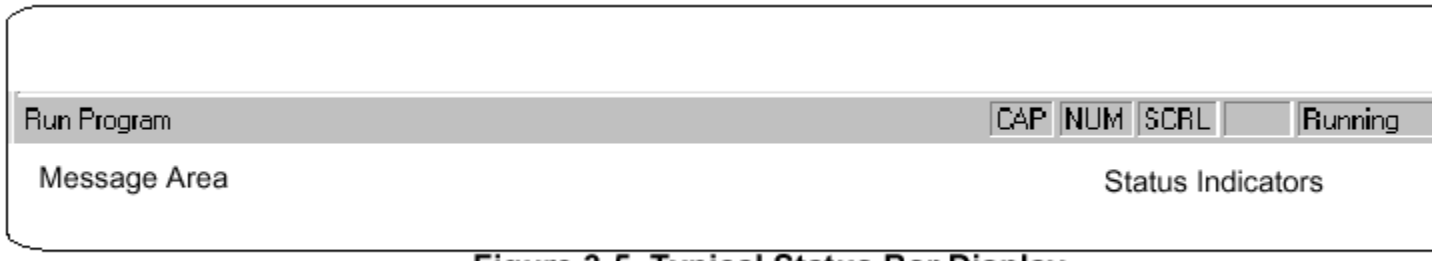


Figure 3-5. Typical Status Bar Display

The status bar includes two areas: message area and status indicators.

The message area at the left side of the status bar displays the name of the control toolbar icon when it is highlighted by the mouse (Run Program icon in Figure 3-5). It also displays the Tooltip when hovering over a toolbar button or a menu item.

The status indicators display the operating state of several keyboard keys including capslock, num lock, insert mode, or scroll lock. The status indicator also provides the current program state indicating whether the HTBasic program is running, paused, stopped, idle, or waiting for input.

Using the Toolbars

The toolbar contains images that represent shortcuts for certain menu items. By clicking a toolbar button, the action performed is the same action as if selected from the menu. When you click a toolbar icon, the icon title (such as New, Open, Find, etc.) appears and the icon description is displayed in the status bar message area. There are three toolbars: the Control, the Search, and the Debug Toolbars. The toolbars can be displayed on the application window. When the application window is in an inactive state, all toolbar buttons except "New" and "Open" are greyed out.

The Control Toolbar is shown when the View | Control Toolbar menu item is checked or is hidden when View | Control Toolbar is unchecked. The Bookmark Toolbar is shown when the View | Search Toolbar menu item is checked or is hidden when View | Search Toolbar is unchecked. The Debug Toolbar is shown when the View | Debug Toolbar menu item is checked or is hidden when View | Debug Toolbar is unchecked. These toolbars are shown in Figure 3-6. Please see the tables following figure 3-6 for complete toolbar functionality.

In addition to the ability to turn the toolbars on or off through the View menu check boxes, this may also be accomplished with the following GESCAPE codes:

```
GESCAPE CRT,46    !turns the Control Toolbar off
GESCAPE CRT,47    !turns the Control Toolbar on
GESCAPE CRT,60    !turns the Search Toolbar off
GESCAPE CRT,61    !turns the Search Toolbar on
GESCAPE CRT,62    !turns the Debug Toolbar off
GESCAPE CRT,63    !turns the Debug Toolbar on
```

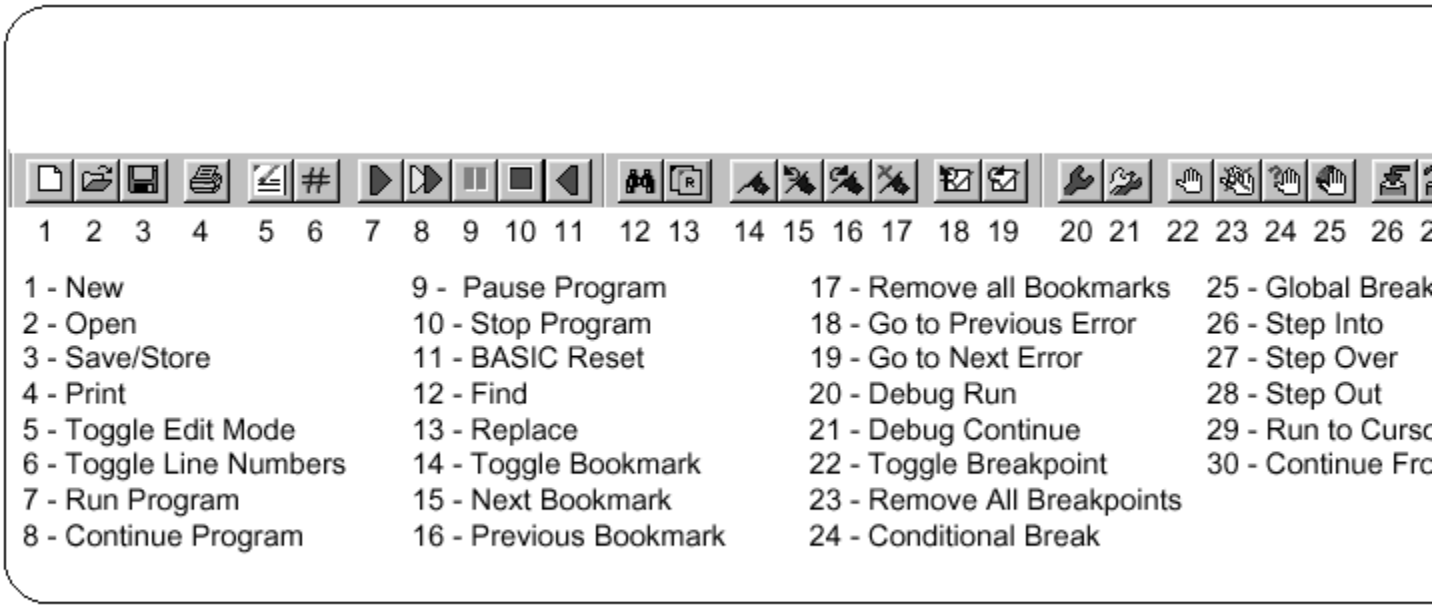


Figure 3-6. Guide to the new Toolbars

Control Toolbar Buttons

Group	Button	Name	Matches Menu / Option
File Handling	1	New	File / New
	2	Open	File / Open
	3	Save / Store	File / Save or File / Store
Print	4	Print	File / Print
Mass Edit	5	Toggle Edit Mode	Edit / Edit Mode
	6	Toggle Line Number	View / Line Numbers
Run	7	Run Program	Run / Run
	8	Continue Program	Run / Continue
	9	Pause Program	Run / Pause
	10	Stop Program	Run / Stop
	11	BASIC Reset	Run / BASIC Reset

Search Toolbar Buttons

Group	Button	Name	Matches Menu / Option
Search	12	Find	Search / Find
	13	Relplace	Search / Replace
	14	Toggle Bookmark	Search / Bookmark /Toggle Bookmark
	15	Previous Bookmark	Search / Bookmark / Previous Bookmark
	16	Next Bookmark	Search / Bookmark / Next Bookmark
	17	Remove all Bookmarks	Search / Bookmark / Remove All Bookmarks
	18	Go to Previous Error	Search / Bookmark / Previous Error
	19	Go to Net Error	Search / Bookmark / Next Error

Debug Toolbar Buttons

Group	Button	Name	Matches Menu / Option
Debug	20	Debug Run	Debug / Run Debugger
	21	Debug Continue	Debug / Continue Debugger
Breakpoint	22	Toggle Breakpoint	Debug / Toggle Breakpoint
	23	Remove All Breakpoints	Debug / Remove All Breakpoints
	24	Condition Break	Debug / Conditional Breakpoint
	25	Global Break	Debug / Global Breakpoint
Step	26	Step Into (Ctrl+F1)	Debug / Step Into (Ctrl+F1)
	27	Step Over (Ctrl+F2)	Debug / Step Over (Ctrl+F2)
	28	Step Out (Ctrl+Shft+F1)	Debug / Step Out (Ctrl+Shft+F1)
	29	Run to Cursor	Debug / Run To Cursor
	30	Continue from Cursor	Debug / Continue From Cursor

Example: Using the Control Toolbar

Placing the arrow cursor over the Run icon (icon #7) as shown in Figure 3-7 causes the icon title (Run) to appear beneath the icon and the icon action (Run program) to appear at the left-hand side of the status bar. Clicking the Run icon executes the icon action (Runs the Program).

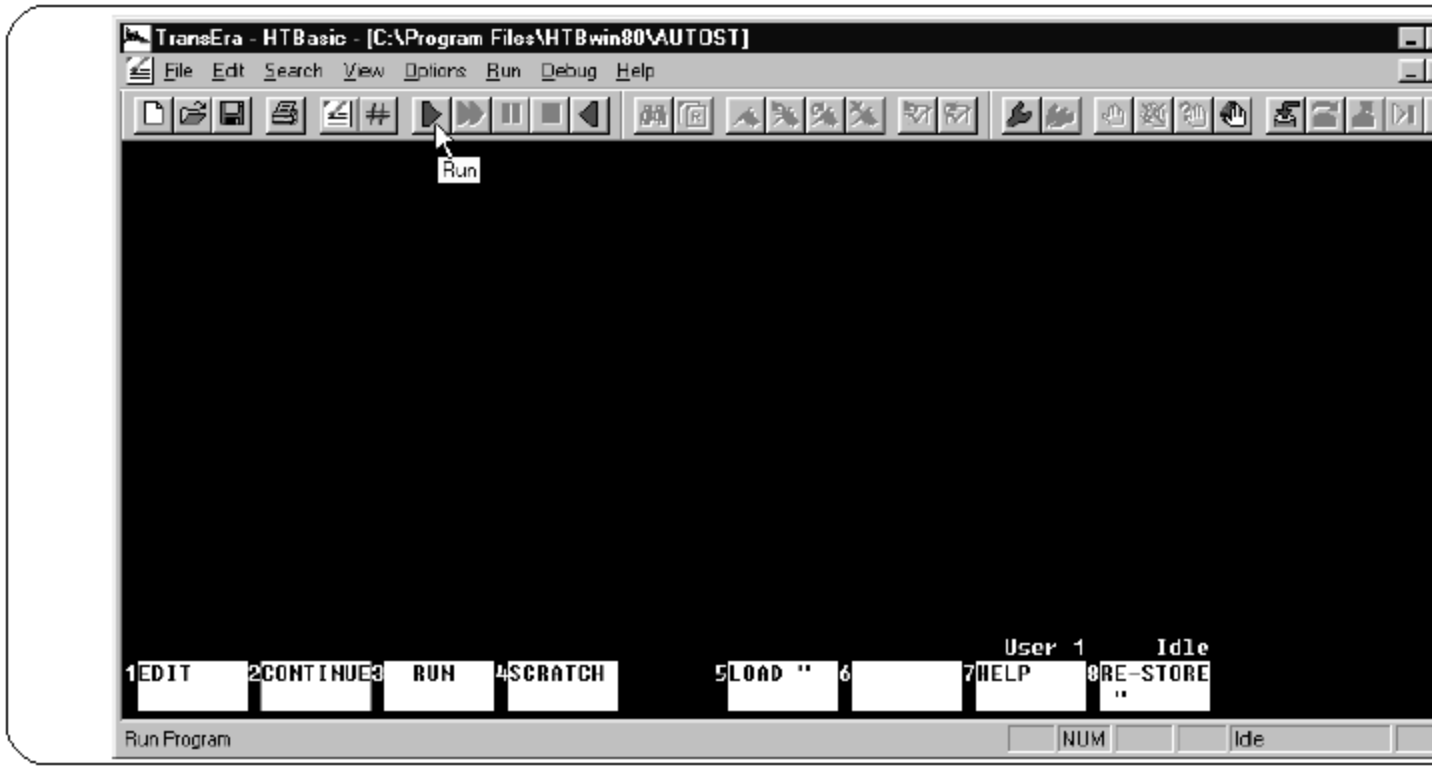


Figure 3-7. Example: using the Control Toolbar

Moving the Toolbar

When a Toolbar is displayed, it can be displayed below the menu bar (default), it can be docked to any side of the application window border, or it can be floating anywhere on the screen, including outside the main application window. The title bar for the toolbar is present when the toolbar is in a floating state.

To move a toolbar, click anywhere inside the toolbar outline (except on an icon), hold the mouse button down, drag the outline to the desired location and then release the mouse button. Figure 3-8 shows a floating Control Toolbar in the right margin of the application window when the application is in the active state. Since the Control Toolbar is floating, the title bar for the Control Toolbar is displayed.

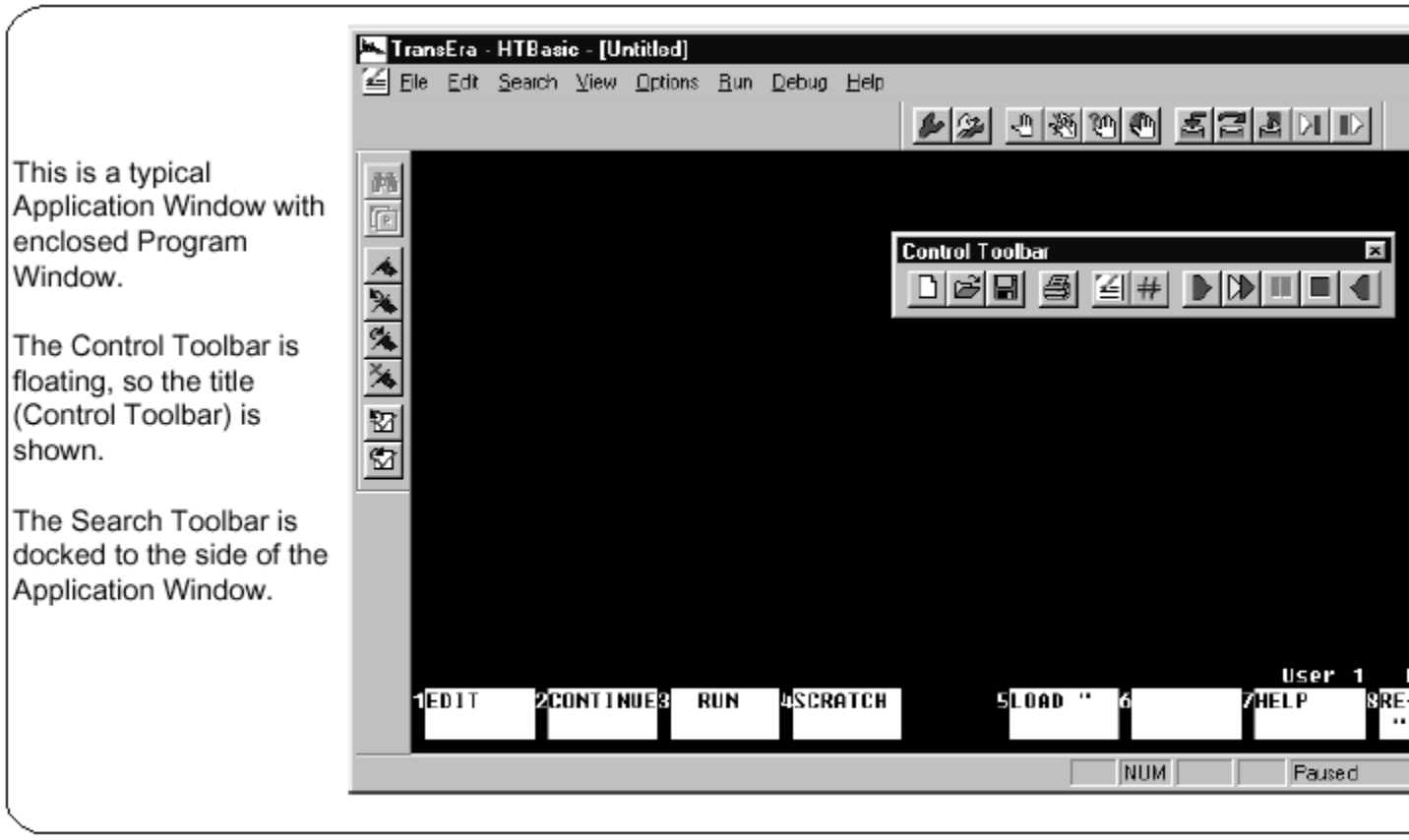


Figure 3-8. Example: Moving the Toolbar

To dock a toolbar to the left or right side or to the top or bottom of the application window, hold the mouse button down and drag the outline left or right until the outline changes to a vertical rectangle. Release the mouse button and the toolbar is docked to the side of the window. When the application is exited and then restarted, the toolbars are displayed in their exit state locations.

GUI Menus

This section describes the menus for HTBasic for Windows, including:

- Menus Overview
- File Menu
- Edit Menu
- Search Menu
- View Menu
- Options Menu
- Run Menu
- Debug Menu
- Tools Menu
- Help Menu

Menus Overview

HTBasic for Windows includes eight pulldown menus you can use to develop HTBasic for Windows programs. Figure 3-9 shows a typical menu display and an expanded view of the Run menu. The actual display for all the pulldown menus depends on the type of window displayed. Each pulldown menu uses standard Windows displays, including shortcut keys (Atl+F+O for Open... etc.) and accelerator keys (Ctrl+O for Open... etc.).

All menu items are displayed for each pulldown menu, but the inactive items for a specific mode of operation are greyed out. For example, in Figure 3-9 the menu items Continue, Pause, and Stop, are shown as inactive.

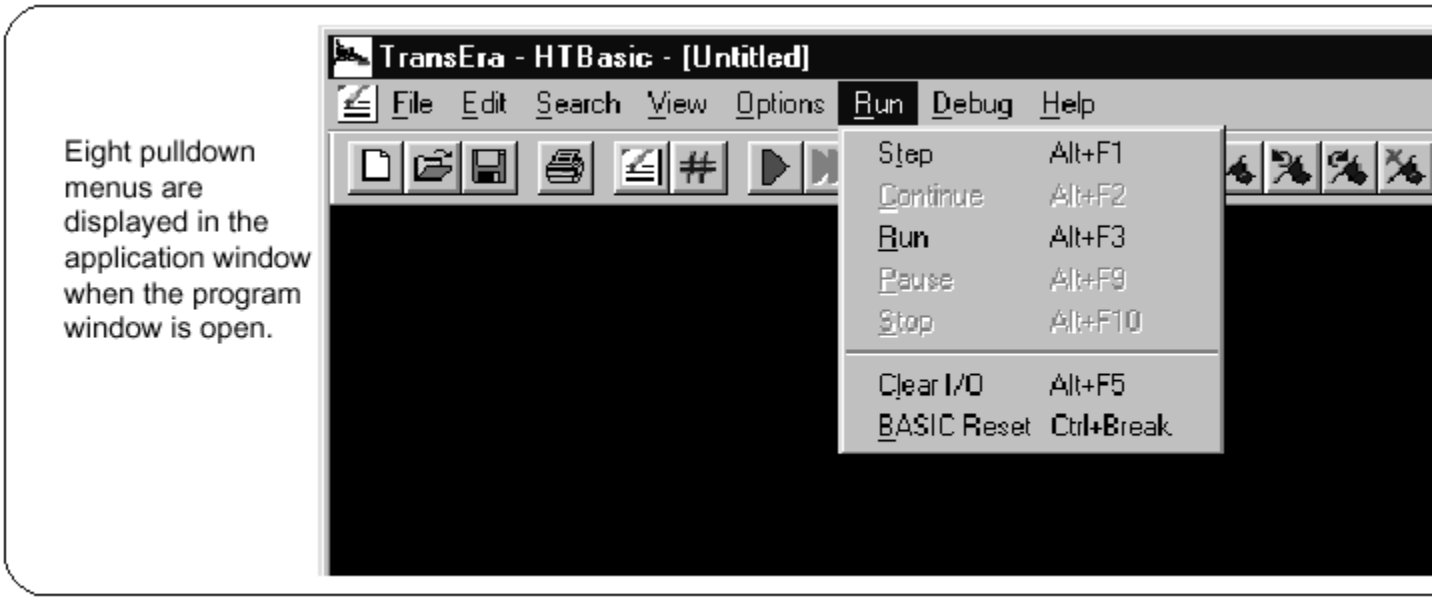


Figure 3-9. Typical Menu Display

For each menu or menu item, the convention used is Menu | Menu Item. For example, File | Open... refers to the Open menu item in the File menu. A menu item followed by three dots (such as File | Open...) means that a dialog box is opened when the menu item is selected. The underlined letter is the menu mnemonic key (such as *F*) or menu item mnemonic key (such as *O*). Accelerator keys are things like Ctrl+O.

There are four main ways to access a menu item as described in the following table.

Method	Description
Use the Mouse Cursor	You can use the mouse cursor to open the menu and then highlight the menu item. Click the mouse button or press Enter to perform the menu item action.
Use the Alt Key to get the mnemonic	For some menu items, you can use Alt+menu item accelerator key to perform a menu item action. To do this, first open the menu with Alt+menu accelerator key but do not release the Alt key. Continue to hold down the Alt key and press the menu item accelerator key. The key sequence shown in the menu tables is Alt+menu accelerator key+menu item accelerator key.
Use Accelerator Key	Accelerator keys are similar to Alt keys except they are not associated to a menu mnemonic. These keys provide direct access to the operation.
Use Toolbar Button	The toolbar contains images that represent shortcuts for certain menu items. By clicking a toolbar button, the action performed is the same action as if selected from the menu.

Example: Using File | Open Menu

For example, suppose you want to open a file using the File | Open menu item. You can use any of the following four methods, each of which results in the Open dialog box being displayed.

Use the mouse cursor to open the File menu and to highlight File | Open... Click the mouse button or press Enter to display the Open dialog box.

Use Alt+F to open the File menu - do not release the Alt key.

Then, press O (or o) to display the Open dialog box.
Press Ctrl-O to display the Open dialog box.
Click on the Toolbar Button to display the Open dialog box.

File Menu

The File menu appears in both the application and program windows.

File Menu (Application Window)

Menu Item	Shortcut	Description	Alt
New	Ctrl + N	Opens a new Program Window	Alt+F+N
Open...	Ctrl + O	Allows user to open a new file	Alt+F+O
Run Program...		Allows user to run a program directly from a file	Alt+F+R
Recent File List 1...10		Allows opening one of up to ten recently opened files	Alt+F+1...10
Change MSI on open		Allows user to change Mass Storage Is location on opening a new file from the Recent File List. (This is not set by default)	Alt+F+h
Reset File List...		Allows user to clear the recent file list	Alt+F+L
Exit		Exit the application	Alt+F+x

File Menu (Program Window)

Menu Item	Shortcut	Description	Alt
New	Ctrl + N	Greyed out - inactive	Alt+F+N
Open...	Ctrl + O	Opens the Open File Dialog box	Alt+F+O
Close	Ctrl + F4	Closes the Program Window	Alt+F+C
Save...		Saves program to existing file	Alt+F+S
Save As...		Saves program to a user-specified file	Alt+F+A
Store...		Stores program to existing file	Alt+F+t
Store As...		Stores program to a user-specified file	Alt+F+e
Revert to last		Reverts program to previously saved version	Alt+F+v
Print Program...		Allows current document to be printed	Alt+F+P Alt+F+D
Recent File List 1...10		Allows opening one of up to ten recently opened files	Alt+F+1...10
Change MSI on open		Allows user to change Mass Storage Is location on opening a file from the Recent File List. (This is not set by default)	Alt+F+h
Reset File List		Allows user to clear the recent file list	Alt+F+L
Exit		Exit the application	Alt+F+x

Edit Menu

The Edit menu appears only in the program window.

Edit Menu (Program Window)

Menu Item	Shortcut	Description	Alt
Edit Mode	Ctrl+E	Allows user to move programs into the Edit Mode or back into normal mode (check box)	Alt+E+E
Undo	Ctrl+Z	Allows user to Undo immediately previous action	Alt+E+U
Redo	Ctrl+Y	Allows user to Redo immediately previous Undo action	Alt+E+R
Cut	Ctrl+X	Allows user to Cut selected text from program to clipboard	Alt+E+t
Copy	Ctrl+C	Allows user to Copy selected text from program to clipboard	Alt+E+C
Paste	Ctrl+V	Allows user to Paste selected text from clipboard to program	Alt+E+P
Delete	Del	Allows user to Delete selected text from program	Alt+E+D
Set Indent...		Displays the Set Indent Dialog Box to determine Indent limits	Alt+E+S
Indent All	Ctrl+I	Allows user to Indent all appropriate lines in the program	Alt+E+I
RENumber		Displays the RENumber Dialog Box	Alt+E+R
Comment		Adds comments to the beginning of currently highlighted lines	Alt+E+m

Search Menu

The Search menu appears only in the program window.

Search Menu (Program Window)

Menu Item	Shortcut	Description	Alt
Find...	Ctrl+F	Allows the user to Find a selected string	Alt+S+F
Find Next	F3	Find the Next occurrence of a selected string	Alt+S+N
Find Previous	Shft+F3	Find the Previous occurrence of a selected string	Alt+S+P
Replace…	Ctrl+H	Allows user to define search and Replace options	Alt+S+R
Goto...	Ctrl+G	Allows user to Goto a line, label, function or a subprogram	Alt+S+G
Bookmarks		Allows users to add, remove, or move to Bookmarks (submenu)	Alt+S+B
Next Error	Ctrl+K	Allows user to find the next error	Alt+S+N
Previous Error	Ctrl+J	Allows user to find the previous error	Alt+S+v

View Menu

The View menu appears in both the application and program windows.

View Menu (Application Window)

Menu Item	Description	Alt
Control Toolbar	When checked, displays the control toolbar	Alt+V+C
Search Toolbar	When checked, displays the search toolbar	Alt+V+S
Debug Toolbar	When checked, displays the debug toolbar	Alt+V+D
Status Bar	When checked, displays the status bar	Alt+V+t

View Menu (Program Window)

Menu Item	Description	Alt
Control Toolbar	When checked, displays the control toolbar	Alt+V+C
Search Toolbar	When checked, displays the search toolbar	Alt+V+S
Debug Toolbar	When checked, displays the debug toolbar	Alt+V+D
Status Bar	When checked, displays the status bar	Alt+V+t
Line Numbers	When checked, displays line numbers within the program	Alt+V+L
XREF...	Allows user to set cross-reference list	Alt+V+X
File Statistics...	Displays file statistics for open program including total line #s, memory (used & free), # of subs, and # functions	Alt+V+F
Debug Windows...	Displays the Debug Window Dialog Box so user may select which debug windows to display	Alt+V+W

Options Menu

The Options menu appears in both the application and program windows.

Options Menu (Application Window)

Menu Item	Description	Alt
Editor Environment	Displays Editor Environment Dialog Box	Alt+O+E
Color Mode	Allows user to set color mode from 256 colors or 16 colors	Alt+O+M
Startup Memory Size...	Allows user to set Startup memory size from 1 to 256 Mbytes	Alt+O+a
Reset to Default Settings	Allows user to reset all option menu items to default	Alt+O+D
HTB Editor	Selects Legacy or Windows Editor (default) on startup	Alt+O+T
Edit on Open	When checked, opens new file in Edit Mode	Alt+O+O

Options Menu (Program Window)

Menu Item	Description	Alt
Editor Environment	Displays the Edit Environment Dialog Box so user may set the edit environment	Alt+O+EAlt+O+RAlt+O+h
Run Environment	Displays the Configure Dialog Box so user may set the run environment	Alt+O+kAlt+O+TAlt+O+O
Change MSI...	Allows user to change Mass Storage Is location path + device (This is not set by default)	
Keyboard Mapping	Allows user to set key behavior to HTBasic for Windows (default) or HTBasic Legacy (has a submenu)	
HTB Editor	Allows user to select Legacy or Windows Editor (default) on startup (has a submenu)	
Edit on Open	When checked (by default), opens new program in Edit Mode	

Run Menu

The Run menu appears only in the program window.

Run Menu (Program Window)

Menu Item	Shortcut	Description	Alt
Step	Alt+F1	Allows user to run program in Step mode	Alt+R+t
Continue	Alt+F2	Continues a paused program (pause and stop mode only)	Alt+R+C
Run	Alt+F3	Runs an active program	Alt+R+R
Pause	Alt+F9	Pauses a running program (run mode only)	Alt+R+P
Stop	Alt+F10	Stops a running program (run mode only)	Alt+R+S
Clear I/O	Alt+F5	Aborts an I/O operation that is in progress	Alt+R+L
BASIC Reset	Ctrl+Break	Performs BASIC reset	Alt+R+B

Debug Menu

The Debug menu appears only in the program window.

Debug Menu (Program Window)

Menu Item	Shortcut	Description	Alt
Run Debugger	Ctrl+F3	Allows user to run program with debugger active	Alt+D+D
Continue Debugger	Ctrl+F2	Allows user user to continue a paused program with the debugger active	Alt+D+C
Add Watch Variable...	Ctrl + W	Allows user to add a variable to the Watch Window	Alt+D+W
Remove Watch Variable		Removes a selected variable from Watch Window	Alt+D+R
Remove All Watch variables		Removes all variables from Watch Window	Alt+D+h
Toggle Breakpoint	Ctrl +F11	Adds a breakpoint at the current line	Alt+D+g
Conditional Breakpoint...		Allows user to establish a conditional breakpoint	Alt+D+k
Global Breakpoint...		Allows user to establish a global breakpoint	Alt+D+o
Remove All Breakpoints		Removes all breakpoints	Alt+D+B
Step Into	Ctrl+F7	Steps into the next program line to be executed	Alt+D+I
Step Over Step Out	Ctrl+F8	Steps over the next context call or program line	Alt+D+S
Run to Cursor	Ctrl+Shft+F7	Steps out of a line in step mode execution	Alt+D+U
Continue From Cursor	Ctrl+F5	Runs program from beginning to cursor	Alt+D+t
	Ctrl+Shft+F5	Allows user to run program from cursor to end	Alt+D+e
Refresh Windows		Refreshes all displayed windows	Alt+D+f
Close All Debug Windows		Allows user to close all debug windows	Alt+D+A
Remove All Debug Info		Allows user to erase all debug info and remove .dbg file	Alt+D+L

Tools Menu

The Tools menu appears menu appears only in the program window.

Tools Menu (Program Window)

Menu Item	Shortcut	Description	Alt
Device Setup...	Ctrl+Alt+D	Allows user to add, remove, or change devices	Alt+T+D

Help Menu

The Help menu appears in both the application and program windows.

Help Menu (Both Application and Program Windows)

Menu Item	Description	Alt
Contents & Index	Displays HTBasic for Windows Help Contents & Index	Alt+H+C
Using Help	Displays Windows Help Contents	Alt+H+U
About HTBasic	Displays HTBasic for Windows Information	Alt+H+A

Control Characters

Control Characters permit character modification as they are displayed or printed by HTBasic. This section describe in detail these control characters for HTBasic for Windows, including:

- Output Area Characters
- Display Line Characters
- Display Enhancement Characters

Output Area Characters

Five character values perform special control actions in the output area.

CHR$(7)	Ring the bell
CHR$(8)	Move the print location back one space
CHR$(10)	Move the print location down one line
CHR$(12)	Print two line feeds, scroll the output area so that the next line is at the top of the output area
CHR$(13)	Move the print location to column one

Display Line Characters

The same control and enhancement characters are active for the Display Line as for the Output area with the following differences:

CHR$(12)	Clears the DISP line
CHR$(13)	Moves the DISP line cursor to column one and clears the DISP line when the next character is sent to the DISP line

Display Enhancement Characters

The characters that control display enhancements are:

CHR$(128)	All enhancements off
CHR$(129)	Inverse video on
CHR$(130)	Blinking video on
CHR$(131)	Inverse and blinking video on
CHR$(132)	Underline mode on
CHR$(133)	Underline and Inverse video on
CHR$(134)	Underline and Blinking video on
CHR$(135)	Underline, Inverse, and Blinking video on
CHR$(136)	White
CHR$(137)	Red
CHR$(138)	Yellow
CHR$(139)	Green
CHR$(140)	Cyan
CHR$(141)	Blue
CHR$(142)	Magenta
CHR$(143)	Black

Because some computers use character sets with character values that conflict with these characters, the CONTROL CRT,100 statement allows these display enhancement control characters to be moved to the range CHR$(16) through CHR$(31).

