# DataTool 3

## Overview

**DataTool** is a program operating under Microsoft **Excel** that manages the FLASH 750 test programs and provides a user interface to the test program.

A FLASH 750 test program consists of an **Excel** workbook containing various sheets of information. A workbook stores this programming information for a single program or for a family of related programs. Except for the pattern information, a workbook includes all information for program execution; patterns are managed through the **PatternTool** software. Users enter the following device information into the appropriate **DataTool** sheets:

- definitions of the device pin names and group names
- mapping of the device package pin and tester channel
- DC voltage and current levels applied to the DUT
- parametric test values for the device
- timing values and pin formats
- test flow and binning

Depending on your preferences or the test program requirements, you can create multiple variations of a sheet type within the same workbook and select combinations of sheets to create an executable test program. Consequently, you can develop and maintain a family of related programs.

FLASH 750 provides several ways to start **DataTool**; refer to *Starting DataTool* on page 3-2.

# Starting DataTool

## Overview

You can start **DataTool** by invoking it from the Windows NT Start menu or a command-line interface. You can also start **DataTool** after you open a workbook by using **Excel** or another FLASH 750 tool.

## Invoking DataTool

### Invoking DataTool from the Start Menu

To invoke **DataTool** from the *Start* menu, select *Programs*, then *Teradyne* **IG-XL.** This submenu has two options:

- *New Test Program*—creates a new workbook, containing the minimum required worksheets, with no data. After entering and editing the data in this new workbook, you must save it by selecting the *Save As* command in the **IG-XL** menu.
- *Open Existing Test Program*—opens a dialog to specify an existing test program workbook (extension *.xls*). After you have entered or edited the data in this workbook, you must save it; refer to *Managing DataTool* on page 3-4.

+ **DataTool** opens a new or existing workbook and enables its macros; **Excel** does not check for macros, you are not prompted when opening a workbook containing macros. Macros must be enabled for **DataTool**.

## Invoking DataTool from a Command-Line Interface

FLASH 750 provides a command-line interface to the *Open Existing Test Program* option on the *Start* menu:

OpenProgram.exe *path\file1* [| *path\file2*]

If you specify more than one file, separate the filenames with |. Any path can be relative.

The **Excel** process is put in the directory specified for the first file, meaning the *curdir* (current directory) for the test program is specified by the first file.

After you have entered or edited the data in this workbook, you must save it; refer to *Managing DataTool* on page 3-4.

# Managing DataTool

## Overview

**DataTool** operates in one of several modes, depending on how you load the workbook. You can open a workbook by using DataTool, Excel, or certain other FLASH 750 applications.

Toolbars and other controls are provided for managing **DataTool** and navigating in the window.

## DataTool Modes

You can open **DataTool** in one of several modes, depending on how you open the workbook:

- *Engineering* mode—If you open a workbook while running **DataTool** on the tester, **DataTool** is in the *engineering* mode. In this mode, the workbook is visible, so you can edit the workbook and execute the edited program on the tester. Also, refer to *Debugging Patterns with PatternTool* on page 3-76.
- *Production* mode—If you load a test program through an operator interface, **DataTool** is in the *production* mode. In this mode, **DataTool** is not visible; consequently, the test program workbook cannot be edited. You run the test program from the operator interface.

+ An operator interface may include a on-screen button for the *engineering* mode, in which, **DataTool** can be made visible.

- *Offline* mode—is like *engineering* mode, but **DataTool** does not run the test program on a tester. In this mode, no tester hardware is used.

See also: *Debug Mode* on page 3-73

# Loading Workbooks

## Opening and Saving Workbooks

Once **DataTool** has started, open an existing **DataTool** workbook by selecting the *Open* command on the *File* menu. The workbook is examined for macros.

+ **DataTool** opens a new or existing workbook and enables its macros; **Excel** does not check for macros, and you are not prompted when opening a workbook containing macros. Macros must be enabled for **DataTool**.

After editing a workbook, you can save it by selecting the *Save* command on the *File* menu or by clicking the *Save*  icon on the **Excel** standard toolbar.

To rename a workbook, select the *SaveAs* command on the *File* menu. In the dialog box that appears, enter the new filename and then click *OK*.

To import a workbook task that has been saved or created in an ASCII format; refer to *Importing and Exporting Worksheets in ASCII Format* on page 3-18.

## Opening a Workbook through Excel

You open a **DataTool** workbook through **Excel** by selecting the *Open* command on the *File* menu or by clicking the *Open*  icon on the **Excel** toolbar. Upon opening the workbook, **DataTool** checks the workbook for macros. You are prompted to enable the macros. **DataTool** requires enabled macros. After the workbook opens, click *Start Datatool* button on the **IG-XL** menu to enable **DataTool** functions.

## Opening a Workbook from FLASH 750 Applications

**DataTool** and another FLASH 750 application, such as the **Pattern Compiler**, may both access a workbook. For example, the **Pattern Compiler** opens a workbook to access the *Pin Map* sheet when it compiles a pattern file.

If more than one FLASH 750 tool accesses a worksheet, and you edit the workbook, such as adding a pin to the Pin Map, the change becomes visible to the other application only *after* the workbook file is saved. In this example, if you add a pin to the *Pin Map* sheet, but do not save the workbook file, and then change the ASCII pattern file to use the new pin, the **Pattern Compiler** will fail, because it cannot recognize the new pin until you have saved the workbook.

# DataTool Window

## Overview

**DataTool**, a program operating under Microsoft **Excel**, manages the FLASH 750 test programs and provides a user interface to the test program. Thus, **DataTool** looks and behaves like **Excel** with the following exceptions or additions:

- **DataTool** adds two special **IG-XL** toolbars to the **Excel** toolbars; refer to *DataTool Toolbars* on page 3-6 and *IG-XL Context Toolbar* on page 3-9.
- To navigate to a specific worksheet by using the **DataTool** window tabs or the *Home* sheet; refer to *Navigating Among the Worksheets* on page 3-12.
- To interpret how certain aspects of the FLASH 750 worksheets are displayed, such as bolded or hatched cells; refer to *Fundamentals of DataTool Worksheets* on page 3-11.

## DataTool Toolbars

The useful **Excel** Formula Bar should remain visible; refer to *Formula Bar* on page 3-10.

Two toolbars are added to the **Excel** toolbar area when the **DataTool** is invoked:

- *IG-XL Toolbar* on page 3-7
- *IG-XL Context Toolbar* on page 3-9

+ If both these toolbars are not visible, be sure that under the *View* menu, *Toolbars* submenu, **IG-XL** *Toolbar* and **IG-XL** *Context*, are both checked.

### *IG-XL Toolbar*

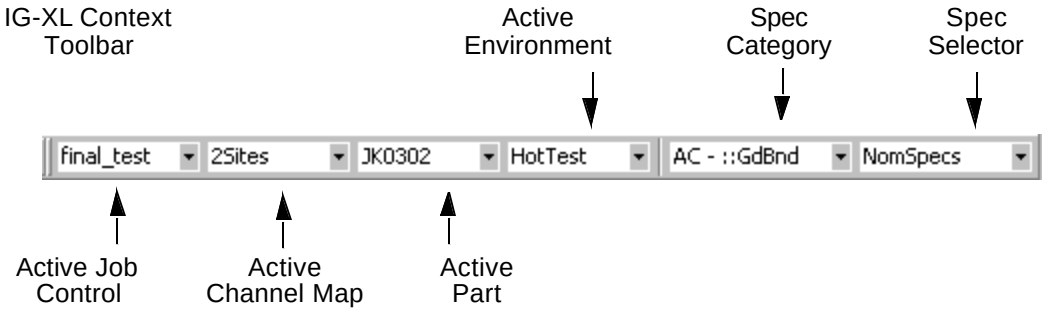
The **IG-XL** Toolbar contains buttons for executing commands that control or manage **DataTool** or a test program workbook, see the figure below. The commands executed by these buttons are also available on the *IG-XL* menu.

- *Start DataTool* —Stops (if necessary) and restarts **DataTool**.
- *Stop DataTool* —Stops **DataTool** without exiting from **Excel**.
- *Home Sheet* —Makes the *Home* sheet of **DataTool** the active sheet.
- *Formula* —Toggles between displaying the **DataTool** formulas or the evaluated values.
- *Validate Job* —Validates the current **DataTool** job; required before execution.
- *Run* —Runs the current job in a test program.
- *Run Options* —Opens a dialog for selecting the execution options for the test program.
- *Output Window* —Opens the *Test Program Output* window to display the execution results of the test program.
- *DataCollect Setup* —Opens a dialog for setting up a datalog or summary report produced while data is collected from an executing test program.
- *PatternTool* —Opens **PatternTool**, for displaying and editing a binary pattern file, or viewing the HRAM.
- *Pattern Compiler* —Opens the **Pattern Compiler**, for compiling an ASCII source file into a binary pattern file.
- *Debug Displays* —Opens the **IG-XL Display Manager**, for selecting the GUI debug tools to view and to edit the tester hardware settings.
- *Calibrate TDR* —Calibrates the tester timing by correcting for the time delays in the interface hardware.

- *Debug Run*  —Runs the current job in the *debug* mode.
- *Debug Stop*  —Stops execution of the current job while in the *debug* mode.
- *Toggle Breakpoint*  —Sets or removes the breakpoint on the current flow step.
- *Step*  —Executes the next test or part of the test instance.
- *Step Over*  —Executes the test instance as a unit.
- *Run DUT Characterization*  —Opens the Characterization Editor to characterize the current test instance.
- *Help*  —Displays the online help on the current active column or sheet.

### IG-XL Context Toolbar

The **IG-XL** *Context* toolbar provides a pulldown menus for selecting the execution context and the specification context for a test program, see the figure below.




The first four pulldown menus are for selecting the execution context:

- *Active Job*
- *Active Channel Map*
- *Active Part*
- *Active Environment*

Use these controls to select which sheets and which flow steps to include in the test program before it is validated and executed; refer to *Debugging a Program* on page 3-72.

Use the last two controls to edit or view an *AC Specs* or *DC Specs* sheet while you are creating or debugging a test program:

- *Spec Category*
- *Spec Selector*

These two controls select the current value of the spec symbols on the *AC Specs* and *DC Specs* sheets; refer to *Fundamentals of AC Specs and DC Specs Sheets* on page 3-41.

### Formula Bar

The **Excel** *Formula Bar* is useful tool. To view it, select *Formula Bar* on the **Excel** *View* menu is checked; see the following figure:


The right side of the *Formula Bar* is an edit box that shows the contents of the current cell, in formula format. You can also edit the formula in this box, as an alternative to editing the cell on the sheet; refer to *Using Formulas* on page 3-32.

The left side of the *Formula Bar* is the *Name Box*. Usually it shows the location of the current cell: column letter and row number. Its drop-down list contains the names of all spec symbols defined in the workbook. If you select a spec symbol name, you go to the sheet and cell where the symbol value is assigned; refer to *Formulas Using Spec Symbols* on page 3-201.

For more information about the *Formula Bar*, refer to the **Excel** online documentation.

## IG-XL Display Manager

The **IG-XL Display Manager** lets you open displays to view and change hardware settings for debugging. This tool is useful during debugging, allowing you to start a debug display at a breakpoint; refer to *Debug Displays*, Chapter 5.*Debug Displays* in *IG-XL Help*.

To start the **IG-XL Display Manager**, click the *Debug Displays* button on the **IG-XL** *Context* toolbar or select the *Debug Display* command from the **IG-XL** menu. Also, By invoking **DataTool** in the *engineering* mode with the **Windows NT** *Start* menu, the **IG-XL Display Manager**, also known as the *Debug Display Manager*, is started and minimized into an icon. By starting the **IG-XL Display Manager** at the same time as **DataTool**, you can start a debug display at a breakpoint.

# Fundamentals of DataTool Worksheets

## Overview

**DataTool** worksheets look nearly identical like the standard **Excel** worksheet; however, the operation and the appearance of a FLASH 750 worksheet has been altered. The following additions and changes to the look and feel of the worksheet improve the efficiency and maximize the ease in developing and maintaining a FLASH 750 test program:

*Worksheets*



*Worksheet Data*

*Test Programs*



# Navigating Among the Worksheets

## Overview

You select a sheet by clicking on its tab at the bottom of the window.

If you don't see the tab, use the tab scrolling buttons, which are to the left of the tabs. The two outer buttons move the display to the far left or far right, while the two inner buttons move the displays one tab to the left or right. When you see the desired tab becomes visible, click on it.

If the workbook contains many worksheets, you can right-click on the tab scrolling buttons to display the list of sheet names. Select the desired sheet from the list by clicking on it.

You can also navigate to a specific sheet from the *Home* sheet. Click on the *Home Sheet* button on the **IG-XL** *Toolbar* to go to the *Home* sheet; then click on the link for the desired sheet name.

# Worksheet Conventions

## Guidelines for Sheet Names

- Worksheet name can contain any character except:

  comma (,)

  left and right curly braces ({ and })

  at symbol (@)

  IG-XL strips these characters from the sheet name.

- Sheet name can contain embedded blanks.
- Sheet names must be unique across the workbook. If you specify a sheet name that already exists, a number is appended to the new sheet name to make it unique.
- Sheet names should be kept as short as possible to minimize the tab width, to ensure successful importing and exporting of worksheets among different operating systems, and to ensure drop-down lists.

  Sheet names sometimes appear in drop-down lists on other sheets: for example, on the Test Instances sheet, names of spec sheets and timing sheets are available on drop-down lists. However, if the number of characters in a list exceeds an **Excel** limit, the drop-down list is disabled, and you must enter sheet names manually. For this reason, you should keep sheet names short.

## Worksheet Row/Column Limits

- Columns—254 (**Excel** restriction)
- Rows—65530 (Excel restriction)

## Appearance of the Data on the Worksheet

**DataTool** uses graphical elements, such as color, hatching, font to convey information about the data on the worksheet.

### Bold Text

A cell value in bold is the name of a group, a pin group, or a pattern group. When you enter the name of a group, **DataTool** bolds the name, both on the sheet defining the group and on any other sheet referencing the group.

On some sheets, bold text may have another meaning: for example, a bold job name on the *Job List* sheet indicates the current *Active Job*.

### Grayed Text

Grayed text signifies a default value or a value that is not applicable. In addition, grayed text is used on the second and subsequent lines of group definitions or other grouped items. For example, on the *Pin Levels* sheet, where the levels definitions for each pin are grouped together, the first line lists the pin name in regular text, while subsequent lines are grayed text.

### Column Headings—Required or Optional

The appearance of the column heading signifies whether the column is required or optional:

- Required—White text on dark green background. You must enter a value in this column.
- Optional—Black text on light gray background. You may leave this column blank.

White text on a black background identifies the first line of a two-line header. The second line indicates whether the column is required or optional.

An optional column may require a value, based on the value in another column. For example, on the *Flow Table* sheet, certain *Group Condition* values require a value in the *Group Name* column. Consequently, this type of column is not considered required; it is usually left blank. A column that always requires a value has white text on a dark background.

### Hatched (Shaded) Cells

A hatched (shaded) cell signifies the cell does not accept entered data. Examples:

- Group definition—If a column applies to the group as a whole (for example, *Type* for pin groups), the column is enabled only for the first line, and the remaining lines are shaded.
- Read-only value—On the *DC Specs* and *AC Specs* sheets, the current value of a symbol is calculated, based on the current category and selector, and is displayed as shaded read-only text. You cannot directly edit the value.
- Disabled cells—On some sheets, columns are interdependent: a certain value entered in one column can cause other columns to become either required or invalid. When you enter such a value, **DataTool** shades the invalid columns; valid columns are not shaded.

