6

Understanding the File System

In this Chapter...

3070 Reference

3070 User and Service manuals are located on 3070 system controllers and on factory-supplied updates.

The Root Directory Environment Variable

This section contains:

Introduction

3070 systems are now available with a choice of these two operating systems:

- HP-UX
- MS Windows®

3070 application file path usage is different between the two operating systems.

The $AGILENT3070_ROOT Environment Variable

Beginning with software revision 3070 04.00pb 0501 WN, an environment variable is used to allow 3070 board files to be easily transferred between 3070 systems running either MS Windows or UNIX.

The environment variable is **$AGILENT3070_ROOT**. It replaces the root directory path (upper path names) on both operating systems.

All subdirectories under `/opt/hp3070/../..` will exist on UNIX systems for the forseeable future.

NOTE

`/opt/hp3070` is replaced by **$AGILENT3070_ROOT** on all MS Windows systems.

$AGILENT3070_ROOT on UNIX

On UNIX systems, the value of **$AGILENT3070_ROOT** is `/var/hp3070`

Identify the Value of $AGILENT3070_ROOT

To identify the value of **$AGILENT3070_ROOT** on a UNIX or Windows system, at a shell window prompt enter:

- `echo $AGILENT3070_ROOT`

New File Path Usage in a UNIX terminal Window

Table 6-1 illustrates new path equivalents using the system config file when working in a UNIX terminal window.

Table 6-1 New file path usage in a UNIX shell window

Before Software Release 3070 04.00 pa	At and After Software Release 3070 04.00 pa
`/var/hp3070/diagnostics/th1/config`	`$AGILENT3070_ROOT/diagnostics/th1/config`
`/hp3070/diagnostics/th1/config`	`$AGILENT3070_ROOT/diagnostics/th1/config`

File Path Usage in BT-BASIC Window

Table 6-2 illustrates new path equivalents using the system config file when working in a **BT-BASIC** window.

NOTE

BT-BASIC usage is the same in both UNIX and MS Windows.

Table 6-2 File path usage in a BT-BASIC window

Pre 3070 Software Release 3070 04.00pa	3070 Software Release 3070 05.00p	At and After 3070 Software Release 3070 04.00pa
`msi "D:/Agilent3070/diagnostics/th1"`	`msi "C:/Agilent3070/diagnostics/th1"`	`msi btgetenv$ ("AGILENT3070_ROOT") & "/diagnostics/th1"`
`get "D:/Agilent3070/diagnostics/th1/config"`	`get "C:/Agilent3070/diagnostics/th1/config"`	`get btgetenv$ ("AGILENT3070_ROOT") & "/diagnostics/th1/config"`

NOTE

The `btgetenv$ ("AGILENT3070_ROOT") &` is only required for BT-BASIC commands which are referenced to the root.

If the text does a BT-BASIC `msi btgetenv$ ("AGILENT3070_ROOT") & <command>` prior to the next BT-BASIC command (for example `compile` or `faon`), then using the environment variable which defines the path from the root is unnecessary. BT-BASIC commands which normally contain paths (`msi`, `load`, `copy`, `save`, `get`, `store`, `unlink`, `rcall`) for example, will require `btgetenv$ ("AGILENT3070_ROOT") & <rest of path>`

$AGILENT3070_ROOT on MS Windows

On MS Windows systems, the factory default value of **$AGILENT3070_ROOT** is `C:/Agilent3070`

Identify the Value of $AGILENT3070_ROOT

To identify the value of **$AGILENT3070_ROOT** on a MS Windows system,

at a shell window prompt enter:

- `echo $AGILENT3070_ROOT`

File Path Usage in a MS Windows Korn Shell Window

When working in a **Korn shell** window, follow the UNIX syntax by:

- Using `$variable` (instead of `%variable%`).
- Using the correct case.
- Using `/` (forward slash) instead of `\` (backslash)

Table 6-3 illustrates new path equivalents when working in a MS Windows **Korn shell** window.

Table 6-3 New file path usage in a MS Windows **Korn shell** window

Pre 3070 Software Release 3070 04.00pa	3070 Software Release 3070 05.00p	At and After 3070 Software Release 3070 04.00pb 0501 WN
`D:/Agilent3070/diagnostics/th1`	`C:/Agilent3070/diagnostics/th1`	`$AGILENT3070_ROOT/diagnostics/th1`

New File Path Usage in a MS-DOS Command Prompt Window

When working in a **Command Prompt** window:

- Use `%variable%` (instead of `$variable`).
- Use \ (backslash) instead of / (forward slash).

Table 6-4 on page 6-6 illustrates new path equivalents using the `dev` directory when working in a **Command Prompt** window.

Table 6-4 New file path usage in a **MS-DOS Command Prompt** window

Pre 3070 Software Release 3070 04.00pa	At and After 3070 Software Release3070 04.00pb 0501 WN
`D:\Agilent3070\dev`	`%AGILENT3070_ROOT%\dev`

NOTE

In MS Windows® 2000 Professional, the **MS-DOS** window is now the **Command Prompt** window. To open the **Command Prompt**: point to **Start**, then **Programs**, then **Accessories,** and choose **Command Prompt.**

The .hp3070 File

The system first searches the current working directory for a `.hp3070` file. If one is not found, the user's home directory is searched. This method allows a `.hp3070` file for each board.

The `.hp3070` file can affect system behavior in many ways.

Some Descriptions of .hp3070 File Keywords

Some descriptions of `.hp3070` file keywords are given in Table 6-5. This is not a complete description of this file. Other options are described beneath appropriate topics throughout the 3070 User documentation.