Setting the Environment

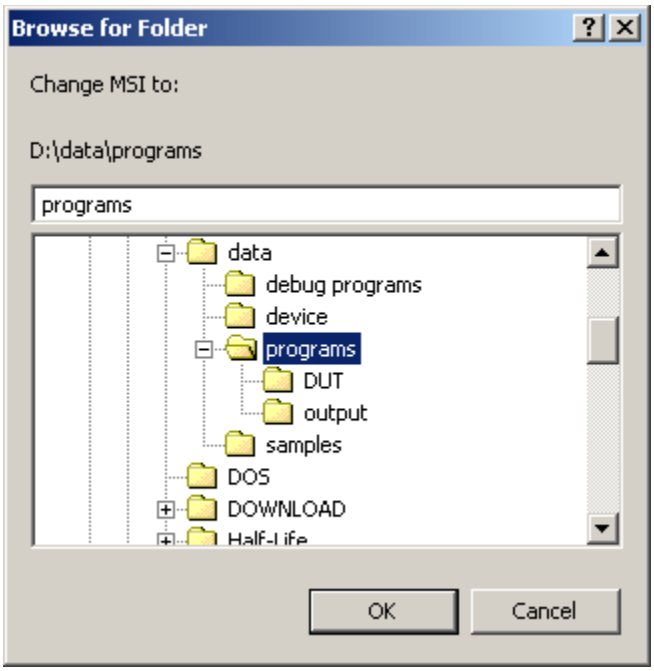
This section gives guidelines to set the environment for HTBasic for Windows, including the following topics:

- Changing MSI
- Keyboard Mapping
- Defining Cross-Reference lists (XREF)
- Aborting I/O Operation
- Basic Reset

Changing the MSI

You can use the Options | Change MSI... menu from the program window to change the MASS STORAGE IS (MSI) path and device specifier. The current MSI includes both the device and the current directory. This current directory is searched first to find any specified files.

To change the MSI, Select the Options | Change MSI... menu to display a Change MSI dialog box. Either browse to the desired folder and select OK or type or paste the desired MSI into the entry box.



To change the MSI when opening a file from the Recent File List select the menu item File | Change MSI On Open from the program window. A check will appear next to the menu selection.

Keyboard Mapping

You can use the Options | Keyboard Mapping... menu from the program window to assign the behavior of the keyboard to either HTBasic for Windows Editor or the HTBasic Legacy Editor.

To assign the keyboard behavior, use the Options | Keyboard Mapping menu. A check will appear by either the HTBasic for "*Windows*" settings menu option or by the HTBasic "Legacy" Editor menu designating which option has been assigned control.

The keyboard behavior of the HTBasic for Windows Editor include Alt key functions like accessing pull-down menus and windows-like keyboard editing (please see your Windows documentation for a complete description of Windows key assignments).

The keyboard behavior of HTBasic Legacy Editor include invoking System Softkeys with the Alt key, several other Alt key functions, and traditional RMB editing keyboard assignments. Please consult Chapter 4, Using the Keyboard, for a complete list of HTBasic Legacy functions assigned to the keyboard.

Defining Cross- Reference Lists

In addition to setting the program window environment, you can also use the View menu to define Cross-Reference lists. The XREF (Cross-Reference) statement generates a cross-reference list of line labels, I/O path names, numeric and string variables, subprograms, functions and COM block names. It also lists the number of unused symbol table entries.

You can select the cross-reference entries for your application. To define (X-REF) Cross-reference statement:

From the program window Click View | XREF... to open the XREF dialog box
Choose a subprogram to define XREF in
Choose an output device or check "Use Default Output" box
Select one of the options pictured in figure 3-10
Select "OK" to define the XREF list, "Cancel" to return to HTBasic, or choose "Default" to reselect XREF options

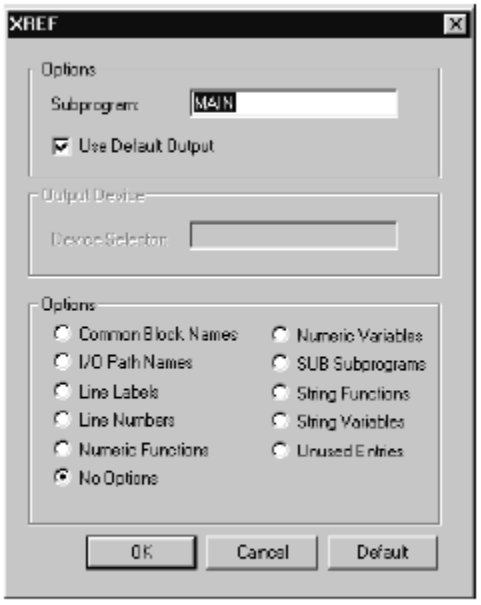


Figure 3-10. Defining Cross-Reference Lists

Aborting an I/O Operation

Run | Clear I/O aborts an I/O operation that is in progress. Unless timeouts have been enabled, the system will wait indefinitely for an I/O operation to complete. Executing Run | Clear I/O (or using Alt + F5) forces the program to return to a paused condition. Executing a CONTINUE function (use Alt + F2) causes the I/O statement to be re-executed.

Basic Reset

Run | BASIC Reset resets HTBasic. If a program is running, it is stopped. If a program is in memory, it is not discarded.

Using the Keyboard

This chapter explains how the different keyboard functions available in HTBasic are invoked. The last half of the chapter gives detailed descriptions of each function.

Keyboard Functions

Each keyboard function has a generic name by which it is referred to in all the manuals. To access the function, you must know which key to press on your keyboard. The generic names may or may not match the physical labels printed on the keyboard keys. Use this chapter, the system softkey menu or the "HELP Keyword" command, to look up key assignments.
Each of these methods allows you to quickly find which keys correspond to which functions. The assignments are logical and easy to remember.

Second Character

Internally, each keyboard function is represented by two characters. The first, CHR$(255), tells HTBasic that a function key is being pressed. The second character identifies the function. A program can execute any of the function keys by outputting the two characters to the keyboard (See the User's Guide, Chapter 6, "CRT, Keyboard and Printer."). The second character for each function is listed in the last column of the alphabetical keyboard functions table.

Ctrl Key

If the Ctrl key is pressed while a function key is pressed, the function key is not executed, but is entered into the keyboard buffer. This is useful when defining keyboard macros that expand into several key presses and when composing OUTPUT KBD statements to execute function keys from a program.

Softkeys

A softkey is a function key whose function can be changed under software control. Rocky Mountain BASIC defines 24 softkeys. Softkeys are programmed with the ON KEY statement to provide convenient user/program interaction. When not used by an ON KEY statement, a softkey can be assigned a keyboard macro (also known as a "Typing Aid"). A macro is a key that is assigned one or more keystrokes; thus by pressing one key you can mimic pressing several keys. Softkey Macros are explained in Chapter 8, "Customizing the Environment."

A softkey menu is displayed at the bottom of the screen. The labels in the menu are numbered to correspond to the numbers printed on the function keys. The label marked "1" corresponds to the F1 key. The number is not meant to be the softkey number, but is printed to help the user locate the correct key to press. For example, the first label is marked "1" and corresponds to the F1 key, though depending on KBD CMODE (explained below), it might be softkey K0 or softkey K1.

If your HP workstation does not display the softkey menu before a program defines any softkeys, then it does not have any keyboard macros defined or does not have the KBD binary loaded. To set HTBasic to this condition, execute a SCRATCH KEY statement in your AUTOST file.

KBD CMODE

Over the years, Rocky Mountain BASIC has supported two major softkey layouts: ITF and Nimitz. HTBasic can run programs written for either keyboard layout, regardless of the physical keyboard in use. The KBD CMODE statement selects the layout to use. When developing new programs, either style may be used at your preference.

Use KBD CMODE OFF for programs written for the ITF keyboard (46021A). With KBD CMODE OFF, eight function keys act as softkeys. The softkey labels are displayed at the bottom of the screen in two groups, four on the left and four on the right. Each label is eight characters wide and two lines high. With KBD CMODE OFF, the softkeys do quadruple duty.

Each softkey has four meanings, depending on which softkey menu is active when the key is pressed. The four menus are System, User 1, User 2 and User 3. An indicator is displayed immediately above the softkey menu to show which menu is active. Another function key is used to cycle through the menus. KBD CMODE OFF is the default softkey mode.
Use KBD CMODE ON for programs written for the Nimitz keyboard (98203). With KBD CMODE ON, ten function keys act as softkeys. The softkey labels are displayed at the bottom of the screen in two rows. Each row contains five labels and each label is 14 characters wide.

Alphabetical Keyboard Functions List

The following table lists alphabetically all the keyboard functions available in HTBasic, the generic names and the key pressed on a PC keyboard to invoke that function.

Keyboard Function	Generic Name	Windows Keyboard	Legacy Keyboard	2nd Char
Add Watch Variable	-	Ctrl-W	-	-
Alpha screen	ALPHA	Alt-F11	Alt-F11	M
Any character input	ANY CHAR	-	Alt= or Alt-K	$
Backspace	BACKSPACE	-	not assigned	B
Begin of line	BOL	Home	Shift-←	H
Begin of output area	BEGIN	Shift-↓	Shift-↓	T
Begin of program	-	Ctrl-Home	Shift-↓	—
Call Stack Window	-	Alt-9	-	-
CAPS state toggle	CAPS LOCK	Caps Lock	Caps Lock	U
Clear alpha screen	CLR SCR	Home→ outside editor	Home	K
Clear I/O	CLR I/O	Alt-F5	Alt-F5	I
Clear line	CLR LN	End→ outside editor	End	#
Clear tab under cursor	CLR TAB	-	Alt-C	[
Clear to end of line	CLR->END	-	Shift-End	%
Close child window	-	Ctrl-F4	-	-
Code window	-	Alt-0	-	-
Continue debugger	-	Ctrl-F2	-	-
Continue program	CONTINUE	Alt-F2	Alt-F2	C
Copy	-	Ctrl-C	-	-
Cut	-	Ctrl-X	-	-
Delete left of cursor	DEL LEFT	Backspace	Backspace	.
Delete program line	DEL LN	-	Shift-Del	/
Delete under cursor	DEL CHR	Del	Del	-
Display functions	DISPLAY FCTNS	-	Alt-F	F
Dump alpha screen	DUMP ALPHA	Alt-Shift-F11	Alt-A	O
Dump graphics screen	DUMP GRAPHICS	Alt-Shift-F12	Alt-G	N
"EDIT" key macro	EDIT	Ctrl-E	Alt-E	D
Enable/disable breakpoint	-	Ctrl-Shift-F11	-	-
End of line	EOL	End→if in editor	Shift-→	G
End of output area	s-HOME	Shift-Home	Shift-Home	_
End of output	END	Shift-↑	Shift-↑	W
End of Program	-	Ctrl-End	Shift-↓	-
Enter	ENTER	Enter	Enter	E
Execute	EXECUTE	-	not assigned	X
Execute selected	-	Ctrl-B	-	-
Find	-	Ctrl-F	-	-
Global breakpoint window	-	Alt-3	-	-
Goto	-	Ctrl-G	-	
Graph screen	GRAPHICS	Alt-F12	Alt-F12	L
Home position	HOME	-	not assigned	\
Increment menu labels	INCR LABELS	Shift-F11	Shift-F11	~
Indent all	-	Ctrl-I	-	-
Insert program line	INS LN	-	Shift-Ins	*
Insert/replace	INS CHR	Ins	Ins	+
Katakana mode	KATAKANA	-	not assigned	J
Left	LEFT	←	←	<

Note: The Alt-F1...F8 keys are present on the System Menu and can be pressed without the Alt key when the System Menu is visible.
The following paragraphs present the keyboard function assignments by physical grouping. Detailed descriptions of each function are given at the end of this chapter.

Softkeys

KBD CMODE OFF. With KBD CMODE OFF, Function keys F1 to F8 are used as softkeys. As stated before, each softkey has four meanings, depending on which softkey menu is active when the key is pressed. The key assignments for each menu are given in the following table.

Keyboard Function	Generic Name	Keyboard
Step program	STEP	System-F1
Continue program	CONTINUE	System-F2
Run program	RUN	System-F3
Pause program	PAUSE	System-F4
Clear I/O	CLR I/O	System-F5
Alpha screen	ALPHA	System-F6
Graph screen	GRAPHICS	System-F7
Recall older line	RECALL	System-F8

Keyboard Function	Generic Name	Keyboard
Stop program	STOP	Shift-System-F4
Dump alpha screen	DUMP ALPHA	Shift-System-F6
Dump graphics screen	DUMP GRAPHICS	Shift-System-F7
Recall more recent line	RECALL NEW	Shift-System-F8
Softkeys 1 to 8	K1 to K8	User 1- F1 to F8
Softkeys 9 to 16	K9 to K16	User 2- F1 to F8
Softkeys 17 to 23	K17 to K23	User 3- F1 to F7
Softkey 0	K0	User 3-F8

Pushing INCR LABELS (Shift-F11) will cycle through the four menus. Pushing SYSTEM (F12) will immediately display the System Menu and USER (Shift-F12) will immediately display the User 1 Menu. Pushing the MENU (F11) key will toggle the menu on and off. These same operations can be done using the BASIC statements SYSTEM KEYS, USER *n* KEYS and KEY LABELS ON/OFF. A short cut exists for the System Menu keys. They can be activated, even if a User Menu is displayed, by holding the Alt key down while the function key is pressed.

Keyboard Function	Generic Name	Windows Keyboard	Legacy Keyboard
Menu labels on/off	MENU	F11	F9
Increment menu labels	INCR LABELS	Shift-F11	Shift-F9
System softkeys	SYSTEM	F12	F10
User softkeys	USER	Shift-F12	Shift-F10

KBD CMODE ON. With KBD CMODE ON, ten function keys are used as softkeys. PC function keys F1 to F10 correspond to keys k0 to k9 of the Nimitz keyboard. The labels on the screen are numbered to correspond to the number printed on the function keys. The number is not meant to be the softkey number, but is printed to help the user locate the correct key to press. For example, the first label is marked "1" and corresponds to the F1 key, though it is softkey K0.

Pushing the Shift key with a function key activates K10 to K19, though no labels are displayed for these keys. Pushing the Alt key with a function key activates System functions listed in the following table.

Keyboard Function	Generic Name	Keyboard
Softkeys 0 to 9	K0 to K9	F1 to F10
Softkeys 10 to 19	K10 to K19	Shift- F1 to F10
Softkeys 20 to 23	K20 to K23	not assigned

Keyboard Function	Generic Name	Windows Keyboard	Legacy Keyboard
Step program	STEP	Alt-F1	Alt-F1
Continue program	CONTINUE	Alt-F2	Alt-F2
Run program	RUN	Alt-F3	Alt-F3
Pause program	PAUSE	Alt-F9	Alt-F4
Clear I/O	CLR I/O	Alt-F5	Alt-F5
Alpha screen	ALPHA	Alt-F11	Alt-F6
Graph screen	GRAPHICS	Alt-F12	Alt-F7
Recall older line	RECALL	Alt-F8	Alt-F8
Recall more recent line	RECALL NEW	Alt-Shift-F8	Alt-F9
Stop program	STOP	Alt-F10	Alt-F10

The Keypad

The PC keyboard has a keypad that has both numbers (for numeric keypad use) and edit functions: arrow keys, Home, PgUp, End, PgDn, Ins and Del. Some PC keyboards have separate numeric keypads and edit-function keys. The NumLock key is pressed to switch the keypad between numeric use and edit use. With NumLock off, the keypad is set for edit use. The keys produce a different function when used with the Shift key. The following table shows what keys are assigned to the keypad.

Keyboard Function	Generic Name	Windows Editor	Legacy Editor
Clear Alpha Screen	CLR SCR	Home, if in editor	Home
Clear line	CLR LN	End, if not in editor	End
Scroll output up	NEXT	PgUp	PgUp
Scroll output down	PREV	PgDn	PgDn
Insert/replace	INS CHR	Ins	Ins
Delete under cursor	DEL CHR	Del	Del
Pause program	PAUSE	Pause	Pause
Left	LEFT	←	←
Right	RIGHT	→	→
Next line	UP	↑	↑
Previous line	DOWN	↓	↓
End of output area	s-HOME	Shift-Home	Shift-Home
Clear to end of line	CLR->END	-	Shift-End
Recall older line	RECALL	Alt-F8	Shift-PgUp
Recall more recent line	RECALL NEW	Alt-Shift-F8	Shift-PgDn
Insert program line	INS LN	-	Shift-Ins
Delete program line	DEL LN	-	Shift-Del
Begin of line	BOL	Home, in editor	Shift-←
End of line	EOL	End, in editor	Shift-→
End of output	END	Shift-↑	Shift-↑
End of Program		Ctrl-End	Shift-↓
Begin of output	BEGIN	Shift-↓	Shift-↓
Begin of Program		Ctrl-Home	Shift-↑
Next word	NEXT WORD	-	Alt-→
Previous word	PREV WORD	-	Alt-←

Functions Invoked by the Alt Key

Besides the System Softkeys, which can be invoked with the Alt key, in the Legacy Editor or Ctrl in the Windows Editor, several other functions are invoked by holding down the Alt or Ctrl key while pressing one of the regular alphabetic keys on the keyboard. These are listed below. As you can see, they are mnemonic in nature:

Keyboard Function	Generic Name		Windows Keyboard	Legacy Keyboard
Any character input	ANY CHAR	-	Alt-= or Alt-K	
Dump alpha screen	DUMP ALPHA	-	Alt-A	
Clear tab under cursor	CLR TAB	-	Alt-C	
"EDIT" key macro	EDIT	Ctrl-E	Alt-E	
Display functions	DISPLAY FCTNS	-	Alt-F	
Dump graphics screen	DUMP GRAPHICS	-	Alt-G	
Print all output	PRT ALL	Ctrl-P	Alt-P	
Set tab under cursor	SET TAB	-	Alt-S	
Result of last calculation	RESULT	Ctrl-R	Alt-R	
Toggle SUB mode	SUB MODE	-	Alt-T	

Other Function Keys

Keyboard Function	Generic Name	Windows Keyboard	Legacy Keyboard
CAPS state toggle	CAPS LOCK	Caps Lock	Caps Lock
Delete left of cursor	DEL LEFT	Backspace	Backspace
Enter	ENTER	Enter	Enter
Reset BASIC	RESET	Ctrl-Break	Ctrl-Break
Tab forward	TAB	Tab	Tab
Tab backwards	TAB BACK	Shift-Tab	Shift-Tab

Keyboard Function	Generic Name	Keyboard
Backspace	BACKSPACE	not assigned
Execute	EXECUTE	not assigned
Home position	HOME	not assigned
Katakana mode	KATAKANA	not assigned
Roman mode	ROMAN	not assigned
Select(bell)	SELECT	not assigned

Additional Keyboard Features

Windows gives the Keyboard some built-in functionality that is available to almost all programs that run under Windows. The Windows documentation is the best source of information in this area. The following two features, however, are worth noting here.

The first feature is the Print Screen key. While HTBasic is the active window, pressing Alt-Print Screen places a "snapshot" of the HTBasic window onto the clipboard. The image can then be pasted into any application that accepts the bitmap format. Pressing Print Screen (without Alt) places a snapshot of the entire screen onto the clipboard.

The second feature is the generation of any ASCII value using the numeric keypad. If Num Lock is on and you hold the Alt key down while typing a number on the keypad, the keyboard will automatically generate one keystroke corresponding to that value. For example, typing 1 3 on the keypad while holding down the Alt key will enter a CHR$(13) character. This feature, when combined with the HTBasic ANY CHAR (Alt-= or Alt-K) function allows you to generate almost any character to be included in a string literal, even if the character is not available on the keyboard.

The ASCII value of a character can be entered in either the Windows (ISO 8859, Latin 1) character set or the OEM (code page 437) character set. To use the Windows character set, type a leading 0. For example, Alt-(0163) enters the character "£" while Alt-(163) enters "ú". Character set tables are found in the User's Guide.

Note: A null character cannot be entered from the keyboard into a string literal. Use CHR$(0) instead.

Detailed Descriptions

The previous sections described the key assignments for each editor function. As stated, each function has a generic name. The following sections describe, by generic name or Windows name, what each editor function does.

ADD WATCH VARIABLE

Allows the user to add a new variable to the debug watch window. The watch window will display the current value of the variable.

ALPHA

ALPHA makes the Alpha Screen visible. Pushing the key twice makes the Graphics Screen invisible. If ALPHA and GRAPHICS are merged, this key has no effect.

ANY CHAR

ANY CHAR allows any character to be entered. If the next character pressed is a Function key, then two characters are entered. The first character has the value CHR$(255). The second character is a code that identifies which editor function was pressed. You also can enter regular characters or control characters using this function.

BACKSPACE

BACKSPACE moves the cursor one place to the left. This is identical to the LEFT function. While this is the action of the Backspace key in HP BASIC, the Backspace key in HTBasic does a DEL LEFT.

BEGIN

BEGIN moves to the beginning of the program (if in edit mode) or the top most line in the Extended Output Area.

BOL

BOL moves the cursor to the beginning of the line.

CALL STACK WINDOW

Allows the user to enable or disable the debug call stack window. This window allows you to view the CALL stack.

CAPS LOCK

CAPS LOCK toggles the capital letter state of the keyboard. With CAPS LOCK on, uppercase letters will be produced when typing on the keyboard. When off, lowercase letters will be produced. On some computers the shift key temporarily inverts the state of the CAPS LOCK. The state of the CAPS LOCK is displayed on the status bar.

CLOSE CHILD WINDOW

Exits the child (program) window. This function is the same as performing QUIT. To start a new child window, the user needs to select NEW.

CLR->END

CLR->END clears to the end of the line.

CLR I/O

CLR I/O aborts an I/O operation that is in progress. Unless timeouts have been enabled, the system will wait forever for an I/O operation to complete. Pressing this key forces the program to return to a paused condition. Pressing CONTINUE will cause the I/O statement to be reexecuted.

CLR LN

CLR LN clears the input line.

CLR SCR

CLR SCR clears the Alpha Screen. On systems where Alpha and Graph have been merged, this also will clear any graphics that are present. This key also is used to leave EDIT mode.

CLR TAB

CLR TAB clears a tab stop if one exists at the present cursor location.

CODE WINDOW

Allows the user to enable or disable the debug code window. This window allows you to view each line of code as it is executing in the debugger.

CONTINUE

CONTINUE resumes program execution if the program is in a paused state.

CONTINUE DEBUGGER

Enables the debugger after it has stopped due to a breakpoint. The debugger will continue until it hits either the end of the program or another breakpoint.

COPY

Will copy the selected text within the HTBasic Windows editor to the clipboard. The selection can then be pasted to another section of the program.

CUT

Will cut the selected text within the HTBasic Windows editor. The cut text can then be pasted to another section of code.

DEL CHR

DEL CHR deletes the character where the cursor is.

DEL LEFT

DEL LEFT deletes the character to the left of the cursor. This is the default action of the Backspace key in HTBasic. This function is not available in HP BASIC.

DEL LN

DEL LN deletes the program line that the cursor is on, if in Edit Mode.

DISPLAY FCTNS

DISPLAY FCTNS displays characters that would normally be interpreted as control codes, including control characters that might normally be thrown away. This mode is useful for debugging I/O operations.

DOWN

DOWN scrolls the output area down one line. In edit mode, scrolling the program down one line leaves the cursor on the previous line.

DUMP ALPHA

DUMP ALPHA sends the contents of the Alpha Screen to the printer or file specified by the DUMP DEVICE IS statement.

DUMP GRAPHICS

DUMP GRAPHICS sends the contents of the Graphics Screen to the printer or file specified by the DUMP DEVICE IS statement. When Alpha and Graphics have been merged, this will also DUMP the contents of the ALPHA screen.

EDIT

EDIT clears the input line and enters "EDIT" for you. You may then hit ENTER to start EDIT mode or you may enter a program line and hit ENTER to start EDIT mode at a particular line number. To exit EDIT mode, press the CLR SCR key. Likewise, you may press EDIT and then press a softkey to begin editing that softkey. To finish editing the key, press the ENTER key.

ENABLE/DISABLE BREAKPOINT

Allows the user to enable or disable a breakpoint. If a breakpoint is enabled, the debugger will stop once it hits the breakpoint. If it is disabled, the debugger will run through the line as if the breakpoint weren't there.

END

END moves to the end of the program (if in Edit mode) or the last line in the Extended Output Area.

ENTER

ENTER executes a command or enters a line into a program.

EOL

EOL moves the cursor to the end of the line.

EXECUTE

EXECUTE is the same as ENTER. This function exists for compatibility with old HP keyboards that had separate keys for executing lines and entering information.

EXECUTE SELECTED

Runs the piece of code selected by the user. The selected piece of code must be commands that can be executed from the command line. If a programmable only command is selected, an error will occur.

FIND

Will search through the code and find a specified string. You can either search up or down through the document.

GLOBAL BREAKPOINT WINDOW

Allows the user to enable or disable the debug global breakpoint window. This window lets you view any global breakpoints you have set. It will indicate whether they are enabled or disabled and in which line number they occur.

GOTO

Allows you to go directly to a specific line, label, sub or function.

GRAPHICS

GRAPHICS makes the Graphics Screen visible. Pressing the key twice makes the Alpha Screen invisible. If ALPHA and GRAPHICS are merged, this key has no effect.

HOME

HOME moves the print position to the home position on the screen. This is the upper-left corner of the screen.

s-HOME

Shift+ HOME sets the print position to the first blank line at the end of the Extended Output Area. If this position is not visible, the screen is scrolled to move it onto the screen.

INCR LABELS

INCR LABELS cycles through the softkey menu labels at the bottom of the screen.

INDENT ALL

Indents your program either to the default settings or those specified directly by the user.

INS CHR

INS CHR toggles Insert/Replace mode. The default is replace mode, where new characters overwrite any existing characters at the position of the cursor. In insert mode, old characters are shifted to the right to open up a place for each new character typed.

INS LN

INS LN inserts a program line before the current line while in Edit mode. A new line number is generated automatically.

K0 to K23

K0 to K23 are user Softkeys 0 to 23. User softkeys can be used as either keyboard macros (sometimes called "typing aids") or for generating events in a running program. By default, a user softkey acts as a keyboard macro. When you push a user softkey, one or more keystrokes that have been assigned to that key will be entered just as if you had typed them.

See "Softkey Macros" in Chapter 8, "Customizing the Environment."

If the key is used in an ON KEY statement, then it is no longer active for softkey macros, but instead generates an event that causes a GOTO, GOSUB, CALL or RECOVER.

LEFT

LEFT moves the cursor one position to the left.

LEGACY KEY MAPPING

Maps your keyboard to the HTBasic Legacy keys. Windows hot keys will not work under this mode.

LINE BREAKPOINT WINDOW

Allows the user to enable or disable the debug line breakpoint window. This window lets you view any line breakpoints you have set. It also indicates whether they are enabled or disabled as well as their line number.

MENU

MENU alternately turns the softkey menu on or off.

NEW

Creates a new instance of HTBasic if the child window is closed. If the child window is active, it will act as a SCRATCH.

NEXT

NEXT allows viewing of the next part of the Extended Output Area by scrolling the output area up.

NEXT BOOKMARK

Will take you to the next bookmark set within the code.

NEXT ERROR

Will take you to the next error within the code.

NEXT WORD

NEXT WORD moves the cursor right to the first letter of the next word.

OPEN

Will bring up the open dialog box. This enables you to open a new program.

PASTE

Will paste at the current cursor location whatever has been cut or is currently on the clipboard.

PAUSE

PAUSE pauses a running program. The program can be continued by pressing CONTINUE.

PREV

PREV allows viewing of the previous part of the Extended Output Area by scrolling the output area down.

PREVIOUS BOOKMARK

Will take you to a previous bookmark within the code.

PREVIOUS ERROR

Will take you to a previous error within the code.

PREV WORD

PREV WORD moves the cursor left to the first letter of the previous word.

PRT ALL

PRT ALL causes subsequent screen output to the message line, display line, input line and output area also to be sent to the PRINTALL device.

QUIT ALL

Will completely close the HTBasic window. This includes the parent and child window.

RECALL

RECALL recalls the last line entered. Several lines are saved and repeated pressing of this key recalls progressively older lines.

RECALL NEW

RECALL NEW recalls a more recent line. This is the opposite of the RECALL key. You can use RECALL and RECALL NEW to search back and forth through the saved lines.

REDO

Will bring back anything that has been removed with an UNDO command.

REPLACE

Will replace a specified word or string with a new word or string.

RESET

RESET resets HTBasic. If a program is running, it is stopped. If a program is in memory, it is not deleted.

RESULT

RESULT recalls the result of the last numeric calculation into the input line.

RIGHT

RIGHT moves the cursor one position to the right.

ROMAN

ROMAN turns on Roman Mode (as opposed to Katakana Mode).

RUN

RUN starts a program running at the first line in the program.

RUN FROM CURSOR

Is a debugger tool. With the debugger running, you can set your cursor at a specific point within the code window and start execution from the cursor. With this feature, you can avoid running through parts of the code that are unnecessary.

RUN THE DEBUGGER

Starts execution of the debugger. The debugger gives you a variety of tools to help debug code.

RUN TO CURSOR

Is a debugger tool. With the debugger running, you can set your cursor at a specific point within the code window and execution of the program will run until it encounters the cursor.

SAVE/STORE

Allows you to either save or store your program.

SELECT

SELECT rings the bell.

SELECT ALL

Selects all of the code within the new Windows editor.

SET TAB

SET TAB sets a tab stop at the present cursor position.

STEP

STEP executes one line of the program and then pauses. If a program is not currently running when you press this key, the first press of the key causes the program to be prerun in preparation for the next press of the STEP key. With each press of the STEP key, the statement that will execute next is displayed.

STEP INTO

Is a debugger tool, which executes each line of code one at a time.

STEP OUT

Is a debugger tool that continues to the end of context, stopping when entering the calling context.

STEP OVER

Is a debugger tool that runs the entire sub context and stops at the next executable line of the current context.

STOP

STOP stops a running program. If the program is executing in an I/O operation, you may have to press CLR I/O first.

SUB MODE

SUB MODE toggles between regular edit mode and SUB mode. In SUB mode, only the first line of each context is shown. This allows rapid movement to any of the contexts in the program. Pressing the key again returns to the regular edit mode where all lines of the program are shown. SUB MODE is only allowed in the Legacy Editor.

SYSTEM

SYSTEM displays the System Softkey Menu.

SYSTEM MENU

Pulls up the system menu at the top of the window. This menu allows you to minimize, maximize or close your application.

TAB

TAB moves the cursor forward to the next tab stop.

TAB BACK

TAB BACK moves the cursor backward to the previous tab stop.

TOGGLE BOOKMARK

Either sets or removes a bookmark on the current cursor line.

TOGGLE BREAKPOINT

Either sets or removes a breakpoint on the current cursor line.

TRACE WINDOW

Enables or disables the debug trace window. This window permits the user to observe which commands are being executed in the running program. There is nothing to "set" in this window.

UNDO

Will undo the previous edit action.

UP

UP scrolls the output area up one line. In Edit mode, scrolling the program up one line leaves the cursor on the previous line.

USER

USER displays the User 1 Softkey menu.

WATCH WINDOW

Enables or disables the debug watch window. This window permits the programmer to watch the values change during the program run for the list of user-defined variables at each step of the program.

WINDOWS KEY MAPPING

Changes the keyboard hot keys to those used by a standard windows system. It is best used along with the HTBasic Windows editor.

CRT and Graphic Drivers

HTBasic supports several graphic drivers. Graphic drivers are classified as CRT or Graphic Output drivers.
The CRT driver provides all the routines necessary for controlling output to the screen. This includes displaying text, writing screen labels, controlling the cursor, scrolling lines and clearing the screen. All CRT drivers also include a graphic output driver. The graphic output driver includes routines to move and draw, load and store portions of the screen, fill and edge an area, dither an area and control the color palette.

The following table lists the drivers available at the time of this manual printing.

Name	Type	Graphic Driver
INTERNAL	CRT	Reuse last display driver specified
WIN	CRT	Microsoft Windows Display Driver
HPGL	Graphic	Hewlett-Packard Graphic Language
PS	Graphic	PostScript printers, plotters and files

The PLOTTER IS Statement

The PLOTTER IS statement is used both to load drivers and switch among them. The PLOTTER IS statement directs vector graphics to a device or file. (Use the DUMP DEVICE IS statement to print bit-mapped graphics from the screen to a device or file.) The default PLOTTER IS device is the CRT. Executing a PLOTTER IS statement directs all subsequent graphics output to the specified target.

Loading Drivers

Driver files can be loaded at any point. It is recommended that PLOTTER IS statements be included in your AUTOST file to load any necessary drivers. Up to ten graphic and dump drivers can be loaded at a time.

To find the driver file, HTBasic takes the language specified in the PLOTTER IS statement and performs several operations upon it to find the correct driver file. ".DW6" is appended to the name. Then the following locations are searched, in the specified order:

1. The directory containing the HTBasic executable.
2. The current directory.
3. The Windows system directory (such as \WINDOWS\SYSTEM).
4. The Windows directory.
5. The directories listed in the PATH environment variable.