# Inserting a Worksheet

A workbook contains only one of each sheet type, unless a Job List sheet defines multiple jobs. **DataTool**, however, does not prevent you from inserting multiple copies of one type of sheet, except for the *Job List* sheet, in a workbook, even if you have not defined multiple jobs. **DataTool** checks for these restrictions only when it validates a test program. As noted, **DataTool** does prevent you from inserting multiple copies of the *Job List* sheet: only one is permitted per workbook.

You can import a worksheet into a workbook from an ASCII file; refer to *Importing and Exporting Worksheets in ASCII Format* on page 3-18.

To insert a new worksheet, with column headers but no data:

1. On the *Insert* menu, select *Worksheet*. An *Insert Worksheet* window appears.
2. From the *Sheet Type* drop-down list, select the type of sheet to be inserted. Most of the types are for **DataTool** sheets. You can also insert either an **Excel** Worksheet (empty spreadsheet) or an **Excel** Chart (blank chart area).
3. In the *Sheet Name* area, enter a name for the sheet; refer to *Guidelines for Sheet Names* on page 3-13. If you do not enter a name, a default name is used. You can rename the sheet later; refer to *Renaming a Worksheet* on page 3-17.
4. Some sheets require additional information. For example:
    a. If you insert a *Channel Map* sheet, you are prompted to enter the number of sites number of sites per station, how to assign the pins of a site, and whether to fill the sheet with the pin map information.
    b. If you insert a *Time Sets* sheet, *Time Sets (Basic)* sheet, or *Edge Sets* sheet, you are prompted to select either *Normal Timing* (100 MHz) or *Extended Timing* (50 MHz).
    c. If you insert a *DC Specs* or *AC Specs* sheet, you are prompted to enter the initial number of categories to be created.
5. After inserting or importing a worksheet, you can move it within the workbook by left-clicking on the sheet tab and dragging it along the row of sheet tabs to the desired location.

## Renaming a Worksheet

To rename a sheet, double-click on the sheet name on the worksheet tab and enter the new name. Or, right-click on the tab, select *Rename* from the menu, and then enter the new name.

When renaming a sheet, follow the *Inserting a Worksheet* on page 3-16.

## Annotating a Worksheet Title

Each sheet has a title in the bar in the first row across the top of the data area. The title describes the type of sheet—for example, *Pin Map* or *DC Specs*—rather than the user-supplied name, which appears on the tab.

Teradyne recommends that you not change this title; instead, you can add text in the right half of the title bar:

1. Click in the right half of the title bar.
2. If the selected area does not contain the sheet title, enter the desired text in this area.
3. If the selected area does contain the sheet title, click again further to the right, and then enter the desired text.

## Deleting a Worksheet

**DataTool** does not check whether the sheet you are deleting is referenced by any other sheet. These deleted references are not detected until the test program is validated; refer to *Validating the Test Program* on page 3-46.

To delete a sheet:

- Select the sheet by clicking on the tab. On the *Edit* menu, select *Delete Sheet.* You are prompted to confirm the deletion.
- You can also right-click on a sheet tab and select *Delete* from the menu.

# Importing and Exporting Worksheets in ASCII Format

## Overview

**DataTool** can import worksheet data from an ASCII file, and export a worksheet to an ASCII file. The ASCII file consists of tab-delimited lines and must conform to a **DataTool** worksheet ASCII format.

### Exporting Worksheets

You can export a single worksheet or an entire workbook by selecting the following command on the *File* menu: *Export Worksheet* | *Export Workbook.*

When you export a worksheet or workbook, the standard Windows file browser opens and prompts you to specify a folder as the export location. The ASCII file is named *sheetname.txt*.

Exporting a workbook creates a separate ASCII file for each worksheet in the workbook. Exporting a worksheet exports the active sheet.

### Importing Worksheets

You can import a single worksheet or an entire workbook by selecting the following command on the *File* menu: *Import Worksheet* | *Import Workbook.*

When you import a worksheet or workbook, the standard Windows file browser, prompting you to specify the folder and text file to be imported. For a worksheet, you can specify only one file; for a workbook, you can specify multiple files, one for each sheet in the workbook. The files must have the *.txt* extension. Each new worksheet will have the name of the source file, minus the *.txt*.

When you import a new sheet, **DataTool** will check for some errors, based on the format it expects to find in the ASCII file. Other errors are discovered during validation.

+ Even though you can use the **Excel** commands *Open* and *Save As* to import from and write to an ASCII file, Teradyne recommends that you use the **DataTool** commands. The **Excel** command *Open* can read only one file at a time; *Save As* changes the name of the active workbook.

## Format

File format is tab-delimited ASCII text. Both imported and exported ASCII files have same format specified in this section. For the details of specific worksheets, such as column order, name, and type, follow the format of each individual sheet type.

+ The easiest way to determine the format for any worksheet is to export the worksheet to an ASCII file, and then to use a text editor to examine the file.

### Delimiters and Terminators

- Each column is delimited by a tab. The last column in a row must also be followed by a tab.
- Each row of data is terminated with a carriage return followed by a line feed. Because the last value in a row must be followed by a tab, each line must end with the sequence *<tab><cr><lf>*.
- Two tabs mean the column is empty for that row.

### Column 0

The first column of the worksheet data area begins after the first tab. Any value before the first tab in a line is in column known as *column 0*, which is not part of the data area of the worksheet. In the ASCII file, only row 1 has a defined value in column 0. In the remaining rows of the data area, column 0 is blank.

### Row Format

- row 1, column 0

  The string *DFF v.v,* identifies the **DataTool** File Format version. If you creating an ASCII worksheet file with an editor, this cell can be blank; **DataTool** assumes that it is the current format.

- row 1, column 1

  The sheet type as it appears at the top of the worksheet, such as *Pin Map* or *Time Sets (Basic)*. This is the standard sheet type, not the user-definable sheet name.

  Some worksheets have additional information in this row. For example, *Time Sets (Basic)* and *Edge Sets* sheets have the string *Normal* or *Extended* in a column of row 1. For these sheets, Teradyne recommends that you verify that the string appears in the correct column of the exported file.

- Row 2 must be blank.

- Row 3 may have a special field in columns 1 and 2. For example, the *Channel Map* sheet has *DIB ID:* followed by a value; the timing sheets have *Timing Mode:* followed by *Normal* or *Extended*. In this case, row 4 is blank and the column headings begin in line 5. Otherwise, they begin in row 3, as described in the next paragraph.
- Row 3 and row 4, if necessary

  On most sheets, column headers occupy the next one or two lines.

  Some worksheets, like the *Pin Map*, have one-row headers. In these cases, the headers occupy row 3, starting with column 1 (after the first tab), and the headers are separated by a tab.

  Other worksheets, like the *Flow Table*, have two-row headers; refer to *Two-Row Headings* on page 3-21.

  Some sheets contain a field between the sheet title and the header rows. For example, *Time Sets (Basic)* has a *Timing Mode* field. These fields occupy row 3, with a blank row before and after. The column headers on such sheets begin in row 5.
- Remaining rows

  Data for the rows begins in column 1 (after the first tab). Two tabs indicate a column with no value.

  Do not include blank lines. A blank line marks the end of the data area. Anything after a blank line may be imported, but it will not be formatted as part of the worksheet data area.

  Enter data as you would in the workbook. Do not enclose strings by quotes. You can use formulas based on spec variables.

### Two-Row Headings

Two-row headings are unique. The first row can be considered a group heading, extending over two or more columns. The first row of each such heading must be in the same column as the first column in that group. You must insert the required number of tabs in the first heading row to keep the columns properly matched. Teradyne recommends that you export an existing worksheet and use the header rows from the ASCII file.

For example, on the *Time Sets (Basic)* sheet, the first column, *Time Set*, is a single-row heading, which would appear on the second heading row. The second and third columns (*Period* and *CPP*) are under the *Cycle* group heading, which must appear in the second column; thus, *Cycle* is preceded by two tabs. The fourth and fifth columns (*Name* and *Setup*) are under the *Pin/Group* group heading, which must be in the fourth column; two tabs separate it from *Cycle*. Using this example, rows 5 and 6 appear like this:

<tab><tab>Cycle<tab><tab>Pin/Group<tab>...

<tab>Time Set<tab>Period<tab>CPP<tab>Name<tab>Setup<tab> ...

# Entering and Editing Data

## Overview

**DataTool** is used mainly for entering data or editing existing data in a worksheet.

## Entering Data

Entered data can be grouped into the following categories:

- Entering user-defined names—such as, pins, test instances, and edge sets; refer to *Rules for User-Created Names* on page 3-25.
- Entering numeric values—such as, timing or voltage levels. These values are usually based on the variables defined on the *AC Specs* and *DC Specs* sheets. Teradyne recommends using formulas; refer to *Using Formulas* on page 3-32.

+ **DataTool** accepts engineering units added to numeric values; refer to *Engineering Units in Formulas* on page 3-34.

- Choosing values from a drop-down list—Values restricted to a predefined set of values, such as, drive formats or pin types, the valid column values are available on a drop-down list.

  If a column refers to other user-defined items within the workbook, those items are available on the drop-down list. For example, on the *Test Instances* sheet, you specify the *Time Sets* for each test instance; the names of all currently defined *Times Sets* and *Time Sets (Basic)* sheets appear on the drop-down list.

  When you select one of these cells, a button with a down-arrow appears. Click on the button to display the list; then, select an item.

+ The number of characters in the drop-down list is limited because of an **Excel** restriction. If the limit is exceeded, the drop-down list is disabled; you will have to enter the values manually. However, if you reduce the number of characters in the list, the drop-down list is re-enabled. Also, the limit can be exceeded is if the sheet names are long.

+ **DataTool** checks for data entry errors when you try to exit from a cell after you enter data in the cell; refer to *Data Entry Errors* on page 3-26.

## Editing a Cell

+ When editing a cell, following the *Editing Restrictions and Rules* on page 3-24.

To edit a cell, first, select it by clicking on it or navigating to it with the arrow keys. After selecting the cell, you have the following options:

- To find text, use the *Find* command; refer to *Finding Data* on page 3-27.
- To replace text found by using the *Find* command, use the *Replace* command; refer to *Replacing Data* on page 3-29.
- To replace the contents of the current cell, enter the desired value.
- To edit the contents of the current cell, double-click on the cell; or, with the cell selected, press the F2 key. The edit cursor appears in the cell.
- If the cell has a drop-down list, selecting an item from the list replaces any current value.
- To edit the contents of the current cell by using the edit area in the **Excel** formula bar; refer to *Formula Bar* on page 3-10.

+ **DataTool** checks for data entry errors when you try to exit from a cell after editing it; refer to *Data Entry Errors* on page 3-26.

## Editing Restrictions and Rules

### Overview

Obey the following column and blank row restrictions and rules for editing only the data area of the worksheet, which are the rows and columns containing data.

**Excel** does not usually prevent you from violating the editing restrictions and rules; however, the worksheet may be corrupted, making it unreadable by **DataTool**.

### Column Restrictions

- Positions of the predefined columns within the data area cannot be changed.
- Any predefined columns from the data area cannot be deleted.
- New columns within the data area cannot be inserted.
- Predefined column names are reserved and cannot be changed, except for the category name on the *DC Specs* and *AC Specs* sheets. When a category is created, it has a default name; however, you can directly edit the column header to change the name or to add an environment string.

### Blank Row Restrictions

- A blank row signifies the end of the data area. If you insert a blank row into the middle of existing data, any rows after the blank row are ignored. Deleting the blank row causes the following rows to rejoin the data area.
- If you enter data into a blank row, the following rows become part of the data area.

You can use this restriction during debugging. For example, to execute only the first part of the *Flow Table* during debugging, insert a blank row to truncate the table. When you delete the blank row, the *Flow Table* is complete again.

### Editing Rules

- Standard spreadsheet values, such as user formulas, calculations, and notes, can be added in the areas to the right of and below the data area.
- If information is added to the right of or below the data area, it must be separated from the data area by at least one blank row or column; otherwise, the **DataTool** interprets it as part of the data area; refer to *Blank Row Restrictions* on page 3-24.
- To add text to the sheet title bar in the first row above the data area, refer to *Annotating a Worksheet Title* on page 3-17.

## Rules for User-Created Names

### Overview

Most of the items in a test program require a name: pins and pin groups, time and edge sets, spec variables, test instances, pattern sets and groups, and characterization setups. The names you enter must meet the following rules:

- *Character Set* on page 3-25
- *Case Sensitivity* on page 3-25
- *Scope* on page 3-25

+ Worksheets have different naming requirements; refer to *Inserting a Worksheet* on page 3-16.

### Character Set

- Names can use ASCII alphabetic characters, upper or lower case, as well as digits and underscores ( _ ).
- First character of the name must be a letter. It cannot be a digit or underscore.
- Embedded spaces are not allowed.
- Names can be long, but Teradyne recommends names 12 characters or less.

Text in a *Comment* column can use any character set, including embedded spaces.

### Case Sensitivity

**Excel** is case insensitive, but it enforces consistency. Names retain the case they were defined with; thus, any name reference is converted to its defined form, ensuring consistency. For example, if *cs* is a pin defined on the *Pin Map* sheet, and *CS* is entered on the *Channel Map* sheet, **Excel** converts the *CS* to *cs*. Thus, if you redefine *cs* as *Cs* on the *Pin Map* sheet, the *Channel Map* sheet reference is updated to *Cs*. The name may not be converted immediately, but it is converted when the job is restarted or the next job is loaded.

### Scope

The scope of any name is the entire workbook. Any other worksheet can reference a pin name. A symbol specific to a sheet is not allowed.

## Data Entry Errors

**DataTool** checks for data entry errors when you try to exit from the active cell after entering or editing data. For example, if the cell has a drop-down list of valid values, and you type in a value that isn't on the list, **DataTool** does not allow you to exit from the cell until you selected a valid value.

Likewise, **Excel** can discover some numeric mistakes during data entry, such as illegal calculations or references to undefined variables. When you enter a formula that contains an error, the cell will display (in the non-formula mode) a special code, such as #NAME? if the equation uses an undefined value or #DIV/0! if the equation causes a divide by zero.

In most cases, any non-numeric errors you have entered are not caught until you explicitly validate the test program; refer to *Validating the Test Program* on page 3-46.

## Using the Excel Feature Versus the DataTool Feature

Even though the **Excel** and **DataTool** user interfaces have common features, Teradyne recommends that you use the **DataTool** feature when **DataTool** seems to duplicate an **Excel** feature. The **DataTool** feature is preferred because **DataTool** formats the worksheets differently than **Excel** does; consequently, the formatting of a worksheet could be unpredictable if you used an **Excel** feature rather than the **DataTool** feature. For example, the **DataTool** feature for hiding and exposing columns might seem redundant with the **Excel** *Hide* and *Unhide* commands; however, the **Excel** commands interfere with the **DataTool** formatting. Consequently, these **Excel** commands should not be used.

On the other hand, knowledgeable of the many built-in **Excel** editing features can decrease the time to enter and edit data and result in fewer data entry errors. For example, knowledge of the **Excel** autofill feature can help you create the pins in a data bus; refer to *Using Excel AutoFill* on page 3-97 and *Excel AutoFilter* on page 3-37.

# Finding Data

## Overview

**DataTool** can search and locate text in a worksheet; refer to *Search Rules* on page 3-27. To open the *Find* dialog, select the *Find* command on the *Edit* menu or use the keyboard shortcut: Ctrl + F; refer to *Find Dialog Box* on page 3-27.