Table 6-5 Some descriptions of `.hp3070` file keywords

keyword	Description
`.BackupLevel`	The value of this option sets the global backup style for this user and determines whether the system compilers keep an unchanged copy (a backup) of files before modifying them, and how the backup is stored. The backup style can be: **none** – No file backup is made. **numbered** – Multiple backups are made as files change. To identify a numbered backup file, its name has a period, a tilde (~), and a unique number from 1 to 9 appended to its name; for example, `file.1~`. Number 1 is the most recent backup, and number 9 is the oldest. When more than 9 backups occur, the oldest backup file in the set is discarded and those remaining are renumbered. **unnumbered** – A single backup is made as files change. Each new backup file overwrites the contents of the previous backup file. To identify an unnumbered backup file, a tilde (~) is appended to its name; for example, `file~`. For example, `.BackupLevel: unnumbered` Besides the global value for backup style, you can individually specify a backup style for some of the software modules in your system. For example, `Mpa.BackupLevel: numbered` overrides the global default and sets the backup style for the pin assignment software to **numbered**.

Table 6-5 Some descriptions of `.hp3070` file keywords (continued)

keyword	Description
`.ProgramAction`	The value of this option determines whether a new window is automatically opened when some commands are executed (such as `execute` – see **Syntax Reference**. `.ProgramAction` can be either `window` (a new window is automatically opened) or `nowindow` (a new window is not automatically opened). For example, `.ProgramAction: nowindow`
`Debug.Source`	The value of this option determines whether the Agilent Pushbutton Debug environment is automatically invoked when a `debug` statement is executed on the BT-BASIC command line. The value of this option can be: **Debug.Source: no** – Use the standard debug environment by default. **Debug.Source: yes** – Use the Agilent Pushbutton Debug environment by default. For more information, see **Test Methods: Digital**.
`FXT.WIRECOLORS`	This option lets you specify user-defined wire colors for fixturing. The values following this variable are the colors that are requested in fixture building reports. This lets you customize wiring reports so they ask for colors (in any language) matching the colors of the wires being used. The first color is used for all non-ground wiring (positive and negative) and should be the local word for red. The second color is used for all ground wiring and should be the local word for black. The remaining colors are used in sequence, one per node. The sequence of colors repeat after the last color has been used. For example, `FXT.WIRECOLORS: "red black blue green yellow aqua white"` In the example, `red` is used for all non-ground wires, and `black` is used for all ground wires. The color of wires specified for wiring nodes cycle through the list from `blue` to `white`. After `white` has been used, the sequence starts over with `blue`. For more information, see **Test and Fixture Development**.
`Operator.ForceWidgets`	This option lets you specify whether the operator keypad appears on the screen for operator logins. Specify `Yes` to have the keypad automatically appear, or `No` to have it not appear. For example, `Operator.ForceWidgets: Yes`

Table 6-5 Some descriptions of `.hp3070` file keywords (continued)

keyword	Description
`Operator.Footswitch`	This option lets you specify whether the foot switch on the Agilent 3070 is enabled. Specify `Yes` to have the foot switch enabled, or `No` to have it disabled. For example, `Operator.footswitch: Yes`

The majority of the `.hp3070` file contains definitions for the operator keypad, which are invoked by an `operator` statement if the `Operator.ForceWidgets` option is set to `Yes` (see **Syntax Reference**). When a set of labels is specified in the `operator` statement — for example, `operator waitforstart` invokes the set of labels and functions defined as `waitforstart` — that set of label definitions becomes active. The boxes in the operator keypad are labeled with those definitions, and selecting a box (with the mouse or the touchscreen) invokes the function associated with the label in that box.

If no label specifier is included in an `operator` statement, the default label definition is determined by the value of either of two variables in this file. The `Operator.Default` variable sets the default for a user who is not using a board handler with the system, and the `Operator.ABH_Default` variable sets the default for a user who is. The values of both of these variables are typically defined as `standard`.

The label and function definitions are arranged into groups that each contain three specifications:

- A physical description of the operator keypad, including the X and Y coordinates and how many boxes should appear in the keypad.
- What label (text) should appear in each box. Labels can contain any combination of upper or lowercase letters and are treated as lowercase when invoked in an `operator` statement.
- Which function is invoked by selecting a particular box. Function names are case-sensitive.

For example, `standard`, which is the default definition for the operator keypad, might look like the following:

```
Standard.Boxes:     8
Standard.X:         10
Standard.Y:         4
Standard.Columns:   8

Standard.Label1:    start
Standard.Label2:    yes
Standard.Label3:    no
Standard.Label4:    .
Standard.Label5:    faon
Standard.Label6:    faoff
Standard.Label7:    stop
Standard.Label8:    exit

Standard.Command1:       CHAR_START
Standard.Command2:       CHAR_YES
Standard.Command3:       CHAR_NO
Standard.Command4:       CHAR_NULL
Standard.Command5:       CHAR_FAON
Standard.Command6:       CHAR_FAOFF
Standard.Command7:       CHAR_STOP
Standard.Command8:       CHAR_EXIT
```

This example defines the keypad boxes which are arranged in eight columns. When the operator keypad is invoked by an `operator` statement that specifies this definition, the boxes are labeled `start`, `stop`, `yes`, `no`, `faonn`, `faoff`, and `exit`. When box #1 (`Label1` or `start`) is selected, the characters `START` are executed on the command line; `start` is the function passed to BT-BASIC.

Installing Software Packages

Introduction

The factory routinely releases new software packages that improve and add capabilities.

Some software packages depend on a previous software package being installed.

Always notify system users when a new software package is installed.

Install a Software Package

Installation instructions accompany software packages. Follow the instructions carefully.