CRT Drivers

The syntax of the PLOTTER IS statement used to load CRT drivers is:

```
PLOTTER IS destination, language [; COLOR MAP ]
```

where *destination* specifies the special interface select codes 1, 3 or 6. Also, the predefined constant CRT can be used for the value 1. *Language* is the driver name or the constant "INTERNAL". "INTERNAL" is a special language string synonymous with the last CRT driver specified. Optionally, the *language* string can include driver options as explained later. For example:

```
PLOTTER IS CRT,"INTERNAL";COLOR MAP
```

Default CRT Driver

HTBasic automatically loads the WIN driver when it starts. It is not necessary to use a PLOTTER IS "WIN" statement, nor a -CRT WIN command line switch.

CRTA and CRTB Modes

The WIN driver only supports CRTB mode. CRTA mode may have been used on the DOS version or a series 200 BASIC workstation. Briefly, CRTA mode uses a true text mode to display the ALPHA screen. The CRTB mode uses bits written into a graphics screen to display the ALPHA screen.

The mode is specified using the interface select code in the PLOTTER IS statement. Specifying interface select code 3 selects CRTA mode and 6 selects CRTB mode. If the specified mode is not supported, the value is ignored. Specifying CRT or 1 in the PLOTTER IS statement reselects the last mode used.

Graphic Drivers

The syntax of the PLOTTER IS statement for graphic drivers is:

```
PLOTTER IS destination, language [,hard-clip] [; APPEND ]
```

Language is the driver name, which optionally can be followed by driver options. Options are included by appending a semicolon to the driver name, followed by the options. Each driver has its own options. Consult the driver documentation, later in this chapter, for the legal options of each driver.
Hard-clip is composed of four values separated by commas that specify the size of the drawing surface. The four values are *xmin*, *xmax*, *ymin* and *ymax*, respectively. For example:

```
PLOTTER IS 10,"HPGL",2,268,0,190
PLOTTER IS "Pictfile","HPGL",5.75,250.50,7.25,136.875
```

The *destination* of a graphic driver can be a device or file, although not every driver can send output to all targets. For example, it doesn't make sense to send GIF output to anything but a file.

Devices

To send graphic output to a device such as a plotter or a printer capable of vector graphics, use the interface select code of the interface connecting the device. Use the device-selector if the device is on the IEEE-488 bus. If hard-clip limits are specified, they are given in the order *xmin*, *xmax*, *ymin*, *ymax* and are specified in millimeters. If the hard-clip limits are not specified, they are read from the device when this statement is executed. The specified device must respond to this query or the computer will wait indefinitely for the response. Use the CLR-I/O key to clear the computer if it gets stuck in this state.

The following example sends HPGL commands to a LaserJet III printer. The first line resets the printer, starts landscape printing and switches into HPGL mode. The second line directs plotter output to the LaserJet III and sets the hard-clip units for an 8-1/2 x 11 sheet of paper. Both lines assume that the LaserJet III is connected to interface select code 26 at Lpt1.

```
OUTPUT 26;CHR$(27)&"E"&CHR$(27)&"&l1O"&CHR$(27)&"%1B";
PLOTTER IS 26,"HPGL",2,268,0,190
```

Files

To send graphics output to a file, *destination* should be replaced with the file name. The file must be an existing, ordinary or BDAT file. The hard-clip limits may be specified or defaulted to? ±392.75 mm on the *x* axis and ±251.5 mm on the *y* axis. If APPEND is not specified, the file is positioned to the beginning and truncated. The file is closed when another PLOTTER IS, GINIT or SCRATCH A statement is executed. For example:

```
CREATE "DRAW.PLT",0
PLOTTER IS "DRAW.PLT","HPGL"
```

WIN Driver

The WIN driver is a CRT driver that uses the Microsoft Windows display drivers.

For compatibility with HP BASIC/UX, options for the WIN driver are specified on the command line. Command line switches were explained in Chapter 1. These command line switches are passed to the WIN driver:

Switch	Effect
-fn	Use named font
-geometry	Specify initial size of HTBasic window
-title	Specify the window title

Window Resize

Resizing the HTBasic window using the mouse is supported, but has the following effects. If the number of text columns changes, any text present is discarded. If in edit mode, the screen is redrawn using the new size.

Any graphics present in the window are discarded. The current pen position is left undefined. The VIEWPORT, WINDOW and hard clip limits are unchanged, although GESCAPE CRT,3 returns the new window size. Use the GINIT statement to set the VIEWPORT, WINDOW and hard clip limits to the new window size, or use the

```
PLOTTER IS CRT,"INTERNAL"
```

statement to activate use of the new hard clip limits without the side effects of GINIT.

HPGL Driver

The HPGL graphic output driver provides support for any output device that accepts Hewlett Packard's HPGL language. The driver also can store the HPGL information into a file that can be imported into a number of graphics packages and word processors.

The minimum and maximum hard clip limits can be specified for either a device or file. This allows you to output HPGL information to a printer that can't return P points. If no hard clip units are specified for a device, P points are requested from the device. If no hard clip units are specified for a file, the default hard clip limits are -392.75, 392.75, -251.5, 251.5 (millimeters).

The HPGL plotter driver is loaded with a line like

```
PLOTTER IS device,"HPGL[;options]",[p1x,p2x,p1y,p2y]
```

or

```
PLOTTER IS "file","HPGL[;options]",[p1x,p2x,p1y,p2y]
```

In the above, *device* refers to an HTBasic device number. *File* refers to a file in the computer's file system. The file must already exist when the PLOTTER IS statement is executed.

Plotting Area

The points *(p1x,p1y)* and *(p2x,p2y)* determine the lower left and the upper right corners of a rectangular area the driver will plot to. These points are specified in mm from the lower left corner of the paper. *P2x* and *p2y* must be larger than *p1x* and *p1y*, respectively. All of these coordinates must be positive or zero if the PCL5 option is used (see Options, below). If the plotting area is omitted, the driver reads the plot area from the plotter, if it is connected to a serial or IEEE-488 port. If output is directed to a file, the driver uses the default values from the table below.

PCL5 Option	Orientation	(P1x,P1y)	(P2x,P2y)
No	Landscape	(-393, -252)	(393, 252)
No	Portrait	(-252, -393)	(252, 393)
Yes	Landscape	(0,0)	(254, 184)
Yes	Portrait	(0,0)	(184, 254)

Polygons

The HPGL driver, for compatibility with HP BASIC, outputs polygon fills as separate lines. However, the driver can be instructed to output HPGL/2 polygon fill commands. This is useful if the plotter supports the polygon fill command or if an HPGL file is produced for import into another program that supports polygons. To enable polygon mode, use GESCAPE code 104, operation number 1:

```
10  INTEGER Param(1)
20  Param(0)=1   ! HPGL Operation Number 1 is HPGL/2 Flag
30  Param(1)=1   ! Set HPGL/2 Flag to 1=enable, 0=disable
40  GESCAPE Isc,104,Param(*)
```

If output is to a device, substitute the device ISC for Isc in line 40. If output is to a file, substitute 1 for Isc.

Options

The options are listed after the semicolon in the driver name, within the quotes. If more than one option is specified, the option names are separated by commas. The options are as follows:

COLOR. This option tells the driver that the device used for plotting is a color printer with plotter functions, such as the Hewlett-Packard PaintJet XL-300. This option is ignored unless the PCL5 option is also specified.

FILL. This option tells the driver that the plotter being used can do area filling. Area filling produced by the plotter is generally much faster than that produced by the driver.

GRAY. This option causes the driver to produce grayscale plots when used with a printer. Each color that normally would be plotted is changed to a brightness, using the method explained in the Pen Colors section, before plotting. Note that the brightness level is inverted unless the INVERT option is also used. The GRAY option need not be specified; it is the default. This option is ignored unless the PCL5 option is also used.

HPGL2. By default, the driver produces plots for an HP-GL plotter. This option allows the driver to produce plots for an HP-GL/2 plotter, such as the Hewlett-Packard DraftMaster. Since HP-GL/2 plotters can all do area filling, the HPGL2 option turns on the FILL option.

INVERT. By default, the driver reverses black and white on color plots and reverses all gray levels on grayscale plots when the plots are made on a printer. This is suitable for printers that use dark inks on white paper, but is the opposite of the colors normally shown on the computer screen. The INVERT option causes colors or gray levels to be represented as they are on the computer screen. This option is ignored unless the PCL5 option is also used.

PCL5. This option tells the driver that the plotter is a laser or electrostatic printer with built-in plotter emulation using the PCL-5 language. This causes the driver to send escape sequences at the beginning and end of plots to enable and disable the plotter emulation. When this option is used, a PLOTTER IS CRT,"INTERNAL" statement should be executed at the end of plotting to make the printer eject the page containing the plot. Since all PCL-5 devices use the HP-GL/2 plotter language, this option turns on the HPGL2 and FILL options.

PORTRAIT. The PORTRAIT option causes the driver to produce plots in portrait orientation, that is, with the long edge of the paper vertical. Without this option, the driver produces plots in landscape orientation, with the long edge of the paper horizontal.

1600. The 1600 option provides compatibility for most newer HP Deskjet printers. This option is ignored unless the PCL5 and COLOR options are included.

Choosing Options. The following table may help in choosing from the FILL, HPGL2, PCL5 and COLOR options for many plotter models. Model numbers refer to Hewlett-Packard products except where noted.

Plotter Model	Options
7470A, 7475A, 7580A, 7585A, older 7580B and 7585B, 7090A, LaserJet II printer with Pacific Data Plotter-in-a-Cartridge, Sweet-Pea, Houston Instruments ImageMaker, other plotters that emulate the 7470A or 7475A, plot files for use in WordPerfect, Ventura Publisher, Microsoft Word, Pagemaker, Ami Pro, etc.	none
17440A, ColorPro, 7510A, 7550A, 7570A, newer 7580B, 7585B, 7586B, 7595A, 7596B, Houston Instruments DMP-60, JetPro	FILL
7550 Plus, DraftMaster series, CalComp Classic, JDL Model 4000E, Houston Instruments DMP-160, DMP-162R	HPGL2
LaserJet III printers, LaserJet IV printers, LaserJet IIP printer with LaserJet III emulation cartridge 7600 series electrostatic plotter, DesignJet and DesignJet 600 plotters	PCL5
PaintJet XL-300 printer, PaintJet XL printer with HP-GL/2 cartridge	PCL5,COLOR

Pen Colors

When the HPGL driver is used with a pen plotter, the HTBasic PEN command selects the indicated pen on the plotter. However, when the driver is used with a printer (as indicated by the PCL5 option), the effect of the PEN command is that described in the following text.

The colors or grayscales produced by each pen depend on the states of the COLOR and INVERT options used in loading the driver, as well as the state of the COLOR MAP option of the HTBasic CRT driver. If the COLOR MAP option is off, the following gray levels or colors are used:

					GRAY	COLOR
PEN	**GRAY**		**COLOR**		**INVERT**	**INVERT**
0	white	white	black	black		
1	black	black	white	white		
2	30% black	red	70% black	red		
3	89% black	yellow	21% black	yellow		
4	59% black	green	41% black	green		
5	70% black	cyan	30% black	cyan		
6	11% black	blue	89% black	blue		
7	40% black	violet	60% black	violet		
8	black	black	white	white		
9	30% black	red	70% black	red		
10	89% black	yellow	21% black	yellow		
11	59% black	green	41% black	green		
12	70% black	cyan	30% black	cyan		
13	11% black	blue	89% black	blue		
14	40% black	violet	60% black	violet		
15	black	black	white	white		

If the COLOR MAP option of the CRT driver is on, the plot is made using the colors in the HTBasic color map if the COLOR option is used. If the INVERT option is not used, black and white are reversed. If the COLOR option is not used, the colors in the HTBasic color map are converted to shades of gray using the NTSC equation:
brightness = 11% blue + 59% green + 30% red

If the INVERT option is not used, the brightness is inverted before plotting is done. With both pen plotters and printers, the sign of the pen is ignored; the absolute value determines the pen used.

Drawing Mode

When the PCL5 option is specified, the HTBasic statement GESCAPE CRT,5 sets alternate drawing mode for the driver. Normally, the driver replaces anything previously at a location with what is currently drawn. In the alternate drawing mode, the previous black or colored areas show through the white areas of the new plot. The HTBasic statement GESCAPE CRT,4 returns the driver to normal drawing mode.

Line Thickness

If the PCL5 option is specified, line thicknesses can be set in the driver. Lines default to 0.35 mm thick. The line thickness for all pens can be changed by the GESCAPE CRT,104 statement as in either of the examples below:

```
INTEGER Param(1:2)              ! an array for the command
Param(1) = 10                   ! line thickness code
Param(2) = thickness            ! desired thickness (in 1/100 GDU's)
GESCAPE CRT,104,Param(*)        ! send thickness
INTEGER Param(1:2)              ! an array for the command
Param(1) = 11                   ! line thickness code
Param(2) = thickness            ! desired thickness (in 1/100 mm)
GESCAPE CRT,104,Param(*)        ! send thickness
```

Line Caps and Joins

When the PCL5 option is specified, line cap and join styles can be specified. By default, the device driver uses round caps to end lines and round joins to connect lines, which simulates the round pens used on pen plotters. This can be changed with the following statements.

```
INTEGER Param(1:3) ! an array for the command
Param(1) = 12                    ! line thickness code
Param(2) = cap                   ! desired line cap
Param(3) = join                  ! desired line join
GESCAPE CRT,104,Param(*)         ! set cap and join
```

The values for *cap* and *join* can be selected from the following tables.

cap	meaning	join	meaning
1	butt cap	1	mitered join
2	square cap	2	mitered, beveled if too long
3	triangular cap	3	triangular join
4	round cap	4	round join
		5	beveled join
		6	no join

Note that many low-resolution PCL-5 devices use a butt cap and no join with lines less than 0.35 mm thick, regardless of the cap and join settings.

Crosshatching

The HPGL driver can crosshatch areas meant to be filled. This is its default behavior unless the FILL or PCL5 option is specified, in which case the default is to use solid fills.

If the FILL or PCL5 options are specified, the driver can be made to crosshatch filled areas with the following statements:

```
INTEGER Param(1:2)              ! an array for the command
Param(1) = 1                    ! set fill type
Param(2) = state                ! turn solid filling on or off
GESCAPE CRT,104,Param(*)        ! send command
```

State is 0 to use crosshatching and any other value to use solid filling. For compatibility with older drivers, if *state* is nonzero, this command turns on the FILL option if neither the FILL nor the PCL5 option was specified when the driver was loaded.

When crosshatching is turned on, the following sets of statements can be used to control the crosshatch parameters. If these statements are not executed, crosshatching is done with solid horizontal lines spaced 0.01 in. (0.25 mm) apart, which is useful on most devices for producing a solid fill.

```
INTEGER Param(1:2)              ! an array for the command
Param(1) = 2                    ! set crosshatch type
Param(2) = type
GESCAPE CRT,104,Param(*)        ! send command
```

Type is 1 for single hatching, 2 for crosshatching.

```
INTEGER Param(1:2)              ! an array for the command
Param(1) = 3                    ! set hatch angle
Param(2) = angle                ! desired angle, degrees
GESCAPE CRT,104,Param(*)        ! send command
```

Angle is the angle in degrees (regardless of the HTBasic RAD or DEG setting) for hatching. *Angle* is rounded to the nearest multiple of 45 degrees.

```
INTEGER Param(1:2)              ! an array for the command
Param(1) = 4                    ! set line spacing
Param(2) = spacing              ! desired spacing (in 1/100 GDU's)
GESCAPE CRT,104,Param(*)        ! send command
INTEGER Param(1:2)              ! an array for the command
Param(1) = 5                    ! set line spacing
Param(2) = spacing              ! desired spacing (in 1/100 mm)
GESCAPE CRT,104,Param(*)        ! send command
```

The above commands are equivalent except that in the first command, *spacing* is expressed in 1/100 GDU and in the second in 1/100 mm.

```
INTEGER Param(1:3)              ! an array for the command
Param(1) = 6                    ! set line type for hatching
Param(2) = type                 ! desired line type for crosshatching
Param(3) = size                 ! desired pattern repetition size
GESCAPE CRT,104,Param(*)        ! send command
```

Type is the type of line, as listed in the LINE TYPE section of the on-line *HTBasic Reference Manual*. *Size* is the pattern repetition length in 1/100 GDU's. This would be 100 times the pattern repetition length specified in a LINE TYPE statement.

Pages

The GCLEAR statement causes subsequent plotting to be done on a new page. If the PCL5 option is specified, the GCLEAR statement causes the printer to eject the old plot. Also, opening a file with

```
PLOTTER IS "file","HPGL";APPEND
```

causes the driver to append new pages of plot information to the current file if it exists already. Note that most word processor programs and other programs that can import files will probably superimpose the plots imported from a file containing more than one plot.

Ending Plots

If the PCL5 option is used, the HPGL driver will not eject a plot until a GCLEAR statement is executed, HTBasic is ended or when the PLOTTER IS device is set to a different device. It is recommended that a statement like

```
PLOTTER IS CRT,"INTERNAL"
```

be placed at the end of each program section that produces a plot using the PCL5 option driver.

PostScript Driver

The PostScript graphics output driver generates PostScript-language files from HTBasic plotting commands. These files are suitable for printing on PostScript-language printers and photographic equipment and for importing into documents using the PostScript file format. The PostScript graphics output driver is loaded with the following statement:

```
PLOTTER IS destination,"PS[;options]",[p1x,p2x,p1y,p2y]
```

Destination refers to a device or file. If it is a file, the file must already exist when the PLOTTER IS statement is executed and it should be an *ordinary file*. Otherwise the HTBasic file header will appear as bad data at the start of the file.

The points *(p1x,p1y)* and *(p2x,p2y)* determine the lower left and the upper right corners of a rectangular area the driver will plot to. These points are specified in mm from the lower left corner of the paper. All of these coordinates must be positive or zero and *p2x* and *p2y* must be larger than *p1x* and *p1y*, respectively. If omitted, the driver uses *(p1x,p1y)* = (25.4 mm, 25.4 mm) and *(p2x,p2y)* = (262.7 mm, 190.5 mm) in landscape mode and *(p2x,p2y)* = (190.5 mm, 262.7 mm) in portrait mode, which produces a plot with adequate margins on US "A" or European A4 size paper.

Note that most PostScript printers cannot print to the edges of the paper. Because of this, the points specified should include a small (about 1 cm) margin on each side when the driver is used with a printer.

Options

The options are listed after the semicolon in the driver name, within the quotes. If more than one option is specified, the option names are separated by commas. The options are as follows:

COLOR. This option causes the driver to produce color plots. Note that black and white are inverted from their values on the screen unless the INVERT option is also used. Color plots require a PostScript level 2 output device or a PostScript level 1 device with color language extensions.

GRAY. This option causes the driver to produce grayscale plots. Each color that normally would be plotted is changed to a brightness using the method explained in the Pen Colors section, below, before plotting. Note that the brightness level is inverted unless the INVERT option is also used. The GRAY option need not be specified; it is the default.

INVERT. By default, the driver reverses black and white on color plots and reverses all gray levels on grayscale plots. This is suitable for printers that use dark inks on white paper, but is the opposite of the colors normally shown on the computer screen. The INVERT option causes colors or gray levels to be represented as they are on the computer screen.

PORTRAIT. The PORTRAIT option causes the driver to produce plots in portrait orientation, that is, with the long edge of the paper vertical. Without this option, the driver produces plots in landscape orientation, with the long edge of the paper horizontal.

Pen Colors

The colors or grayscales produced by each pen depend on the states of the COLOR and INVERT options used in loading the driver, as well as the state of the COLOR MAP option of the HTBasic CRT driver. If the COLOR MAP option is off, the following gray levels or colors are used:

					GRAY	COLOR
PEN	**GRAY**		**COLOR**		**INVERT**	**INVERT**
0	white	white	black	black		
1	black	black	white	white		
2	30% black	red	70% black	red		
3	89% black	yellow	21% black	yellow		
4	59% black	green	41% black	green		
5	70% black	cyan	30% black	cyan		
6	11% black	blue	89% black	blue		
7	40% black	violet	60% black	violet		
8	black	black	white	white		
9	30% black	red	70% black	red		
10	89% black	yellow	21% black	yellow		
11	59% black	green	41% black	green		
12	70% black	cyan	30% black	cyan		
13	11% black	blue	89% black	blue		
14	40% black	violet	60% black	violet		
15	black	black	white	white		

If the COLOR MAP option of the CRT driver is on, the plot is made using the colors in the HTBasic color map if the COLOR option is used. If the INVERT option is not used, black and white are reversed. If the COLOR option is not used, the colors in the HTBasic color map are converted to shades of gray using the HTSC equation:
brightness = 11% blue + 59% green + 30% red

If the INVERT option is not used, the brightness is inverted before plotting is done. GESCAPE codes 4 and 5 are ignored as is the sign of the PEN. Graphics always overwrite existing graphics.

Line Thickness

Lines default to 0.35 mm thick. The line thickness can be changed by the GESCAPE CRT,104 statement as in either of the examples below:

```
INTEGER Param(1:2)              ! an array for the command
Param(1) = 10                   ! line thickness code
Param(2) = thickness            ! desired thickness (in 1/100 GDU's)
GESCAPE CRT,104,Param(*)         ! send thickness
INTEGER Param(1:2)              ! an array for the command
Param(1) = 11                   ! line thickness code
Param(2) = thickness            ! desired thickness (in 1/100 mm)
GESCAPE CRT,104,Param(*)         ! send thickness
```

Line Caps and Joins

By default, the device driver uses round caps to end lines and round joins to end lines, which simulates the round pens used on pen plotters. This can be changed with the following statements.

```
INTEGER Param(1:3)                ! an array for the command
Param(1) = 12                     ! set line cap and join
Param(2) = cap                    ! desired line cap
Param(3) = join                   ! desired line join
GESCAPE CRT,104,Param(*)          ! set cap and join
```

The values for *cap* and *join* can be selected from the following tables.

cap	meaning	join	meaning
1	butt cap	1,2	mitered join, beveled if too long
2	square cap	3,4	round join
3,4	round cap	5,6	beveled join

Crosshatching

By default, the PostScript plotter driver fills areas with shades of gray or color (if the COLOR option has been specified). The driver can be made to crosshatch filled areas with the following statements.

```
INTEGER Param(1:2)              ! an array for the command
Param(1) = 1                    ! set fill type
Param(2) = state                ! turn solid filling on or off
GESCAPE CRT,104,Param(*)        ! send command
```

State is 0 to use crosshatching and any other value to use solid filling.
When crosshatching is turned on, the following sets of statements can be used to control the crosshatch parameters. If these statements are not executed, crosshatching is done with solid horizontal lines spaced 0.01 in. (0.4 mm) apart.

```
INTEGER Param(1:2)              ! an array for the command
Param(1) = 2                    ! set crosshatch type
Param(2) = type
GESCAPE CRT,104,Param(*)        ! send command
```

Type is 1 for single hatching, 2 for crosshatching.

```
INTEGER Param(1:2)              ! an array for the command
Param(1) = 3                    ! set crosshatch angle
Param(2) = angle                ! desired angle, degrees
GESCAPE CRT,104,Param(*)        ! send command
```

Angle is the angle in degrees (regardless of the HTBasic RAD or DEG setting) for hatching. *Angle* is rounded to the nearest integer.

```
INTEGER Param(1:2)              ! an array for the command
Param(1) = 4                    ! set line spacing
Param(2) = spacing              ! desired spacing (in 1/100 GDU's)
GESCAPE CRT,104,Param(*)        ! send command
INTEGER Param(1:2)              ! an array for the command
Param(1) = 5                    ! set line spacing
Param(2) = spacing              ! desired spacing (in 1/100 mm)
GESCAPE CRT,104,Param(*)        ! send command
```

The above commands are equivalent except that in the first command, spacing is expressed in 1/100 GDU and in the second in 1/100 mm.

```
INTEGER Param(1:3)              ! an array for the command
Param(1) = 6                    ! set line type for hatching
Param(2) = type                 ! desired line type
Param(3) = size                 ! desired pattern repetition size
GESCAPE CRT,104,Param(*)        ! send command
```

Type is the type of line, as listed under the LINE TYPE topic in the on-line *HTBasic Reference Manual*. *Size* is the pattern repetition length in 1/100 GDU's. This would be 100 times the pattern repetition length specified in a LINE TYPE command.

Pages

The GCLEAR statement causes subsequent plotting to be done on a new page. The driver inserts a PostScript "%%Page" comment at the beginning of each page. The comments are used by some print spooling software. Also, opening a file with

```
PLOTTER IS "file","PS";APPEND
```

causes the driver to append new pages of plot information to the current file if it exists already. Since the driver doesn't know how many pages are already in the file, it begins its "%%Page" comments with page 1. This may cause problems with some print spooling software.

Ending Plots

The PostScript language requires information at the end of a plot to cause the plot to be printed. This information is output when the GCLEAR statement is executed, HTBasic is exited or when the PLOTTER IS device is set to a different device. It is recommended that a statement like

```
PLOTTER IS CRT,"INTERNAL"
```

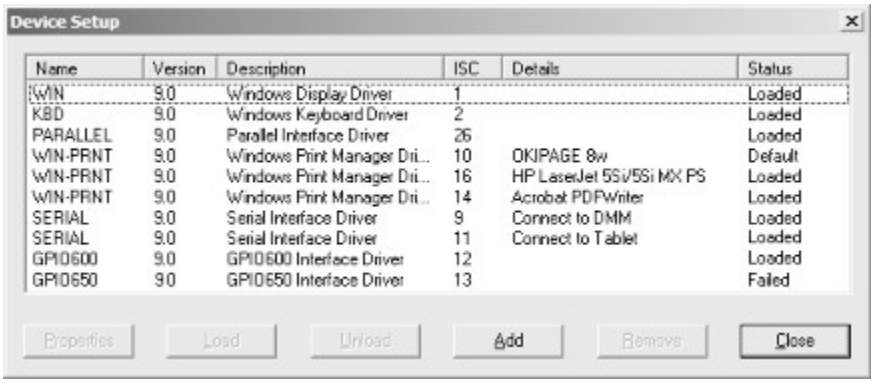
be placed at the end of each program section that produces a plot using the PostScript driver.

I/O Device Drivers

HTBasic provides loadable drivers for support of different interfaces and I/O ports. I/O drivers for the CRT, KBD, processor, parallel port and printer are built into HTBasic. An additional 27 I/O drivers can be loaded for a total of 32 drivers, one for each available ISC. These other drivers should be specified in the Device Setup Dialog or in the AUTOST file.
This chapter describes how to load and use the "WIN-PRINT" driver, an interface to the Windows Print Manager, and the parallel driver, an interface to the parallel ports, the Serial, IEEE-488 I/O GPIB, and GPIO drivers bundled with HTBasic.

Device Setup

To load an I/O Device Drivers using the Device Setup Dialog box, select Tools | Device Setup. This will cause the "Device Setup" dialog box to appear. Loading drivers using LOAD BIN or equivalent statements, will cause them to appear in the list of drivers listed in the "Device Setup" dialog box. The "Device Setup" dialog box displays the Name, Version, Description, Status, and the associated ISC (if applicable) of the driver. The Status of the drivers will indicate whether or not they are loaded (See Figure 6-1).



Click on the "Add" button of the dialog box. This will cause the "Device Driver Selection" dialog box to appear. Highlight the desired device driver. To add a printer, select "WIN-PRNT: Windows Print Manager Driver". Once "Add" has been selected, the "Configure Print Driver Properties" dialog box will appear (see Figure 6-2).

In this dialog box, you can select the print driver configuration you desire. For the print driver, you can choose associations for a printer, a font, a page orientation and an ISC, as well as set properties for the printer. Once you have made the desired selections, click on the "OK" button of the dialog to confirm your choices. The new print driver will now appear among the list of drivers.

Unloading Drivers

To remove a driver, select File | Device Setup. This will cause the "Device Setup" dialog box to appear. Select the print driver you wish to remove from the list of drivers. Next Unload the driver if it is loaded. Then to remove the diver, click on the "Remove" button of the dialog box. Note: the default print driver cannot be removed. The selected print driver will now be removed from the list of drivers.

To modify the configuration for a driver, select File | Device Setup. This will cause the "Device Setup" dialog box to appear. Select the driver you wish to modify from the list of drivers and click on the "Properties" button of the dialog box. Once "Properties" has been selected, the driver's properties dialog box will appear. In this dialog box, you can modify the selected driver.

For the print driver, you can choose to modify any of the settings associated with the print driver, including the printer, font, page orientation, ISC and the properties associated with the printer itself. Once you have made the desired modifications, click on the "OK" button of the dialog to confirm your choices. The modified print driver will now appear among the list of drivers.

WIN-PRINT Driver

The WIN-PRINT device driver is built into HTBasic and allows access to standard windows printer drivers. If a default Windows printer is available, a Win-Print driver will be loaded and assigned ISC 10 automatically. This ISC can be changed as explained later.

The OUTPUT, CONTROL and STATUS statements are supported by this driver, ENTER, READIO, WRITEIO, ON INTR and ENABLE INTR are not.

Printing to the Printer

The WIN-PRINT device driver is built into HTBasic and allows access to standard windows printer drivers. If a default Windows printer is available, a Win-Print driver will be loaded and assigned ISC 10 automatically. This ISC can be changed as explained later.

The OUTPUT, CONTROL and STATUS statements are supported by this driver, ENTER, READIO, WRITEIO, ON INTR and ENABLE INTR are not.

```
20  ASSIGN @Prn TO 10         ! Assign an I/O Path
30  OUTPUT @Prn;"PI = ";PI    ! Now use the I/O Path
40  ! Insert other printer statements here
50  ASSIGN @Prn TO *          ! End of print job, print it
60  END
```

Because Windows is a multitasking environment in which several programs may try to print at once, Print Manager collects printer output into "jobs." Only when a job is done is it printed. Normally, the WIN-PRINT driver ends the job when the I/O path is closed. If you specify the ISC explicitly, you must also explicitly end the print job. The following conditions end a print job: changing the PRINTER IS device, closing the I/O path, executing the RESET statement, resetting the interface through control register 0, or writing a value to control register 111. For example:

```
10  OUTPUT 10;"MAXREAL = ";MAXREAL
20  CONTROL 10,111;1
30  END
```

Printer Control

Regardless of the actual printer type, the WIN-PRINT driver responds to the control characters in the following table. All other characters will be printed (if possible).

Character	Function
CHR$(10)	Move the print location down one line
CHR$(12)	Move the print location to the top of the next page
CHR$(13)	Move the print location to column one

The default margins are 1.27 cm (1/2 inch). Characters printed past the right margin are discarded. When the line advances past the bottom margin, the print location moves to the top of the next page.

Change the margins using CONTROL registers 104 to 107 as shown in the example below. Margins are specified in 1/100's mm. In other words, a value of 100 specifies 1 mm. To specify values in inches, take the desired value in inches and multiply by 2540. Some printers have minimum margins. Attempting to set the margins smaller than allowed by the printer results in an error.

```
10  CONTROL 10,104;1*2540         ! Left   margin: 1 inch
20  CONTROL 10,105;2*2540         ! Top    margin: 2 inches
30  CONTROL 10,106;1/2*2540       ! Right  margin: 1/2 inch
40  CONTROL 10,107;4000           ! Bottom margin: 4 cm
```

Selecting a Printer

Two options are available for printing in HTBasic for Windows. Either direct printing to the parallel port or routing through the Windows Print Manager.

Direct access to the parallel port is available through ISC 26. This is particularly helpful when sending escape codes to the printer that the print manager normally removes from the print command.

When you OUTPUT to ISC 10, HTBasic sends the data to the default printer. At least one printer must be installed. To install a Windows printer driver, please consult your Windows documentation. Briefly, double-click on "My Computer," double-click on "Printers Folder," and double-click on "Add Printer."

Printing in HTBasic is divided into jobs. A job begins when the print spool file is opened and ends when the file, if not empty, is sent to the printer. A new print job begins on an ISC when no current print job exists and one of the following occurs:

- An OUTPUT Isc command is executed
- The first point is plotted after PLOTTER IS Isc
- A DUMP GRAPHICS command is executed and the current DUMP DEVICE is Isc
- A CAT, LIST, or PRINT command is executed after PRINTER IS Isc

If there is a print job associated with Isc, it ends when:

- CONTROL Isc,111;1 (FLUSH) command is executed
- RESET Isc command is executed
- PLOTTER IS Isc command is executed after PLOTTER IS Isc
- PRINTER IS Isc command is executed after PRINTER IS Isc
- DUMP GRAPHICS command finishes when the current DUMP DEVICE is Isc
- CAT or LIST command with PRINTER IS Isc terminates

To send data to a different printer, change the default printer using the Device Setup dialog or use CONTROL register 102 to access the Printer Setup dialog. The user can then select the desired printer. The user can also change the orientation of the print job, portrait or landscape. If the printer does not support the current margin settings, they are automatically adjusted. The following statement causes a Printer Setup dialog box to pop-up.
CONTROL 10,102;1 ! Let user choose printer

```
CONTROL 10,102;1 ! Let user choose printer
```

WIN-PRINT CONTROL Registers

The following CONTROL registers are supported for ISC 10 and the Windows print manager. When using ISC 26 for direct routing to the parallel port only control registers 0, 101, 102, 111, 112, 113 and status register 0 are available.