## Search Rules

**DataTool** uses the following search rules:

- Searches are case-insensitive.
- Search string cannot contain wildcard characters.
- A substring of the cell data, not just the whole cell will produce a match.
- Text in hidden cells or rows are searched; however, matches in a hidden cell are not always exposed; refer to *Hiding and Exposing Columns* on page 3-36.
- If the found text is a hypertext link, such as, a test name on the *Home* sheet, the selected cell is the first non-link cell to the right of the actual matching cell. In this case, clicking *Find Next* on the *Find* dialog may cause the same text to be found again.
- If a search is successful, the dialog box closes and the cell containing the found text is selected. If no match is found, a message is displayed.

## Find Dialog Box

Use this dialog box to set any options and to enter the string to be located:

1. Enter the text string to be searched for in the *Find What* field.
2. Set any non-default options; refer to *Find Options* on page 3-28.
3. Click the *Find Next* button. **DataTool** searches the current workbook, including formulas and values.

   If a search is successful, the dialog box closes and the cell containing the found text is selected. If no match is found, a message is displayed.
4. To find the next match, click *Find Next*.
5. To open the *Replace* dialog box, click *Replace*; refer to *Replacing Data* on page 3-29.

### Find Options

*Find What*

Enter the string to find or click the down-arrow to select from the last four strings you entered.

The search string cannot contain wildcard characters. By default, the string is used as a template, as if it had a wildcard character at both ends. To match only whole cells, check *Whole Cell Only*.

*Search*

Select the search range:

*Current Sheet*—Searches only the current worksheet, starting from the top of the sheet.

*Current Workbook*—(default) Searches the entire current workbook, starting from the far-left sheet, usually the *Home* sheet.

*All Workbooks*—Searches all workbooks currently opened in this **DataTool** session.

*Look In*

Search or not search for formulas in a non-numeric string:

By default, *Formulas* is checked; all data cells are searched for a non-numeric string, including those entered as a formula.

If *Value* is checked, no search in cells for a non-numeric string entered as a formula.

If string is a numeric value, all cells are examined, regardless of this setting.

*Match Case*

If checked, searches for all exact matches of the combination of uppercase and lowercase letters specified in the *Find What* box. By default, case is ignored.

*Whole Cell Only*

If checked, the search string must match the entire data string in the cell, not just part of the string. By default, the search string is considered a template, meaning a substring of the cell data produces a match.

# Replacing Data

## Overview

**DataTool** can replace text found by the *Find* command.

**DataTool** can search and locate text in a worksheet; refer to *Search Rules* on page 3-27. To open the *Find* dialog, select the *Find* command on the *Edit* menu or use the keyboard shortcut: Ctrl + F; refer to *Find Dialog Box* on page 3-27.

You have two method to replace text:

- Use the *Replace* dialog box to replace the text. Open the dialog box by selecting *Replace* in the *Edit* menu or by using the keyboard shortcut Ctrl + H; refer to *Replace Dialog Box* on page 3-30.
- Use the *Find* dialog to search for the text. After the text is found, click *Replace* in the *Find* dialog to open the *Replace* dialog. This method transfers the search string in the *Find What* field of the *Find* dialog box to the *Find What* field of the *Replace* dialog; however, the non-default options from the *Find* dialog box are not transferred to the *Replace* dialog box; refer to *Replace Dialog Box* on page 3-30.

## Replace Dialog Box

Use this dialog box to enter the string to replace, the replacement string, and to set any replacement options:

1. Enter the search string in the *Find What* field.
2. Enter the replacement string in the *Replace With* field.
3. Set any non-default options; refer to *Replace Options* on page 3-30.
4. Click *Find Next* to find the first occurrence of the search string. **DataTool** searches the current workbook, looking in both formulas and other values. For the list of searching rules, refer to *Search Rules* on page 3-27.
5. To replace this occurrence of the search string, click *Replace*.
6. To replace all occurrences of the search string, click *Replace All*.
7. To skip this occurrence of the search string without replacing it, click *Find Next*; the next occurrence, if any, is found.
8. To close the dialog box without replacing text, click *Cancel*.

### Replace Options

*Find What*

Enter the string to be replaced or click the down-arrow to select from the last four strings you entered. You must find the text before replacing it.

The search string cannot contain wildcard characters. By default, the string is used as a template, as if it had a wildcard character at both ends. To match only whole cells, check *Whole Cell Only*.

*Replace With*

Enter the replacement text. The replaced text appears exactly as you enter it here.

To delete the text in the *Find What* field from the document, leave the *Replace With* field empty.

| | |
|---|---|
| *Search* | Select the search range:<br><br>*Current Sheet*—Searches only the current worksheet, starting from the top of the sheet.<br><br>*Current Workbook*—(default) Searches the entire current workbook, starting from the far-left sheet (usually the *Home* sheet).<br><br>*All Workbooks*—Searches all workbooks currently opened in this **DataTool** session. |
| *Look In* | Search or not search for formulas in a non-numeric string:<br><br>By default, *Formulas* is checked; all data cells are searched for a non-numeric string, including those entered as a formula.<br><br>If *Value* is checked**,** no search in cells for a non-numeric string entered as a formula.<br><br>If string is a numeric value, all cells are examined, regardless of this setting. |
| *Match Case* | If checked, searches for all exact matches of the combination of uppercase and lowercase letters specified in the *Find What* box. By default, case is ignored. |
| *Whole Cell Only* | If checked, the search string must match the entire data string in the cell, not just part of the string. By default, the search string is considered a template, meaning a substring of the cell data produces a match. |

# Using Formulas

## Overview

You can quickly enter test program values into **DataTool** by using formulas based on a set of values defined on the *AC Specs* or DC Specs sheet; *Creating Formulas* on page 3-32. For example, values for various parameters on the *Pin Levels* sheet can be defined by using variables from a *DC Specs* sheet, while the waveforms defined on the *Edge Sets* sheet or *Time Sets* (*Basic*) sheets can be defined by using variables from an *AC Specs* sheet.

In addition, **DataTool** formulas support:

- *Variables in Formulas* on page 3-33
- *Using the Excel IF Function* on page 3-33
- *Engineering Units in Formulas* on page 3-34

All cells on a worksheet are displayed in either formula form or evaluated cells; refer to *Displaying the Worksheet Data* on page 3-35.

## Creating Formulas

### Overview

Any column that accepts a numeric value accepts a formula. Like in **Excel**, you enter a **DataTool** formula by typing an equal sign (=) followed by the formula. Consequently, any valid **Excel** formula is a valid **DataTool** formula. An operand in an **Excel** formula is a constant, a cell or range reference, or a worksheet function. For information about **Excel** formulas, refer to the online **Excel** documentation.

Additionally, **DataTool** formulas support:

- *Variables in Formulas* on page 3-33
- *Using the Excel IF Function* on page 3-33
- *Engineering Units in Formulas* on page 3-34

### Variables in Formulas

**DataTool** supports variables defined on the *AC Specs* or *DC Specs* sheet. Its name is defined in the *Symbol* column of the sheet. Other sheets can refer to the variable in a formula by placing an underscore in front of the variable name:

| | |
|---|---|
| *AC Specs* or *DC Specs* sheet variable: | *tpd* |
| Reference from other sheet: | *_tpd* |
| In formula: | *=_tpd* |
| | *=_tpd*2* |

By using underscores in the variable, the test program accepts symbols that are otherwise reserved for **Excel**, such as, cell names. For example, *=T13* is a reference to an **Excel** cell, while *=_T13* refers to a user-defined variable.

### Using the Excel IF Function

The **Excel** function **IF** is useful in a **DataTool** formula. For example, a spec symbol can be used as a pin level value only when the symbol evaluates to a certain minimum value when the **Excel** function **IF** is used in a formula:

=IF(*_Vps_val* < 0.1,0.1,*_Vps_val*)

In this sample formula, if the current value of *Vps_val* is less that 0.1, the value 0.1 is used; otherwise, the value of *Vps_val* is used.

## Displaying Formulas

All cells on a worksheet are displayed in either formula form or evaluated cells. Click on the *Formula* button  on **IG-XL** toolbar to toggle between displaying the formulas and the result values or select the *Toggle Cell Formula* command on the **IG-XL** menu. This command affects all cells in the current worksheet only, other worksheets are not affected. Even if the current cell displays the evaluated value, the formula is always displayed in the editing area of the Formula Bar.

In **Excel**, you can use Ctrl + ` (left quotation mark) to toggle the display of formulas and values; however, this function can affect the column widths of the **DataTool** display. Consequently, Teradyne recommends that you use the IG-XL *Formula* button or the *Toggle Cell Formula* to toggle the formula display.

## Engineering Units in Formulas

By default, numeric values are entered and displayed without units, such as milliamps or nanoseconds. Engineering units are a special in **IG-XL** formulas. They are entered by using a numeric formula, which is a value that includes a unit:

=*value***unit*

where *unit* is a predefined scaling values:

**ps**, **ns**, **us**, **ms**, **s**, **pA**, **nA**, **uA**, **mA**, **A**, **nV**, **uV**, **mV**, **V**

Examples:

=20*mA

=0.6*ns

### Rules and Restrictions

- *pV* is not a valid scaling value: it is reserved by **Excel**. Do not use it to represent picovolts.
- Case of the scaling values is not significant because **DataTool** converts all values to the proper form.
- Engineering units are allowed in a formula containing variables. For example, the following formula adds 2 nanoseconds to the current value of *tpd*:

  =_*tpd*+2*ns

### Viewing Engineering Units

The units displayed depend on the formula mode, whether the formulas or the result values are selected. In the formula mode, values are displayed with their units. For resulting values, the units are displayed in engineering notation, as a value in base units, multiplied by the scaling factor. For example, the formula *=20*mA* is displayed in engineering notation as 20.E-03.

# Displaying the Worksheet Data

## Overview

DataTool provides the following features for displaying the worksheet data:



## Collapsing and Expanding Groups

### Overview

Several worksheet types use successive rows to define the items in a group. For example, a pin group on the *Pin Map* is defined on several rows, each row lists one pin in the group, while the parameter definitions for each pin or pin group on the *Levels* sheet are on successive rows. These successive rows can be collapsed or expanded; refer to *DataTool Controls* on page 3-35 and *Excel Hide and Unhide Commands* on page 3-36.

### DataTool Controls

Sheets with successive rows that define the items in a group have a plus or minus sign in the bar at the left edge of the window. One plus or minus sign is displayed for each group of rows, providing separate control over each group:

- plus sign (+)—means the group is collapsed, appearing to occupy only a single row. To expand the group into the individual rows, click the plus sign.
- minus sign (-)—means the group is already expanded into its individual rows. To collapse the expanded group into a single row, click the minus sign.
- Sheets with expanded or collapsed groups will have a 1 and 2 displayed horizontally at the top left corner of the window. Click the *1* to collapse all groups into single rows; click the *2* to expand all groups into the multi-row format. These numbers signify the outline level to display: Level *1* displays just the group names, Level *2* displays the group names and group members.

These controls—plus and minus signs, and *1* and *2* in boxes—are **Excel** outline symbols. If they are not visible, on the *Tools* menu, *Options* submenu, *View* tab, *Windows* options, check the *Outline symbols*.

### Excel Hide and Unhide Commands

Even though **Excel** provides *Hide* and *Unhide* commands on the *Row* submenu in the *Format* menu, Teradyne recommends the **DataTool** features for collapsing and expanding rows because if you use the **Excel** controls to expand or collapse the display of the rows, **DataTool** may displayed the rows in a unpredictable format because the **DataTool** and **Excel** worksheet formatting is different.

## Hiding and Exposing Columns

### Overview

Several types of worksheet have many columns, some are used less often than others; thus, you can hide or expose the least used columns. The worksheets that allow you to hide columns include the *Test Instances* and *Flow Table* sheets.

### DataTool Controls

+ Teradyne recommends that you use the **DataTool** features for collapsing and expanding columns; refer to *Excel Hide and Unhide Commands* on page 3-36.

As shown below, sheets that support hiding and exposing columns have a plus or minus sign in the bar below the toolbar and above the data area:


- plus sign (+)—hidden columns. To expand the columns, click the plus sign.
- minus sign (-)—all columns are exposed. To collapse them, click the minus sign.
- Sheets with expanded or collapsed groups display a number 1 and 2, which are stacked at the left edge of the top bar; see figure below. These numbers indicate the outline level to be displayed: click *1* to hide all columns; click on the *2* to expand all columns. These controls—plus and minus signs, and *1* and *2* in boxes are the **Excel** outline symbols:


If they are not visible, under the *Tools* menu, *Options* submenu, *View* tab, make sure that *Outline symbols* is checked on the *Window options*.

## Filtering Data in a Column

### Overview

Some sheets can be filtered to display only a certain value in a column; refer to *Filtering Controls* on page 3-37. For example, all rows on the *Pin Map* sheet that contain a specific pin name in the *Pin Name* column can be display, which lets you view all pin groups that contain this pin.

### Filtering Controls

A column supporting filtering has a drop-down list button in its column heading. Clicking this button displays a drop-list containing all current values in the column, plus keywords.

The drop-down list for filtering has the following rules:

- Filter drop-down list includes only those values that are currently displayed; thus, values for hidden rows are not included. To display hidden values, expand all groups; refer to *Collapsing and Expanding Groups* on page 3-35. The easiest way to expand the groups is to click on the *2* in the upper left edge of the display area, beneath the toolbars.
- To display only those rows that contain a specific value, click on the drop-down list button, and select the value to filter on.
- Top10 option in drop-down.
- To redisplay all rows, click on the drop-down list button and select *All.*
- For information about the *Custom* option, refer to the **Excel** online help for the *Custom AutoFilter*; also, refer to *Excel AutoFilter* on page 3-37.

### Excel AutoFilter

**DataTool** has predefined the columns that support filtering; consequently, if you use the *AutoFilter* command in the *Filter* submenu of the *Data* menu, the drop-down filtering lists will be removed from the predefined columns. If you repeat the same command, filtering will be enabled for all columns. Teradyne recommends that you do not use the **Excel** *AutoFilter* command.

# Spec Sheets

## Overview

Spec sheets define the spec symbols used on other sheets in the formulas for timing values and pin level values. By separating a spec symbol used in formulas from the assignment of a value to the spec symbol, **DataTool** simplifies the creation and maintenance of test programs. Thus, if a specification changes, you can change the definition of a single symbol: the formulas using the symbol do not have to be manually changed. Several types of spec sheets are provided; refer to *Types of Spec Sheets* on page 3-39.

In addition, a single spec symbol can be different values, depending on the job or the test. For example, an *AC Specs* sheet defines a spec symbol named *tpd*. Other sheets can reference *tpd* by prefixing an underscore ( _ ) to it and then inserting it in a formula; refer to *Choosing a Spec Symbol* on page 3-39. In the following example, an *Edge Set* sheet defines a timing value:

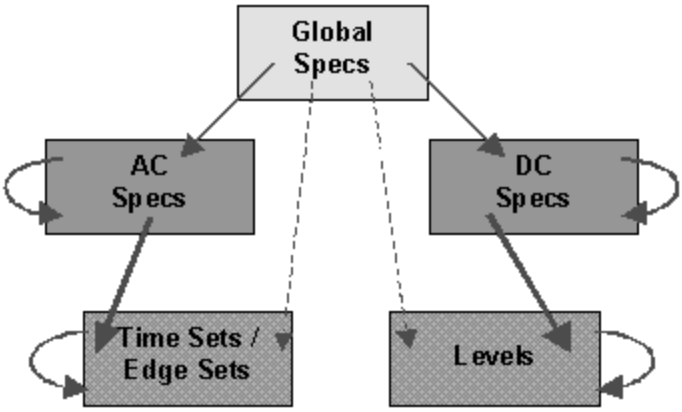
*=_tpd+3*ns*

The value of *tpd* at test execution time depends on the current job context and on the version of the spec selected by the test instance. The meaning of this timing value, however, remains the same because the test program has defined the value in terms of a spec symbol, instead of a specific numeric value.