0 Reset. Ejects all queued print jobs on the ISC specified. The value must be non-zero.

101 Change interface select code. For example, to change the interface select code from 10 to 12, you would use:
CONTROL 10,101;12
To later change it back to 10, you would use:
CONTROL 12,101;10

102 Invoke Printer Setup dialog box. The value must be one. The action of other values is undefined.
For 26, this allows changing of the LPT port number - i.e. `CONTROL 26,102;4` for LPT 4.
The port must be available in Windows.

103 Invoke Printer Setup dialog box. The value must be one. The action of other values is undefined.
For 26, this allows changing of the LPT port number - i.e. `CONTROL 26,102;4` for LPT 4.
The port must be available in Windows.

104 Set left margin. Units are 1/100's mm. Default is 1270. Sets position to left margin, not top of page, so may be used as an indent.

105 Set top margin. Units are 1/100's mm. Default is 1270.

106 Set right margin. Units are 1/100's mm. Default is 1270. Does not move print position, so may be used as an outdent.

107 Set bottom margin. Units are 1/100's mm. Default is 1270. Does not move print position.

108 Set line spacing. Units are 1/100's mm.

109 Set the current print position. Units are 1/100's mm. Register 109 sets the X and register 110 sets the Y values.

110 Set the current print position. Units are 1/100's mm. Register 109 sets the X and register 110 sets the Y values.

111 End the current print job and flush spooled output. The value must be 1.

112 PRINTER IS auto eject: 0 = turns auto eject off; 1 = turns auto eject on (this is ON by default).

113 DUMP auto eject: 0 = turns auto eject off for DUMP commands; 1 = turns auto eject on for DUMP commands
(this is ON by default).

114 Paper orientation: 1 = Set the selected printer's paper orientation to portrait; 2 = Set the selected printer's paper
orientation to landscape.

115 Changes the font associated with the printer to be the name specified in "Fontname". The font name must be truetype font name, else the original font will remain. The printer associated with the ISC will have its font changed:
CONTROL Isc,115;"Fontname"

116 Font style control. Changes the font style associated with the printer to the specified attribute:
0 = Normal;
1 = Italic;
2 = Bold;3 = Bold Italic;
4 = Underline;
5 = Underline Italic; 6 = Bold Underline;
7 = Bold Italic Underline.

117 Changes the font point size for the associated printer to the size recorded in Value:
CONTROL Isc,117;Value

150 Executes a DUMP GRAPHICS

151 INVERT Option for DUMP GRAPHICS: 1 = Inverts the dump, 0= Non-inverted dump.

152 Places a frame around the DUMP GRAPHICS area.

154 Sets X1 or the beginning row for specifying partial screen dumps see GESCAPE PRT,106

155 Sets Y1 or ending row for specifying partial screen dumps

156 X2 or beginning column for specifying partial screen dumps

157 Y2 or ending column for specifying partial screen dumps

WIN-PRINT STATUS Registers

The following STATUS registers are supported.

0	Interface identification. Returns a 302 for ISC 10; 300 for ISC 26.
104	Get left margin. Units are 1/100's mm. Default is 1270.
105	Get top margin. Units are 1/100's mm. Default is 1270.
106	Get right margin. Units are 1/100's mm. Default is 1270.
107	Get bottom margin. Units are 1/100's mm. Default is 1270.
108	Get line spacing. Units are 1/100's mm.
109	Get the current print position. Units are 1/100's mm. Register 109 gets the X and register 110 gets the Y values.
110	Get the current print position. Units are 1/100's mm. Register 109 sets the X and register 110 sets the Y values.
111	Retrieve the "page dirty" flag. A value of 0 means no output exists yet on this page. A value of 1 indicates output has been made to the current page, but the print job is not yet complete.
112	Returns the PRINTER IS auto eject status: 0 = auto eject off; 1 = auto eject on.
113	Returns the DUMP auto eject status: 0 = auto eject off; 1 = auto eject on.
114	Returns the associated printer's paper orientation: 1 = portrait; 2 = landscape.
116	Returns the font style for the associated printer: 0 = Normal 1 = *Italic* 2 = **Bold** 3 = ***Bold Italic*** 4 = Underline 5 = *Underline Italic* 6 = **Bold Underline** 7 = ***Bold Italic Underline***
117	Returns the font point size for the associated printer.
154	Returns X1 or the beginning row for specifying partial screen dumps see GESCAPE PRT,106
155	Returns Y1 or the ending row for specifying partial screen dumps
156	Returns X2 or the beginning column for specifying partial screen dumps
157	Returns Y2 or the ending column for specifying partial screen dumps

Serial (RS-232) Driver

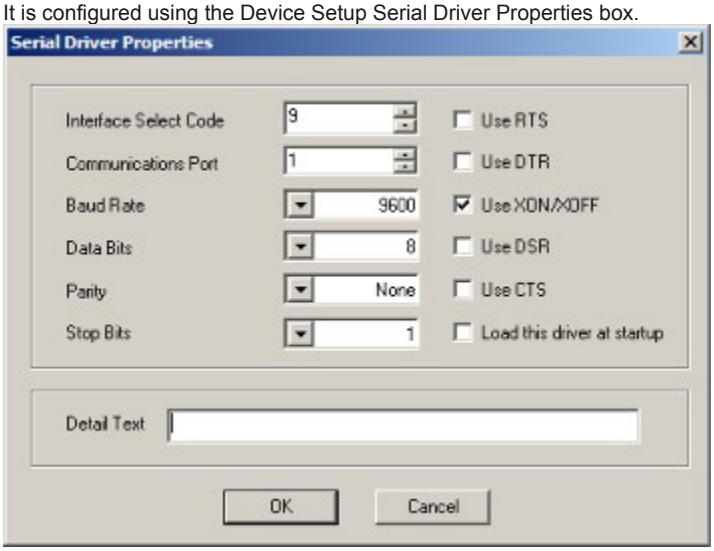
A serial interface driver named "SERIAL" is included with "HTBasic". Using Device Setup, this driver supports Windows compatible serial interfaces. It will work with third-party Windows communications drivers if they adhere to the Windows interface standard for serial drivers. Many devices can be connected to the serial ports and controlled with this driver. Previously in HTBasic for Windows the SERIAL32 driver was used. This has been replaced with the new SERIAL driver.

Loading

The SERIAL driver is loaded by the HTBasic command

```
LOAD BIN "SERIAL[;COM1 Isc][COM2 Isc]…[COMX Isc]"
```

It is configured using the Device Setup Serial Driver Properties box.



This driver controls up to twenty-seven serial ports, named COM1 to COM256. After the serial driver is loaded, HTBasic takes control of the serial ports assigned to it. Until the driver is unloaded, or HTBasic terminates, no other process can use those serial ports. If loading the driver from the command line, or programmatically, it is necessary to specify the ISC associated with each COM port.

Modes of Operation

Baud rates. The baud rate is specified in the Properties box or may be changed by the CONTROL n,3;rate command, where n is the ISC of the serial port and rate is the baud rate. The possible baud rates are dependent on the hardware.

Character Framing. When the driver is loaded the default character framing is undefined. The character framing (bits per character and parity) is set using the Properties box, or by using the CONTROL n,4;value command, where n is the ISC of the serial port and value is one of the values listed in Chapter 9 of the HTBasic User's Guide for control register 4. The possible character framing is dependent on the hardware.

Handshaking. When the driver is first loaded, the serial port is set to use XON/XOFF handshaking and to ignore the CTS and RTS lines.

The use of the RTS and CTS lines can be enabled using the Properties box, or with CONTROL n,5;value and CONTROL n,12;value commands. XON/XOFF software handshaking can be enabled or disabled using the CONTROL n,100;value command. These commands are explained in the HTBasic User's Guide, Chapter 9.

Interrupts

Interrupts are handled by the operating system and by HTBasic. The ON INTR and ENABLE INTR commands are supported for the serial ports and the CONTROL and STATUS registers dealing with interrupts are detailed in Chapter 9 of the HTBasic Users Guide.

READIO/WRITEIO

READIO and WRITEIO are supported and can be referenced in Chapter 9 of the HTBasic User's Guide.

Note: If you are using a serial port only for a printer, you should use the "WIN-PRINT" interface rather than the "SERIAL" interface. "WIN-PRINT" was described earlier in this section. If a mouse or printer is not using the serial port and if the required Windows drivers are present for that port, HTBasic can drive that port with its serial port driver, described in this section.

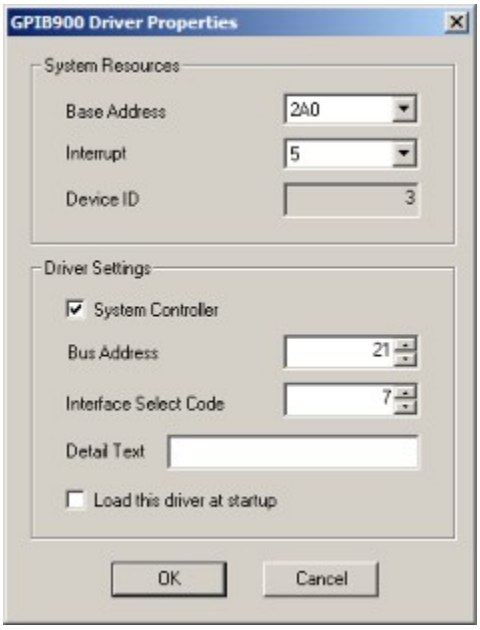
IEEE-488 "GPIB900" Driver

The HTBasic loadable device drivers support many IEEE-488 cards. The "GPIB900" driver is one of three drivers distributed with HTBasic for Windows, the other two drivers are "HPIBS" and "GPIBNI". The GPIB900 driver is designed for the TransEra Model 900 and compatible cards. Previously two separate drivers supported the TransEra 900 card. The GPIB driver for Windows 95/98/Me and HPIBS for NT/2000/XP. The new GPIB900 driver supports the TransEra 900 card under all operating systems. The GPIB driver is still provided for legacy systems.

Loading

The driver is configured using Device Setup. Once the driver is configured it may be loaded using the Device Setup, or by including a line like the following in your AUTOST file:

```
LOAD BIN "GPIB900;DEVICE Id [ISC Isc] [BASE Address][INTERRUPT Interrupt][BUS Bus][NOTSYS|
SYSTEM]"
```



Options

The legal options for the IEEE-488 drivers are:

DEVICE Device_idBASE AddressINTERRUPT InterruptISC IscBUS BusNOTSYSSYSTEM

One or more options can be specified, each separated by a space. The option may be abbreviated, as long as the abbreviation is unique.

NOTE: Under NT/2000/XP it is necessary to reboot the system prior to loading the driver for proper device driver configuration.

DEVICE ID

Each driver in the Device setup is uniquely identified by its Device ID. This ID or identification number must be specified when loading the GPIB900 driver from the command line or programmatically.

BASE I/O Address Option

The BASE option tells the driver the beginning I/O address of your board. You must specify the address in hexadecimal, with no leading "&H".

The default Chip Address for the HM900 board is &H02B8. Addresses in the range &H02A0 to &H02BF are used. The value the LOAD BIN statement is expecting is the lower limit of this address range, 2A0. If any address in the Address Range conflicts with other hardware installed in your computer, the address must be changed. It can be set to any address between &H0018 - &H03F8, in increments of &H20. The board uses 24 I/O addresses below and 8 above the specified address. This range is listed as the Address Range in the following table:

Chip	Switch Number							Address
Address	7	6	5	4	3	2	1	Range
018	on	on	on	on	on	on	on	000-01F
038	on	on	on	on	on	on	off	020-03F
058	on	on	on	on	on	off	on	040-05F
078	on	on	on	on	on	off	off	060-07F
098	on	on	on	on	off	on	on	080-09F
0B8	on	on	on	on	off	on	off	0A0-0BF
0D8	on	on	on	on	off	off	on	0C0-0DF
0F8	on	on	on	on	off	off	off	0E0-0FF
118	on	on	on	off	on	on	on	100-11F
138	on	on	on	off	on	on	off	120-13F
158	on	on	on	off	on	off	on	140-15F
178	on	on	on	off	on	off	off	160-17F
198	on	on	on	off	off	on	on	180-19F
1B8	on	on	on	off	off	on	off	1A0-1BF
1D8	on	on	on	off	off	off	on	1C0-1DF
1F8	on	on	on	off	off	off	off	1E0-1FF
218	on	on	off	on	on	on	on	200-21F
238	on	on	off	on	on	on	off	220-23F
258	on	on	off	on	on	off	on	240-25F
278	on	on	off	on	on	off	off	260-27F
298	on	on	off	on	off	on	on	280-29F
2B8	on	on	off	on	off	on	off	2A0-2BF
2D8	on	on	off	on	off	off	on	2C0-2DF
2F8	on	on	off	on	off	off	off	2E0-2FF
318	on	on	off	off	on	on	on	300-31F
338	on	on	off	off	on	on	off	320-33F
358	on	on	off	off	on	off	on	340-35F
378	on	on	off	off	on	off	off	360-37F
398	on	on	off	off	off	on	on	380-39F
3B8	on	on	off	off	off	on	off	3A0-3BF
3D8	on	on	off	off	off	off	on	3C0-3DF
3F8	on	on	off	off	off	off	off	3E0-3FF

If the I/O address is changed, it must also be specified in the software.

```
LOAD BIN "GPIB900;DEVICE 3 BASE 3A0"
```

This statement corresponds to an I/O address of &H3B8 and the options should be set as follows:

Chip	Switch Number							Address
Address	7	6	5	4	3	2	1	Range
3B8	on	on	off	off	off	on	off	3A0-3BF

INTERRUPT Option

The INTERRUPT option tells the driver the interrupt number used by the board. Again, consult the manufacturer's documentation if you do not know the interrupt used by your board. If you do not specify an interrupt, then 5 is assumed.
To expand our example, if your TransEra board at 2B8(hex) is set to use interrupt 7, you would use the following LOAD BIN statement:

```
LOAD BIN "GPIB900;DEVICE 3 BASE 2A0 INTERRUPT 7"
```

ISC Option

The ISC option is used to specify the interface select code that HTBasic will use when referring to the board. Normally, you do not specify an ISC and it is automatically set to 7 to match an HP BASIC workstation. But if you are using multiple IEEE-488 boards in your PC, you must use the ISC option when loading additional drivers so that each has a unique ISC.

.

For example, suppose in addition to our example 900 board at 2A0(hex) we also have a TransEra board at 3A0(hex) using interrupt 5. The following two lines would load drivers for the two boards:

```
LOAD BIN "GPIB900;DEVICE 5 BASE 2A0 ISC 7"LOAD BIN "GPIB900;DEVICE 6 BASE 3A0 ISC 8"
```

The first board would have an ISC of 7 and the second board would have an ISC of 8.

SYSTEM and NOTSYS Options

The NOTSYS option is used if another computer on the bus is system controller. The option stands for "NOT SYStem controller" and causes the board to act as a Talker/Listener Device that can become active controller if control is passed to it.

The SYSTEM option is used to make the board act as system controller. These options are useful for overriding the hardware options on the board. HTBasic assigns a primary address of 21 to the board if it is system controller and 20 if not. The primary address may be changed using Device Setup or with CONTROL register 3 Remember: Only one system controller can be connected to the bus.

BUS Option

The BUS option specifies the cards BUS address. This must be set if it is anything other than the default of 21. If the card is loaded as NOTSYS, by default the card's BUS address is set to 20.

IEEE-488 "HPIBS" Driver

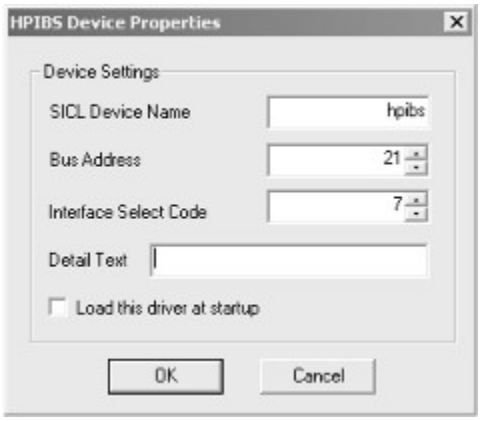
The HTBasic loadable device drivers support many IEEE-488 cards. The "HPIBS" driver is one of three drivers distributed with HTBasic for Windows, the other two drivers are "GPIB900" and "GPIBNI". The HPIBS driver supports Hewlett-Packard / Agilent IEEE and compatible interfaces. Before this driver can be used, the Agilent I/O Libraries (SICL) software must be installed, and the interface must be configured correctly for SICL. Many parameters and configuration options such as default timeouts are controlled in the SICL software..

Loading

To use this driver, the Agilent I/O Libraries software must be first installed. With this installed, load the driver from inside of HTBasic from Device Setup or with the following statement:

```
LOAD BIN   "HPIBS[;DEVICE Device_name][ISC Isc][BUS Bus][TIMEOUT Timeout][NOTSYS|SYSTEM]"
```

where Device_name is the name of the device assigned to the GPIB card in the I/O Libraries setup.



Options

The legal options for the IEEE-488 drivers are:

DEVICE Device_nameISC IscBUS BusTIMEOUT TimeoutNOTSYSSYSTEM

DEVICE Option

The general technique of using options with the LOAD BIN command is explained earlier in this chapter.
This option specifies the SICL symbolic name of the interface to be controlled. The symbolic name of each interface is set by the SICL I/O Configuration Utility. If you do not specify a DEVICE name, the default name "hpib" is used. An example of specifying a different symbolic name is:

```
LOAD BIN "HPIBS;DEV ieee7"
```

ISC Option

The ISC option is used to specify the interface select code that HTBasic will use when referring to the board. Normally, you do not specify an ISC and it is automatically set to 7 to match an HP BASIC workstation. But if you are using multiple IEEE-488 boards in your PC, you must use the ISC option when loading additional drivers so that each has a unique ISC.

BUS Option

The BUS option specifies the cards BUS address. This must be set if it is anything other than the default of 21. If the card is loaded as NOTSYS, by default it's BUS address is set to 20.

SYSTEM and NOTSYS Options

To control the SYSTEM vs NOTSYStem option for the HPIBS driver, select this information in the Agilent I/O Configuration Utility. These options may be specified in LOAD BIN statement, however if they conflict with the settings in the configuration utility, an error will be generated.

```
LOAD BIN "HPIBS;TIME 12"
```

Interface Registers

CONTROL Registers. The IEEE-488 chapter of the *User's Guide* documents the STATUS and CONTROL registers normally available for GPIB interfaces. The HPIBS driver supports all of these registers, with the following exceptions:

STATUS Registers. The HPIBS driver does not support user control of NDAC holdoff. CONTROL register 4 is not implemented and will give Error 55.

The following ENABLE INTR events are not supported by the HPIBS driver. Status register 5 always reports them as 0:

Bit	Value	Meaning
6	64	Handshake Error
5	32	Unrecognized universal command
4	16	Secondary command while addressed
2	4	Unrecognized addressed command

All other HP-IB interrupt events are supported and reported properly in STATUS registers 4 and 5.
The "LSB of last address" bit (bit 8) in STATUS register 6 is not implemented by the HPIBS driver and is always reported as 0.

STATUS register 7 has a high byte that reports bus control lines and a low byte that reports bus data lines. The HPIBS driver does not support the direct reading of bus data lines the upper byte of this register is supported, but the lower byte is always 0.

CONTROL and STATUS Register 255. In addition to the STATUS and CONTROL registers described in the *User's Guide*, the HPIBS driver provides register 255. This register is similar to register 255 of HP BASIC/UX drivers. Unused bits are ignored by the CONTROL statement and return 0 to the STATUS statement. The register map follows.

Bit	Value	Meaning
7	128	Not Used
6	64	Not Used
5	32	Reserved for future use
4	16	Reserved for future use
3	8	Reserved for future use
2	4	ENTER buffering
1	2	Not Used
0	1	Interface Lock

Some examples using register 255 follow:

```
CONTROL 7,255;4    !Enables buffering for ENTER
CONTROL 7,255;0    !Disables buffering for ENTER
CONTROL 7,255;4+1  !Lock GPIB, enable buffering
STATUS7,255;Stat   !Get status of register 255
```

Interface locking can be used to help HTBasic cooperate with other Windows applications that might also access SICL-based interfaces. Setting bit 0 locks the interface, while clearing bit 0 unlocks the interface.

ENTER buffering can increase the speed of free-field ENTER statements. A free-field ENTER is one without a USING clause, such as:

```
ENTER @Dev;A$
```

Unsupported Keywords

READIO and WRITE/IO of interface registers are not supported by the HPIBS driver. Attempts to use READIO or WRITE/IO cause Error 170.

PPOLL Note. Parallel Poll configuration is automatic with the HPIBS driver. The Active Controller can configure a non-controller PC GPIB interface for parallel poll, and the PC interface will respond correctly based upon that configuration.
If bit 14 of the Interrupt Enable register is set, HTBasic for Windows will receive an interrupt indicating that a parallel poll configuration has occurred. There is no way, however, to know the value of the configuration byte that came from the Active Controller.

You can set the parallel poll response by using CONTROL register 2 or 5, or you can automatically accept the configuration sent by an Active Controller.

IEEE-488 "GPIBNI" Driver

The HTBasic loadable device drivers support many IEEE-488 cards. The "GPIBNI" driver is one of three drivers distributed with HTBasic for Windows, the other two drivers are "GPIB900" and "HPIBS". This driver supports NI and compatible cards. This Windows driver calls the National Instruments Software that is shipped with the NI or compatible GPIB card. Many parameters and configuration options such as default timeouts are controlled in the NI software.

Compatible cards include many manufacturers including Capitol Equipment, Computer Boards, I/OTech and Keithley. Please contact you card manufacturer for details. For details on installation of this software, see your Getting Started Guide for your GPIB card.

Loading

To use this driver, the National Instruments software must be first installed. With the NI configuration software installed and configured, load the driver from inside of HTBasic from Device Setup or with the following statement:

```
LOAD BIN   "GPIBNI[;DEVICE Device_name] [ISC Isc] [BUS Bus] [NOTSYS|SYSTEM]"
```

where Device_name is the name of the device assigned to the GPIB card in the NI GPIB control panel.

Options

The legal options for the IEEE-488 drivers are:

DEVICE Device_nameISC IscBUS BusNOTSYSSYSTEM

DEVICE Option

The general technique of using options with the LOAD BIN command is explained earlier in this chapter.

This option specifies the card name as specified by the National Instruments' card Configuration Utility. If you do not specify a DEVICE name, the default name "GPIB0" is used. An example of specifying a different card name is:

```
LOAD BIN "GPIBNI;DEV GPIB7"
```

ISC Option

The ISC option is used to specify the interface select code that HTBasic will use when referring to the board. Normally, you do not specify an ISC and it is automatically set to 7 to match an HP BASIC workstation. But if you are using multiple IEEE-488 boards in your PC, you must use the ISC option when loading additional drivers so that each has a unique ISC.

BUS Option

The BUS option specifies the cards BUS address. This must be set if it is anything other than the default of 21. If the card is loaded as NOTSYS, by default it's BUS address is set to 20.

System and NOTSYS Options

The NOTSYS option is used if another computer on the bus is system controller. The option stands for "NOT SYStem controller" and causes the board to act as a Talker/Listener Device that can become active controller if control is passed to it.

The SYSTEM option is used to make the board act as system controller. These options are useful for overriding the hardware options on the board. HTBasic assigns a primary address of 21 to the board if it is system controller and 20 if not. CONTROL register 3 can be used to change the primary address. Remember: Only one system controller can be connected to the bus.

GPIO 600/650 Drivers

The "GPIO600" and the "GPIO650" drivers are distributed with HTBasic for Windows for the TransEra Model 600 and 650 GPIO cards. The function of the General Purpose Input Output (GPIO) interface is to provide communication with digital peripherals. For detailed installing and usage instructions please see the "GPIO Interface Installing & Using Manual"

Loading

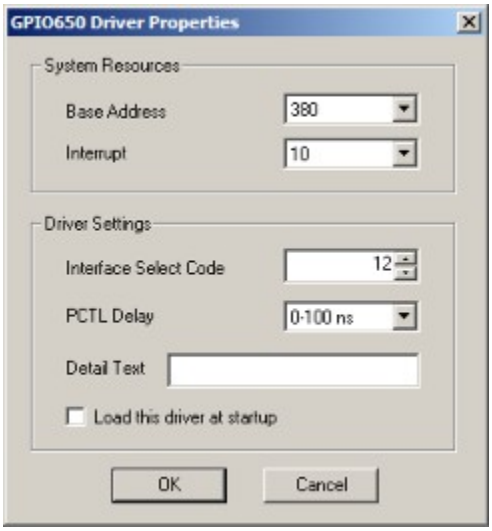
The drivers are configured using Device Setup. Once a driver is configured it may be loaded using the Device Setup, or by including a line like the following in your AUTOST file:

Model 600 Board

```
LOAD BIN "GPIO600[;ISC Isc][BASE Address][INTERRUPT Int]"
```

Model 650 Board

```
LOAD BIN "GPIO650[;ISC Isc][BASE Address][INTERRUPT Int] [DELAY Delay]"
```



Printer and Pixel Image Device Drivers

HTBasic supports several dump drivers. The CONFIGURE DUMP statement is used both to load drivers and switch among them. The following table lists all available drivers.

Name	For these printers
WIN-DUMP	Send the dump to the default Windows printer
PCL	Advanced HP-PCL driver
PS-DUMP	PostScript printers, devices and files
GIF	Graphic Interchange Format files

The CONFIGURE DUMP Statement

The CONFIGURE DUMP statement specifies what graphic printer language or bitmapped file format to use for the DUMP statement. When the DUMP GRAPHICS statement is executed or the DUMP GRAPHICS Key is pressed, the contents of the screen are copied to a printer, device or file. If a file is specified, verify that it is an *ordinary file*; otherwise the HTBasic file header will appear as bad data at the start of the file. The syntax is

```
CONFIGURE DUMP TO language
```

where *language* is a string expression naming the printer language and driver options. For example,

```
CONFIGURE DUMP TO "PCL"
```

DUMP Device Statement

The DUMP DEVICE statement controls where the output of the DUMP is directed. There are several choices for this. When using the WIN-DUMP driver, it is necessary to direct the CONFIGURE DUMP to an ISC which is set as a Windows printer. ISC is set by default as the default Windows printer. When using any other option, is it necessary to use the ISC of the device where the output is to be directed. For the parallel port, this is ISC 26. For a device on the IEEE-488 bus, this would be the device's address. For a file, this would be the file name.

Loading Drivers

It is recommended that CONFIGURE DUMP statements be included in your AUTOST file to load any necessary drivers. Up to ten graphic and dump drivers can be loaded.

The first time a driver is specified in a CONFIGURE DUMP statement, the driver is loaded and graphics are directed to it. When the driver is subsequently specified, it is not loaded again, but graphics are again directed to it.

As an example, if you wish to use an HP LaserJet II for screen dumps, use the following command to change to the HP printer control language:

```
CONFIGURE DUMP TO "PCL"
```

If a DUMP is made before doing a CONFIGURE DUMP, HTBasic automatically loads and uses the WIN-DUMP driver.

Dumping Rows

To specify a portion of the screen to dump when DUMP GRAPHICS is executed, the GESCAPE PRT,106,param(*) statement is used. The param array must be a one dimensional INTEGER array of five elements. The first element is the operation number. The remaining elements specify the boundary for the DUMP. The boundary is specified in screen units:

param(1) - 1
param(2) - Beginning Row
param(3) - Ending Row
param(4) - Beginning Collumn
param(5) - Ending Collumn

The full screen will be dumped if any of the following conditions occur:

1. The beginning row is greater than ending row.
2. A new Plotter, Graphics or Dump driver is loaded.
3. GINIT, SCRATCH A, PLOTTER IS, GRAPHICS INPUT, CONFIGURE DUMP command or a Basic Reset is executed.

Number of Colors

The number of colors in the DUMP depends on both the dump driver and the CRT driver. All *dump* drivers support black and white dumps. Some dump drivers also can handle 16 or 256 colors. The same is true of *display* drivers. The number of colors in the dump will be the largest number both drivers support.

Options

It is sometimes necessary to specify options to the drivers. Options are included by appending a semicolon to the driver name, followed by the options. Since options vary from driver to driver, the following driver sections contain more details on specific options.

WIN-DUMP Driver

The WIN-DUMP dump driver provides support for any printer supported by Windows that accepts bitmaps. The command to load the WIN-DUMP dump driver is:

```
CONFIGURE DUMP TO "WIN-DUMP[;options]"
```

If a DUMP is made before doing a CONFIGURE DUMP, HTBasic automatically loads and uses the WIN-DUMP driver.

Print Manager

The default interface select code (ISC) for DUMP DEVICE IS is 10, the Windows print manager interface. The WIN-DUMP driver can send dumps to any ISC that is set to a valid Windows printer using the Device Setup Dialog. To bypass the print manager when DUMPing, use ISC 26 in the DUMP DEVICE IS statement.

Because Windows is a multitasking environment in which several programs may try to print at once, Print Manager collects printer output into "jobs." Only when a job is done is it printed. Normally, the WIN-DUMP driver prints a single dump per print job. To mix text and screen dumps or multiple screen dumps on a single page, use the following control statement to toggle off this automatic page eject:

```
CONTROL 26,113;0    ! toggle auto eject off
```

For example, to output some text to the page before doing the dump:

```
10    CONFIGURE DUMP TO "PCL"
20    DUMP DEVICE IS 26
30    CONTROL 26,113;0    ! toggle auto eject off
40    ASSIGN @I TO 26
50    OUTPUT @I;"This is a  screen dump:"
60    DUMP GRAPHICS
70    ASSIGN @I TO *
80    CONTROL 26,113;1    ! toggle auto eject on
90    DUMP GRAPHICS
100   END
```

The various settings, such as margins and line height, made in the WIN-PRINT driver are honored by the WIN-DUMP driver. See the WIN-PRINT driver documentation in Chapter 4 for more information.

The EXPANDED keyword in the DUMP statement is ignored. The DUMP is made in landscape or portrait mode depending on the paper orientation specified in the printer settings, or by Control Register 114 as detailed in Chapter 4.

DUMP Size

By default, the screen image is scaled until it fills 100% of the width between the left and right margins. The size can be changed using GESCAPE code 39. This example sets the scaling to 20% of the width between the margins:

```
10    INTEGER S(1:1)
20    S(1)=20
30    GESCAPE CRT,39,S(*)
40    DUMP GRAPHICS
50    END
```

INVERT Option

By default, black and white are exchanged while other colors remain. This is often suitable for output to a black and white printer, where printing is done with black ink on white paper. The INVERT option causes the colors or gray levels to be dumped exactly as they are on the screen.

PCL Driver

The PCL dump driver provides support for devices and software that accept the Hewlett-Packard PCL printer language or HP Raster Interface Standard graphic commands. The driver supports both DUMP ALPHA and DUMP GRAPHICS from bitmapped displays.

The PCL driver is loaded with a line like:

```
CONFIGURE DUMP TO "PCL[;options]"
```

Options

The options are listed after the semicolon in the driver name, within the quotes. If more than one option is specified, the option names are separated by commas. The options are as follows:

ADJUST. The ADJUST option is ignored in the Windows version of HTBasic. All pixels are considered to be square and the dump is made using the aspect ratio of the window running HTBasic.

BW. This option causes the driver to dump using the paper color for the areas on the screen that were drawn using pen 0 and the ink color (usually black) for the areas on the screen drawn with any other pen. This is reversed if the INVERT option is also used. The BW option need not be specified explicitly; it is the default.

COLOR, CCMY, C16 and C256. These options cause the dump to be done in color to a color printer. The COLOR option uses the printer's default 8-color solid-color palette (black, white, red, green, blue, cyan, magenta and yellow), mapping each color on the screen to the closest one from the palette. COLOR uses the default RGB palette to dump the screen; CCMY uses the default CMY palette. The C16 and C256 options use a 16- or 256-color palette on the printer and only work with printers that have settable color palettes, such as the PaintJet series and the DeskJet 1200C. With printers that use dithering to print mixed colors, you may have to specify a coarser resolution than the printer is capable of in order to enable the dithering; for example, on the original PaintJet printer, C16 and DPI90 together are needed to produce dithering; C16 and DPI180 cause the printer to use only the 8 default colors when printing.