## Types of Spec Sheets

- Global Specs Sheet—defines global (usually hardware-specific) spec symbols, plus any user-defined constants.
- AC Specs Sheet—defines spec symbols used for timing values.
- DC Specs Sheet—defines spec symbols used for pin level values.

The following figure shows the relationship between the global specs, AC and DC specs, and timing and levels sheets:




## Choosing a Spec Symbol

All spec symbols defined in the workbook are available in a drop-down list in the **Excel** *Formula Bar*. To view the *Formula Bar*, select *Formula Bar* on the **Excel** *View* menu is checked; refer to *Formula Bar* on page 3-10. On the left side of *Formula Bar* is the *Name Box*, which usually displays the column letter and row number of the current cell.

If you select a spec symbol from this list, you are taken to the sheet and cell where the symbol value is assigned.

## Global Specs Sheet

The *Global Specs* sheet defines global spec symbols, which are usually hardware. On this sheet, you can also define any constants that can be referenced throughout the program, including the *AC Specs* and *DC Specs* sheets. Even though global specs can define the timing and levels values; Teradyne does not recommend this practice.

A workbook for a test program must have only one *Global Specs* sheet. When a new workbook is created, **DataTool** creates one Global Specs sheet, with a set of global spec names and values already defined.

In most cases, the spec symbols defined on the *Global Specs* sheet should be truly global: they should always have the same value. Note that you can assign values to the spec symbols for specific jobs.

# Fundamentals of AC Specs and DC Specs Sheets

### Overview

The *AC Specs* and *DC Specs* sheets have the same structure and restrictions, although they are distinct types:

- Specifications on the *AC Specs* sheet define the values on the *Time Sets*, *Time Sets* (Basic), and *Edge Sets* sheets.
- Specifications on the *DC Specs* sheet define values on the *Pin Levels* sheet.

Both these sheets conform to the STIL (Standard Test Interface Language) specification for Digital Test Vector Data. By following the STIL format, these sheets use categories and selectors for structuring the specification data on these sheets.

### Rules

- AC specs may define other specs on the same *AC Specs* sheet.
- DC specs may define other specs on the same *DC Specs* sheet.
- Timing values may define other specs on the same *Time Sets* or *Edge Sets* sheet.
- Levels values may define other specs on the same *Pin Levels* sheet.

### Restrictions

- Only one *AC Specs*, *DC Specs*, *Time Sets*, *Edge Sets*, and *Pin Levels* sheet is active; thus, specs or values on one sheet cannot define specs or values on another sheet of the same kind. For example, a spec on one *AC Specs* sheet cannot define a spec on another *AC Specs* sheet.
- Definitions cannot be used on different sheet types. Example: an AC spec cannot be defined on a *Pin Levels* sheet, and a DC spec cannot be defined on a *Timing Sets* sheet.
- Definitions cannot move upward. Example: an AC or DC spec cannot define a global spec.

### Creating Categories

A category is a named set of values for the spec symbols. These categories and their values correspond to the data on the manufacturer's specification sheet. You should use this data when creating spec sheets; also, refer to *Creating Selectors* on page 3-42. Each category has a set of *Typ*, *Min*, and *Max* values for each spec symbol. Example:

| | Category 1 | | | Category 2 | | |
|---|---|---|---|---|---|---|
| **Symbol** | **Typ** | **Min** | **Max** | **Typ** | **Min** | **Max** |
| *sp1* | 4 | 2 | 6 | 17 | 15 | 21 |
| *sp2* | -10 | -15 | -8 | -5 | -7 | -1 |

For *AC Specs*, a category could correspond to a different speed grade. For *DC Specs*, one category could correspond to a high performance part with one set of voltage specs, and another category could represent a lower-performance part with a different set of specifications. Also, refer to *Categories and Selectors in a Test Instance* on page 3-43.

### Creating Selectors

A selector chooses the *Typ*, *Min*, or *Max* value for each spec symbol. Example:

| *Symbol* | *Sel1* | *Sel2* |
|---|---|---|
| *sp1* | *Typ* | *Min* |
| *sp2* | *Max* | *Max* |

If *Sel1* is specified, *sp1* is assigned the *Typ* value, and *sp2* is assigned the *Max* value.

Selectors are usually created for specific tests. For a given test, you know the proper value for a symbol: *Typ*, *Min*, or *Max*. For example, when testing for narrow limits, you would use mostly the minimum values; for testing wide limits, you would use mostly maximum values. In each case, you choose either the maximum or the minimum values, regardless of the actual values. For this reason, creating the selector is independent from creating the categories and their values. Also, refer to *Categories and Selectors in a Test Instance* on page 3-43.

### *Categories and Selectors in a Test Instance*

Each test instance specifies the category and the selector, thereby defining the values for the spec symbols in the *AC Specs* and *DC Specs*. The value for each spec symbol is a combination of a category and a selector. Using the examples in *Creating Selectors* on page 3-42, if the test instance specifies *Category2* and *Sel1*, the following table lists the available combinations of categories and selectors:

**Category 2**

| Symbol | Sel1 | Typ | Min | Max |
|---|---|---|---|---|
| *sp1* | *Typ* | 17 | 15 | 21 |
| *sp2* | *Max* | -5 | -7 | -1 |

For this test instance, *sp1* = 17 (*Typ*) and *sp2* = -1 (*Max*).

To specify a category and selector, use the *Spec Category* and *Spec Selector* controls; refer to *Spec Category and Spec Selector Controls* on page 3-44.

### Spec Category and Spec Selector Controls

The **IG-XL** *Context* toolbar provides controls to select the current category and selector that you want to rename, edit, or display. These controls must be enabled before you can use them; refer to *Enabling the Spec Category and Spec Selector Controls* on page 3-44.

- Enabling the *Spec Category* and *Spec Selector* Controls

  The Spec Category and Spec Selector controls are enabled only when you are on an active *AC Specs* or *DC Specs* sheet or when the active sheet uses spec symbols defined on a spec sheet; see the following figure:


  Examples of sheets using spec symbol include the *Pin Levels* sheet, *Time Sets* (*Basic*) sheet, or *Test Instances* sheet. In this context, *active* means a sheet used in the active job. You can change the category and selector on one of these sheets and view the effect of the values defined by the spec symbols.

- *Spec Category Control*

  This drop-down menu lists all categories defined on the active *AC Specs* and *DC Specs* sheets. If a category name is preceded by two colons (*::*), an environment name is required; refer to *Environment Name as Part of Category Name* on page 3-45.

  In addition, the *Spec Category* lists all test instances on the active *Test Instances* sheet. The test instances are listed before the categories. By specifying a test instance, you implicitly specify both the categories and selectors because each test instance specifies the AC and DC categories and selectors. Consequently, when you select a test instance, you are also selecting the spec context in which the test instance evaluates spec symbols.

- *Spec Selector Control*

  This drop-down menu lists the available *Spec Selector* values for the selected *Spec Category*:

  a. If the *Spec Category* control specifies a *category*, the *Spec Selector* control displays the selectors defined on the spec sheet where the category has been defined; the list is restricted to the valid selectors for the specified category.

  b. If the *Spec Category* control specifies a *test instance*, the *Spec Selector* control is disabled because the test instance already implies a selector has been selected.

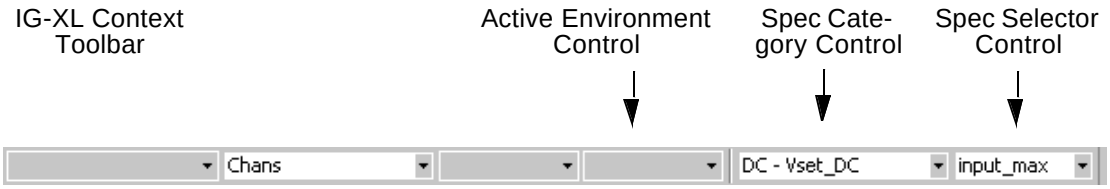
### *Environment Name as Part of Category Name*

A category name may begin with an environment name, with the two names separated by two colons (::). Example:

*Cold::40MHz*

where *Cold* is the environment name, and *40MHz* is the category name. Environment names define different but related sets of spec values for different conditions; these names can also be used on the *Flow Table* sheet for test selection by gating; refer to *Selecting the Test by Gating* on page 3-226.

If a category name beginning with an environment name is displayed on the *Spec Category* control, you must also use the *Active Environment* control on the IG-XL Context toolbar to specify an environment name for the category because both a category name and the environment name are required. The following figure shows this control:




Changing either the environment or category name may temporary create an invalid category name. After **DataTool** displays an error message; acknowledge the error message by clicking *OK* on the error dialog and then select a valid category name or environment name to complete the name.

# Validating the Test Program

## Overview

Even though **DataTool** discovers some types of data entry errors when numeric data is entered in a cell or the cell is edited; non-numeric programming errors are discovered only when the test program is validated. These errors include overusing tester resources, illegal pin names, and incomplete or multiple definitions; also, refer to *Typical Checks Found During Validation* on page 3-48 and *Errors Worksheet* on page 3-48.

Test programs are validated only when you select the validation command, or you try to run a program that has not been validated; refer to *Invoking Validation* on page 3-47. Consequently, the workbook can be in a temporarily invalid state while you are editing a test program.

# Invoking Validation

Before invoking validation, you must set the execution context by using the *Active Job*, *Active Channel Map*, *Active Part*, and *Active Environment* controls on the *IG-XL Context* toolbar; refer to *Debugging a Program* on page 3-72. Validation does not check all possible combinations; it checks only the combinations defined by the current execution context; refer to *Typical Checks Found During Validation* on page 3-48 and *Optimizing Validation* on page 3-49.

A test program can be directly or indirectly invoked:

- You can directly invoke validation by clicking on the *Validate Job* button  on the **IG-XL** toolbar or select the *Validate Job* command from the **IG-XL** menu.

  If the test program has not changed since the previous validation, another validation is not needed; consequently, the *Validate Job* button and command are disabled. When you first open the workbook or make a change in the job that requires revalidation, the validation button and command are enabled, meaning that you must validate the job before running it.

- Validation is invoked by **DataTool** when you try to run a unvalidated test program. An unvalidated program cannot be run.

  If a FLASH 750 test program is run in the off-line mode, the test program is not completely validated because **DataTool** cannot access the system files, pattern files, and other files comprising a complete test system. Consequently, the FLASH 750 must fully validate the test program before it is run on a tester.

## Typical Checks Found During Validation

Some, but not all, types of possible checks when verifying a test program workbook:

- Each referenced pin or group has a corresponding entry in the *Pin Map* sheet.
- Pins are defined before they are included in pin groups.
- Pin groups do not contain circular references.
- Each sheet selected by a Test Instance actually exists.
- Values on the *Pin Levels*, *Edge Sets*, and *Time Sets* sheets are within the capabilities of the tester.

Also, refer to *Errors Worksheet* on page 3-48.

## Errors Worksheet

If errors are found during validation, **IG-XL** creates an *Errors* worksheet, which lists the errors found during validation. This is the right-most sheet. If this sheet is not created, no validation errors were found. On the *Errors* sheet, the *Sheet* column has a hyperlink to the sheet with the error; click on the hyperlink to go to the sheet.

If an item has multiple errors, such as, a specific test instance on the *Test Instances* sheet, the multiple error rows for the item are collapsed into a single row. To display all the errors for the item, click on the plus sign (+) in the left margin next to the row.

# Optimizing Validation

Validation of a workbook is optimized so the minimum items are verified. For example, if you change only the *Flow Table* sheet, the rest of the workbook does not have to be re-verified because no other sheets depend on the *Flow Table* sheet. On the other hand, if you change the *Pin Map* sheet, a thorough re-validation is required because most of the other sheets depend on the *Pin Map* sheet.

Editing different sheets require different degrees of validation:

- Edits requiring an instance to be revalidated:
  a. *Flow Table*
  b. *Test Instance* arguments
- Edits requiring all test instances using the sheet to be revalidated:
  a. *Pin Levels*
  b. *Time Set*
  c. *Edge Set*
- Edits requiring the entire job to be revalidated:
  a. *Job List* (whether or not the active job was modified)
  b. *Channel Map* (affects tester resource checks)
  c. *Pin Map* (most sheets have dependencies on pin map)
  d. *AC Specs*, *DC Specs*, or *Global Specs*
  e. Pattern group
  f. Pattern set
  g. *Test Instances*, only if adding or deleting instances
- Changing these toolbar selections requires the entire job to be revalidated:
  a. *Channel Map* (affects check of tester resources)
  b. *Environment* (affects spec sheets)
  c. *Job* (affects active sheet set)
- Adding or deleting active sheets requires a full revalidation.

# Calibrating TDR

## Overview

Before a production test program can be executed on a test system, the system TDR (Time Domain Reflectometry) must be calibrated. TDR calibrates the electrical path length from the end of the pogo pins to the device contacts on the Prober Interface Board (PIB) or Device Interface Board (DIB), plus the length back. The result of TDR calibration is to apply a correction to the programmed received edges of the DUT.

If a a test program is run in the *engineering* mode, TDR calibration is optional.

See also:


- *Requirements for TDR Calibration* on page 3-50
- *Running TDR Calibration* on page 3-51
- *Modifying TDR Calibration* on page 3-51


## Requirements for TDR Calibration

+ If any of the following conditions are not met, TDR calibration will fail:

- FLASH 750 hardware must have passed AC calibration. If the tester passes AC calibration, it should also pass TDR calibration.
- Test program is loaded
- DIB/PIB is engaged
- All device sockets are open; no device in any socket

# Running TDR Calibration

To run TDR calibration in *engineering* mode with **DataTool** visible, click *Calibrate TDR*  on the **DataTool** toolbar or select *Calibrate TDR* command on the *IG-XL* menu.

To run TDR calibration in the *production* mode, click the appropriate button on the operator interface.

# Modifying TDR Calibration

By default, all pins are calibrated, and calibration voltages of 3V (high) and 0V (low) are used.

# Running a Test Program from DataTool

## Overview

During test program development, a test program is run in the engineering or offline mode. In these modes, **DataTool** is visible, and the toolbar buttons are enabled; thus, the test program can be executed.

+ The other **DataTool** mode, the production mode, is used for running a test program from the **IG-XL** operator interface.

To run a test program:

1. Edit the tester configuration file, if necessary; refer to *Tester Configuration File* on page 3-66.
2. Run TDR calibration. Before the program is run in the production mode, you must mount the DIB or PIB, with no devices in the sockets, and run TDR calibration; refer to *Calibrating TDR* on page 3-50.
3. Set the execution context for validating and running the test program. Use the controls on the **IG-XL** *Context* toolbar to set the *Active Job*, *Active Channel Map*, *Active Part*, and *Active Environment*; refer to *Debugging a Program* on page 3-72.
4. Validate the program. A test program must be validated before it can be run. If you edit the program or attempt to run an unvalidated program, **DataTool** validates the program before running it; refer to *Validating the Test Program* on page 3-46.
5. Set up datalog reporting, if desired. Use *DataCollect Setup* to set up datalog reporting. You can enable datalog reporting, specify the output destinations (window, ASCII file, or STDF file), and specify what data to include in a report. By default, datalog reporting is not enabled; refer to *Enabling Datalogging* on page 4-7, *DataCollect Setup,* Chapter 4. Also, you can use *DataCollect Setup* to set up summary reporting and, during program execution, to request interim summary reports during program execution, and to signal the end-of-lot and generate a final summary report.

6. Set any *Run Options*, which are options for running the program: what sites to use, whether to print the results or the execution time, loop over the test program, select debugging options, and others; refer to *Run Options Dialog* on page 3-56.

   If you do not use *Run Options*, the following defaults are used:

   - All sites are enabled as a result of validation.
   - No results are printed.
   - Assume capability for debugging the flow is not enabled.
   - Flow words for enabling flow steps are disabled.
   - Execution stops when all devices are removed from the active device list.

7. Open the *Test Program Output* window, if desired. This window is not the same window that displays the datalog and summary reports, which are set up by **DataCollect Setup**. For more information, refer to *Test Program Output Window* on page 3-64.

   If you select *Run Options* that display the output from a test program, the *Test Program Output* window displays the output. You can open this window before running the program; however, selecting the *Print Results* option always opens this window when you run the program.

8. Run the program. If the program has not been validated, **DataTool** validates the program before running it; refer to *Invoking Validation* on page 3-47. If error occur when running the test program, refer to *Debugging a Program* on page 3-72.

# Setting the Execution Context

## Overview

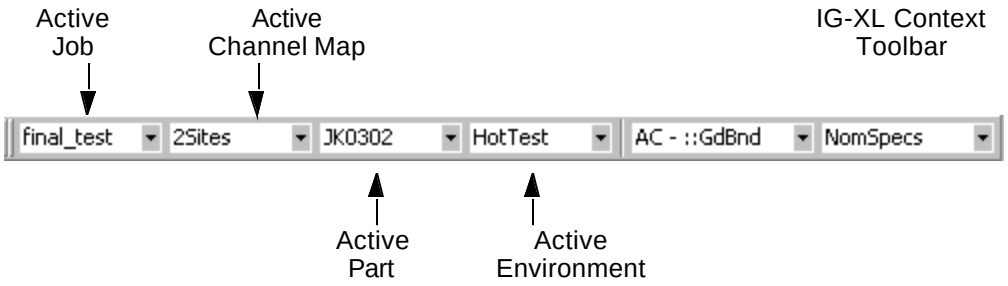
You can select the execution context for the program to be validated or run. The execution context is set before validating a program. If you change any of the execution context settings after validating a program, the program may require revalidation; refer to *Validating the Test Program* on page 3-46. You can select the following environmental items:

- Active job
- Active *Channel Map*
- Active part
- Active environment

Each item has a control in the **IG-XL** *Context Toolbar*; refer to *Execution Context Controls* on page 3-55.

## Execution Context Controls

As shown below, the controls for the execution context are the first four on the *IG-XL Context* toolbar:




+ The selected *Active Job* establishes the settings for the *Active Part* and *Active Environment* controls; therefore, select the *Active Job* before setting the *Active Part* or *Active Environment*.

- *Active Job* specifies the job defined on the Job List sheet; thus, it specifies the **DataTool** sheets that validate and execute the test program.
- *Active Channel Map* selects the Channel Map sheet for validating and executing the test program. The workbook may contain a separate *Channel Map* sheet for each DIB.

  If the *Channel Map* sheet specifies a DIB ID, the tester expects this identification number on the DIB mounted on the tester. If the IDs do not match, the program will not be executed; refer to *DIB ID* on page 3-90.
- *Active Job*, *Active Part*, and *Active Environment* determine which steps of the *Flow Table* sheet are executed; refer to *Selecting the Test by Gating* on page 3-226.
- *Active Environment* specifies the categories used on the AC Specs and DC Specs sheets; thus, it determines the values of the spec symbols. Also, refer to *Environment Name as Part of Category Name* on page 3-45.

# Run Options Dialog

## Overview

Options for running a **IG-XL** program are set with the *Run Options* dialog box. To open the *Run Options* dialog box, click the *Run Options* button on the **IG-XL** toolbar, or select the *Run Options* command on the *IG-XL* menu. If run options are not set, the tester uses defaults listed in *Run Options Defaults* on page 3-56.

Options are set in the Run, Loop, and Stop on Event frames; refer to the following topics:

*Run Options Defaults* on page 3-56

*Run Frame* on page 3-57

*Loop Frame* on page 3-61

*Stop On Event Frame* on page 3-62

*Run/OK/Cancel Buttons* on page 3-63

## Run Options Defaults

If you do not use *Run* Options, the following defaults are used when a test program is run:

- All sites are enabled (as a result of validation).
- Flow words for enabling flow steps are disabled.
- No results are printed.
- Assume capability (for debugging the flow) is not enabled.
- Test program is executed once (no looping).
- Execution stops when all devices are removed from the active device list; no event-based debugging options are enabled.

## Run Frame

The options in this frame are available for any program, even in the *debug* mode.

*Do All*

Check this box to execute all the tests; do not stop on fail.

The default is to stop testing after all devices have been removed from the active device list and the *Pass/Fail Result* have been set for all devices.

If you select Print Results and *Do All*, the Test Program Output window shows two sets of results for each site. The first set of columns shows the results of the *Do All*, which executes all tests. The second set shows the results when *Do All* is not checked: stop testing when the site is removed from the active device list. Example:

```
S R  B SL|R  B SL
i s  i rg|s  i rg
t l  n tc|l  n tc
-------------------------
0 F  4 62 C  F  4  2 C
```

*Do All* results are on the left; the results of normal execution are on the right. To interpret the columns, refer to Print Results.

*Starting Sites*

This window specifies the list of starting sites, those used in executing the test program. All sites (starting with site 0) defined on the *Channel Map* are listed. By default, **DataTool** selects all sites when the *Run Options* window is opened after a test program is validated. Select the sites by clicking on them. Click the *Clear* button to clear all selected sites; click directly on a selected site to clear or de-select just that one site.

As testing begins, all starting sites are on the active site list. As testing proceeds, **DataTool** removes a site from the active sites list after its *Pass/Fail Result* on the *Flow Table* sheet is set; refer to *Active Site List* on page 3-232.

*Assume*

The *assume* capability is for debugging the flow; refer to *Assume Capability* on page 3-243. On the *Flow Table* sheet, the *Debug Assume* and *Debug Sites* columns specify whether a site should be assumed to pass or fail a test.

Check *Assume* on *Run Options* to enable the assume capability. If this item is not checked, any values in *Debug Assume* and *Debug Sites* are ignored, and the test program is executed offline.

### *Enable Words*

This window lists all flow words in the *Enable* column of the *Flow Table* sheet. By selecting the flow words in this window, execution is enabled for any step where the selected word appears in the *Enable* column of the *Flow Table* sheet. To enable the flow words in the *Enable Words* window, click on them. Click the *Clear* button to clear all selected flow words; click directly on a selected word to clear or de-select just that one word. For more information, refer to *Enable* on page 3-270. Note that opcodes can also enable or disable flow words from within the flow.

Enable words must be set up before test execution begins. They cannot be changed during program execution, such as, when stopped at a breakpoint.

### *Execution Time*

Checking this box prints in the Test Program Output window the time, in seconds, to execute the entire flow. By comparison, the Detailed Execution Time option in the *Debug* frame prints the time for executing each step of the flow.

*Print Results*

Check this box to print in the Test Program Output window the results of the test program execution. By default, following print format is used:

```
S R  B S L
i s  i r g
t l  n t c
--------------
0 F  4 62 C
1 P  1  1 C
```

Passing result is green; failing result is red. From left to right, the columns are designated:

- S—site number
- R—Pass/Fail Result
- B—Bin Number
- S—Sort Number
- L—Logic State—True, False, or Clear

For more information, refer to *Processing Results* on page 3-231.

Check any of the following *Run Options* to print additional information:

- *Do All*
- *Execution Time*
- *Detailed Execution Time*
- *Trace Execution*

Execution results from these options is appended to any results already printed in the Test Program Output window.

*Debug*

+ All other *Run Options* can also be used when debugging.

The *Run* frame includes several *Debug* options, which are enabled when the test program is in the *debug* mode. You have three ways to enter the *debug* mode:

- Click the *Debug Run* button on the **IG-XL** *Toolbar*
- Select the *Debug Run* command on the **IG-XL** menu.
- If a debug option is checked, clicking the *Run* button on the *Run Options* dialog executes the program in *debug* mode; refer to *Debugging Mode Commands* on page 3-73.

The two debug options, *Detailed Execution Time* and *Execution Time*, label each test step with the test instance name, followed by the name of the test template section—*prebody*, *body*, or *postbody*—and then write this information to the Test Program Output window:

+ If a flow step is not a test instance, it is labeled with the string that is listed in the *Command Parameter* column of the *Flow Table* sheet, followed by the string Prebody, which, in this case should be ignored.

- *Detailed Execution Time*—prints the time, in seconds, to execute each step of the flow:
  a. failing test—three separate lines are printed: time to execute the prebody, body, and postbody.
  b. passing test—one line is printed, labeled Prebody.

+ By comparison, Execution Time prints once for the entire flow.

- *Trace Execution*—prints the results for each step (*prebody*, *body*, and *postbody*) as it is executed. By comparison, the Print Results option in the in *Run* frame prints only the final results for the entire test program.

## Loop Frame

### Overview

These options are for looping over a test program or a test instance, executing it a specified number of times.

+ All other *Run Options* can also be used when looping.

### Looping Over a Test Program

To loop over the entire job:

1. In *Loop Count*, enter the number of times to execute the test program. To loop forever, enter 0 or leave this field blank.
2. To stop execution when at least one site fails, check the *Until Fail* box. If this box is not checked, execution continues until the loop count is reached.

   A site can fail only when the *Until Fail* box is checked and the *Pass/Fail Result* variable in the *Flow* table for the site is set to *Fail*; refer to *Processing Results* on page 3-231. Note that failing a single test may not cause the site to fail.
3. To start looping, click *Start*.

   The number of loops executed is displayed in the *Loops Executed* field.

+ Clicking *Run* starts program execution.

4. To stop looping, click *Stop*.

### Looping Over a Test Instance

Additional looping options are enabled when the test program is at a breakpoint on a **Test** opcode in the *Flow Table*. With these controls, a test program can loop over a single test instance or parts of a test instance:

1. If the program is stopped on a test that is an instance of a template, the following options are enabled: *PreBody*, *Body*, *PostBody*, and *All*. You can loop over the entire test instance (*All*) or over one or more parts of the template.
2. If the program is stopped on something other than a template, only the *All* box is enabled.
3. After making the looping selections, click *Loop Instance*. To stop looping, click *Stop*.

+ If the *Stop on Fail* box in the *Stop On Event* frame is checked, looping will terminate if this test instance reports a failure.

## Stop On Event Frame

### Overview

These options specify the events that stop execution of the test program: on the first failing test, on a specific test number, or on the first failing test after a specific test number. These options are also called event-based debugging. After the test program is stopped, you are prompted to abort the test, also known as suspending execution.

### Suspended Execution of a Test Program (Aborting the Test)

If a selected stop-on-event occurs, the **DataTool** aborts the test, which suspends the executing test program. While the test execution is suspended:

- **DataTool** and the **IG-XL** Debug Display Manager remain active. You can examine data in the workbook or open a debug display to examine or change a hardware setting.
- A message box appears, prompting you to abort execution of the test program. To abort text execution, click *Yes*; to continue test execution from where the test program was suspended, click *No*.
- Aborting execution of the test program affects only the current test. If the test program is looping, aborting the test does not break the loop; the loop continues; only the current test aborts. To stop looping, click the *Stop* button in the *Loop* frame.

### Controlling the Event

+ If you check more than one box, execution stops on the first selected event.

Check the appropriate box for the event or events that will suspend execution:

| | |
|---|---|
| *Stop on Test #* | Suspends testing after the specified test has been executed. Enter a single test number in the field below this box. Program execution stops as soon as the specific test passes or fails. If this field is blank, execution does not stop. |
| *Stop on Fail* | Suspends testing when the first failing pin is encountered, even if the failure occurs in the middle of a test instance. Check this box to enable this feature. |
| *Stop on Fail After Test #* | Executes the test program through the specified test; after that point, it suspends testing on the first fail. You enter a single test number. If the program doesn't include a test with the specified number, execution does not stop. |

+ If the test program is stopped at a breakpoint, these additional controls are enabled: *PreBody*/*Body*/*PostBody*/*All.*

## Run/OK/Cancel Buttons

Once you have selected all options:

- Click *Run* to execute the program immediately, using the selected options.

  To run the program in a loop, you must click the *Start* button in the *Loop* frame. The *Run* button will not execute the loop.

- Click *OK* to save the options, close this window, and return to **DataTool**. When you execute the program later, the saved options will be used.

- Click *Cancel* to close this window and return to **DataTool**. The options are not saved.

# Test Program Output Window

## Overview

The *Test Program Output* window prints any results from the selected *Run Options*; *Managing the Test Program Output Window* on page 3-64 and *Displayed Test Results* on page 3-65. This window is different from the *Datalog* and *Summary Output* windows:

- *Test Program Output* window displays a small set of test results specified by the Run Options window.
- *Datalog* and *Summary Output* windows display more comprehensive test results, which are specified by **DataCollect Setup**.

## Managing the Test Program Output Window

If you select any Run Option that generates output, **DataTool** opens this window to display the results; *Displayed Test Results* on page 3-65. Or, you can open this window by clicking the *Output Window* button  on the **IG-XL** toolbar or selecting the *Output Window* command on the **IG-XL** menu.

The results from the execution of each test program are appended to any results already displayed in the window, unless you first clear the window.

To clear the window, click *Clear*.

To close the window, click *Close*.

To save, print, select, or copy the text in this window, use the standard Windows commands in the *File* and *Edit* menus of this window.

## Displayed Test Results

The results displayed in this window depend on the selected Run Options:

+ The test results for the *Debug* options in the *Run Options* window are printed only if the test program is run in the *debug* mode.

### *Any Mode*

| | |
|---|---|
| *Print Results* | Displays the final pass/fail, bin, sort, and logic results for each site. Passing sites are green; failing sites are red. The *Do All* option prints additional results. |
| *Execution Time* | Displays the time, in seconds, for executing the entire test program. |

### *Debug Mode*

| | |
|---|---|
| *Detailed Execution Time* | Displays the execution time, in seconds, for each step of the flow. |
| *Trace Execution* | Prints the results (pass/fail, bin, sort, logic) for each step as it is executed. |

# Tester Configuration File

## Overview

Each time a test program is loaded and run on the FLASH 750 hardware (online) or the test system power is cycled, **IG-XL** queries the ID PROMs on the system boards to determine the tester configuration. **IG-XL** then writes the configuration information to an ASCII file. This information ensures the tester has the required hardware to execute the test program. If the hardware is not present, validation errors are generated and the test program cannot be executed.

The tester configuration file is also required when running a test program offline (not on the tester). In this case, the tester configuration is always read from a tester configuration file.

You can override the tester hardware configuration by creating a configuration file. For example, even though a test program requires hardware that is not present in the tester, you can create a configuration file to override the actual hardware configuration, so the test program can be run.