Printing using the COLOR and CCMY options swaps black and white colors when printing, unless the INVERT option is also used.

When using the solid-color palette with older PaintJet printers, the COLOR option should be used, as these printers do not support the CMY color model. The DeskJet 500C and 550C models can only generate color screen dumps with the CCMY option.

COMPRESS. The COMPRESS option specifies that the printer being used can do "packbits"-style data compression. If this option is specified, the screen dump is transmitted to the printer using fewer data bytes. The COMPRESS option can be used with all the LaserJet IIP and IIP+ printers, all LaserJet III and IV series printers, all DeskJet series printers, the PaintJet XL300 printer (but not the older PaintJets), and the DesignJet printers, as well as other brands of printers that emulate these. Note, however, that the printers with slower CPU's will print 2-4 times slower when printing compressed data, so COMPRESS may not be a good option to use with these printers.

EXPANDED. When the EXPANDED keyword is used on the DUMP DEVICE IS statement, graphics screen dumps are placed in landscape. When grayscale dumps are done, the PCL dump driver scales the dump to 10 x 7 inches, if EXPANDED, or 4 x 7 inches, if not EXPANDED. With color or black/white dumps, each screen pixel is dumped to a single printer pixel; use DPI150, DPI100, or DPI75 to increase the dump size in these cases.

DPI*nnn.* This option tells the driver to use *nnn* dots per inch when dumping graphics. Without this option, the printer's default resolution is used. This option is required for the GRAY option, explained below. The resolution specified must be one supported by the printer. For most newer devices, DPI75, DPI100, DPI150 and DPI300 are the legal values for this option. Some older printers, like the Hewlett-Packard ThinkJet, don't support this option.

With the COLOR and BW options, this option controls the size of the dump, by mapping each pixel on the screen to one of the specified-sized dots on the printer; with the GRAY option, this options controls the size of the sub-pixels used to create the printed image, as explained in the GRAY option section.

GRAY. The GRAY option causes the driver to consult the screen's color map and calculate a gray shade for each color using the NTSC grayscale equation. Screen dumps are produced using the resulting shades of gray. If the INVERT option is not also specified, white and black are reversed after the gray shade is calculated, so that lighter colors on the screen become darker colors on the printer.

When dumps are made using this option, the driver calculates the number of printer pixels, as specified in the DPI*nnn* option, required to print a single screen pixel to make a 9 x 6 3/4 inch (23 x 17 mm) plot, up to 4 x 4 printer pixels per screen pixel. The driver sets the appropriate number of printer pixels to black to represent the gray shade of the corresponding screen pixel. The GRAY option is ignored unless the DPI*nnn* option is also specified.
The NTSC grayscale equation is:

brightness = 11% blue + 59% green + 30% red.

INVERT. By default, the driver makes images with black and white exchanged from the values used on the screen. If the GRAY option is used, the driver by default reverses the gray level of all pixels dumped from that seen on the display. This is often suitable for output to a printer, where printing is done with colored inks on white paper, but may not be suitable for film output devices, where an exact image of the screen is wanted. The INVERT option causes the colors or gray levels to be dumped exactly as they are on the screen.

RELATIVE. Normally, the driver begins each dump at the left margin. The RELATIVE option causes the driver to begin each dump at the printer's current print position.

EJECT

Typically, the driver ejects the page after a dump is finished. Control Register 113 toggles the driver to send a Form Feed character to the printer or file at the end of each dump. To turn off Auto Page Eject, use CONTROL PRT,113;0.

APPEND

If the APPEND keyword is used with the DUMP DEVICE IS command and if the dump device is a file, the driver appends dumps to the file, separated by form feeds.

ALPHA Dumps

The DUMP ALPHA command has the exact same function as DUMP GRAPHICS if the ALPHA and GRAPHICS planes are combined.

DUMP ALPHA produces a text dump at the top of a US "A" or European A4- sized sheet of paper if done from a text screen (as selected by PLOTTER IS 3), if ALPHA and GRAPHICS are separated.

Option Tables

The following tables will help in choosing options to use with specific printer models. Two screen sizes are shown, a 640 x 480 pixel screen common on PC's and a 1024 x 768 pixel window size common on UNIX workstations and Windows. The proper DPI*nnn* option for other screen sizes can be determined by experimentation.

OPTIONS for 640x480 Screen

Printer Type	Dump Type	Portrait	Landscape
ThinkJet		too large	none
LaserJet	B/W	DPI100	none
LaserJet	grayscale	GRAY,DPI300	GRAY,DPI300
LaserJet II-IV	B/W	DPI100, COMPRESS	DPI75, COMPRESS
LaserJet II-IV	grayscale	GRAY,DPI300, COMPRESS	GRAY,DPI300, COMPRESS
DeskJet	B/W	DPI100	none
DeskJet	grayscale	GRAY,DPI300	GRAY,DPI300
DeskJet 500/550C	color	CCMY,DPI100	CCMY,DPI75
DeskJet	color	C256,DPI100	C256,DPI75
PaintJet & XL	B/W	DPI90	DPI90
PaintJet & XL	grayscale	GRAY,DPI180	GRAY,DPI180
PaintJet & XL	color	C16,DPI90 or COLOR,DPI90	C16,DPI90 or COLOR,DPI90
PaintJet XL300	B/W	DPI100	none
PaintJet XL300	grayscale	GRAY,DPI300	GRAY,DPI300
PaintJet XL300	color	C256,DPI100	C256,DPI75
DesignJet 600	B/W	DPI100	none
DesignJet 600	grayscale	GRAY,DPI300	GRAY,DPI300
DesignJet 600	color	C256,DPI100	C256,DPI75

OPTIONS for 1024x768 Screen

Printer Type	Dump Type	Portrait	Landscape
ThinkJet		too large	none
LaserJet	B/W	DPI150	DPI100
LaserJet	grayscale	GRAY,DPI300	GRAY,DPI300
LaserJet II-IV	B/W	DPI150, COMPRESS	DPI100, COMPRESS
LaserJet II-IV	grayscale	GRAY,DPI300, COMPRESS	GRAY,DPI300, COMPRESS
DeskJet	B/W	DPI150	DPI100
DeskJet	grayscale	GRAY,DPI300	GRAY,DPI300
DeskJet 500/550C	color	CCMY,DPI150	CCMY,DPI100
DeskJet 560C	color	CCMY,DPI150	CCMY,DPI100
DeskJet 1200C	color	C256,DPI150	C256,DPI100
PaintJet & XL	B/W	DPI180	DPI90
PaintJet & XL	grayscale	GRAY,DPI180	GRAY,DPI180
PaintJet & XL	color	COLOR,DPI180	COLOR,DPI180
PaintJet XL300	B/W	DPI150	DPI100
PaintJet XL300	grayscale	GRAY,DPI300	GRAY,DPI300
PaintJet XL300	color	C256,DPI150 or COLOR,DPI150	C256,DPI100
DesignJet 600	B/W	DPI150	DPI100
DesignJet 600	grayscale	GRAY,DPI300	GRAY,DPI300
DesignJet 600	color	C256,DPI150	C256,DPI100

PS-DUMP Driver

The PostScript dump driver provides support for devices and software that accept the PostScript graphics language. It provides support for both the DUMP ALPHA and DUMP GRAPHICS commands. The DUMP ALPHA and DUMP GRAPHICS commands both dump any text and graphics visible in the window if ALPHA and GRAPHICS are merged.

The PostScript dump driver produces a screen image intended to be rendered on a US "A" size or European A4 size page. It scales the image so that its longest dimension fits in the shortest dimension of the paper with an adequate margin. When the EXPANDED keyword is used on the DUMP DEVICE IS statement, screen dumps change from their normal portrait orientation to landscape orientation.

The PostScript dump driver is loaded with the following statement:

```
CONFIGURE DUMP TO "PS-DUMP[;options]"
```

Options

The options are listed after the semicolon in the driver name, within the quotes. If more than one option is specified, the option names are separated by commas. The GRAY and COLOR options are ignored in ALPHA dumps. The options are as follows:

BW. This option causes the driver to dump using the paper color for the areas on the screen that were drawn using pen 0 and the ink color (usually black) for the areas on the screen drawn with any other pen. This is reversed if the INVERT option is also used. The BW option need not be specified explicitly; it is the default.

GRAY. This option causes the driver to render colors on the computer screen as shades of gray on the printer. Each shade of gray is calculated using the NTSC grayscale equation:

brightness = 11% blue + 59% green + 30% red.

Unless the INVERT option is used, the resulting brightness is inverted before printing, so that dark colors on the computer screen print as light colors and vice-versa.

COLOR. The COLOR option causes the driver to output a color image of the screen. The resulting PostScript screen image can only be rendered on a device that supports Level 2 PostScript or the color extensions of Level 1.

INVERT. By default, the driver makes images with black and white exchanged from the values used on the screen. If the GRAY option is used, the driver by default reverses the gray level of all pixels dumped from that seen on the display. This is often suitable for output to a printer, where printing is done with colored inks on white paper, but may not be suitable for film output devices, where an exact image of the screen is wanted. The INVERT option causes the colors or gray levels to be dumped exactly as they are on the screen.

ADJUST. The ADJUST option is ignored in the windowed versions of HTBasic. All pixels are considered to be square and the dump is made using the aspect ratio of the window running HTBasic.

The APPEND Keyword

If the APPEND keyword is used in the DUMP DEVICE IS statement, the dump driver appends all dump images after the first one to the exseisting file as new pages. The driver inserts "%%Page" comments, used by some print spooling software, into the file at the beginning of each page. If the dumps are done in separate HTBasic sessions, the driver doesn't know which page it is on, so it starts over with page 1. This may be a problem with some spooling software. Also note that only one page can be present in a file that will be imported into a word processor document.

ALPHA Dumps

The DUMP ALPHA command produces a black-and-white graphics image of the entire screen if MERGE ALPHA is in effect. DUMP ALPHA produces a black-and-white image of the text on the screen if SEPARATE ALPHA is in effect. The image is in black and white even if there are three ALPHA pens (as with 256-color displays).

DUMP ALPHA produces a text dump at the top of a US "A" or European A4- sized sheet of paper if done from a text screen.

GIF Driver

The GIF dump driver provides support for software that accepts CompuServe Graphics Interchange Format (GIF) files.
With this driver, the DUMP ALPHA command produces a black-and-white graphics image of the entire screen if MERGE ALPHA is in effect. DUMP ALPHA produces a black-and-white image of the text on the screen if SEPARATE ALPHA is in effect. The image is in black and white even if there are three ALPHA pens (as with 256-color displays).

When the EXPANDED keyword is used on the DUMP DEVICE IS statement, graphics screen dumps are rotated 90 degrees clockwise from their normal orientation.

The GIF dump driver is loaded with the following statement:

```
CONFIGURE DUMP TO "GIF[;options]"
```

Options

The options are listed after the semicolon in the driver name, within the quotes. If more than one option is specified, the option names are separated by commas. The options are as follows:

BW. The driver normally produces a 16- or 256-color screen dump when used with a color screen. The BW option causes the driver to produce a black-and-white screen dump with color screens. In this dump, pixels of color zero are dumped as black and pixels of any other color are dumped as white. (This is reversed if the INVERT option is also specified.)

COLOR. This option causes the driver to produce a color dump. This option is the default; it need not be explicitly specified.

INVERT. The driver normally dumps an image in the colors shown on the screen. The INVERT option causes the driver to reverse black and white in the dump. All other colors are unchanged.

The APPEND Keyword

If the APPEND keyword is used in the DUMP DEVICE IS statement, the GIF dump driver appends all dump images after the first one to the existing file. Note, however, that the screen type and colormap are stored when the first image is dumped. If the screen type or colormap changes, the dump images after the first one will not be correct. Also note that most software that uses the GIF format cannot process multiple images in one file.

Graphic Input Drivers

HTBasic supports loadable GRAPHIC INPUT drivers. The GRAPHIC INPUT driver is used by the DIGITIZE, READ LOCATOR and SET LOCATOR statements. The following table lists the drivers available at the time of this manual printing.

Name	For These Devices
KBD	Keyboard arrow keys or Mouse
KBDA	Keyboard arrow keys or Mouse (Absolute)
HPGL	HPGL Plotters or Digitizers
TABLET	Most available digitizing tablets

The GRAPHICS INPUT IS Statement

The GRAPHICS INPUT IS statement is used both to load drivers and switch among them. The GRAPHICS INPUT IS statement also specifies the interface connecting the device to the computer. The syntax for loading the driver is

```
GRAPHICS INPUT IS device-selector, "driver-name[;options]"
```

The *device-selector* specifies the device or interface to use to communicate with the graphic input device. This is usually KBD, an IEEE-488 device selector or the Serial interface select code. The *driver name* and *options*, shown in literal form above, can be specified with a string expression. *driver-name* is from the table above and *options* are described in the following descriptions. Here are some examples of GRAPHIC INPUT IS statements:

```
GRAPHICS INPUT IS KBD,"KBD"
GRAPHICS INPUT IS KBD,"KBDA"
GRAPHICS INPUT IS 705,"HPGL"
GRAPHICS INPUT IS 705,"TABLET;BIN-2,0,5000,0,5000"
```

Loading Drivers

The first time a driver is specified in a GRAPHICS INPUT IS statement, the driver is loaded and used for graphics input. When the driver is subsequently specified, it is not loaded again, but is again used for graphics input.

It is recommended that GRAPHICS INPUT statements be included in your AUTOST file to load any necessary drivers. Up to ten graphic and dump drivers can be loaded at a time. HTBasic automatically loads the "KBD" driver when it starts.

To find the driver file HTBasic takes the language specified in the GRAPHICS INPUT IS statement and performs several operations upon it to find the correct file. ".DW6" is appended to the name. Then the following locations are searched, in the specified order:

1. The directory containing the HTBasic executable.
2. The current directory.
3. The Windows system directory (such as \WINDOWS\SYSTEM).
4. The Windows directory.
5. The directories listed in the PATH environment variable.

KBD Driver

The keyboard (KBD) graphics input driver provides support for input of X and Y coordinates from the mouse. The KBD driver is loaded at start up. The command to switch back to the KBD graphics input driver from another driver is:

GRAPHICS INPUT IS KBD,"KBD"

The following example program shows how to set up the KBD driver and get coordinate information from the input device.

```
10  PLOTTER IS CRT,"INTERNAL"
20  GRAPHICS INPUT IS KBD,"KBD"
30  TRACK CRT IS ON
40  FRAME
50  DIGITIZE X,Y,S$
60  PRINT X,Y,S$
70  END
```

HPGL Driver

The HPGL graphics input driver provides support for any input device that accepts Hewlett Packard's HPGL language. Some HPGL compatible devices are the HP 9111A and HPGL plotters. The following example assumes an HPGL capable device is attached to the IEEE-488 bus at primary address 5:

```
GRAPHICS INPUT IS 705,"HPGL"
```

TABLET Driver

The TABLET graphics input driver provides support for most digitizers currently available. It can use either the serial port or the IEEE-488 (GPIB or HP-IB) bus to communicate with the tablet. The following guidelines will help you in loading the driver and in selecting the proper tablet configuration and data communication options. The command to load the TABLET graphics input driver is as follows.

```
GRAPHICS INPUT IS Isc,"TABLET;[mode[,]][resolution]"
```

The *mode* option allows you to specify the method in which the tablet's data is interpreted by the driver. If both *mode* and *resolution* options are specified, specify the *mode* option first and separate the two by a comma. The following table gives the legal values for *mode*:

Mode	Meaning
(None)	Comma separated ASCII
BIN-1	Summagraphics MM Binary Format
BIN-2	Hitachi Binary Format
BIN-3	UIOF Binary Format.

If no *mode* is specified, the driver assumes the tablet is using a comma separated, CR/LF terminated, ASCII data format. The ASCII format and the different binary formats are discussed below.

The *resolution* option is of the form Xmin,Xmax,Ymin,Ymax. The *resolution* option is only necessary if the tablets range of X & Y values are different from the default values of 0-11000 in both the X & Y directions. The *resolution* option is discussed in greater detail below.

Examples:

```
GRAPHICS INPUT IS 9,"TABLET"
GRAPHICS INPUT IS 9,"TABLET;BIN-1"
GRAPHICS INPUT IS 705,"TABLET;0,5000,0,5000"
GRAPHICS INPUT IS 705,"TABLET;BIN-2,0,5000,0,5000"
```

Communication

The TABLET driver can use either the serial port or the IEEE-488 bus to communicate with the digitize tablet. This is specified by the interface-select-code in the GRAPHICS INPUT IS statement. For example:

```
GRAPHICS INPUT IS 702,"TABLET"    !IEEE-488 Address 2
GRAPHICS INPUT IS 9,"TABLET"      !First Serial Port
```

Communication with the tablet over the IEEE-488 bus is straightforward. You specify the device-selector (i.e. 702) and the control and data messages proceed without further setup.

Communication with the tablet over the serial port is more involved because of the many serial configuration options. The SERIAL driver can change the number of data bits, parity, stop bits and the baud rate. Make sure that the switches on the tablet are set to match the values used by the SERIAL driver or use CONTROL statements after loading the SERIAL driver to make the SERIAL driver use the same settings as the tablet.

With the SERIAL driver, the tablet may support either XON/XOFF handshaking or hardware handshaking. Find out which method your tablet supports and set the SERIAL driver to use the same handshaking. By default the SERIAL driver uses XON/XOFF handshaking, the following line is all that is needed to set the driver to this method.

```
10  LOAD BIN "SERIAL32"           !Loads SERIAL device driver
```

If you need to use hardware handshaking, you will have to set a number of other registers within the SERIAL driver. The following program lines specify hardware handshaking.

```
10  LOAD BIN "SERIAL32"           !Loads SERIAL device driver
20  CONTROL 9,5;0  !Use DTR and RTS
30  CONTROL 9,12;0 !Read DSR, CD and CTS
40  CONTROL 9,100;0               !Disable XON/XOFF handshaking
```

With some digitizers the RTS line must be held active to make the TABLET driver work correctly, otherwise an error will occur after several successful reads. To hold the RTS line active change program line 20 to CONTROL 9,5;2.

ASCII Data Format

The ASCII method of data transmission is preferred over binary; it is easier to set up and get working. The ASCII format can be used with either XON/XOFF handshaking or hardware handshaking. XON/XOFF handshaking is the preferred method. The ASCII data needs to be comma separated and CR/LF terminated.

The ASCII data format needs to look something like the following line.

```
Sxxxx,Syyyy,F<CR><LF>
```

The meaning of these symbols is given in the following table:

Symbol	Meaning
S	sign flag
xxxx	X value
yyyy	Y value
F	button flag
<CR>	Carriage Return
<LF>	Line Feed

The sign flag doesn't need to be present and the number of X & Y digits doesn't matter. The data cannot contain any decimal points within the string.

Binary Data Formats

Three binary data formats are supported: Summagraphics MM format, Hitachi format, and UIOF format. The type of binary format is selected by specifying the BIN-1, BIN-2 and BIN-3 strings respectively. When using the binary format, XON/XOFF handshaking cannot be used, only hardware handshaking is allowed. The meaning of each bit in the binary formats is listed in the tables below:

BIN-1 - MM Binary Data Format

Byte	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1st	1	PX	T	Sx	Sy	Fc	Fb	Fa
2nd	0	x6	x5	x4	x3	x2	x1	x0
3rd	0	x13	x12	x11	x10	x9	x8	x7
4th	0	y6	y5	y4	y3	y2	y1	y0
5th	0	y13	y12	y11	y10	y9	y8	y7

BIN-2 - Hitachi Binary Data Format

Byte	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1st	1	PX	Fd	Fc	Fb	Fa	x15	x14
2nd	0	x13	x12	x11	x10	x9	x8	x7
3rd	0	x6	x5	x4	x3	x2	x1	x0
4th	0	0	0	0	0	0	y15	y14
5th	0	y13	y12	y11	y10	y9	y8	y7
6th	0	y6	y5	y4	y3	y2	y1	y0

BIN-3 - UIOF Binary Data Format

Byte	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1st	P	1	0	0	0	0	T	PX
2nd	P	0	0	Fe	Fd	Fc	Fb	Fa
3rd	P	0	x5	x4	x3	x2	x1	x0
4th	P	0	x11	x10	x9	x8	x7	x6
5th	P	0	0	Sx	x15	x14	x13	x12
6th	P	0	y5	y4	y3	y2	y1	y0
7th	P	0	y11	y10	y9	y8	y7	y6
8th	P	0	0	Sy	y15	y14	y13	y12

The meaning of each of these symbols is given in this table:

Symbol	Meaning
Sx & Sy	sign flag for X & Y coordinates
x15,...,x0	X coordinate, x0 is least significant bit
y15,...,y0	Y coordinate, y0 is least significant bit
Fe,...,Fa	button flag, Fa is least significant bit
PX	proximity bit
T	tablet identifier

Resolution

The TABLET driver assumes a default maximum resolution of 11000 units in both the X and Y directions. This value is used to scale the digitizer X & Y coordinates to the display WINDOW coordinates. If this value is not correct for your digitizer or if you want to adjust for any distortion, you can change the scaling values with the following command:

```
GRAPHICS INPUT IS 9,"TABLET;Xmin,Xmax,Ymin,Ymax"
```

Xmin and *Xmax* are the digitizer's X values that correspond to the display's minimum and maximum X values respectively. *Ymin* and *Ymax* are the digitizer's Y values that correspond to the display's minimum and maximum Y values respectively.

Please note that these values are specified in device units.

The TABLET driver scales the digitizer X & Y coordinates into the display WINDOW coordinates. For example, suppose the screen's WINDOW resolution is 0-133 in the X direction and 0-100 in the Y direction and the digitizer's X & Y resolution is 0-11000. If the digitizer returned 11000,11000 as the current X & Y location, the DIGITIZE statement will return a value of 100,133 to the user. If you want the X & Y values to be the same for equal movements in the X & Y directions, specify a square WINDOW.

For example:

```
WINDOW 0,100,0,100
```

Option Configuration

The digitizer has several options that are critical to make it work properly with HTBasic. They are:

- Handshaking Mode
- CTS Handshaking (if hardware handshaking is used)
- Absolute coordinates

If a binary format is specified, make sure that the tablet format and the TABLET driver format match. If the ASCII format is specified, then the data needs to be comma separated and CR/LF terminated. XON/XOFF handshaking may only be used when the tablet is set to ASCII format. Hardware handshaking may be used for either format.

Some other tablet settings that are not critical, but recommended are as follows:

- Data transmitted only in proximity
- Disable Increment mode
- Disable leading zeros
- Enable RUN mode
- Enable Maximum report rate

Please consult your digitizer documentation for the correct switch settings for these options.

HP Mode

```
10  LOAD BIN "SERIAL32"
20  CONTROL 9,5;0
30  CONTROL 9,12;0
40  CONTROL 9,100;0
50  GRAPHICS INPUT IS 9,"TABLET"
60  READ LOCATOR X,Y,S$
70  PRINT X,Y,S$
80  END
```

Summagraphics MM1103 Emulation

```
10  LOAD BIN "SERIAL32"
20  CONTROL 9,5;2
30  CONTROL 9,12;0
40  CONTROL 9,100;0
45  CONTROL 9,4;2
50  GRAPHICS INPUT IS 9,"TABLET"
60  READ LOCATOR X,Y,S$
70  PRINT X,Y,S$
80  END
```

Hitachi HDG-1111B Emulation

```
10  LOAD BIN "SERIAL32"
20  CONTROL 9,5;2
30  CONTROL 9,12;0
40  CONTROL 9,100;0
50  GRAPHICS INPUT IS 9,"TABLET"
60  READ LOCATOR X,Y,S$
70  PRINT X,Y,S$
80  END
```

You can bypass the tablet driver and display the byte sequence returned by the tablet by deleting line 10 and substituting the following program lines for line numbers 50, 60 and 70 in the above program examples.

```
50  ASSIGN @Dev to 9;FORMAT OFF
60  ENTER @Dev USING "#,B";A
70  PRINT IVAL$(A,16)
```

Customizing the Environment

HTBasic provides a keyboard function key macro facility and a powerful CONFIGURE statement that can both be used to customize HTBasic to more nearly match the configuration of an HP 9000 Series 200/300 computer and to alter the way the HTBasic environment works. Each of these features will be discussed on the following pages.

Softkey Macros

The softkey macro facility allows you to define a macro sequence and associate it with a function key. When you press that function key the macro sequence is entered into the input buffer just as if you had typed it in from the keyboard. A softkey definition can be quite complex and it can even call other softkey definitions. You can assign, edit, read, delete, list, store and load softkey macros.

Assigning Softkeys

The HTBasic softkey keyboard macro facility allows great flexibility in defining special purpose function keys. For Example, to define a softkey macro to RENumber and INDENT the current program and then put you into Edit mode, you could define function key 9 as follows:

```
10 L$ = CHR$(255)&"#"   !Clear Line
20 S$ = CHR$(255)&"K"   !Clear Screen
30 E$ = CHR$(255)&"E"   !Execute Input
40 D$ = CHR$(255)&"D"   !EDIT
50 SET KEY 9,"INDENT  RENUMBER"&L$&S$&"REN"&E$"INDENT"&E$&D$&E$
```

You are only limited by your imagination in defining softkey macros.

Editing Softkeys

The EDIT KEY statement allows interactive editing of softkey macros. Enter EDIT KEY *n*, where *n* is the softkey number or press EDIT followed by the softkey function key you wish to edit and then the ENTER key. If the softkey has been defined, the old definition will be displayed. You can modify the old definition by overstriking the existing characters or inserting new characters. Pressing ENTER saves the new softkey definition.

Reading Softkeys

READ KEY returns one or more softkey macro definitions. Specify the key number of the softkey macro to read. If a simple string or array element is specified, then only one key is returned. If a string array is specified, then successive keys, starting with the one specified, are returned into the elements of the string array. For example

```
READ KEY 2,Keytwo$          ! One macro
READ KEY First_key,Keys$(*) ! Several macros
```

Deleting Softkey Macros

Softkey macro definitions are removed from memory with the SCRATCH KEY statement. For example:

```
SCRATCH KEY 3
```

removes the definition for softkey macro number 3. If no key number is specified all softkey macro definitions are removed from memory.

Listing Softkey Macros

The LIST KEY statement lists the current softkey macro definitions on the PRINTER IS device. For example:

```
LIST KEY
```

outputs all the softkey macro definitions. You also may specify an interface select code for the output.

Storing Softkey Macros

The STORE KEY statement saves all the current softkey macro definitions in a new file of type BDAT. HTBasic softkey BDAT files are compatible with HP BASIC softkey BDAT files when HPCOPY is used to move the files. The definition for each defined softkey is written to the file by outputting two items using FORMAT OFF. The first item is an integer, specifying the key number. The second item is a string, giving the key definition. For example:

```
STORE KEY "Helpers"
```

saves the current softkey macro definitions into a new BDAT file named Helpers. If the file already exists, an error is reported. Use RE-STORE KEY to update an existing file.

Loading Softkey Macros

Use the LOAD KEY statement to reenter the softkey macros into the computer from a BDAT file. An HP BASIC softkey BDAT file is compatible with HTBasic and may be directly loaded. Each softkey macro defined in the file replaces any previous definition. The following example reloads the softkey macros defined in the file Helpers into the computer:

```
LOAD KEY "Helpers"
```

CONFIGURE

CONFIGURE allows you to customize HTBasic to more nearly match the configuration of an HP 9000 Series 200/300 computer and to alter the way the HTBasic environment works. Using CONFIGURE will allow many existing HP BASIC programs to run without alteration.

You can use the Options | Run Environment menu from the program window to change the Run Environment. Any changes made to the Run Environment dialog box will appear in all programs when they are opened.

To set the Run Environment, select the Options | Run Environment menu to display an Run Environment dialog box similar to that shown in Figure 9-1. To change an item, click the desired item on the configuration tab.

To return all settings to the default state, click the "Default" button at the bottom of the dialog box.

The following items can be configured using either the CONFIGURE dialog box, or they can be set programmatically:

CONFIGURE BDAT	HP Compatible BDAT files
CONFIGURE CREATE	Controlling File Header Type
CONFIGURE DIM	Controlling Implicit Dimensioning
CONFIGURE LONGCATDATES	Toggles between two and four digit year display in CATalog
CONFIGURE LONGFILENAMES	Enabling use of Long File Names
CONFIGURE MSI	HP MSUS Windows disk substitutions
CONFIGURE SAVE ASCII	Windows Compatible ASCII files
RUN WITH ERRORS IN CODE	Allows or disallows running with sytnax errors

The following CONFIGURE statements can only be configured programmatically. These statements should be placed in your AUTOST file so that they are executed each time you use HTBasic:

CONFIGURE DUMP	DUMP GRAPHICS Printer Type
CONFIGURE KBD	Non-Latin-1 Character Set Keyboard Remapping
CONFIGURE KEY	Redefining Function Keys
CONFIGURE LABEL	Defining New LABEL Characters
CONFIGURE PRT	Changing the value of PRT

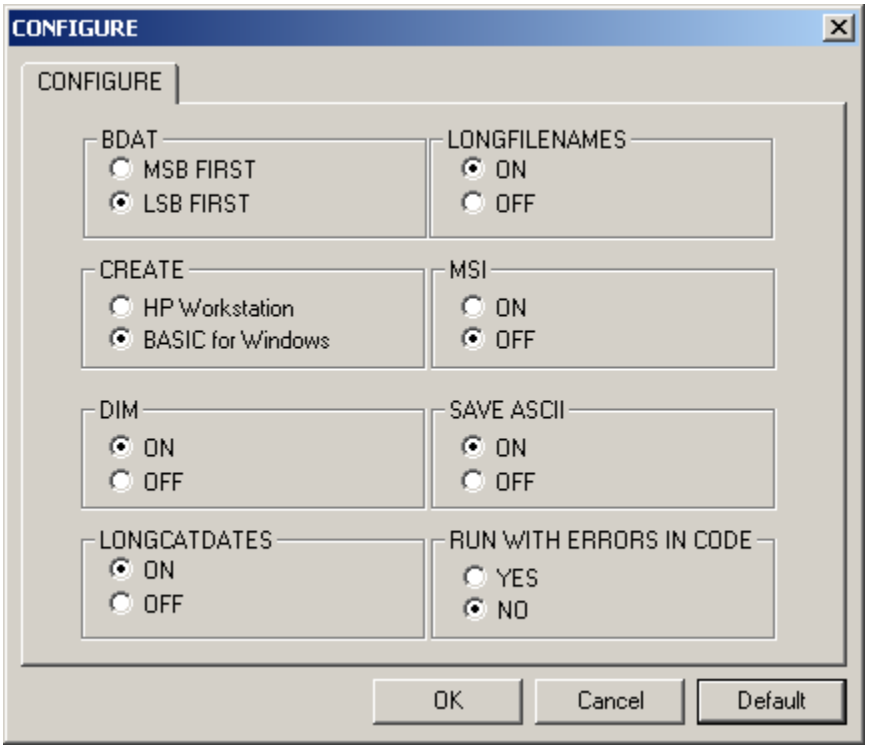


Figure 9-1. Configure Dialog Box

HP Compatible BDAT Files

CONFIGURE BDAT {MSB|LSB} FIRST specifies the byte ordering to use with each BDAT file created after this statement is executed. By default, BDAT files are created with the native byte order of the computer. HTBasic for Windows uses LSB first. This statement was previously used to create BDAT files that could be copied (using HPCOPY) to an HP BASIC workstation. CONFIGURE CREATE is now the recommended method for creating files for interchange with HP BASIC. CONFIGURE BDAT only affects files created with HTBasic headers; files with HP LIF headers are not affected.

Controlling File Header Type

CONFIGURE CREATE specifies the kind of file header ("HP" or "HTB") to use when creating a typed file (LIF ASCII or BDAT). HTBasic can always use files with either header, regardless of the setting of CONFIGURE CREATE. The setting affects file creation only. A CAT listing in SRM format shows the kind of file header of each file in the System Type column.