## Creating a Configuration File for Online Execution

Before a test program is run online (on a tester), the system creates a tester configuration file:

1. Reads the ID PROMs on the system boards to learn the tester configuration.
2. Searches for a user-created configuration file in the following order and goes to the next step after finding the first file:

   *<current_dir>\TesterConfig.txt*

   *<install_dir>\Tester\TesterConfig.txt*

   *<install_dir>\Bin\TesterConfig.txt*
3. Merges the ID PROM information gathered in Step 1 with the configuration file found in Step 2 and writes this information to the following ASCII file:

   *<install_dir>\Tester\CurrentConfig.txt*

   This file represents the current configuration; refer to *Merging the Configuration Information* on page 3-67.

   If no user-created configuration file is found, the information in the *CurrentConfig.txt* is based only on the ID PROM information.

   If you suspect a configuration problem, examine this file with any text editor.

## Merging the Configuration Information

The tester does not replace the hardware configuration information with a user-created configuration file; instead, the tester merges slot by slot the information from the ID PROMs and the user-created configuration file. Thus, if the information for a particular slot in the configuration file matches the information for the same slot gathered by the system, the slot information from the configuration file replaces the slot information from the ID PROMs. In addition, you can add slot information to the user-created configuration file to create a hardware configuration that does not actually exist, so you can run a test offline or override the existing hardware configuration; refer to the *Configuration File Format* on page 3-68.

## Creating a Configuration File for Offline Execution

Before a test program is run offline (not on a tester), the system creates a tester configuration file:

1. Searches for a user-created configuration file in the following order and goes to the next step after finding the first file:

   *<current_dir>\SimulatedConfig.txt*

   *<install_dir>\Tester\SimulatedConfig.txt*

   *<install_dir>\Bin\SimulatedConfig.txt*

2. Writes the configuration information from the user-created configuration file to the following ASCII file:

   *<install_dir>\Tester\CurrentConfig.txt*

   If no user-created configuration file is found, an empty *CurrentConfig.txt* is created. This empty file does not affect the offline execution of a test program.

3. When you open an existing test program and no user-created configuration file exists, a **DataTool** error message appears. Click *OK* to acknowledge the system is using the *SimulatedConfig.txt* file.

## Configuration File Format

### Overview

The ASCII configuration file, whether created by **IG-XL** or by a user, contains any number of lines with fields. If you are creating or editing a configuration file, be sure to be aware of the *Rules and Guidelines* on page 3-68.

### Rules and Guidelines

- The configuration file does not need to duplicate any slot numbers from the hardware configuration that you want to use as is.
- Fields are separated by any number of blank spaces or tabs.
- Any line or portion of a line beginning with a pound sign (#) is ignored up to the carriage return and can be used as a comment.
- Blank lines (lines with carriage returns with or without preceding white space) are ignored.
- Later lines in the file can override earlier lines without error or warning.
- Any line beginning with < is expected to contain the system name and should contains the label <system name> followed by the system name string separated by tabs.
- MTM entries have two extra fields following the IDPROM fields; these contains the <dbm size> and <ecr size>.
- To cancel a board in the configuration list, use a 0 ID-PROM on the system board (0-0-0-0 or just 0) or place the word *delete* in the *ID-PROM* field of the configuration file.

### Fields

<*station*>[.<*subslot*>]<*board_type*> <*boardtype*>
<*serial*><*rev*><*company*><*datecode*> <dbm size> <ecr size>

| | |
|---|---|
| <*station*> | integer from 0 to 31. Slot number –1 designates a slotless board (*DIB, PCIT*, *SLI*, backplane, or *CalCUB*). |
| <*subslot*> | integer, sub-slot number, designates a daughterboard on the parent board that is identified by a *station*. Use 0 to 1 for backplane, *CalCUB*, *SLI*, or *PE32*. |
| <*boardtype*> | string, one of the following board types: *backplane*, *calcub*, *cdd*, *dib*, *mtm*, *pcit*, *pe32*, *sli*, *station* |
| <*id-prom-value*> | consists of four subfields separated by blanks, which have the following formats, where *X* is a hex nibble, *A* is a alphanumeric character, and the hyphens are required:<br><br><*boardtype*>: X*XX-XXX-XX*<br><br><*serial*>: *XXXXXXX*<br><br><*rev*>: *XXXX-X* for *pcit* and *sli; AAAA-A* for all others<br><br><*company*>: *XXXX* (always *5445*)<br><br><*datacode*>: *WWYY*, where *WW* is work week and *YY* is year |
| <*dbm size*> | integer, specifies the DBM size on the MTM only |
| <*ecr size*> | integer, specifies the ECR size on the MTM only |

# Optimizing Test Programs

## Overview

The following suggestions are for optimizing a FLASH 750 test program.

## set-device Opcode with Blank Result

The **set-device** opcode at the end of the program flow should not have anything in the *Result* column on the *Flow table*. If the column has *All* or *Pass*, the software cycles through the passing sites, shutting them down serially. Furthermore, when sites are shut down serially, download cannot be used. As a result, this process is even slower, and has 0% parallel efficiency.

If the *Result* column is blank, the **set-device** opcode just sets the bins and the result; refer to *set-device* on page 3-255 and *set-device-new* on page 3-257. When the flow terminates, all sites are shut down at once, in parallel, using download. This technique saves 100 to 400 ms, depending on the number of sites.

Note that **set-device** operates as it does because it can be also used in the middle of the flow, where stopping only a subset of the sites would be required.

## Default Power-up Sequence on the Pin Levels Sheet

The *Pin Levels* sheet determines the order of how the pins are powered up and down. By default, pins are always powered up or down in a sequence that avoids damage to the DUT.

On the *Pin Levels* sheet, you can specify a non-default sequence for powering up and down the pins; however, this can significantly degrade performance. Specifically, if a sequence number is listed for a signal pin, **IG-XL** cannot download the levels prior to execution; consequently, performance cannot be optimized. If possible, you should use the default sequence, and should not enter sequence numbers for input or I/O pins; refer to *Power-Up and Power-Down Sequencing* on page 3-145.

## Conditional Execution of Timing Adjusts

A timing *Adjust*, also known as *edge find*, can be performed on every device tested, but testing is faster by adjusting the timing only when it is required. One test approach uses the adjusted timing value generated by a previous *Adjust*. If the test passes, the adjusted value is adequately, and another *Adjust* is not required. But, if the test fails, a new *Adjust* is required; refer to *Overlay* on page 3-206.

## Sort the Flow

In the flow, test instances are usually order-independent. Once a test program is working, the flow should be able to be changed without affecting the functionality of the test program, except when the flow has intentional branching or test instances that use an adjusted spec value from an earlier characterization.

Because the flow should be order-independent, you can sort the flow so instances set up with the same timing or levels, or both, are grouped together. This sorting minimizes the number of the tester setup changed. Depending on the characteristics of the particular test program, this can significantly improve performance.

Even though FLASH 750 cannot be programmed to generate a list of instances sorted based on their context, you can manually generate this information:

1. Stop **DataTool** by clicking the *Stop* button on the **IG-XL** toolbar or by selecting the *Stop DataTool* command on the **IG-XL** menu.
2. On the *Test Instances* sheet, use **Excel** to sort the timing and levels columns.
3. Print out the sorted sheets and use them to reorder the flow manually.

   The reordering must be manual because some intentional ordering is always required, such as, continuity tests are always performed first.

## Powered Relay Mode

If the device currents and device characteristics can tolerate hot-switching, select this mode. This mode improves performance because the tester does not power down and then power up when switching relays.

# Debugging a Program

## Overview

**DataTool** provides several methods for debugging an **IG-XL** test program:



## Debugging with the Assume Capability

You can use the assume capability to debug the *Flow Table* sheet. This feature lets you assume that each test will pass or fail certain site. Consequently, you can ensure that flow control and binning are operating properly.

## Debugging with the Run Options

Anther way of debugging a program is to review the results of executing the program. By setting up the Run Options, you can print the execution results in the Test Program Output window.

You can also use *Run Options* to debug a particular event by suspending test execution when any test fails or when a particular test is executed. While test execution is suspended, you can examine the data in the **DataTool** workbook and loop or open a *Debug Display* to examine and edit the tester hardware setting.

# Debugging Using Datalogging

You can enable detailed datalogging of test program execution by using **DataCollect Setup**; refer to *Enabling Datalogging* on page 4-7, *DataCollect Setup*, Chapter 4. The datalog report is printed to the Test Program Output window, and can be written to a file.

# Debug Mode

## Overview

**DataTool** provides a *debug* mode for an executing test program; refer to *Run Options Dialog* on page 3-56. In the *debug* mode, you can also use the Run Options to print additional debugging information, such as, the result of executing each step and the execution time for each step.

## Debugging Mode Commands

These buttons and commands are enabled when the current sheet is the *Flow Table* sheet. The buttons cannot be used from other sheets. The debug mode buttons on the **IG-XL** toolbar have corresponding commands on the *IG-XL* menu.

### Starting and Stopping Execution in the Debug Mode

*Debug Run*  runs the test program in the *debug* mode. When you are running in this mode, you can use the *Toggle Breakpoint*, *Step*, and *Step Over* buttons.

You can also run a test program in the *debug* mode from the Run Options dialog box. By selecting one of the *Debug* options on this box and then clicking the *Run* button on this box, the test program will be executed in *debug* mode.

*Debug Stop* button  stops execution of the test program during the *debug* mode. Used while pausing at a breakpoint or stepping through the flow.

### Setting or Removing a Breakpoint

*Toggle Breakpoint* button sets or removes a breakpoint from the current step of the test flow.

To set a breakpoint, select a cell in the row for the step where execution should pause, click *Toggle Breakpoint*. The breakpoint is represented by a red dot next to the step, in the first column of the *Flow Table* sheet.

### Running to, Stepping, Stepping Over, and Stepping Through Breakpoints

To run a test program to the first breakpoint, where the test program stops, click *Debug Run*. The current point of execution is represented by a yellow arrow; refer to *Parts of Test Templates* on page 3-75. If the breakpoint is set for a *Test* step, execution stops before the prebody.

After the test program is paused at a breakpoint, you have the following options:

- To run to the next breakpoint, click *Debug Run* .
- To step through to the next test or part of a test in the flow, click *Step* .
- To execute the test instance as a unit without stepping into it and to move to the next item in the flow, click *Step Over* . It executes the current test without stepping into the prebody, body, and postbody. If the flow is currently in the middle of a test, *Step Over* executes the rest of the test as a unit, and then moves to the next item.
- To remove a breakpoint, select a cell in the row containing the breakpoint, click *Toggle Breakpoint* . The selected cell must be in the data area, not in column *A*, even though the breakpoint red dot is in Column *A*.
- To remove all breakpoints, select the *ClearAllBreakpoints* command from the **IG-XL** menu or use the keyboard shortcut *Ctrl + Shift + F9*.

### Parts of Test Templates

A test template has three parts: prebody, body, and postbody; thus, you must take three steps to step through a test template. Your current position in the test template is represented by a yellow arrow in the first column of the *Flow* sheet:

 (up)—current position is before the prebody of the test template.

 (across)—current position is before the body of the test template.

 (down)—current position is before the postbody of the test template.

If the step contains a different opcode rather than a test template, the up arrow  is shown.

# Debugging the Tester Hardware Settings with the Debug Displays

## Overview

The **Debug Display** tools examine the tester operations at the hardware level. With these tools, you can view the tester hardware settings and edit them under certain circumstances, and then observe the effects on the test program execution. The following debug displays are provided:

- *Board PMU Debug*
- *DIB ID Prom Utility*
- *Digital Channel* (timing and format of the pin electronics levels for each channel)
- *FLASH 750 DPS* (Device Power Supply)
- *Pattern Control Display* (run patterns and set up the pattern generator and HRAM)
- *Pin PMU Debug*
- *Program Flash*
- *RRT_DD*

To start the **Debug Display** tools, refer to *IG-XL Display Manager* on page 3-10.

# Debugging Patterns with PatternTool

**PatternTool** displays the binary pattern files in a readable and editable form. You can update the tester with the edited binary file without recompiling. **PatternTool** can also display HRAM from the most recent test execution.

To open **PatternTool**, click the *PatternTool* button  on the IG-XL toolbar or select the *PatternTool* command in the **IG-XL** menu.

For more information, refer to *PatternTool* in *IG-XL Help*.

# Types of DataTool Worksheets

# Home Sheet

## Overview

The *Home* sheet provides an overview of the contents of the workbook, organized by job.

Each row of the sheet lists one component of the job: either a sheet specified on the *Job List* sheet or a test instance specified on the *Test Instances* sheet for this job. Each item is an active link to the actual sheet or test.

In addition to job components, the *Home* sheet also lists the *Global Specs* sheet and the current *Channel Map*.

## Using the Home Sheet

### Managing the Home Sheet

This sheet is created and updated by **DataTool**. You cannot insert, edit, or export it.

### Accessing the Home sheet

- Click the *Home* tab on the bottom of the worksheet, the far-left sheet.
- Click the *Home Sheet*  icon on the **IG-XL** toolbar.
- Select the *Home Sheet*  command from the **IG-XL** menu.

### Viewing the Home Sheet

The *Home* sheet is organized by the jobs in the workbook. Each job is displayed as a collapsed group; refer to *Collapsing and Expanding Groups* on page 3-35. Toggle buttons along the left side of the spreadsheet can expand (+) or collapse (-) the rows for each job. The buttons labeled 1 and 2 at the top left can expand (2) and collapse (1) all rows.

### Filtering on a Column

You can filter on either the *Element* column, to see which sheet is used in all jobs, such as the *AC Specs* sheet, or on the *Name* column, to see which jobs use a specific item, such as a specific *Flow Table* sheet. Also, refer to *Filtering Data in a Column* on page 3-37.

### Test Instances

If the job component is a test, the row also displays the sheets that the test instance has specified for its time sets, edge sets, and pin levels. Each of these values is also an active link to the specified sheet:

- Click the test name in the *Name* column to go to the *Test Instances* sheet that defines the test.
- Click a value in the *Time Sets, Edge Sets,* or *Pin Levels* column to go to the sheet for the type that the test instance has specified.

## Home Sheet Columns

### Job

Lists the jobs contained in this workbook. Expand a job to show its contents by clicking on the plus sign at the left edge of the window. To collapse an expanded job, click on the minus sign.

### Element

Identifies the job component—sheet type or test, such as keywords *Pinmap, Flow Table, AC Specs,* and *Test*. It identifies the type of element in the *Name* column. Most keywords identify the types of sheets: *Test* keyword identifies a test instance defined on the *Test Instances* sheet for this job.

## Name

Name of the job component described by this row. The type of component named in this column is listed in the *Element* column. A value in this column is either a test instance name, whose *Element* is *Test*, or a sheet name.

You can filter on a name to see all items of that name in all jobs; refer to *Filtering Data in a Column* on page 3-37.

## Time Sets

Lists the *Time Sets* or *Time Set (Basic)* sheet used by the test instance in this row. It is used only if *Element* column value is *Test*.

## Edge Sets

Identifies the *Edge Sets* sheet used by the test instance. It is used only if *Element* column value is *Test*. Column is blank if the *Time Sets* column specifies a *Time Set (Basic)* sheet rather than a *Time Set* sheet.

## Pin Levels

Lists the *Pin Levels* sheet used by the test instance. It is used only if *Element* column value is *Test*.

## Redundancy Tables

Lists the *Redundancy Table* sheet used by the test instance. It is used only if *Element* column value is *Test*.

## SDA Tables

Lists the *SDA Table* sheet used by the test instance. It is used only if *Element* column value is *Test*.

## Scramble Program

Lists the *Scramble Program* sheet used by the test instance.

## Frame Select

Lists the *Frame Select* sheet used by the test instance.

## MTM Resource Map

Lists the *MTM Resource Map* sheet used by the test instance.

## Data Generator

Lists the *Data Generator* sheet used by the test instance.

## Pin Func

Lists the *Pin Function Set* sheet used by the test instance.

## ECR/DBM Config

Lists the optional *ECR/DBM Configuration* sheet used by the test instance.

## Comment

Lists any additional information about this item. If *Element* column value is *Test*, the AC and DC categories and selectors specified by the test instance are listed in the following format:

*{ac-category.ac-selector }{dc-category.dc-selector}*

# Channel Map Sheet

## Overview

This sheet specifies how the DUT is connected to the tester by mapping the individual device pin names to the test system channels and power supplies. You can define up to 128 parallel test sites; 128 DPSs, each with two outputs; and up to 128 High-Voltage force and measure pins.

The information on this sheet describes the wiring for a specific Prober Interface Board (PIB). The tester channels are accessed through the PIB.

## Using the Channel Map Sheet

### Key Concept: Pin Map and Channel Maps

The difference between the *Pin Map* and the *Channel Map* is important to know:

- Pin Map describes the device and its pins.
- *Channel Map* describes the tester and its resources.

For instance, the *Type* column on the *Pin Map* sheet refers to device pin types, such as a power pin, while the *Type* column on the *Channel Map* refers to specific tester resources, such as the Device Power Supply (DPS).

### Support for Multiple Channel Maps

**IG-XL** supports multiple instances of *Channel Map* sheets in a device test program. For example, a device may be issued in different packages; thus, each wiring variation is defined in a separate *Channel Map* sheet.

The **IG-XL** *Context Toolbar* includes an *Active Channel Map* control, so you can use this control to select the one of the multiple *Channel Maps* before validating or running the test program; refer to *IG-XL Context Toolbar* on page 3-9.

## Matching the DIB ID

Number entered in the *DIB Field* of the *Channel Map* for an executed test program must match the ID of the PIB on the tester; refer to *DIB ID* on page 3-90.

## Inserting a New Channel Map

When you select the *Worksheet* command in the *Insert* menu to create a new *Channel Map*, you are prompted to specify how many total sites will be supported and if pin name will be automatically filled in with values from the *Pin Map* sheet:

1. How pins are assigned:

   a. *Channels within a station only*: Pins of a site are assigned to channels within a station only. In this mode, the test program is run on the host or the embedded processor (EP), and branch-on-error features is supported.

   b. *Channels within an even-odd station pair.* Pins of a site are assigned to channels of an even-odd station pair. In this mode, neither the EP can run a test program nor automatic fill-in is supported.

   c. *Channels within a card-cage*. Pins of a site are assigned to channels within a card-cage pair. In this mode, neither the EP can run a test program nor automatic fill-in is supported.

   DPS restrictions: DPS resources within a station must be used the same way across all sites for DPS ratcheting to work properly. For example, if the first DPS of a station (DPS0) is assigned to a Vcc pin for a site, then DPS0 for all stations must be assigned to Vcc pins. Consequently, when DSP0 is ratcheted, all DPS0's in all stations are ratched at the same time.

2. *Total Number of Sites*: enter 1 to 128 if the DUT pins are confined to a station; otherwise, enter 32.

3. *Number of Sites Per Station*:

   a. 1, 2, or 4 if the DUT pins are confined to a station.

   b. 1 or 3 if the DUT pins are confined to a station pair.

   This field is disabled if the DUT pins can be randomly assigned to any pin in a card cage.

   Enter the number of sites per station required because you cannot add or delete sites later. All sites on the *Channel Map* must be filled.

   Number of sites per station selection also determines how the Failbus hardware is set up.

4. To automatically fill in of the *Pin Name* column with pins from the *Pin Map* sheet, check the box *Fill with pins from the Pin Map*.

   If you check this box, all pins, not the pin groups, from the *Pin Map* sheet are entered in the *Pin Name* column of the new *Channel Map* sheet. The *Type* column is also filled in, based on the pin type. Be aware that this default mapping of pin type to tester resource type is an approximation, and should be reviewed for accuracy, and you should change any of these channel type assignments, as needed.

   Default mappings of pin type to tester resource type:

| *Pin type* | *Channel type* |
|---|---|
| *Power* | *DPS* |
| *Analog* | *N/C* (not connected or not used) |
| *HVF* | *HVF* (High Voltage Force) |
| *HVM* | *HVM* (High Voltage Measure) |

## Channel Map Requirements

- Pin group names are not allowed in the *Pin Name* column.
- Type of the pin defined in the *Pin Map* sheet must be compatible with the type of the channel it is wired to. For example, utility pins must be wired to utility bits and DPS pins to power supplies.
- All site columns (*Site 0* and up) must be filled in. Thus, if you created 3 sites, the channel mapping for 3 sites must be entered, even if only two sites are actually tested. You can set how many sites are tested by selecting the *Run Options* before running the program; refer to the *Run Options Dialog* on page 3-56.
- All pins in the *Pin Map* must be listed in the *Channel Map*. Pins not wired are assigned to channel type *N/C*.

## Multiplexed Pins

### Overview

The multiplex (*mux*) mode connects pairs of adjacent channels, so signals can be generated at twice the normal clock rate.

### Guidelines and Rules

The following description uses as an example pin *xyz* driven by the multiplexed channels *ch8* and *ch9*.

- The tester multiplexes only adjacent pairs of even-odd channels. Channel pairs start with *ch0* and *ch1*; thus, an even channel is multiplexed with the next highest odd channel. For example, *ch8* can be multiplexed with *ch9*, but not with *ch7*.
- In each multiplexed pair, the odd-numbered channel is the multiplexed drive signal. For example, if *ch8* and *ch9* are multiplexed, *ch9* provide the drive signal.
- Pin Map sheet must define (1) two dummy pins representing each half of the multiplex pair: the even and odd channels and (2) a pin group that contains these dummy pins. For example, if device pin *xyz* is to be driven with multiplexed channels, the pin map can define dummy pins *xyza* and *xyzb*, and also pin group *xyz*, which contains *xyza* and *xyzb*.

  *Channel Map* sheet maps these dummy pins to the multiplexed channels. In the example, the *Channel Map* maps the even number (*ch8*) to dummy pin *xyza*, and the odd channel (*ch9*) to *xyzb*.
- Pin Levels sheet defines values for the pin group**,** not for the individual pin names that make up the halves of the multiplexed pair. In the example, the levels are defined for the group *xyz*, not the dummy pins *xyza* and *xyzb*.
- Timing sheet (*Edge Set* and *Time Set* or *Time Set Basic*) must be in *Extended* mode.
- Timing sheet must also set the individual dummy pins (*xyza* and *xyzb*) to the *mux* mode by selecting *mux* in the *Pin/Group Setup* column.
- The *timing sheet* must program the edges for each dummy pin so the *odd* edges occur *before* the even edges; thus, the edges for the dummy pin mapped to the odd channel must occur before the edges for the dummy pin mapped to the even channel. The edges from both dummy pins are combined to form the actual wave. In the example, *xyzb* is mapped to the odd channel (*ch9*), so its edges occur first.

- The order of the edges means the data for the odd channel is applied to the DUT pin before the data for the even channel. In the example, the data specified for pin *xyzb* (mapped to *ch9*) is applied before the data for pin *xyza*.
- On the PIB, wire the odd-numbered channel (*ch9*) to the device pin (*xyz*). The odd-numbered channel provides the multiplexed signal. The even-channel (*ch8*) is not wired.
- Pattern file must support the *mux* mode. Teradyne recommends, but does not require that the file use a **pin_setup** statement specifying that pins *xyza* and *xyzb* are multiplexed. This setup statement permits the pin data for a single vector to be entered on two lines; refer to *Multiplex Mode* on page 6-48, *Pattern Language Reference*, Chapter 6.

## Ganged Power Supplies

### Overview

Multiple supplies (DPS) may be ganged or wired in parallel (up to 8) to provide more current than is available from a single supply.

### Requirements and Guidelines

- Pin Map sheet must declare an appropriate number of dummy pins of type *Power*. It must also declare a pin group of type *Power* containing all of these dummy pins. Note that this is the only legal use of a pin group of *Power* pins. You cannot define a *Power* pin group as a programming convenience like the other types of pin groups; any *Power* pin group must be a set of ganged power supplies. For example, *Vcc1*, *Vcc2*, and *Vcc3* are declared as individual pins, and *Vcc* is a group that contains *Vcc1*, *Vcc2*, and *Vcc3*. The device only has one physical pin *Vcc*, but the subscripted pin names are individual supplies wired to *Vcc*.
- Pin Levels sheet must define the voltage and current limits for the power supply group. A pin level is assigned only to the entire power supply group, such as example *Vcc*, not to individual pins (example, *Vcc1*, *Vcc2*, and *Vcc3*) in the group.
- *Channel Map* sheet must wire *Vcc1*, *Vcc2*, and *Vcc3* to adjacent channels of type *DPS*. The DPS channels must be within the same group of the eight DPSs.

+ If multiple sites are used, the value specified is forced on each site, not divided up among the sites.

- Ganged DPS supplies must all be on the same DPS board. Each DPS board has 8 supplies. In addition, they must be adjacent supplies. For example, *dps3*, *dps4*, and *dps5* are ganged because they are adjacent and on the same board; however, you cannot gang *dps2*, *dps4*, and *dps6* (not adjacent), or *dps7*, *dps8*, and *dps9* (*dps7* is on one board and *dps8* and *dps9* are on another board).
- A *Power* pin is used in no more than one group.
- *Isc* value must be less than or equal to *n* times the *Isc* value for a single supply, where *n* is the number of paralleled supplies.
- *Vps* or other DPS parameters may not be programmed for pins that are part of a power supply group. Only the group itself may be programmed.

# Replicating Sites and Stations

## Overview

When a *Channel Map* sheet is inserted, the columns for all sites of *Station0*, except for *Site0*, are disabled, to prevent users from entering data into these columns. In the same way, the columns for all sites of all stations greater than 0 are disabled. After you have entered data for *Site0* of *Station0* and have pressed either *Replicate Station* or *Replace Site* button, these buttons are disabled, meaning that the data in all station and sites are synchronized. Furthermore, if you edit any cell in the *Channel Site* column, these buttons become enabled, meaning that the station and site data require resynchronization. Also, if the site data in the row of the edited cell is a different color, meaning that the channel mapping was changed.

After the *Site0* column has been filled with data, pressing the *Replicate Site* button causes the column data to be replicated at other sites within the station. Likewise, pressing the *Replicate Station* button causes the data of *Station0* to be replicated at other stations.

After site entries are replaced, you can edit the cells by using the standard **Excel** methods.

## Channel Assignment

### Channel Assignments in Stations

| *Station* | *Channels* |
|---|---|
| 0 | 0 to 63 |
| 1 | 64 to 127 |
| 2 | 128 to 191 |
| 3 | 192 to 256 |
| ... | ... |
| 29 | 1856 to 1910 |
| 30 | 1920 to 1983 |
| 31 | 1984 to 2047 |

When replicating data across sites and stations, FLASH 750 assigns pins in the following manner.

### *Same Pin on Different Site of a Station*

The same pin on different sites of a station is assigned to a channel number separated by 64/number of sites per station. For example, if pin 0 of site 0 is assigned to ch07, pin 0 of site 1 is assigned:

- *ch39* if 2 sites per station are supported
- *ch23* if 4 sites per station are supported

### *DPS and HVF/HVM Pins*

For automatic fill-in of the *DPS* and *High-Voltage Unit Measure* and *Force* pins, **DataTool** sequentially assigns these types depending on the number of sites per station:

| *Site per station* | *site 0* | *site 1* | *site 2* | *site 3* |
|---|---|---|---|---|
| 1 | dps0 to dps3 | dps4 to dps7 | dps8 to dps11 | dps12 to dps15 |
| 2 | dps0 to dps1 | dps2 to dps3 | dps4 to dps5 | dps6 to dps7 |
| 4 | dps0 | dps1 | dps2 | dps3 |

The same power pin on different sites of a station is assigned to the same relative power supply within the group of power supplies for the site. For example, in the 2-site per station, if pins 39 and 40 are power pins assigned to *dps0* and *dps1* for site 0, then pins 39 and 40 of still 1 will be assigned to *dps4* and *dps5*.

| *Site per station* | *site 0* | *site 1* | *site 2* | *site 3* |
|---|---|---|---|---|
| 1 | hvf/hvm0 to hvf/hvm3 | hvf/hvm4 to hvf/hvm7 | hvf/hvm8 to hvf/hvm11 | hvf/hvm12 to hvf/hvm15 |
| 2 | hvf/hvm0 to hvf/hvm1 | hvf/hvm2 to hvf/hvm3 | hvf/hvm4 to hvf/hvm5 | hvf/hvm6 to hvf/hvm7 |
| 4 | hvf/hvm0 | hvf/hvm1 | hvf/hvm2 | hvf/hvm3 |

# Channel Map Controls

## Replicate Station

Press this control for automatic fill in of the station site data based on the data of all sites in the first station. This feature is not enabled if the tester has only 1 station.

This button is enabled unless 0 or 1 was entered in the *Number of Stations* field in *Insert Channel Map* dialog when the *Channel Map* sheet was inserted. For more information, refer to *Replicating Sites and Stations* on page 3-88.

## Replicate Site

Press this control automatic fill in of the site data based on the data of the first site is supported. This feature is not enabled if the tester has only 1 site per station.

This button is enabled unless 0 or 1 was entered in the *Number of Sites Per Station* field in *Insert Channel Map* dialog when the *Channel Map* sheet was inserted. For more information, refer to *Replicating Sites and Stations* on page 3-88.

# Channel Map Field

## DIB ID

Optional identifier for the Prober Interface Board (PIB) used with this *Channel Map*. Enter the optional ID string of the Prober Interface Board (PIB) used with this *Channel Map* in the following format:

xxx-xxx-xx

Digits are hexadecimal. Dashes are required.

By leaving the field blank or setting it to all zeros, the DIB ID is never checked.

The DIB ID value is not checked when you enter it, or when you validate the program; however, it is checked each time you try to run a test program online. The DIB ID in this field is checked against the ID of the PIB on the tester. If the IDs do not match, an error is generated, and the test program cannot be executed. Note you can change the PIB without turning off power to the test system.

# Channel Map Columns

## Device Under Test

### Device Under Test Pin Name

Enter the required name of a pin specified on the Pin Map sheet. You cannot enter pin group names in this column. When you create the *Channel Map*, you are prompted to let **DataTool** fill in the pin names from the *Pin Map* sheet; refer to *Inserting a New Channel Map* on page 3-83.

**DataTool** has additional requirements for entering the name of a pin used in the *mux* mode or with ganged power supplies:

- If the mapped pin uses the *Mux Mode*, enter the two pin names paired on successive rows (example, *xyza* and *xyzb*) and map them to an even-odd pair of adjacent channels in the site column, such as channels *ch6* and *ch7*. For more information, refer to *Multiplexed Pins* on page 3-85.
- If a power supply pin must be wired to multiple power supplies so that it receives the required current, enter the names of the supply group pins on successive rows and enter the device supply names (*dps0* to *dps32*) under the site column. For more information, refer to *Ganged Power Supplies* on page 3-87.

### Device Under Test Package Pin

Enter an optional alternate name for the pin. This name could be the physical package pin number, bonding pad name, or any string or number. The entered value can be displayed as part of the datalog data.