Use HP LIF headers if you wish to create data files that are simultaneously accessed over a network by HTBasic and HP BASIC. Files with HP LIF headers can also be "binary" copied among DOS or UNIX media for access by the HP Language Coprocessor (Viper card), HP BASIC and HP BASIC/UX. To specify HP LIF headers, use:

```
CONFIGURE CREATE "HP"
```

By default, HTBasic creates HTBasic file headers, since they are two or three times smaller than HP LIF headers. BDAT files with HTB headers can also be created with data in either LSB or MSB byte ordering (see CONFIGURE BDAT). File operations are much faster when the byte ordering of the file matches the byte ordering of the computer. Files with HTB file headers, when copied with HPCOPY, are completely compatible with HP BASIC. Example:

```
CONFIGURE CREATE "HTB"
```

Controlling Implicit Dimensioning

CONFIGURE DIM turns implicit variable and string dimensioning on or off. By default it is on and if a variable is never declared, it is assumed to be of type REAL. If a string is never declared, it is assumed to have a maximum length of 18. If an array is never declared, it is implicitly declared having the number of subscripts found in its first occurrence, with each dimension having the default OPTION BASE lower bound and an upper bound of ten. Example:

```
CONFIGURE DIM ON
```

When CONFIGURE DIM is OFF, then each variable, string and array must be explicitly declared using a REAL, INTEGER, COMPLEX or DIM statement. During prerun, any undeclared variables generate an error message that is written to the message line. Turning off implicit variable and string dimensioning can greatly simplify finding misspelled variable names. If a program has already been prerun, CONFIGURE DIM OFF will not report any undeclared variables until another prerun occurs. To force a prerun to occur, change a program line and press the STEP key.

Controlling Date Display

CONFIGURE LONGCATDATES determines the length of the year displayed in CATalogs. With LONGCATDATES on, four digit years are displayed. With LONGCATDATES off, only two digit years are displayed. When using four digit years, and CATaloging to an array, it is necessary to dimension the array an additional two characters. Typically this means a change from 60 to 62 characters.

Enabling Use of Long File Names

Long filenames are allowed in addition to the standard 8.3 names. The filenames can be up to 256 characters long and can have embedded spaces. However, by default HTBasic removes spaces from file specifiers and CAT listings don't have enough room for long filenames. To enable display and use of long filenames, use the statement

```
CONFIGURE LONGFILENAMES ON
```

With LONGFILENAMES ON, spaces are not deleted from directory and file specifiers since they may be significant. Also, the listing format for CAT is changed to accommodate varying length filenames. It is roughly modeled after the Windows NT DIR command-listing format.

HP MSUS to Windows Disk Substitutions

CONFIGURE MSI defines a table of MSUS substitutions. Whenever an HP MSUS is used as part of a file-specifier, the table is checked for an exact match, including case. It is not sufficient for the MSUS to be equivalent, i.e. ":CS80,700,0" does not match ":CS80,700". If an exact match is found, the Windows path-specifier is used in its place. If no match is found, an error will be reported. The syntax is:

```
CONFIGURE MSI hp-msus TO path-specifier
```

In the DOS path-specifier, directory names must end with a directory separator character "\". To replace an existing entry in the table, specify a new DOS path-specifier. To delete an existing entry in the table, specify a zero length DOS path-specifier. To turn CONFIGURE MSI off or on, use

```
CONFIGURE MSI {ON|OFF}
```

As an example, let's assume that you have lots of programs written that use two HP disk drives on your HP BASIC workstation. They are ":INTERNAL,4", a floppy and ":,700,0", a hard disk. You wish to use Drive A on your PC whenever the programs would have used the internal drive on your workstation. And you wish to use the subdirectory "\HARDDISK" of Drive C on your PC whenever the programs would have used the hard disk on your workstation. This is done by putting the following two lines in your AUTOST file:

```
CONFIGURE MSI ":INTERNAL,4" TO "A:\"
CONFIGURE MSI ":,700,0" TO "C:\HARDDISK\"
```

Windows Compatible ASCII Files

CONFIGURE SAVE ASCII sets the file type SAVE uses when saving a file to disk. CONFIGURE SAVE ASCII ON, the default, produces a LIF compatible ASCII file. This type of file is useful for exchanging programs with older HP workstations that cannot use Ordinary files. See Chapter 10, "Transferring Programs and Data from HP BASIC."

CONFIGURE SAVE ASCII OFF produces Windows ASCII ordinary files. Such files are compatible with all popular program editors, most word processors and recent revisions of HP BASIC. RE-SAVE produces the same file type as an existing file or the file type specified by CONFIGURE SAVE ASCII if no file exists. GET can read either file type, as well as Viper-I ASCII and Viper-II ASCII.

Note: If you use CONFIGURE SAVE ASCII OFF you should not embed carriage-returns or line-feeds in string literals since GET will interpret them as end-of-line indicators.

RUN WITH ERRORS IN CODE

By default syntax errors left in code will be marked as syntax errors, and will cause the program to not run. However, if programs should be allowed run when syntax errors exist, change this option to YES.

DUMP GRAPHICS Printer Type

The CONFIGURE DUMP statement specifies what graphic printer language or image file format the DUMP GRAPHICS statement uses. Chapter 7, "Printer and Pixel Image Device Drivers," explains how to use CONFIGURE DUMP so that DUMP GRAPHICS works with many different printer types and file formats. For example, most HP printers support PCL or the HP Raster Interface Standard. If you wish to use an HP LaserJet for screen dumps, use the following statement to change to the HP printer control language:

```
CONFIGURE DUMP TO "PCL"
```

Non-Latin-1 Character Set Keyboard Remapping

CONFIGURE KBD defines keyboard mappings for character sets other than Latin-1. When in effect, CONFIGURE KBD substitutes characters from the specified string for characters that come from the keyboard. This remapping is good for ASCII characters in the range 0 to 255, but does not apply to function keys. CONFIGURE KBD is not intended to be a complete keyboard driver; it merely substitutes one ASCII value for another. The syntax is:

```
CONFIGURE KBD first-char TO string-name$
```

The range of ASCII values that are remapped starts at the character number specified by *first-char*. The string specifies the ASCII values that should be substituted for values in that range. For example, to remap four keys starting with character number 65 use the following:

```
CONFIGURE KBD 65 TO "DCBA"
```

Redefining Function Keys

The CONFIGURE KEY statement can be used to redefine key assignments. A single key can have more than one definition, if the definitions apply to different shift states or conditions. For example, a key can be defined to have one function when the shift key is pressed, and another definition when the shift is not pressed. A key definition consists of these four parts:

1. The key number,
2. The shift conditions to be examined (masked) when the key is pressed,
3. The value those shift conditions must have,
4. The RMB function to execute when the proper key is pressed with the proper shift conditions.

The CONFIGURE KEY statement has four forms that are used for the different aspects of defining a key. The first form,

```
CONFIGURE KEY key-number TO 256
```

removes all prior definitions a key may have. It is usually a good idea to remove prior definitions of a key to avoid unwanted side effects. *Key-number* specifies what key the action applies to. A utility, keynum, is included with HTBasic to determine the *key-number* for a key.

In the HTBWin program group or program folder, double-click on the KeyNum or KEYNUM.EXE icon. Press the key of interest. The *key-number* for that key is printed in the window. Press Alt-F4 to quit.

A key definition is made using the final three forms of the CONFIGURE KEY statement:

```
CONFIGURE KEY shift-mask TO 257      !Set shift-mask for define
CONFIGURE KEY shift-value TO 258     !Set shift-value for define
CONFIGURE KEY key-number TO function ! Create a definition
```

Shift-mask and *shift-value* specify what shift conditions must be in effect when the key is pressed and what conditions will be ignored. In the table below, find the conditions you wish to have an affect and write down the associated *Shift-mask* and *Shift-value*. Leave out the values for the conditions to be ignored. Then add together all the *shift-mask* values and all the *shift-value* values.

Table for Shift-mask and Shift-value Calculation

Shift Conditions	Shift-mask	Shift-value
Ignore all shift conditions	0	0
Shift key not pressed	1	0
Shift key pressed	1	1
Control key not pressed	2	0
Control key pressed	2	2
Alt key not pressed	8	0
Alt key pressed	8	8
Caps Lock off	64	0
Caps Lock on	64	64
System Menu	1792	0
User 1 Menu	1792	256
User 2 Menu	1792	512
User 3 Menu	1792	768
KBD CMODE OFF	2048	0
KBD CMODE ON	2048	2048

Only one menu may be specified: system, user 1, user 2 or user 3. The Alt key is an alternate shift key and may have different names on different keyboards. It is not a good idea to use the control key, because that prevents its conventional use. (The control key is usually used to insert a function key's two characters into a string literal rather than execute it.)

Function specifies the Keyboard Function to assign to the key. The function is specified using the second character produced when the function key is pressed. Check the "Alphabetical Keyboard Functions List" in Chapter 4 for the second characters used by each HTBasic function. The numeric value of the character is used in the CONFIGURE KEY statement. For example, the table in Chapter 4 lists "V" as the 2nd character for the Previous Line (DOWN) keyboard function. If you wish to define a key to do the DOWN function, substitute NUM("V") for *function*.

A key redefinition is usually done in groups of three CONFIGURE KEY statements. And it is always a good idea to delete any previous definitions made to a key before making any new definitions. The reason is that if a previous definition specified *Shift-mask* and *Shift-value*s that are less restrictive (more inclusive) than the new definition, then the previous definition will be used; the new definition will never be used.

CONFIGURE KEY Example. Suppose you wish to swap the actions of the and ??keys. By default, two functions are assigned to each of these keys:

Keyboard Function	Generic Name	PC Keyboard
Next line	UP	↑
End of output/program	END	Shift-↑
Previous line	DOWN	↓
Begin of output area	BEGIN	Shift-↓

After we redefine the keys the definitions will be

Keyboard Function	Generic Name	PC Keyboard
Next line	UP	↓
End of output/program	END	Shift-↓
Previous line	DOWN	↑
Begin of output area	BEGIN	Shift-↑

Use the keynum utility to find the key numbers for the ↑ and ↓ keys.
Under Windows, these values are usually 38 and 40, respectively.

```
10                                      ! Define Up-Arrow Key
20    CONFIGURE KEY 38 TO 256           ! Delete previous ↑ key defs
30    CONFIGURE KEY 1 TO 257            ! Shift mask: Shift key
40    CONFIGURE KEY 0 TO 258            ! Shift value: Not pressed
50    CONFIGURE KEY 38 TO NUM("V")      ! ↑ key is DOWN
60    CONFIGURE KEY 1 TO 257            ! Shift mask: Shift key
70    CONFIGURE KEY 1 TO 258            ! Shift value: Pressed
80    CONFIGURE KEY 38 TO NUM("T")      ! Shift-↑ is BEGIN
90                                      ! Define Down-Arrow Key
100   CONFIGURE KEY 40 TO 256           ! Delete previous ↓ key defs
110   CONFIGURE KEY 1 TO 257            ! Shift mask: Shift key
120   CONFIGURE KEY 0 TO 258            ! Shift value: Not pressed
130   CONFIGURE KEY 40 TO NUM("^")      ! ↓ key is UP
140   CONFIGURE KEY 1 TO 257            ! Shift mask: Shift key
150   CONFIGURE KEY 1 TO 258            ! Shift value: Pressed
160   CONFIGURE KEY 40 TO NUM("W")      ! Shift-↓ is END
170   END
```

Lines 20 and 100 delete any previous definitions. Then the four definitions are made using groups of three: 30-50, 60-80, 110-130 and 140-160.

If the *Shift-mask* is not specified, the last *Shift-mask* specified will be used. If no *Shift-mask* has ever been specified, a value of 0 is used. The same is true for the *Shift-value*. Thus, lines 60, 110 and 140 in the previous example are redundant. If line 100 is moved to 21, 130 to 51 and 160 to 81, then lines 120 and 150 can also be eliminated.

CONFIGURE KEY Example. CONFIGURE KEY is sometimes useful for eliminating an HTBasic key definition so the key reverts to its default use. For example, HTBasic defines Alt-F4 as Pause Program. If you wish to use it to Quit HTBasic, you must eliminate the HTBasic definition. Assuming F4 is key number 115, the following example eliminates all definitions for F4, including Alt-F4, then redefines all definitions except Alt-F4:

```
10    CONFIGURE KEY 115 TO 256    ! Delete all F4 definitions
30    CONFIGURE KEY 3849 TO 257   ! Mask: KBD CMODE, Menu, Alt, Shift
50    CONFIGURE KEY 0    TO 258   ! Value: KBD CMODE OFF, System
60    CONFIGURE KEY 115  TO 80    ! Def: KBD CMODE OFF, System, F4
70    CONFIGURE KEY 1    TO 258   ! Value: KBD CMODE OFF, System, Shift
80    CONFIGURE KEY 115  TO 33    ! Def: KBD CMODE OFF, System, Shift-F4
90    CONFIGURE KEY 256  TO 258   ! Value: KBD CMODE OFF, User 1
100   CONFIGURE KEY 115  TO 52    ! Def: KBD CMODE OFF, User 1, F4
110   CONFIGURE KEY 512  TO 258   ! Value: KBD CMODE OFF, User 2
120   CONFIGURE KEY 115  TO 99    ! Def: KBD CMODE OFF, User 2, F4
130   CONFIGURE KEY 768  TO 258   ! Value: KBD CMODE OFF, User 3
140   CONFIGURE KEY 115  TO 107   ! Def: KBD CMODE OFF, User 3, F4
160   CONFIGURE KEY 2057 TO 257   ! Mask: KBD CMODE, Alt, Shift
180   CONFIGURE KEY 1    TO 258   ! Value: KBD CMODE OFF, Shift
190   CONFIGURE KEY 115  TO 118   ! Def: KBD CMODE OFF, User n, Shift-F4
200   CONFIGURE KEY 2048 TO 258   ! Value: KBD CMODE ON
210   CONFIGURE KEY 115  TO 51    ! Def: KBD CMODE ON,  F4
220   CONFIGURE KEY 2049 TO 258   ! Value: KBD CMODE ON, Shift
230   CONFIGURE KEY 115  TO 100   ! Def: KBD CMODE ON,  Shift-F4
240   END
```

Defining New LABEL Characters

CONFIGURE LABEL defines additional characters for use with the LABEL statement. Characters in the range 33 to 255 may be defined. You may define one character by giving the character number and a simple string or several characters by giving the starting character number and a string array. To delete a definition, use a zero length string for the definition. Each character in the definition strings has the form CHR$(Move + x*16+ y), where Move is 0 or 128, x ranges from 0 (far left) to 7 and y ranges from 0 (bottom) to 15. The baseline is y=5.

The following example defines the character "H":

```
A$=CHR$(133)&CHR$(14)&CHR$(238)&CHR$(101)&CHR$(138)&CHR$(106)
CONFIGURE LABEL 72 TO A$
```

More information is presented in Chapter 12, "International Language Support" in the User's Guide.

Changing the Value of PRT

On most HP BASIC workstations a printer is hooked to the IEEE-488 interface and is set to primary address 1. Thus, the device selector is 701. This convention is followed so closely that a special function, PRT, was created with the value 701. Many programs use this fact to switch output to the printer with the statement.

```
PRINTER IS PRT
```

Under Windows, printers are accessed through the printer driver, which is accessed in HTBasic through device selector 10. Many existing programs use PRT as the device selector for the printer. So that these programs continue to work, PRT has been changed to 10. If you are using an IEEE-488 printer with ISC 1, you may wish that PRT had the value 701 as it did under HP BASIC. To allow you to change PRT, the CONFIGURE PRT statement has been added. CONFIGURE PRT need only be used if you are porting existing programs that reference PRT and you do not wish to use the Windows printer driver.

CONFIGURE PRT specifies the device-selector that the PRT function returns. If you are using an IEEE-488 bus printer at address one, you will probably want to change PRT back to 701 to match the HP BASIC definition of PRT:

```
CONFIGURE PRT TO 701
```

Transferring Programs and Data from HP BASIC

This chapter presents transferring programs and data files between HP BASIC and HTBasic and details the differences between HP BASIC and HTBasic. It discusses HTBasic file types, remote access across networks or serial link, files created by other computers, hardware and keyword differences, and the LIF diskette file transfer utilities.

If you are transferring programs from HP 9000 Series 200/300 computers to HTBasic, you should read this entire chapter. Differences in hardware, file system and keywords are discussed.

Compatibility

HTBasic is compatible with Hewlett-Packard's 9000 Series 200/300 "Rocky Mountain" BASIC, and runs on a variety of hardware platforms - without the need for a coprocessor board. It includes most of the language features defined in the HP 6.2 LOADable BIN files.

HTBasic is not compatible with Series 80 or 9835/9845 BASIC. But there are translator programs available from third parties to translate from other BASICs to Rocky Mountain BASIC and to transfer files among different systems.
To transfer BASIC programs between HTBasic and RMB use SAVE/GET. Prog files cannot be directly shared between HTBasic & RMB.

Operations which cause an error may not produce exactly the same error number as RMB. For example, if an out-of-range value is passed to an INTEGER parameter, one platform might report error 19 (value out of range) and the other platform might report error 20 (integer overflow).

The destination of an ON...CALL statement can be deleted in HTBasic. In RMB, attempts to delete such SUBs cause an error.

When doing a CAT TO S$(*), the first array element has the text "DIRECTORY:" before the actual directory name.

I/O Differences

Some error handling and EOF handling in the ENTER statement is different in HTBasic and HP BASIC/WS.
The GPIB interface drivers in HTBasic do not use DMA mode.

Mass Storage Differences

Filename wildcards are only available through the CAT statement.

INITIALIZE is not supported, and therefore, HTBasic cannot create RAM volumes. For RAM volumes, use a DOS RAM disk program instead (e.g., VDISK.SYS, RAMDISK.SYS).

There can be differences in file formats and byte ordering within files. See the ASSIGN statement in the online "Reference Manual" for more information.

MSUS Format

RMB format: [directory path][filename][:msus]
HTBasic format: [drive:][directory path][filename]

You can map the RMB format to the HTBasic model using the CONFIGURE MSI statement.

RMB CSUB

The following RMB CSUB utilities are not provided:

PHYREC: no substitute.

GDUMPC: automatically provided by DUMP GRAPHICS if current Windows printer supports color.

BPLOT: Bload() and Bstore() functionality now provided through extensions to the GLOAD and GSTORE statements.

File Types

HTBasic can work with both typed files (such as BDAT, ASCII and PROG) and files without a type or "ordinary" files. Both kinds of files are discussed in the following paragraphs.

Typed Files

Most HTBasic or HP BASIC files have a file type, such as BDAT, ASCII or PROG. The typed file information is stored in a file header at the beginning of the file. Two types of headers are in common use: HP LIF and HTBasic.

HTBasic can identify and use HP BASIC typed files with HP LIF headers. Data can be in BDAT or ASCII files. HP BASIC PROG files must be saved in ASCII or HP-UX format; use the SAVE statement, not the STORE statement to save programs for exchange. HP LIF file headers are either 512 or 768 bytes in length.

HTBasic file headers are 256 bytes in length.

The LIF diskette utilities, described later in this chapter, supports typed files with either type of file header.

Ordinary Files

Recent releases of HP BASIC support files without a type. HP BASIC calls these files "HP-UX" files when present on a LIF or UNIX volume and "DOS" files when present on a DOS volume. HTBasic calls these files "ordinary" files. An ordinary file is created with the CREATE statement (as opposed to CREATE ASCII or CREATE BDAT). In a CATalog of a LIF diskette, an ordinary file is listed as "HP-UX". When an ordinary file is copied to a Windows disk, it remains an ordinary file, but is listed without a type by the HTBasic CAT statement. Files created on the PC by other programs are ordinary files.

Remote Access

HTBasic can access files on other computers over a network by accessing a remote mounted volume or by using file transfer utilities across the network or across a serial link.

Mounted Network Volumes

HTBasic can access files on remote computers directly if the disks containing the files on the remote computer(s) are mounted as network volumes on the computer running HTBasic. To access the remote files, simply include the local name of the remote directory when specifying file names.

Network File Transfer

If there are files on a computer connected to your network but not mounted in your computer's file system, if that computer has an ftp server and if you have TCP/IP utilities for your PC, you can use the *ftp* and *rcp* file transfer utilities to copy the files to your computer's disk and then use them. Note that typed files are considered to be binary files by ftp, so you must use the binary command with ftp when copying these file types.

Other File Transfer Utilities

Two computers connected through a network or with a serial link can transfer files using a file transfer utility, for example, the Kermit program. As with other types of connections, BDAT and LIF ASCII files are considered to be binary files, so you must use the binary mode on both the sending and receiving sides of the file transfer when copying these file types. In Kermit, this is done with the "set file type binary" command.

In some instances, data can be transferred between two computers by hooking them together with a serial or IEEE-488 interface and writing a small program on each computer to transfer programs and data between the two. Transfer programs as if they were ASCII data.

Converting Old HTBasic Program Files

To convert an old PROG format program run the old version of HTBasic and execute the following statements:

```
LOAD "prog_name"
SAVE "temp_name"
QUIT
```

Then run the new release of HTBasic and execute the following statements:

```
GET "temp_name"
       Now press the STEP key (Alt-F1)
RE-STORE "prog_name"
PURGE "temp_name"
```

If the file is not in the current directory, remember to include the full pathname.

Files Created by Other Computers

HTBasic can use LIF ASCII, BDAT and ordinary files created by other computers. PROG files created by other computers generally cannot.

LIF ASCII Files

LIF ASCII files created by HTBasic or by HP BASIC can be directly read and written by HTBasic. If the files are transferred using a file transfer utility, binary mode should be used.

BDAT Files

BDAT files created by HTBasic or by HP BASIC can be read and written by HTBasic. This is true even if the two computers use different byte ordering. If the files are transferred using a file transfer utility, binary mode should be used.

PROG Files

PROG files created by different versions of HTBasic can be exchanged only if the two computers use the same byte ordering. PROG files created by HP BASIC cannot be read or written by HTBasic. LOADing an incompatible PROG file results in error 58, "Improper File Type." ASCII and ordinary format programs can be used to exchange programs between versions with incompatible PROG files. Use GET to load an ASCII or ordinary format program file.

HTBasic PROG files with the wrong byte order are listed by the CAT statement with a file type of "PROGL" or "PROGM", where the trailing L or M indicates the file has LSB or MSB byte ordering. PROG files created by HP BASIC are listed with the file type "HPPRG". Compatible PROG files are listed with a file type of "PROG".

Ordinary Files

Ordinary files (DOS, UNIX or HP-UX ASCII files) can be both read and written by HTBasic. If ordinary files written with FORMAT OFF are transferred using a file transfer utility, binary mode should be used. If ordinary files written with FORMAT ON are transferred using a file transfer utility, use its ASCII mode ("ascii" in *ftp* and "set file type ascii" in Kermit).

HTBasic is able to work with ordinary, FORMAT ON files regardless of whether the file was written with a line termination of CR/LF (carriage-return/line-feed) or just LF. Other programs, such as text editors, may not be as flexible. CR/LF is considered the standard for DOS and Windows. LF is considered the standard for UNIX.

Hardware Differences

HTBasic compensates automatically for many of the hardware differences that exist between an HP 9000 Series 200/300 workstation and other hardware platforms. The following sections outline many of the differences and explain what limitations still apply.

Disk Drives

HP-IB disk drives are not supported. No mass storage device is supported across the HP-IB. HP LIF disks are not compatible with DOS, NT and UNIX disk formats and cannot be read or written directly by HTBasic. On the PC, HPCOPY is used to transfer data and programs back and forth.

Softkey Layout

Different keyboard layouts are used by PCs and HP Workstations. Some keyboards have the softkeys down the left side of the keyboard, and some have them across the top of the keyboard. To make up for these differences and the lack of spatial coherence between the physical keys and the softkey menu, HTBasic menu labels have been numbered. This provides numerical coherence. The label numbered "1" always corresponds to the softkey with the "1" printed on it. *The number is not meant to be the softkey number.*

Note: to provide compatibility with programs written for the 9836 Nimitz keyboard (which starts with softkey K0), the command KBD CMODE can be used to turn on or off compatibility mode.

A New Backspace

Under HP BASIC, the Backspace key was assigned to CHR$(255)&"B". This function does the same thing that the Left Arrow key does. HP BASIC is unique in this treatment of the Backspace key. For compatibility with the remainder of the computing community, HTBasic assigns a new function to the Backspace key. Pushing Backspace deletes the character to the left of the cursor. This function is named DEL LEFT and is equivalent to the function LEFT followed by DEL CHR. If you do not like this definition, you may redefine the key using the CONFIGURE KEY statement.

Series 300 Bit-Mapped (CRTB) Displays

If you are using an HP 9816, 9836, or other HP computer with separate alpha and graphics hardware, some of the differences you find in HTBasic will be the same differences you would find moving to an HP 310 or another HP computer with a fully bit-mapped alpha/graphics display. Bit-mapped displays are driven by the CRTB Mode Driver. A CRTB display has only bit-mapped images. The ALPHA image is written into one or more of the graphic planes. One plane can be separated from the others for use solely for the ALPHA image. Or, all the planes can be merged for shared use between ALPHA and GRAPHICS. See SEPARATE and MERGE ALPHA in the Reference Manual.

When ALPHA and GRAPHICS are merged, ALPHA text is converted to graphic bits and written into the graphic planes, overwriting any graphics data that might be present. Therefore, ALPHA and GRAPHICS cannot be dumped separately. And when the ALPHA text is scrolled, any graphic data present will be scrolled also. This is the default mode for a CRTB display.

The SEPARATE ALPHA statement can be used to simulate 9836C displays with independent ALPHA and GRAPHICS screens. Either image can be turned on or off or DUMPed independently of the other.

9836C (CRTA) Displays

The HP 9836C display is driven by CRTA Mode drivers. A CRTA display has distinct ALPHA and GRAPHICS images. Either the ALPHA or GRAPHICS images can be displayed, or both can be displayed at once, overlapping each other. The hardware for the two images is independent. HTBasic does not directly emulate CRTA Mode. HTBasic does not support CRTA mode. Use the SEPARATE ALPHA command for the best 9836C emulation.

Other Video Circuitry Differences

A list of minor differences in video circuitry that you should be aware of follows:

Underline is not supported by PC color display adapters in text modes.
Blinking is not supported by PC color display adapters in graphics modes.

Processor

The instruction sets for the Intel x86 family of processors, Motorola 680x0 processors and HP Precision Architecture processors are all different. Since a CSUB contains processor code, a CSUB cannot be moved from HP BASIC or HTBasic on one processor and executed under another. The same holds true for assembly routines called with WRITEIO 9827.

Clock

Timing resolution varies from version to version of HTBasic. Millisecond timing resolution is not available on the IBM PC. HTBasic is dependant upon the Operating System's timing resolution, all timing functions within HTBasic have a best resolution of about 55 milliseconds while running under Windows 95/98/ME and 10 milliseconds wile running under Windows NT & 2000.

HP-IB/IEEE-488

The HP-IB bus is more commonly known by the name IEEE-488 or GPIB bus. Most computers are not sold with an integral IEEE-488 interface. An HP compatible IEEE-488 interface card for the PC can be purchased from TransEra, either bundled with HTBasic or as a later upgrade. Various third-party IEEE-488 interfaces are also supported.

The TransEra Model 900 IEEE-488 card gives full IEEE-488 compatibility. Other interfaces usually lack minor functionality. For example, PC IEEE-488 interface cards that use the NEC 7210 IEEE-488 controller chip are incompatible in the following areas: All READIO and WRITEIO registers are different. Bit 0, IFC, is not supported in STATUS register 5 or the ENABLE INTR mask. The REM and LOC bits of STATUS register 6 are not supported as well as the DAV, NDAC, NRFD, and IFC bits of STATUS register 7.

Character Set

The default character set used by the IBM PC is different than the HP Roman-8 character set used by HP BASIC. The differences exist in characters CHR$(128) and above. In HTBasic, all characters from CHR$(128) through CHR$(254) are allowed in variable names. If you will be running a program with HP BASIC and HTBasic, you should restrict variable names to characters from CHR$(161) to CHR$(254) and the legal characters less than CHR$(128). Chapter 12 of the User's Guide, "International Language Support," contains information on converting from the HP character set, or setting up the PC to use the HP character set.

Shared Resource Manager (SRM)

Shared Resource Manager users should consider upgrading to Hewlett-Packard's SRM/UX. By replacing SRM servers with SRM/UX servers, Windows can access the files on the server using NFS. HTBasic is also compatible with most industry standard networks, such as Microsoft NT, NetWare and LAN Manager.

Miscellaneous Differences

On the PC, support for the Centronix/Parallel Printer Interface and the Windows Print Manager has been added. The interface select code for LPT1 is 26, for the Print Manager it is 10. These can be changed if needed (see Chapter 10 of the User's Guide, "Other I/O Destinations" and Chapters 5-9 in this manual).

The serial interface has been enhanced with XON/XOFF flow control. It is enabled by default. If you are porting existing programs that transfer binary data or ^S and ^Q characters as part of the data, you should turn off XON/XOFF flow control and turn on hardware handshaking with the statements:

```
CONTROL 9,100;0
CONTROL 9,5;0
CONTROL 9,12;0
```

The maximum line number in HTBasic has been increased from the HP limit of 32,766 up to 4,194,304. If you plan on also running a program on an HP computer, you should not use line numbers over 32,766.

HP ASCII File Problem

The Hewlett-Packard BASIC Language Reference manual entry for "ENTER" states that "data should be entered into variables of the same type as those used to output it." This general rule applies to HTBasic as well, and if violated can produce unexpected results. Consider the following program:

```
10   CREATE "TEMP",1
20   ASSIGN @I TO "TEMP";FORMAT ON
30   OUTPUT @I;"1,2"
40   OUTPUT @I;"3,4"
50   RESET @I
60   ENTER @I;A,B
70   PRINT A,B
80   ASSIGN @I TO *
90   PURGE "TEMP"
100  END
```

This program violates the matching-type rule by outputting strings and then entering numerics. Intuitively, you expect line 70 to print 1 and 2, which it does. When line 10 is changed to "CREATE BDAT", the same result is produced. But if line 10 is changed to "CREATE ASCII", then HP BASIC prints 1 and 3. Whether or not you consider this a bug in HP BASIC, it is a discrepancy that has been corrected in HTBasic. If you have written programs that use ASCII files and violate the matching-type rule, you should correct them before running them with HTBasic. If your program must run with both HP BASIC and HTBasic, you must either adhere to the matching-type rule, or use another file type.

File Systems

HTBasic uses the native file system of whatever operating system it is running under. HTBasic does not support the LIF file system.

A primary difference between HP BASIC and HTBasic is that HP BASIC/WS does its own file I/O, while HTBasic has the operating system do all file I/O. This has advantages. *Any* disk, diskette, network, or device accessible through the file system is accessible from HTBasic.

A couple of warnings are in order about the way most operating systems work. If a file is currently ASSIGNed, the operating system buffers some data in memory to make I/O faster. This buffering can produce unexpected results if the same file has multiple I/O paths ASSIGNed to it concurrently. Also, you should not remove a diskette, or turn the power off while a file is ASSIGNed.

HTBasic maintains compatibility with LIF file types by keeping a special file header for typed files (BDAT, LIF ASCII, and PROG files). The extra information that must be stored for these file types is kept in the header. The header is kept totally hidden from BASIC programs. However, to programs outside HTBasic, the header will be accessible as the first bytes of the file. Any files without the special header are known as "ordinary files". In a CAT listing, the file type is blank for ordinary files.

In addition to HTBasic file headers, HTBasic can recognize and use HP LIF headers. This allows networked computers to directly interchange data between Series 200/300 systems running HP BASIC and systems running HTBasic. The new CONFIGURE CREATE {"HP" | "HTB"} statement allows the program to specify the type of file header to use when creating a new BDAT or LIF ASCII file.

LIF ASCII vs. Ordinary ASCII

Early versions of HP BASIC did not have a file type that matched DOS ASCII or UNIX ASCII files. Starting with BASIC 5.0, it does have an ordinary, "vanilla" file type that can hold DOS ASCII or UNIX ASCII data. With HP BASIC/WS or HP BASIC/UX, ordinary files are called "HP-UX" files. With the Viper Card, ordinary files are called "DOS" files. No special header or other embedded information is placed in ordinary files. An ordinary file written with FORMAT ON is a DOS ASCII file. An ordinary file written with FORMAT ON, EOL CHR$(10) is a UNIX ASCII file. Use the CREATE statement to create an ordinary file. In a CAT listing, an ordinary file is listed with the file type blank.

PC FAT or NTFS File Systems

The following sections describe some of the differences between the FAT or NTFS file systems and the HP 9000 Series 200/300 BASIC Logical Interchange Format (LIF) file system. The following discussion is not meant to be a substitute for your PC manual. You should read it for complete information on the topics presented here.

The FAT/NTFS file systems, like the HP-UX or Shared Resource Manager (SRM) file system, is a Hierarchical File System (HFS). An understanding of HFS, UNIX, or SRM file systems may aid you in understanding the PC file systems.

File Specifier

On the PC, a file specifier has the form:

[d:] [directory path] filename[.ext]

where: d: is the drive letter. Usually this is A: and B: for the two diskette drives, and C: for the hard disk. Higher drive letters, D:, E:, etc., usually refer to multiple partitions and RAM (MEMORY) disks, CD-ROMs, or network drives. This part of the file specifier corresponds to the MSUS of the HP file system, but is included on the front, *not the end of the file specifier.* It is optional, and if omitted, the current (or default) drive is used.

directory path is optional. It is explained below.

filename is the main part of the file specifier. The following characters are not allowed to be in a filename: ."/\[]:|<>+=;, and control characters whose ASCII value is less than the space character. All other characters are legal. If lowercase letters are used in a filename they are converted to uppercase. Thus, you cannot have a file named "AbC" and another file named "aBC". HTBasic will consider both names to be "ABC". Be aware that a PC file specifier is often referred to simply as a filename.

ext is the filename extension. The same characters that may be used in a filename may also be used in an extension. Certain conventions are used for filename extensions. Most of them are arbitrary, with a few exceptions noted below.

Some conventional extensions are:

Extension	Conventional Use
.PRG	BASIC Prog-type Program
.BAS	BASIC program
.BDT	BDAT file
.ASC	ASCII file
.DAT	BDAT, or DOS data files
.TXT	text files
.LIS	text files, or compiler listings
.LST	another common listing file extension
.DOC	A DOS ASCII file containing documentation
.BAT	DOS batch files MUST have this extension
.COM	An executable command or program
.EXE	An executable command or program. Commands and programs MUST have one of these two extensions

Note that compared to an HP file specifier, the MSUS is replaced by a drive letter and moved from the back to the front. Also, an extension has been added, and no password is used or available. While PC does not have passwords, it does have some access capabilities (similar to SRM access capabilities). These are discussed below. If passwords are present, they will be ignored.

If an HP-style MSUS is present, it can be translated to a PC path if specified by the CONFIGURE MSI statement. If no matching translation can be found, an error is returned. The CONFIGURE MSI statement has been added to HTBasic to allow HP style file specifiers to be used with a PC file system. If PC style file specifiers are used, then the CONFIGURE MSI statement is not needed.

Directory Path

The PC file system is "tree" structured, almost exactly like the HFS and SRM file systems. If you are familiar with the HFS file system, the following differences may be instructional. The "/" character of the HFS file system is replaced with "\" in the PC file system. Passwords do not exist in the PC file system and will cause an error if included in a directory name.
The file system is organized as a tree. Actually, it is usually thought of as an upside-down tree. The top of the tree is, thus, the root. (Not roots. Directory trees are considered to have only one root, and the term trunk is usually not used.) The tree is composed of directories and files. Each directory may contain files and additional directories, which act like branches down the tree. Directory names follow the same rules as filenames with extensions.

A directory path is the path you climb through in the tree to get from the root of the tree to a certain directory. It consists of the names of each directory that must be climbed through in order to get to that certain directory, separating the directory names with the backslash "\" character. If you have not already, you will find it helpful to read the material in your PC manual concerning directories.

PC File Types and Access Capabilities

The PC file systems store certain information *about* a file in addition to storing information *in* the file. This information consists of:

- file name
- number of bytes in the file
- modification date and time
- access capabilities: hidden, read-only, system
- location on the disk of the file contents

The PC does not store a record length or a file type. It does have access capabilities, although they are different from the SRM access capabilities. The access capabilities are called "attributes" by the PC and may be changed by the PC ATTRIB command. They may also be changed by the HTBasic PROTECT command. The meanings of the attributes are:

Attribute	Meaning
hidden	the filename is not shown in a disk catalog, although the file is there
read-only	the file may be read, but not written or deleted
system	the file is a system file

Devices

HP BASIC accesses devices through interface and device select codes. The same is true of HTBasic. However, *Windows also allows access to devices as if they were files.* Windows gives special names to devices, and when used in the place of a filename, access a device instead of a file. These names are called DOS Device Names and are typically: CON, AUX, COM1, COM2, PRN, LPT1, LPT2, and NUL. Often, data acquisition hardware for the PC comes with a device driver. To access such a data acquisition board, treat it as you would a file and use ASSIGN, OUTPUT, and ENTER.

Wildcards

The question mark "?" and the asterisk "*" have special meaning to Windows. They are called wildcards and are used in commands like DIR and the HTBasic CAT command in order to select more than one file. A filename with wildcard characters in it will be compared with existing filenames using special rules and all filenames that "match" will be selected.

These are the rules used to match an actual filename with wildcards:

The "?" character will match any one character in the same position of an actual filename. For example, the string "?AT" will match the strings "CAT", "BAT", "MAT", or any other string three letters long that has an "A" as the second letter, and "T" as the third letter. The "*" character will match zero or more characters starting at that position. For example, "*" will match all filenames. "*.BAS" will match all filenames that have the ".BAS" extension.

Keyword Differences

The following sections present, by keyword, various implementation differences between HTBasic and HP BASIC. Most differences are the result of hardware differences, file system differences, or extensions in HTBasic. The Reference Manual can be consulted for the full explanation of keywords mentioned here. In some cases, Windows commands that relate to BASIC commands are given. This may help you learn Windows faster by associating functionality with commands you already know.

Some differences exist because of enhancements TransEra has made. Many enhancements can be included in programs that run under both HP BASIC and HTBasic, if you are careful. One approach is to hide statements that will not parse under HP BASIC in OUTPUT KBD statements:

```
OUTPUT KBD;"CONFIGURE PRT TO 701"&CHR$(255)&"E";
```

Another approach is to set up a section of code that is executed only by HTBasic. When you GET the program under HP BASIC, the lines with HTBasic syntax enhancements will be commented out and ignored by HP BASIC:

```
10  INTEGER Htbasic
20  Htbasic=SYSTEM$("VERSION:HTB")<>"0"
30  IF Htbasic THEN
40    CONFIGURE DUMP TO "PCL"
50    CONFIGURE PRT TO 701
60  END IF
70  END
```

When porting programs between HP BASIC and HTBasic, note the differences for the following keywords.

ALLOCATE

Under HTBasic, GOSUB and ALLOCATE use the same stack. Intermixing these statements can cause changes in available memory that are different from HP BASIC.

ASSIGN

When an ASSIGN fails, the previous state of the I/O path is not preserved. Also, the CONVERT and PARITY options are not implemented. With HTBasic, if changes are made to an ASSIGNed file, the directory entry is not updated until the file is closed. Windows buffers read and write to disk. You should not remove a diskette or turn the power off while a file is ASSIGNed. Exchanging diskettes while a file is ASSIGNed on the first diskette can destroy the next diskette. Two I/O paths ASSIGNed simultaneously to the same file can produce slightly different results than HP BASIC, depending on the buffering Windows does.

The HTBasic ASSIGN includes two new options, FORMAT LSB FIRST and FORMAT MSB FIRST, to specify byte ordering of binary numeric data transfers. This provides the ability to do binary transfers with any device or computer, regardless of the byte ordering that device uses.

ATN2

ATN2 is a new HTBasic function.

BEEP

Sound generation capabilities vary from version to version. On computers that do not provide control for variable frequency sound generation, BEEP generates a beep or bell sound. The range of the duration and frequency are subject to the limits of the computer hardware. Contrast the following capabilities with HP BASIC. HP BASIC rounds the frequency value to a multiple of 81.38 Hz and supports a range of 81 Hz to 5.208 KHz.

On the IBM PC, the period (not the frequency) is rounded to a multiple of 0.838 microseconds. The range of frequencies is 40.7 Hz to 32.767 KHz.

BINEQV/BINIMP

These are new HTBasic functions.

BLOAD/BSTORE

The functionality of the HP BLOAD, BSTORE compiled subroutines are an integral part of the HTBasic language and do not require CSUBs. See the GLOAD/GSTORE section in this chapter for more information.

BUFFER

In HTBasic, it is usually incorrect to access numeric data in a buffer through the array name. ENTER and OUTPUT should be used instead.

CAT

The format of CAT output varies according to operating system. This behavior is compatible with HP BASIC. The CAT statement supports the use of wildcards. Again, wildcard interpretation varies by operating system. See CAT and WILDCARDS in the Reference Manual for more information.

Under DOS, HTBasic does not allow wildcards to be turned off as HP BASIC allows. Under UNIX, WILDCARDS are ON by default and the escape character is "\". HP BASIC uses wildcards as a primary filter and the SELECT option as a secondary filter in choosing filenames to display. HTBasic is designed to be used with one or the other. Use wildcards or the SELECT option, but not both.

CDIAL

CDIAL is not supported.

CHECKREAD

This command is equivalent to the DOS VERIFY command and is not supported by HTBasic.

CHGRP and CHOWN

CHGRP and CHOWN are useful with an operating system like UNIX in which files are owned by individuals and groups. These commands allow a user with the appropriate privilege to change or assign ownership of files. These commands are not used for Windows, and are not supported. The HTBasic editor will allow these statements to be entered and the syntax checker will check them for correctness.

CINT

CINT is a new HTBasic function.

COMMAND$

COMMAND$ is a new HTBasic function.

CONFIGURE

The CONFIGURE statement is an enhancement to HTBasic that allows the environment to be customized to a user's preference, or to match HP hardware. The CONFIGURE statement is explained in the Installing and Using Manual.

CONVERT

This is an unsupported ASSIGN option.

CONTROL/STATUS

Depending on the hardware interface, some CONTROL and STATUS registers may be different. The PC serial hardware and TransEra's GPIB-900 board have registers that are compatible with the HP registers. For other interfaces, consult the interface register documentation.

TransEra has added capabilities to several of the standard interfaces. The additional registers resulting from these enhancements are always numbered 100 and above. In some instances HTBasic can pass arrays to and from a single register. This capability is used for things like gain control lists in data acquisition drivers.

COPY

HTBasic does not support the copy of a full disk to another disk. Use the operating system for full disk copies.

CREATE

Because Windows supports extendible files, the number of records specified in the CREATE statement is ignored. An invalid number does not generate an error, as it will under HP BASIC. Programs that depend on errors occurring by writing past the last specified record will not function correctly, as HTBasic will simply extend the file as needed. Programs that depend on the pre-allocation of the requested records should write dummy data at the time the file is CREATEd. Under HTBasic, it is sufficient to write data in the last record.

Don't confuse a LIF ASCII file, created with CREATE ASCII, with a DOS ASCII or UNIX ASCII file, created with CREATE. See CREATE in the Reference Manual for more information.

Use the CONFIGURE BDAT MSB FIRST statement before creating BDAT files that will be moved back to HP BASIC.

CREATE DIR

This command is exactly like the HP BASIC command of the same name. It is the equivalent of the DOS MD or MKDIR commands, or the Windows new folder function.

DEF FN

Nested I/O is not allowed under HP BASIC.

Nested I/O does not return an error under HTBasic but should not be used because future improvements may make it illegal.

HTBasic limits the depth that recursion can occur. The depth is limited by the size of the processor stack, not the BASIC workspace size.

DELSUB

HTBasic allows a string variable to specify the name of the subprogram or function to delete.

DUMP

HP BASIC supports only Hewlett-Packard printers, but HTBasic supports many types of printers. For this reason, you may need to tell HTBasic what language to use before doing the DUMP. On a PC, the default language is "WIN-DUMP", which supports both IBM and Epson graphic printers. If you are going to make screen dumps to another type of printer, you must first use the CONFIGURE DUMP statement. You may find it convenient to include this statement in your AUTOST file. Chapter 7, "Printer and Pixel Image Device Drivers," of the Installing and Using manual explains what languages are supported and how to select them.

EDIT

EDIT SUB and EDIT FN are extensions in HTBasic. Several new edit functions are also included. See "OUTPUT KBD" later in this chapter.

ENABLE INTR

Depending on the hardware interface, some ENABLE INTR mask values may be different. On the PC, interfaces supported by the "SERIAL" and TransEra "GPIB" board drivers have interrupt masks that are compatible with the HP masks. For other interfaces, consult the interface register documentation.

ENVIRON$

ENVIRON$ is a new HTBasic function.

ERRDS

ERRDS is not supported.

ERRM$

HTBasic error messages are usually similar to those in HP BASIC. Programs that depend on ERRM$ returning the exact same message as HP BASIC should be modified accordingly. In particular, where an HP BASIC error message has seemed less descriptive than it should be, HTBasic returns a more descriptive message.

ERRN

Any error number of 2000 or greater is an HTBasic extension to HP BASIC. Not all errors that can occur under HP BASIC can occur under HTBasic. The Reference Manual contains a list of errors that can occur.

In general, the error numbers returned for errors are the same as those returned by HP BASIC. But in some instances the operating system or environment in which HTBasic runs makes it impossible or impractical to return the same number.

EXECUTE

The EXECUTE statement has been added to run operating system commands or other programs while HTBasic is running.

FBYTE

FBYTE is not supported.

FIX

FIX is a new HTBasic function.

FRACT

HTBasic allows the FRACT of a complex value, returning the fractional part of the real part of the complex value. HP BASIC gives error 620.

FRE

FRE is a new HTBasic function.

GESCAPE

Only HP BASIC operation selectors 1 to 6 are supported by GESCAPE. Often, where operation selector 7 is used, MERGE or SEPARATE ALPHA can be used instead. Operation selectors greater than 99 are enhancements to HTBasic.

GET

HTBasic turns lines with syntax errors into comments by inserting "!*" instead of just "!" after the line number. This allows FIND "!*" to quickly identify lines needing corrections.

GLOAD/GSTORE

The HP rule that images be GLOADed on the same display and with the same write-enable mask that was used when the image was GSTOREd applies to HTBasic as well. In particular, don't think that you can GSTORE on an HP BASIC display and GLOAD on an HTBasic display.

HTBasic GLOAD and GSTORE have been enhanced with the capabilities of HP BASIC's BLOAD and BSTORE subprograms. These capabilities allow rectangular blocks of the screen to be stored or loaded. The following subprograms can be used in place of the BLOAD and BSTORE subprograms for users that want to continue calling BLOAD/BSTORE instead of switching to the new GLOAD/GSTORE syntax:

```
10    SUB Bstore(INTEGER Array(*),W,H,OPTIONAL Rule,REAL X,Y)
20      SELECT NPAR
30      CASE 3
40        GSTORE CRT,Array(*),W,H
50      CASE 4
60        GSTORE CRT,Array(*),W,H,Rule
70      CASE 5
80        WHERE X0,Y0
90        GSTORE CRT,Array(*),W,H,Rule,X,Y0
100     CASE 6
110       GSTORE CRT,Array(*),W,H,Rule,X,Y
120     END SELECT
130   SUBEND
140   SUB Bload(INTEGER Array(*),W,H,OPTIONAL Rule,REAL X,Y)
150     SELECT NPAR
160     CASE 3
170       GLOAD CRT,Array(*),W,H
180     CASE 4
190       GLOAD CRT,Array(*),W,H,Rule
200     CASE 5
210       WHERE X0,Y0
220       GLOAD CRT,Array(*),W,H,Rule,X,Y0
230     CASE 6
240       GLOAD CRT,Array(*),W,H,Rule,X,Y
250     END SELECT
260   SUBEND
```

Note that only rule 3, replace, is currently supported.

GRAPHICS INPUT IS

Both HP BASIC and HTBasic do an implicit GRAPHICS INPUT IS assignment for you if you attempt to use graphic input statements before an explicit GRAPHICS INPUT IS. The difference is that HTBasic does the implicit GRAPHICS INPUT IS as soon as HTBasic is started, and HP BASIC waits until the first graphic input statement is executed. The only known effect of the different approaches is that under HP BASIC, a SYSTEM$("GRAPHICS INPUT IS") returns "0" until the first graphic statement is executed and HTBasic returns the correct value anytime.

HELP

The HELP statement is a new HTBasic statement. It provides on-line help for all language statements. The Reference Manual has been compressed and is stored in a file. Just enter HELP followed by the keyword of interest and HTBasic brings up the Reference Manual page for the requested keyword. You no longer have to run and get the Reference Manual when you have a question on statement syntax or functionality.

HIL

HIL related statements are not supported.

IMAGE

Entering data from a string using

```
ENTER L$ USING "Y"
```

will always use the internal byte ordering of the computer. For PCs and compatibles, the byte ordering is LSB FIRST. For HP Workstations, the byte ordering is MSB FIRST. This limitation applies to ENTER/OUTPUT with strings only. With devices, the byte ordering can be selected in the ASSIGN statement.

INITIALIZE

HTBasic does not support INITIALIZE. To initialize a new LIF disk, use "INITIALIZE" on an HP BASIC workstation.
RAM disks are not supported with the INITIALIZE ":MEMORY,0" command. Many excellent RAM disk programs are available for the PC that makes the RAM disk available to all Windows programs, including HTBasic. These programs can usually make RAM disks in conventional, expanded, or extended memory.

INP/INPW

INP and INPW are new HTBasic functions for communicating with devices having no HTBasic device driver. They are not supported under protected mode operating systems like NT.

KBD CMODE

HP BASIC and HTBasic both use KBD CMODE ON for Nimitz keyboard softkey compatibility. The Nimitz keyboard is used on the 9836 and has ten softkeys, the lowest of which is labeled k0. The softkey labels are displayed at the bottom of the screen in two rows, each row containing five labels and each label 14 characters wide.

The difference between HP BASIC and HTBasic's implementation of KBD CMODE ON is that HTBasic exactly emulates the screen format for the labels, while HP BASIC uses an emulation that gives physical correspondence with the ITF 4-2-4 softkey layout.

LEXICAL ORDER

Several extensions are present in the LEXICAL ORDER statement of HTBasic to allow user definition of upper/lowercase rules for languages that are not built-in. The rules can also be loaded from a file.

LINE TYPE

In the LINE TYPE statement, the repeat length is ignored by most graphic drivers.

LINK

LINK is not supported.

LIST BIN

LIST BIN is programmable in HTBasic, but not in HP BASIC.

LOAD

HP BASIC PROG files and HTBasic PROG files are not compatible. To move programs between the two environments, use ASCII program files.

LOAD BIN

The LOAD BIN statement has been enhanced to allow software switches to be passed to device drivers. HP BASIC BIN files are not compatible with HTBasic.

LOADSUB

HTBasic allows a string variable to specify the name of the subprogram or function to load.

MASS STORAGE IS

The current "MASS STORAGE IS" file system includes not only the device, but also the current directory. In other words, it specifies not only which tree (device) you are in, but where in the tree (current directory) you are.

OUT/OUTW

OUT and OUTW are new HTBasic statements for communicating with devices having no HTBasic device driver. They are not supported under protected mode operating systems like NT.

OUTPUT KBD

Three editor functions have been added to HTBasic and should not be used in programs that will be executed with HP BASIC: DEL LEFT, NEXT WORD, and PREV WORD. Otherwise, all the two-character function key sequences (CHR$(255)&CHR$(X)) used by HP BASIC are compatible with HTBasic. If multiple statements are output in a single OUTPUT KBD statement, they are all executed before the next BASIC line. HP BASIC sometimes intermixes the execution with multiple BASIC lines, based on the presence or absence of "closure keys."

PARITY

PARITY is an unsupported ASSIGN option.

PERMIT

PERMIT was used under UNIX to set the permissions (mode) of a file, directory, or device. Permissions specify who can read, write, or execute a file, and who can search a directory. To change file attributes under Windows, use the file properties dialog.

PHYREC, Phyread, Phywrite

The PHYREC utilities under HP BASIC allow physical disk sectors to be read or written. Such access usually provides enhancements or file utilities for LIF formatted disks. DOS and UNIX provide methods of accessing physical disk sectors, but no PHYREC utilities are supplied with HTBasic.

PLOTTER IS

Under HP BASIC, PLOTTER IS 3,"INTERNAL" is equivalent to PLOTTER IS 6,"INTERNAL" unless the HP 98546 Display compatibility Interface is installed. Under HTBasic, 3 selects CRTA mode and 6 selects CRTB mode. If this is not your intention, you need to do what HP recommends for users of the 98546: use PLOTTER IS CRT,"INTERNAL" instead of 3 or 6. Note: CTRA mode is not supported under HTBasic.

Both HP BASIC and HTBasic do an implicit PLOTTER IS assignment for you if you attempt to use graphic statements before an explicit PLOTTER IS. The difference is that HTBasic does the implicit PLOTTER IS as soon as HTBasic is started, and HP BASIC waits until the first graphic statement is executed. The only known effect of the different approaches is that under HP BASIC, a SYSTEM$("PLOTTER IS") returns "0" until the first graphic statement is executed and HTBasic returns the correct value anytime.

HP BASIC supports only "INTERNAL" and "HPGL" graphic languages. HTBasic supports loadable graphic device drivers so it is not limited to these two choices. HTBasic also allows clip-limits to be specified when output is directed to a device, allowing use of plotters or printers that are incapable of returning p-points.

PRINT

HTBasic has been extended to allow the displacement of the attribute and color control characters that normally have the values CHR$(128) to CHR$(143). Since the PC has characters in this range that sometimes need to be displayed, HTBasic has the capability of moving the range with the statement CONTROL CRT,100;1.

PRINT LABEL and READ LABEL

PRINT LABEL and READ LABEL are used to set and read the volume label of a disk drive. HTBasic does not support PRINT LABEL. To change the label of a disk from an HTBasic program, use the EXECUTE "LABEL x:" command.

PROTECT

PROTECT is used to set LIF file passwords under HP BASIC and file attributes under HTBasic. A special form of PROTECT is used by HTBasic to change file attributes. The syntax is:

```
PROTECT file-specifier, protect-code
```

where *protect-code* is a string containing zero or more of the following characters:

Character	Meaning
(none)	no protection
R	read-only: File cannot be written or deleted.
S	system file: For the most part, this attribute has no meaning.
H	hidden: File will not be listed by a CAT command.

If a character is not included, that attribute is cleared. If the string is blank, all attributes are cleared.

PRT

Most PC printers are connected to the parallel printer interface. For this reason, PRT returns the value 10 instead of 701. Programs with statements that use PRT explicitly, such as "PRINTER IS PRT" need not be changed if a parallel printer is used on the PC. To change PRT back to 701, for IEEE-488 printers at primary address 1, use the statement "CONFIGURE PRT TO 701". You may find it convenient to include this statement in your AUTOST file. This statement is not necessary if you use the value 701 (or any other value) explicitly.

PURGE

PURGE is similar to a combination of the DELETE and RD or RMDIR commands. Unlike DELETE, PURGE will only delete one file at a time, and will also delete directories. PURGE will not delete a directory unless there are no files in that directory. Also, *HTBasic will allow you to PURGE an ASSIGNed file.* The actual PURGE takes place after the file is closed.

QUIT

QUIT closes the HTBasic child window. Any program or data in memory is lost. You should store any program changes before quitting.

QUIT ALL

QUIT ALL exits HTBasic and returns to the operating system. Any program or data in memory is lost. You should store any program changes before quitting.

READIO/WRITEIO

READIO/WRITEIO access hardware registers directly, and therefore, if the interface hardware is different than the hardware on an HP BASIC Workstation, the READIO/WRITEIO registers will not be compatible. TransEra supplies a PC IEEE-488 bus controller card that duplicates the Series 200/300 HP-IB READIO and WRITEIO registers. Other IEEE-488 bus controller cards are not usually completely compatible. For other interfaces consult the interface register documentation.

RENAME

RENAME is used to change the name of a file, but can also move a file from one directory to another directory on the same disk. RENAME is similar to the DOS RENAME command.

SAVE

The SAVE statement in HTBasic can be set to save programs in either ordinary ASCII or LIF ASCII. This is done with the CONFIGURE SAVE ASCII statement. (DOS ASCII and UNIX ASCII are ordinary ASCII files.)

SCRATCH BIN

SCRATCH BIN is not supported for HTBasic binaries. You must QUIT ALL and re-start HTBasic to scratch all binaries.

SEPARATE ALPHA

HP BASIC assigns green to the alpha plane by assigning green to pens 8 through 15. HTBasic assigns white. If you prefer green, or some other color, you may explicitly set pen values 8 to 15 to the color desired.

SET ALPHA DISPLAY MASK

SET MASK is not supported. Currently, MERGE/SEPARATE ALPHA are the supported methods of changing the masks.

SET CHR

SET CHR is not supported.

SET TIME/TIMEDATE

HP BASIC/UX keeps a BASIC time that is separate from the actual system time. SET TIME and SET TIMEDATE, specified without any time value, resynchronized the two. HTBasic uses SET TIME to set and SET TIMEDATE to read the system clock.

SOUND

SOUND is not supported.

STATUS

STATUS @Iopath,2 always returns a 4.

STATUS @File,3 returns the current length, not the *CREATE* length. Under HTBasic, files are extendible.

The STATUS() function (as opposed to the STATUS statement) is an addition to HTBasic. Any STATUS or CONTROL registers greater than 99 are also additions.

As in HP BASIC, STATUS register 0 of interface cards contains the card ID. Interface cards that are available on a PC, but not on an HP BASIC Workstation are identified with ID numbers greater than or equal to 300.

STORE

HP BASIC PROG files and HTBasic PROG files are not compatible. To move programs between the two environments, use ASCII program files.

STORE SYSTEM

In HP BASIC this statement stores a copy of the operating system with all loaded BINs already linked in. Under HTBasic this is not possible. Use AUTOST to load HTBasic device drivers.

SYMBOL

LORG 5 moves the symbol origin from (0,0) to (5,8). In HP BASIC it moves the origin to (4.5,7.5).

SYSBOOT

HTBasic does not support SYSBOOT, which under HP BASIC reboots the computer. Since HTBasic runs as a guest of the operating system, it is considered inappropriate to reboot the computer.

SYSTEM$

Minor differences in some SYSTEM$ responses exist where appropriate to reflect the hardware differences between the Windows and HP/UX operating systems. See SYSTEM$ in the on-line Reference Manual for more details.

The SYSTEM$("DISP LINE") function is an HTBasic extension that returns the present contents of the display line.
The SYSTEM$("VERSION:HTB") function returns the HTBasic version description, for example, "Windows Release 9.0". This function can be useful for programs that run on both HP BASIC and HTBasic systems, enabling them to determine which system they are currently running on. The following example sets a variable according to the system running the program:

```
10  SUB Which_system
20    COM /Which_system/Htbasic,Hpbasic
30    IF SYSTEM$("VERSION:HTB")="0" THEN
40      Hpbasic=1
50    ELSE
60      Htbasic=1
70    END IF
80  SUBEND
```

TIMEZONE

HTBasic does not require this statement and will return an error if an attempt is made to execute it. The editor will allow it to be entered, and the syntax checker will check it for correctness to allow you to develop programs and run them under HP BASIC. HP BASIC requires this statement for two reasons: 1) HP BASIC/UX keeps a time clock independent of the UNIX time, and 2) it is possible to boot HP BASIC/WS on a computer whose real-time clock is set to Greenwich Mean Time (GMT).

TRANSFER

HTBasic currently supports TRANSFER for files, RS-232 and GPIB.

WRITEIO

See the explanation under READIO for some differences. Other processors cannot execute the Motorola code accessed by WRITEIO 9827. The code must be rewritten.

XREF

HTBasic allows a string variable to specify the name of the subprogram or function.

ZERO

ZERO is an unsupported ASSIGN option.

LIF Diskette Utilities

The LIF diskette utilities are used to copy program and data files between LIF diskettes and PC disks. Where necessary, the file header is converted and then the data is copied to the new file. These utilities work with single or double-sided diskettes, formatted with 256 or 1024 byte sectors.

NOTE: These utilities do not work on protected mode systems such as Windows NT.

HPCAT prints a CATalog of files on an HP LIF diskette. *HPCOPY* copies files between LIF diskettes and PC disks. *HPPURGE* deletes files on LIF diskettes. These commands are used at the DOS prompt. To use them while running HTBasic, use the EXECUTE command, i.e. EXECUTE "hpcat 0:". You may use these commands if they are in the current directory or if a PATH has been set up to the directory in which they are stored (see your DOS manuals).

Problems

It is fairly common to find a LIF floppy that cannot be read by one or more PC floppy disk drives. If you have problems, try several computers from different manufacturers and you can usually find one that works. If you have problems, read the "Common Problems" section later of this manual.

HPCAT

Display a CATalog of files on an HP LIF diskette.

Syntax:

```
HPCAT drive:
```

where:
drive = 0, 1, ...

Sample:

```
HPCAT 0:   ! drive A
HPCAT 1:   ! drive B
```

This command allows you to display a catalogue (directory) of the files on an HP LIF diskette. An HP LIF diskette is one that was initialized with the HP BASIC INITIALIZE command on an HP 9000 Series 200/300 workstation. The diskette must be inserted into a PC diskette drive. Both 5-1/4 and 3-1/2 inch diskette drives are supported. Disk drives connected to the HP-IB are not supported.

The first drive, "A", is number 0, the second drive is number 1. It is recommended that you only use diskette drives "A" and "B". In limited circumstances, other drives will work, but some experimentation is needed to use them. For other drives, you must discover the diskette drive number. Try values from 0 to 9. If none work, your drive or system may not support the 256 or 1024 byte sectors required by HP diskettes.

HPCOPY

Copies files between an HP LIF diskette and a DOS disk.

Syntax:

```
HPCOPY [drive:]lif-filename [disk:]dos-filename [-LIF]
HPCOPY [disk:]dos-filename [drive:]lif-filename
```

where:
drive = 0, 1, 2,...
disk = A, B, C,...
lif-filename = a legal LIF filename, may include wildcards
dos-filename = a legal DOS filename, may include wildcards

Sample:

```
HPCOPY 0:hpfile C:DOSFILE
HPCOPY C:GOOD.BYE 3:Hello
HPCOPY A:DOSFILE 1:HPbdat
HPCOPY 0:lifASCII C:\DIR2\DOSFILE
HPCOPY 0:* C:
HPCOPY C:\HTB\DATA\D?T* 0:
```

This command allows you to copy ASCII, BDAT and ordinary files from HP LIF diskettes to DOS disks or vice-versa. An HP LIF diskette is one that was initialized with the HP BASIC INITIALIZE command on an HP 9000 Series 200/300 workstation. The diskette must be inserted into a PC diskette drive. Both 5-1/4 and 3-1/2 inch diskette drives are supported. Disk drives connected to the HP-IB are not supported.

Programs must be in ASCII format. PROG files are not supported. BDAT files can be transferred with the limitations noted below.

When files are copied no translation is done on the file contents; a LIF ASCII file remains a LIF ASCII file. However, you may write simple HTB programs that do the translation if you need to use the data files with other DOS programs. This conversion is not required if the data files will be used only by HTBasic, because HTBasic knows how to use HP BASIC files.

Filenames

The lif-filename should be the legal name (including correct capitalization) of a LIF file. If the LIF diskette is in drive A or B, prefix the name with "0:" or "1:". If the LIF diskette is in another drive, specify the drive number (as explained above under "HPCAT").

The dos-filename should be the legal name of a file. This name optionally can include a drive letter and a full path. If no drive is specified, the default drive is used. This drive must be different from the drive containing the LIF diskette. If no path is given, the present directory is used.

If the destination file already exists on the PC, it is overwritten. If the destination file already exists on a LIF diskette, an error is reported and the file is left unchanged.

Copying Files to HP BASIC

HTBasic can create typed files that are compatible with Series 200/300 computers, but that is not the default. If you plan on transferring files back to HP BASIC systems or sharing the files with HP BASIC systems on a network, you should execute the CONFIGURE CREATE "HP" statement before creating any files. Any BDAT or ASCII files created after this statement is executed are completely compatible with Series 200/300 computers.

If you do not plan on transferring files back to HP systems, it is best to use CONFIGURE CREATE "HTB" (the default), since the file header is smaller and the fastest byte ordering will be used.

Using Wildcards

Wildcards can be used to transfer more than one file at a time. If a wildcard is used, it should only be used in the source filename, not the destination filename. The destination should specify only the drive (and directory if the destination is a PC disk). Legal wildcards are "*" and "?". An asterisk will match any one or more characters starting at that location. A question mark will match any one character at that location. These conventions are also explained in your Windows manual.

Because of the differences in legal LIF and PC filenames, filenames may be translated. A LIF filename is limited to at most ten characters and all ASCII characters except "space,:,<,|" are legal LIF filename characters. A DOS filename is limited to 8 characters, a period and 3 characters and all characters except "\/:|<>+=;," and control characters whose ASCII value is less than the space character are legal DOS filename characters. Also, lowercase letters are converted to uppercase by DOS.

When transferring a file from a DOS to a LIF disk the first ten characters of the filename, including the period, are used for the LIF name. Any illegal LIF characters in the DOS filename are translated to an underscore character.

When transferring a file from a LIF disk to a DOS disk the filename is converted to uppercase and if necessary, a period is inserted after the eighth character. Because DOS converts lowercase letters to uppercase, two LIF files named "Aa" and "aa" will be transferred into the DOS filename "AA". The second file transferred will overwrite the first file transferred.

Note: Wildcards are not supported by the HPCOPY that is supplied with the Demonstration Version of HTBasic.

-LIF Option

With the -LIF header option, when copying files from a LIF diskette to DOS, if the file type is ASCII or BDAT, the file is created on the DOS disk with an HP LIF file header rather than an HTBasic file header. When copying files from a LIF diskette to DOS, all other file types besides HP-UX are automatically created on the DOS disk with an HP LIF file header. When copying files from a DOS disk to LIF, this option is ignored; files with HP LIF file headers are handled automatically.

HPPURGE

Deletes files from an HP LIF diskette.

Syntax:

```
HPPURGE [drive:]lif-filename
```

where:
drive = 0, 1, ...
lif-filename = a legal LIF filename

Sample:

```
HPPURGE 0:LIFFILE
HPPURGE 1:Hello
```

This command allows you to delete ASCII, BDAT and ordinary files from HP LIF diskettes. An HP LIF diskette is one that was initialized with the HP BASIC INITIALIZE command on an HP 9000 Series 200/300 workstation. The diskette must be inserted into a PC diskette drive. Both 5-1/4 and 3-1/2 inch drives are supported. Diskette drives connected to the HP-IB are not supported.

The lif-filename should be the legal name (including correct capitalization!) of a LIF file. The first drive, "A", is number 0, the second drive is number 1. It is recommended that you only use diskette drives "A" and "B". In limited circumstances, other drives will work, but some experimentation is needed to use them. For other drives, you must discover the diskette drive number. Try values from 0 to 9. If none works, your drive or system may not support the 256 or 1024 byte sectors required by HP LIF diskettes.

Common Problems

The following paragraphs document some common problems you may experience trying to use HPCAT, HPCOPY and HPPURGE. If you experience a problem, glance through the headings to find the answer to your question.

HPCOPY Says the File is Not Present, But It Is. Use HPCAT to find the *exact* spelling, including upper and lower case. Remember that LIF filenames are case-sensitive and DOS filenames are not. "HELLO" and "hello" refer to different LIF files, but the same DOS file.

HPCOPY Gives An Error Part Way Through the File and Then Stops. This error is common on double-sided 3-1/2 inch diskettes. It usually means your computer ROM BIOS is incapable of reading some LIF diskettes. Try another PC or another LIF diskette. Often a single sided LIF diskette will work where a double sided will not. A 9122 drive can be instructed to initialize a double-sided (or single sided) diskette as if it were single sided by using INITIALIZE option 4. If the diskette has previously been initialized, you need first to remove the HP format information from the second side by formatting on your PC (or by using INITIALIZE option 2) before initializing as single sided.

The Error "Bad command or filename" Is Reported. HPCAT or HPCOPY is not in the current directory and no PATH is set up to point to them.

Error 910 Is Reported. You are currently running HTBasic, not a DOS Window. HPCAT, HPCOPY and HPPURGE are DOS command line commands, not BASIC commands. Issue the commands in a MS-DOS window. Or use the QUIT ALL command to exit to Windows and start a DOS window before proceeding. Or use the HTBasic EXECUTE statement:
EXECUTE "HPCAT 0:"

The Error "Sector not found..., Abort, Retry, Ignore?" Is Reported. This or similar errors usually mean that you have inserted a LIF diskette in the current DOS drive. While your DOS prompt is "A" you cannot put a LIF diskette in drive A.

The Light on the Drive Does Not Turn On. If your diskette drive has a letter other than "A" or "B", you may not be able to use it. In limited circumstances, other drives will work, but some experimentation is needed to use them. For other drives, you must discover the diskette drive number. Try HPCAT with values from 0 to 9 until the light comes on the drive you are trying to use.

Changes From Earlier Releases

The following sections document the differences between the current release and earlier releases. Changes from 8.0 to 9.0 "Changes From 7.0 to 8.0," "Changes From 6.0 to 7.0," "Changes From 5.0 to 6.0," "Changes From 4.0 to 5.0," "Changes From 3.0 to 4.0" and "Changes From 1.x/2.x to 3.0" are the main sections. If you are upgrading from more than one release, you may need to read more than one section to see what changes affect you.

Changes from 8.0 to 9.0

If you are upgrading from 8.x to this release, read the following material to see what changes may affect you.

The Numeric Compiler has now been fully integrated into HTBasic. It is available on the Tools menu. It has been enhanced for full support of LONG variables, a GUI interface allowing for option checking, and sub selection. The Numeric Compiler can greatly increase execution speed of numerically intensive subroutines and can provide an additional level of security.

In addition to _cdecl type DLLs, _stdCall DLLs may now be called with the DLL Toolkit. They syntax is:

```
DLL GET "STDCALL VOID DLL:FUNCTION" AS "ALIAS"
DLL GET "CDECL VOID DLL:FUNCTION" AS "ALIAS"
DLL GET"VOID DLL:FUNCTION" AS "ALIAS" ! defaults to cdecl
```

Because the stdcall support was a last minute change, examples will be forthcoming. They will be made available on our website.

Timing resolution is improved for most Operating Systems down to 1ms. Use STATUS(32,5) to detect timing resolution. A 1 indicates the faster timing resolution is in effect.

GFONT now has full support for double byte fonts such as MS Mincho for the Japanese language. Start-up command line is: "C:\Program Files\HTBWin90\Htbwin.exe" -fn "MS Mincho",14,128

Corrected GFONT to properly handle LORG modification to LABEL.

HTBasic has been optimized for faster execution of programs. The Runtime version has been optimized so that it now runs programs as quickly as the Development version.

Using Ctrl + A to select all code in the editor then using copy and paste into a new document would only get the first page of code. It now properly copies all lines of the program.

Using Ctrl+C to copy the contents of the command line in the Windows editor now properly copies the contents to the clipboard.

The status bar is now properly displaying run state while in debug mode.

The wait cursor no longer comes up when trying to run a file where there is no security access for write.

The Windows print monitor now properly displays the filename of the program producing the printed output. Previously it would only display the source as HTBasic.

XREF no longer causes program errors when run with no program in memory.

XREF, when run with the SUB option, now properly only displays the subs.

Continuing a paused program now properly clears the display line of the paused program.

When using the performance tuning register with CONTROL

KBD,207;3 HTBasic now properly returns to normal priority when stopped. This prevents an erroneous wait cursor when opening a new file.

The >= expression evaluator is now properly working for LONG variable types.

Corrected LONG variable behavior when comparing to zero in IF statements.

ALLOCATE now functions with all LONG variable conditions.

LONG support added to the BIT functions, DET, IVAL, and IVAL$ statements.

Corrected STATIC declaration of multiple variables.

Corrected editor behavior when dealing with SUBs, CSUBs, out of context areas.

Properly allow deletion of CSUBs when highlighting the CSUB and pressing the delete key. Corrected deletion of CSUBs using the DEL command.

Write to Display CSUB usage is slightly changed. For the dispxy.c second byte color and enhancement attributes use the following values. These were updated to provide more flexibility in the color enhancement attribute.
0x0F => 0x00ff
0x010 => 0x0100
0x020 => 0x0200
0x040 => 0x0400

Debugger correctly handles breakpoints set on first line of a program. Previously it would not always catch the breakpoint.

Breakpoints for string variables now properly handle variables 15 and more characters in length.

In Legacy Keyboard mode, ON KBD ALL now properly processes Ctrl+N and Ctrl+O. Previously these were processed as menu items.

Non-colormap mode was restored. In versions 8.x HTBasic was always in colormap mode. GESCAPE CRT,4 now properly returns to normal drawing mode.

For programmatic control of the NUM LOCK key, CONTROL KBD,211;X and STATUS(Kbd,211) were implemented. Where X=0 turns off NUM LOCK, and X=1 turns on NUM LOCK.

CONTROL KBD,16;1 now properly turns off keyboard scrolling. Previously this feature would only work if the program was paused or stopped.

In the DEBUGGER using Step Out now properly Debug Pauses at first line after return from sub call. Previously it would run to the end of the program.

Opening a new file after opening HTBasic by double clicking on an associated file type such as .prg, now properly clears the old file out of memory before creating the blank document.

DLL toolkit was updated to support passing of full arrays. To pass a full array to a DLL, pass the first element in the array and then in the DLL

receive it as a pointer to an array, and access it as such.

DLL toolkit was repaired to properly process large programs with large quantities of variables.

One line IF statements that call DLLs with the DLL toolkit now properly evaluate.

When attempting to edit a program while in Debug mode, HTBasic now properly stops debug and returns to edit mode. Previously it would appear to be in edit mode but would not accept changes to the program.

The TRACK CRT crosshair now properly appears when alpha and graphics are separate.

Editor text colors now properly change colors as per the Edit Environment Dialog box.

Dump Device Is to a file with EXPANDED option using the PS-DUMP driver now properly rotates instead of mirroring the output.

The PS driver now will properly output to paper sizes larger than 8.5 X 11. Previously any size specifications in the PLOTTER IS statement were ignored.

The BASIC Plus CONFG file is now properly changing all colors. Previously some colors would not change to their specifications in the CONFIG file.

The BASIC Plus File Dialog now properly displays a full list of drives on the drives dropdown. The ESC key is properly treated like the cancel button. Not selecting a file properly returns just the directory. Previously the HTBTree DLL was used for traversing directories.

The BASIC Plus List widget now properly automatically adjusts line height as necessary based upon the FONT attribute.

The BASIC Plus Combo Widget now properly displays the dropdown under all OS's. Previously under Windows 98 it would place the dropdown in an unpredictable location.

Close functionality via X in top right corner added to all BASIC Plus widgets. Added a "CLOSEABLE" attribute and an "ON SYSTEM CLOSE" event.
For example:
CONTROL @Main;SET ("CLOSEABLE":1)

ON EVENT @Main,"SYSTEM CLOSE" GOTO Finis

The BASIC Plus TIMER widget now allows LONG values for the VALUE attribute.

The BASIC Plus widget Bitmap was corrected to work with all screen resolutions and color depths. Corrected DUMP WINDOW and DUMP AREA attributes to function as documented.

The default color for LABEL statements is again a pure white. In version 8.3 it was 1% grey.

The memory leak caused by loading multiple instances of BPLUS was corrected.

Using lengthy ASSIGN statements with the BASIC Plus Bar Widget no longer causes Runtime Errors.

In the Save As dialog saving to a text file no longer requires CONFIGURE SAVE ASCII OFF to be set. The file created now by default is readable by word processors and source saving programs.

Using re-save to save a file that is in memory but not available on disk now properly saves the file, or prompts for a disk if the disk has been removed.

Error messages may now be displayed in a dialog box. This may be enabled/disabled in the run environment dialog.

Using RE-STORE SUB Subname TO "filename" to store subroutines to a file no longer improperly creates files with unknown passwords.

Values that fit into the INTEGER variable range, not specifically declared as REAL variable types no longer automatically convert to LONG variables.

Using ENTER from a Buffer created by the CAT command now properly displays the catalog.

The GUI GOTO selection dialog has been repaired to properly move to Functions that have names 15 characters in length.

Using LIST KEY with PRT set to 10 now properly lists the softkeys to the default printer.

The Numeric Compiler now compiles the calling of CSUBs from ON EVENT statements. However, since this is not supported, it generates error 2007 when running. It is recommended to call a sub from the ON EVENT that in turn calls the CSUB.

The Application Runtime version now properly accepts –geometry switches specified in character size, i.e. less than 100.

The Application Runtime version now properly produces an error when a file load attempt from the file menu fails.

In the Runtime version, GESCAPE codes 64 & 65 now properly toggle the program name in the title bar.

DEVICE SETUP

The Device Setup menu option was moved to the proper location on the Tools menu. In the runtime version, it is on the File menu.

A SCRATCH BIN command was added for removing all drivers that can be unloaded from memory.

CONFIGURE SYSTEM ("DEVICE SETUP") brings up the Device Setup Programatically.

The number of loadable drivers in the Device Setup has been increased from 16 to 32.

Shortcut keys were added for adding devices to the Device Setup.

GPIBNI

The GPIBNI Driver has been added to the Device Setup, and can now be loaded and configured from the Device Setup.

The driver now properly reports SRQ status even after a REQUEST. Previously using STATUS (7,7) after a REQUEST statement would give bad data.

Using REQUEST no longer causes a program crash when used with a board loaded as NOT SYSTEM.

The ENTER statement may now be configured to terminate upon receiving a CR/LF

The TRANSFER statement was enhanced for stability.

All Interrupts are now supported with the GPIBNI driver except for the IFC, PPOLL, and SPAS Interrupts. Supported Interrupts will consistently interrupt an unlimited number of times.

Multiple National Instrument cards can now be loaded and used concurrently by the GPIBNI driver. Previously using more than one card with the same driver would cause unpredictable results.

SERIAL

Serial TRANSFER no longer stops when transferring to large buffers.

Serial status register six no longer changes when using COM statements. Previously COM statements would alter the contents of the register.

Serial status register five now reports correct values. Previously the values were the opposite of what was expected.

The Serial Interrupt thread now gets started correctly. Supported Interrupts will consistently interrupt an unlimited number of times. Serial Drivers Modem status Interrupt is now Interrupting correctly.

When changing Handshaking options using control statements, the check box's in the Device setup now get changed to match the current configuration.

GPIB900

The GPIB900 Driver has been added to the Device Setup, and can now be loaded and configured from the Device Setup. This driver is a kernel level driver and is supported by all Operating Systems supported by HTBasic. It is no longer necessary to use Agilent's SICL I/O libraries to use this card under NT.

Multiple TransEra 900 cards can now be loaded and used concurrently using the GPIB900 driver.

Supported Interrupts will consistently interrupt an unlimited number of times. PPOLL Interrupts are now supported using the GPIB900 Driver.

Transfer statements are now fully supported using the GPIB900 driver.

GPIO

The GPIO Drivers have been added to the Device Setup,

and can now be loaded and configured from the Device Setup. The GPIO600 driver is for the TransEra Model 600 GPIO board. The GPIO650 driver is for the TransEra Model 650 GPIO board. Both of these drivers are kernel level drivers and are supported by all Operating Systems supported by HTBasic.

Changes From 7.0 to 8.0

If you are upgrading from 7.x to this release, read the following material to see what changes may affect you.

- New Windows style editor
- New HTBasic Debugger
- DLL Toolkit to allow users to call Dynamic Link Libraries from HTBasic
- TRANSFER function for GPIB, Serial, and File is fully supported
- New LONG integers variable type with a range of -2,147,483,648 to 2,149,483,647 (See the on-line **Reference Manual** for more information)
- STATIC variables added as another variable type (See the on-line **Reference Manual** for more information)
- Enhanced printer support for multiple types of drivers (See Chapter 6 of this manual)

New Editor

In addition to the traditional HP BASIC-style editor (Legacy Editor), HTBasic 8.0 also includes a new Windows style editor. The new editor has the ability to toggle line number On and Off. When "On", line number behavior essentially remains consistent with the Legacy Editor except the HTBasic for Windows editor line numbers can be edited only through the RENumber, COPYLINES, and MOVELINES commands. When in the "Off" position, the numbers are not removed, but the line numbers are no longer displayed.

A primary addition to the new editor is the implementation of standard Windows mouse functions. Source code is more easily edited with cut, copy, and paste functions. Another standard Windows feature now implemented is the Undo and Redo function.

Bookmarking functions have been added to the new editor for improved code navigation. The Windows editor also provides the user the ability to customize the color and font selection in the editor environment. Color allows individualized adaptation of the source code to fit each user's requirements.

Debugger

The new HTBasic Debugger was designed to promote optimal programming effectiveness and flexibility using HTBasic within the Windows operating system. The Debugger tools allow the user to view the program in specific detail.

A key feature of the Debugger is Breakpoints, which provide a "pause" in program execution so that variable values and other parameter changes may be observed. The HTBasic debugger supports line, conditional, and global breakpoints. Line breakpoints pause execution in a specified line. Conditional breakpoints pause execution at a specific line if a specified condition is met. Global breakpoints pause program execution when a specified variable reaches a specific value and condition regardless of where the program is.

Another feature of the debugger is Step functions, which allow the user to move through and observe program execution. There are three step functions: step in, step over, and step out. "Step in" steps one line of code at a time, following all program branches. "Step out" continues to end of context, stopping when entering the calling context. "Step over" runs the entire sub context and stops at the next executable line of the current context. "Step out" and "Step over" will stop at all breakpoints in sub contexts.

Running to the cursor and running from the cursor provide a fast and flexible way to move through code. Run to Cursor and Continuing from the Cursor are similar in functionality to a breakpoint with the cursor acting as a breakpoint. Run to / from Cursor is governed by the same rules as the CONTINUE command.

One of the most powerful tools in the Debugger is the new Debug Window. Six new windows are provided to assist in debugging programs by permitting the user to monitor variables, subroutines, breakpoints, and the BASIC source code as it executes. All Debug Windows may be moved and rearranged to suit the working style of any programmer.

The Watch Window allows the user to observe a list of user-defined variables and their values during each step of program execution. There are "Variable" (variable name), "Type" and "Value" columns in the Watch Window. Type can be Array, Integer, Real, Complex, String, Long, Static, or I/O Path.

The Line Breakpoints Window allows the user to observe the line breakpoints as the program is run. Breakpoint parameters monitored are "Enabled," "Type," "Line," "Subroutine," "Variable," "Condition," and "Value."

The Global Breakpoints Window permits the user to observe the global breakpoints as the program is running. The Global Breakpoint Window monitors "Enabled," "Subroutine," "Variable," "Condition," and "Value" status.

The Trace Window permits the user to observe which commands are being executed in a running program. There is nothing to "set" in this window. It automatically monitors and notes each HTBasic command line as it is executed. Only a "Command" column exists in the Trace Window. The Trace Window differs fundamentally from HTBasic's well-known TRACE Statement in that the Trace Window provides a running trail of commands executed. The TRACE Statement is limited to only what appears on the message line before it scrolls away.

The CALL Stack Window was designed to open a view into the CALL Stack (the CALL Stack is used by BASIC to track subroutines accessed by CALL statements) so that one can see it operate at each step of the running program. There is nothing to "set" in this window. It automatically monitors and notes what is going on in the program defined CALL Stack as the program is running. Only a "Subroutine" column exists in the CALL Stack Window.

The Code Window displays the program source code as the program is running. There are "Line" (Number) and "BASIC Code Lines" columns in the Code Window. These columns show the line number within the BASIC source code program and the actual text of the code lines. Breakpoints, Bookmarks, and the program pointer are seen in this window as well.

DLL Toolkit

The new DLL Toolkit allows users to call Dynamic Link Libraries (DLLs) from HTBasic. The DLL Loader calls precompiled functions created in other programming languages, most notably C/C++, using the _cdecl calling convention. This will allow the user to have the object-oriented flexibility of C and C++ while retaining the simplicity of HTBasic.

TRANSFER

The TRANSFER function for GPIB, Serial, and File is fully supported in the HTBasic 8.0 release. The TRANSFER statement sets up a data transfer between memory and a device. This transfer typically occurs in the background while HTBasic continues to run in the foreground. The GPIB TRANSFER is supported under all of the current GPIB drivers including GPIB, GPIBNI, and HPIBS (SICL) drivers.

Changes From 6.0 to 7.0

If you are upgrading from 6.x to this release, many new features have been added. The most significant are listed here.

- Added Display Functions, which control the display for control characters on the CRT. See on-line **Reference Manual**
- Support for SEPARATE ALPHA FROM GRAPHICS. See on-line **Reference Manual**.
- New drivers for National Instruments' line of Data Acquisition (DAQ) cards including their AT, PCI, PXI, and PCMCIA platforms
- New GESCAPE CRT CODES were added for manipulation of the HTBasic program window. See on-line *Reference Manual*
- KBD Control Registers were added for directory CAT and mouse control. See Chapter 4.
- Version 7.0 has added improved functionality to the SERIAL32 driver, which include inbound TRANSFER, clearing of the receive buffer using RESET, and BREAK detection.
- The HP SICL driver for GPIB (HPIBS) now supports the TransEra's Model 900 GPIB card under Windows NT.

Changes From 5.0 to 6.0

If you are upgrading from 5.x to this release, read the following material to see what changes may affect you.

- HTBasic for Windows no longer requires the use of a hardware key; it now uses a software solution for protection against unauthorized reproduction. There is a serial number provided in the packaging that is required to be input during installation. Proper installation requires the correct serial number.
- Other changes to Version 6.0 include enhancements to the SERIAL32 device driver, including:
- Support for STATUS register 5, which is the Read hardware handshaking, output line
- Support for STATUS register 11, which is the Modem Status line
- Increased Baud rates up to 115200 bps

Changes from 9.0 to 9.1

An auto-save feature was added. On the Run Environment dialog a new tab AUTO SAVE provides automatic saving of programs. Options may be set for the duration and directory to save the backup file into. Files will be saved with an .AutoSave extension.

The Secure Utility (previously only available with the HTBasic Workshop) is now included as an option on the Tools menu. The SECURE utility processes an HTBasic PROG file for distribution to make it smaller and harder to reverse engineer It removes most of the embedded program information, compresses the unused control table space, secures the program lines to prevent listing.

The CSUB Toolkit (previously only available with the add-on package CSUB Toolkit) is now included as an option on the Tools menu. The CSUB Toolkit allows one to build compiled subprograms for HTBasic. A compiled subprogram, or CSUB, runs directly on the processor hardware and has access to all of it's power and functionality.

The Advanced Math Library is now included with the HTBasic package instead of being available as an add-on component. The HTBasic Math Library is a collection of subroutines that give users of the HTBasic programming language access to fast versions of higher mathematical and signal processing functions. Most of these routines are compiled, so they run at a much higher speed than equivalent BASIC subroutines. The routines are meant to be incorporated into user BASIC programs to enhance their speed, and to save the user's writing the subroutines. The routines are installed into the mathlib folder, and samples for each of them are found in the math_ex folder. The Advanced Math Library Help file also details each of the routines, their syntax and usage.

An option was added to the Tools menu to directly access the BASIC PLUS Screen Builder Utility.

The BASIC PLUS PUSHBUTTON widget now wraps text if the pushbutton is not wide enough to handle the entire text.

The BASIC PLUS Number Dialog now properly accepts all numbers when running in Octal mode.

The BASIC PLUS Panel Widget now properly shows a minimized icon.

The BASIC PLUS System Widget now accepts the ON EVENT @Sys,"SYSTEM CLOSE" syntax for programmatically closing System Widgets.

An option was added to the Run Environment Dialog to allow for sending PURGEd files to the Window's recycle bin rather than permanently deleting the files.

A copy option was added to the Debug Watch window.

A dll sample that returns the current Window's username, HTBUserName.dll, was added.

The DLL toolkit was enhanced to support returning more than 256 string characters.

An icon was added to the toolbar to provide quicker access to the Device Setup Dialog.

The ability to call CSUB's from an ON EVENT call was added.

It is now possible to print code in color using the Print Program option on the file menu.

A BW option was added to the WIN-DUMP driver for dumping output to color printers in Black & White. The default option was also changed to COLOR. To print in grayscale, use the GRAY option.

GESCAPE PRT,106 was expanded to allow for DUMPing portions of the screen with the WIN-DUMP driver. New Control and Status registers were added for specifying the DUMP region.

The new SERIAL driver introduced with the 9.0 release may now be loaded with the LOAD BIN "SERIAL" syntax as well as from the Device Setup dialog.

An option was added to return the name of the COM port that a particular ISC has been assigned to using the SERIAL driver. The syntax for this is: STATUS (Isc,102).

Performing a RESET on a SERIAL ISC now properly retains setup information for the baud rate. Previously this was being improperly reset.

The SYSTEM$("PRINTER NAME") returns the current Description of the Driver currently assigned to the PRINTER IS. If this driver is the WIN-PRINT driver, it returns the printer description.

Functions and subprograms are now properly separated in the XREF output.

The WAIT statement was optomized to not dominate CPU cycles.

The GLOAD command was enhanced to properly allow for repositioned re-loaded images to different locations on the BASIC screen.

Find from the output window now properly scrolls. Previously it would not scroll to the next found item.

When reading in an ASCII file blank lines no longer read with a "!" comment character.

The TIMEDATE function has been enhanced to match the 1 millisecond resolution for timing functions implemented with the 9.0 release. These will only be effective on NT based operating systems.

Match Whole Word and Match Case options under the Search/Find menu are now working correctly.

Blank line editing has been improved for stability.

DISPLAY FUNCTIONS is now fully supported on localized versions of Windows.

The GPIB driver is now properly handling multiple user accounts. Previoulsy only Administrator accounts could use the GPIB driver on setups other than the default.

Interrupts with the HPIBS Driver were enhanced to provide support for all interrupt types.

The TIME option in the HPIBS driver is now properly setting duration for TIMEOUT conditions.

The Default SICL name in the HPIBS Properties dialog was changed to gpib0 to match the currently available Agilent SICL package. The default name was changed from hpib7 to gpib0.

When Entering a String array using GPIBNI 9.0 Driver an Error 153 "Insufficient data for ENTER" was being received unless USING "K" was used. The driver was reporting EOI when an EOS character was seen even if EOI did not really occur. This has been corrected to properly terminate with out needing to use USING "K".

Doing an incoming TRANSFER using the GPIBNI driver, using a string that had a Line feed in the middle of it, the transfer would terminate when hitting the line feed. This was corrected so the TRANSFER would terminate properly.

Changes from 9.1 to 9.2

A function called LoadSub has been added to the Tools menu. This function allows the user to LOADSUB subprograms and functions from a PROG file using a list of all available SUBs in a list box.

A new DLL for creating custom toolbars has been added to the samples folder in the DLL Toolkit.

Control Register 158 was added to clear the Dump Graphics Custom Coordinates set using registers 154 – 157. After setting the control registers 154 - 157, their values are always set in that particular instance of HTBasic. This register clears those values. This will also give the user the ability to do a partial dump during one part of his program, and a full dump in another part.

Full support for LONGs has been added to the Numeric Compiler.

Recursive Cat, Dump Plus and GLOAD ON and OFF has now been added to the Runtime Options Dialog.

Blank lines no longer have !~! added to them when saving an HTBasic program as an ASCII file. Previously when opening the file in an editor other than HTBasic, all blank lines contained the !~!.

CAT on a non-existent filename from the command line HTBasic generates error 56. The old HP Workstations generate an error 56 when executing the same code. To help those that are used to the HP Workstation behavior a configure system command was added. To set this switch click on the "HP Skip" checkbox under the Options/Run Environment menu, or by using the CONFIGURE SYSTEM("HPSKIP ON") and CONFIGURE SYSTEM("HPSKIP OFF") options.

Changing the properties of the Serial driver with CONTROL statements, will now properly update the properties for the Serial driver entry in the Device Setup dialog.

CONTROL KBD, 16 was changed to have multiple options.
They are as follows:
CONTROL KBD, 16;0 --> enable scroll keys
CONTROL KBD, 16;1 --> disable scroll keys
CONTROL KBD, 16;2 --> disable scroll keys, but allow scroll keys to trigger ON KBD event.

Control statement was added to allow use of the PPA type Printers when doing a DUMP GRAPHICS. CONTROL 10,160;1 now enables the use of PPA Printers. CONTROL 10,160;0 disables use of PPA Printers. PPA printers include the HP DeskJet 710C, 712C, 720C, 722C, 820Cxi, 820Cse,1000Cxi, 1000Cse.

Dithering is now working properly when using Area Intensity and Area Color.
Focus is now correctly set to String Widgets that are children widgets to a panel.
HTBasic now generates error 951 instead of hanging when trying to execute an incomplete save command. Ex. SAVE "filename",

GLOAD / GSTORE command functionality when using optional parameters has been changed. Prior to 9.1 it was not possible to GSTORE using positioning parameters and then GLOAD using those same parameters. To correct this issue, the GLOAD command began to require parameters when loading in an image. It is now possible to GLOAD with or without parameters, however to change between the New and Old GLOADs it is necessary to use the following syntax:
CONFIGURE SYSTEM("GLOAD UP")
or
CONFIGURE SYSTEM("GLOAD DOWN")
Additionally these settings may be set in the Run Environment dialog.

GPIBNI Driver no longer gives a "Missing option or configuration" error when executing a SEND command with the TALK option.

HPIBS Driver performing an ENTER, the ON TIMEOUT statement now works properly. Previously the ON TIMEOUT statement was ignored.

Lines of code are no longer lost when highlighting and printing a section of code that extends to 2 or more pages.

LONG variables are now fully supported in the Numeric Compiler. Previously, certain computations such as two LONG variables being added and the value assigned to a REAL, or a LONG being divided by a REAL, would cause compile time errors.

Pasting text onto the first line of a SUB program is no longer possible. This prevents users from inadvertently altering the SUB definition in a way that would cause errors.

Printing a program that contains the SYSTEM$ command, lines of code that contain this syntax are no longer omitted.

Program lines that exceed the printer width now print on multiple lines. Previously when printing out a program line that extended off the page, the line cut off instead of wrapping around to the next line.

Saving as "Text without line numbers" now shows the new file name in the HTBasic header immediately after re-opening.

Saving as Text without line numbers no longer causes all !'s that appear at the end of a line to shift position when reopening the file.

Serial driver using the WAIT statement now works properly when used after the ON TIMEOUT statement.

The Alt+L keyboard shortcut now accesses the Tools menu.

The Alt+T keyboard shortcut now switches between sub and full program listing only. Previously the Tools menu was activated using <Alt>+T as well.
Editor now scrolls correctly after setting a bookmark with CTRL+F9 and then accessing the bookmark using CTRL+F10.

TIMEDATE functions now is dynamically updated from the system time if the system time changes during HTBasic execution.

XY WIDGET attribute BIG NUMBERING is not a supported attribute. They syntax is supported for porting functionality only.

Changes From 9.2 to 9.3

The WIN-PRINT driver now allows color printing. This may be turned on/off by checking/Unchecking the "Color Printing using WIN-PRINT Driver" box under the Options|Run Environment menu, or by using the CONFIGURE SYSTEM("WINPRINT COLOR ON") or CONFIGURE SYSTEM("WINPRINT COLOR OFF") statements. By default, when doing color printing using the WIN-PRINT driver, all white text will be inverted to black. If this is not desired, the add the INVERT option to the above statement. CONFIGURE SYSTEM("WINPRINT COLOR ON;INVERT")

Comment and UnComment are now available from the right click context menu in the HTBasic for Windows Editor.

Basic Plus Widgets may now be dumped to a GIF file using the GIF driver, after having enabled Basic Plus dumping by using either the CONFIGURE SYSTEM("DUMP;PLUS") statement, or by checking the Dump Plus checkbox under the Options|Run Environment menu.

Custom baud rates for the serial driver may now be entered using the "CONTROL 9,13;baudrate" statement.

Specifying array elements is now allowed in the Debug Watch Window. The array elements should be specified the same as they would be in a program. e.g. Intarray(3,2) Note: All declared elements of the array must be specified.

Array elements may now be specified in the Conditional Breakpoints and Global Breakpoints dialogs. After selecting the appropriate variable from the dropdown list, specify the correct element using the following syntax: (2,3)

The Numeric Compiler now allows file paths to be longer than 80 characters.

Clicking on Ascii files with the .BAS extension will now launch HTBasic, and GET the program. Previously only HTBasic Prog files had this feature.

Long Hex constants are now supported.

Deleting lines that contain breakpoints will now correctly remove the breakpoints and the lines. Previously this would sometimes cause HTBasic to close.

Event Handlers are no longer disabled when using the Autosave feature.

Dumping to a GIF file will now allow the BW and EXPANDED options to be used together, producing correct results.

Combining Dump Graphics and Printing using PPA printers will now correctly print each page. Previously, after printing and doing a dump graphics using PPA printers, no more Printing or Dumping would correctly spool to the printer.

A Pause character followed by a Step character using CHR$(255)&"P"&CHR$(255)&"S" will now correctly execute the pause followed by the step. Previously, this caused an error.

The Numeric Compiler now supports the combination of LONGs and INTEGERs in numeric expressions.

Doing a STORE AS or SAVE AS from the File menu now consistently checks for duplicate file names before performing the Store or Save Operation.

Printing a program larger than 2 pages using File|Print Program now correctly prints the entire program. Previously it would print a couple of pages and then close.

The GPIO600 and GPIO650 Drivers now correctly support the ON TIMEOUT statement.

Print program now correctly calculates the number of lines being printed on each page, given a page size. Previously when selecting A4, or any other non-default page size, lines would not be correctly calculated, and sometimes printed off the page.

Debugging through code that uses Loadsubs and Delsubs is now supported. Previously after debugging through the first Loadsub and Delsub, all subsequent Loadsubs would cause HTBasic to freeze.

The Debugger code window's current line indicator now correctly updates after a Loadsub statement. Previously the current line indicator would remain on the loadsub statement, and a new current line indicator would be created.

Continuous Signal Events from a DLL will no longer cause stack corruption.

Japanese Kanji characters are now correctly supported when using the Label statement.