## Channel

### Channel Type

Describes the characteristics of the tester hardware resource wired to this pin. This information is compared during validation to the type of device pin. Enter a valid channel type in this required cell or select a channel type from the drop-down list:

- *I/O*—bi-directional digital channel
- *DPS*—device power supply
- *Gnd*—ground channel
- *HVF*—High-voltage force
- *HVM*—High-voltage measure
- *Utility*—utility bit.
- *N/C*—unused or channel not connected. Any pin can be assigned to a channel of type *N/C*, in which case the information in the *Site* columns is ignored.

When creating a *Channel Map*, you are prompted to let **DataTool** fill in the channel type, based on the pin type; refer to *Inserting a New Channel Map* on page 3-83.

### *Channel Site*

Enter the channel name, power supply name, high-voltage measure/force, or utility bit to which the pin to wired to.

Every *Channel Map* has at least a *Site 0*. Additional sites are for parallel testing. When you insert the *Channel Map*, you specify the number of sites; refer to *Inserting a New Channel Map* on page 3-83. Sites are named *site0* to *site127*, while stations are numbered 0 to 31.

All site columns must be filled in, meaning that if you created a sheet for three sites, you must map the channels for three sites, even if only two sites are actually tested. To specify how many sites are tested, refer to *Run Options Dialog* on page 3-56.

Use the following site naming requirements, which correspond to the value entered in the *Channel Type* column:

- I/O—*ch0* to *ch2047*
- DPS—*dps0a* to *dps127a* and *dps0b* to *dps127b*. Each DPS has two outputs, designated *a* and *b*.
- *Gnd*—Ground pins are not mapped to channels. If the *Type* column is *Gnd*, the *Site* column is grayed out. A channel number is not required.
- *HVF*—high-voltage force—*hvf0* to *hvf127*
- *HVM*—high-voltage measure—*hvm0* to *hvm127*
- Utility—*u0* to *u127*

## *Comment*

Enter any optional notes or comments as a string.

# Pin Map Sheet

## Overview

This sheet defines the device pin names and pin group names and their grouped pins.

+ Because the pins and pin groups are referred by other **DataTool** sheets and are used by the pattern compiler and other tools, Teradyne recommends that you first create the *Pin Map* sheet when building a new test program.

## Using the Pin Map Sheet

### Pin Map Concepts

#### Key Concept: Symbolic Pin Name

You should understand the difference between the *Pin Map* and the *Channel Map* because a pin on a DUT may have several designators:

- *symbolic pin name*—typically related to the pin function, such as *Clk*, *D14*, *A0*, or $V_{CC}$.
- *package pin* or *bonding pad designator*—*AF13*
- pin wired to the *tester channel*—*ch27*

The *Pin Map* sheet defines the symbolic pin name, while the package pin and tester channel are defined by the Channel Map sheet. Also, refer to *Key Concept: Pin Map and Channel Maps* on page 3-82.

#### Number of Pin Maps

**DataTool** creates an empty *Pin Map* sheet as part of a new test program workbook. At least one *Pin Map* is required for each test program. If a workbook contains several *Pin Map* Sheets; the workbook must have a Job List sheet to specify a *Pin Map* for a specific job.

#### Sources for Pin Map Information

Data for this sheet is usually entered from information on specification sheets and from the pin list and group names used in creating the test patterns.

### Editing the Pin Map Sheet and Recompiling Test Patterns

Pattern files and other worksheets refer to the pin and group names defined on this sheet. Consequently, certain edits to the *Pin Map* sheet require that you recompile the test patterns. For instance, removing or reordering pins or redefining a pin group requires that you recompile the test program. On the contrary, if you add pins to a pin map, you do not have to recompile the test patterns.

### Defining Pins Before Defining Pin Groups

You must define all pins before any pin groups are defined, meaning that the *Pin Map* rows defining the pins must come before the rows defining the pin groups.

### Data Entry Verification

**DataTool** does not verify that every device pin has been entered in the *Pin Map*.

### Displaying the Pin Map Data

Pin groups with more than one pin can be expanded and collapsed; refer to *Collapsing and Expanding Groups* on page 3-35.

You can filter the *Pin Name* column to identify all groups belonging to a given name; refer to *Filtering Data in a Column* on page 3-37.

### Utility Bits

Pins on this sheet usually correspond to physical ports of the DUT, but this sheet can define any *Utility Bits* programmed as part of the test. The *Type* column supports the value *Utility* for *Utility Bits*; refer to *Type* on page 3-99.

### Parallel Testing

Devices tested in parallel require only a single *Pin Map* sheet because the Channel Map sheet specifies the package pins and the handler wiring for single device or parallel device testing.

## Pin Group Concepts

### Pin Group Rules

- All pins must be defined before any pin groups are defined.

- A pin group contains pins, groups, or a mixture of pins and groups; refer to *Restrictions for Nesting Pin Groups* on page 3-96.
- Pins in a group must be of the same type as the type of the group. For example, if the group type is *Input*, all pins in the group must also be type *Input*. Exception is the *I/O* type, which supports digital pins of mixed types: *Input*, *Output*, and *I/O*; refer to *Type* on page 3-99.
- An individual pin can be a member of any number of pin groups, but a pin can appear only once in a given pin group.
- Pin and pin group cannot have the same name.

### Pin Group of Power Pins

Pin groups of type *Power* have a unique **IG-XL** function, which is a pin group that gangs the power supplies; refer to *Ganged Power Supplies* on page 3-87. You cannot define a *Power* pin group simply as a programming convenience like the other types of pin groups.

### Mux Mode

A pin group of two *dummy* pins is required for multiplexed channels; refer to *Multiplexed Pins* on page 3-85.

### Restrictions for Nesting Pin Groups

A pin group can contain other pin groups. This nesting of pin group definitions is useful in defining large buses that consist of smaller buses. Note the following restrictions:

- Pin group must not contain a circular reference: such as, *Group A* includes *Group B*, and *Group B* includes *Group A*.
- Pin group must be defined before it can be nested in other group definitions.
- A pin may be mentioned only once in a given group definition, which disallows groups with repeated pins.

### Pin Groups as Pin Synonym

You can implement a pin synonyms by defining a pin group that contains one pin; thus, the group name is effectively a synonym.

### Creating a Pin Group

1. In the *Group Name* column, enter a name for the group.
2. In the *Pin Name* column for this row, enter the most significant bit of the group or the name of a previously defined pin group.

3. In the *Type* column for this row, enter a type for the group; be sure to follow the guidelines in *Pin Group Rules* on page 3-95.
4. Enter successive bits in the group on subsequent rows in the *Pin Name* column. until the least significant bit is reached. The group name column is filled in as you add pins to the group.
5. You can use the **Excel** *AutoFill* feature to fill in the pins in a bus; refer to *Using Excel AutoFill* on page 3-97.

## Using Excel AutoFill

You can quickly generated the pins for buses by using the Excel *Autofill* functions or the *Fill* command on the **IG-XL** *Edit* menu. For example, enter a name like *p1* in a cell, and use the *Fill* or *AutoFill* command to create *p2*, *p3*, and the sequential names. Note that you use *AutoFill* by clicking on the lower right corner of the cell and dragging down.

You can also define two cells to establish a pattern that *AutoFill* will continue. For example, if you create *p2* and *p4* and highlight both cells, *AutoFill* will continue with *p6*, *p8*, and so on. Or, if you create *p10* and *p9*, *AutoFill* will continue with *p8*.

# Pin Map Columns

## Group Name

Enter the name of the pin group to be created, such as *Dbus*; otherwise, if it is an individual pin, leave the cell blank. Group names are strings. They are **bolded** to distinguish them from pin names. Also, refer to *Pin Group Concepts* on page 3-95.

## Pin Name

Enter the user name for the defined pin. If it is part of a pin group definition, enter a pin name or pin group name that has already been defined on an earlier row of this sheet. Pin names are strings and are required. For example, enter *Q7*, *vcc*, or *A13*. When defining pins, Teradyne recommends entering the pins in an sequential order, ensuring that you define all pins.

+ **Excel** *Autofill* function is useful in generating pin names; refer to *Using Excel AutoFill* on page 3-97.

## Type

Describes the type of DUT pin, not the tester and its resources, which are defined by the Channel Map. This value is checked during validation of the test program. The string value for this cell is required for a pin definition or for the first row of a pin group definition. Except for the *I/O* pin, all pins in a pin group must be of the same type as the group; refer to *Pin Group Rules* on page 3-95.

Enter a valid pin type or group in this required cell or select a pin type or group from the drop-down list:

- *I/O*—bi-directional digital pin or pin group
- *Input*—digital pin or pin group, input only
- *Output*—digital pin or pin group, output only
- *Analog*—analog pin or pin group
- *HVF*—High-voltage force pin or pin group
- *HVM*—High-voltage measure pin or pin group
- *Power*—power supply pin or pin group. A pin group of type *Power* is for ganging the tester power supplies; it is not a programming convenience like other types of pin groups; refer to *Ganged Power Supplies* on page 3-87.
- *Gnd*—ground pin
- *Utility*—Pin or pin group with *type utility*; refer to *Utility Bits* on page 3-95.
- *Unknown*—Pin with unknown *type*. This type can be used for pins named *N/C* that have no function; these pins could also be set to *Gnd*. On the *Channel Map*, set the corresponding channel type to *N/C*.

## Comment

Enter any optional notes or comments as a string.

# MTM Resource Map

## Overview

This sheet maps the resources of the FLASH 750 Memory Test Module (MTM) to the DUT pins by specifying which MTM resource is sent to a specified DUT pin and which DUT pins are sending data back to the MTM. It shows inputs as X and Y physical addresses and outputs as tester channels.

At the hardware level, this sheet configures the Alternate Data Bus (ADB) Interface, which connects the 64 drive and expect data and the physical X and Y address bits to the ADB bits and the 36 error capture data bits from the ADB bits.

## Using the MTM Resource Map Sheet

### Multiple MTM Resource Maps

This sheet is required for all jobs. A job can specify multiple named resource maps. You select which resource map to use for a given test instance on the Test Instances sheet. Also, refer to *Inserting a New MTM Resource Map* on page 3-101.

### MTM Resource Categories

The MTM resource maps have two categories:

- Source—X address, Y address, or data sent to a DUT pin.
- Capture—which DUT pins are sending back data to the MTM. Only one set of capture information may be specified.

## Inserting a New MTM Resource Map

One of these sheets is required for all jobs; however, you can create resource maps:

1. On the *Insert* menu, select the *Worksheet* command. The *Insert Worksheet* popup appears.
2. Under *Sheet Type*, select *MTM Resource Map*. Under *Sheet Name*, enter the name of the sheet. Click *OK*. The *Insert MTM Resource Map* popup appears.
3. On the *Insert MTM Resource Map* popup:
   a. To use the pin fill-in option, check the checkbox *Fill with pins from the Pin Map*. Click *OK*.
   b. To manually enter the pins names, uncheck the checkbox *Fill with pins from the Pin Map*. Click *OK*.

## Rules

- Only digital pins may be referenced on this sheet.
- Pins with entries in the *Source* column must be *DUT Input* or *I/O* pins.
- Pins with entries in the *Capture* column must be DUT *Output* or *I/O* pins (per-map).
- Each pin in *Pin Name* column has an entry in at least one of the other columns (per-map).
- Each row containing capture information must be unique. For example, 2 rows cannot contain *C2*.

## Displaying the Data

Each resource map, which contains a row for each pin, can be expanded and collapsed; refer to *Collapsing and Expanding Groups* on page 3-35.

# MTM Resource Map Columns

## Map Name

Enter the name of the resource map to create. The name is repeated in gray on each subsequent line of the map as you add pins. Entering a new name in this column creates a new resource map. These required names are strings.

## Pin Name

On successive rows, enter a valid DUT pin to be mapped to an MTM resource in this map or select a pin from the drop-down list. As you enter each pin name after the first, **DataTool** adds the map name to the row.

Only digital pins (*Input*, *Output*, *I/O*) are used; pin group names are not allowed. The pin name should match one of the entries on the *Channel Map* and *Pin Map* sheets. Map name can be used only once.

You can use the pin fill-in option to enter the pin names, refer to *Inserting a New MTM Resource Map* on page 3-101.

## MTM Resource

### MTM Resource Source

Enter a string in this required field or select from the pull-down menu the MTM resource to send to the pin specified on this row of the resource map:

- blank—Empty cells are considered unused
- *Patt*—MTM pattern opcode
- *X0* to *X15*—X Scramble RAM
- *Y0* to *Y15*—Y Scramble RAM
- *D0* to *D31*—Data Generator
- *FS0* to *FS15*—Frame Select output

Within each resource map, at least one cell must have a source value.

### MTM Resource Capture

Enter a string in this optional field or select from the pull-down menu the bit to capture from the pin specified on this row of the resource map:

- blank—Empty cells are considered unused
- *C0* to *C35*

Within each resource map, each row containing capture information must be unique: for example, two rows cannot contain *C2*.

## Comment

Enter any optional notes or comments as a string.

# Errors Sheet

## Overview

This sheet displays any errors found during validation, including active links to the sheet on which the errors occurred. It is the far-right sheet. If a test program had no validation errors, this sheet is not created.

+ This sheet is different than the *Error Sources* sheet, which specifies the conditional branching in a test program; refer to *Error Sources Sheet* on page 3-110.

## Using the Errors Sheet

### Creating and Deleting the Errors Sheet

**DataTool** creates this sheet if it discovers errors while validating the test program. You cannot insert, edit, or export this sheet. it. When you validate a program, any existing *Errors* sheet is deleted, and another is created if errors are found during validation.

### Displaying Multiple Errors

If a sheet has more than one error, it is displayed on the *Errors* sheet as a collapsed group. To expand it, refer to *Collapsing and Expanding Groups* on page 3-35.

### Test Instances Errors

Errors discovered in a test instance during validation are displayed on this sheet with the test instance name followed by the name of the *Test Instances* sheet:

*sheet-name*.*test-name*

For example, EngrTests.Continuity means a test instance named *Continuity* and a *Test Instances* sheet named *EngrTest*.

### Active Links

The error source is displayed as the sheet name or test name, which is an active link to the sheet, and where possible, to the cell. The links behave differently depending on the type of sheet and the error source:

- For a sheet other than the Test Instances sheet, click on the link to go to the sheet and to the cell where the error occurred the active cell. By clicking on the link, any collapsed groups or hidden columns are exposed to make the active cell visible.
- For an error in a test instance, click on the link to go to the active *Test Instances* sheet, where the active cell is the button for invoking the test instance editor. After clicking this button, the instance editor opens and displays in color the fields causing the error.
- For errors caused by missing information, click on the link to go to the appropriate sheet; the active cell is at the bottom of the data area.

## Errors Sheet Columns

### Sheet

Displays the name of the sheet with the error. Name is a hypertext link; refer to *Active Links* on page 3-105.

### Error Code

Displays the Teradyne error code for the error.

### Error Message

Briefly describes the error.

# ECR/DBM Configuration Sheet

## Overview

This sheet is required for capturing errors with the Error Catch RAM (ECR) or for using the data buffer memory (DBM). Even though only one sheet may exist, and most users specify only one configuration, you can create another ECR/DBM configuration sheet; refer to *Inserting a New ECR/DBM Configuration Sheet* on page 3-107.

Even though the MTM Resource Map sheet specifies the resources for all devices, only the *ECR/EBM Configuration* sheet specifies the devices using the ECR/DBM resources. It specifies the device specific information needed to setup the DBM and ECR **amode** and ECR **dmode** hardware logic.

# Using the ECR/DBM Configuration Sheet

+ For more information about the ECR, refer to *Error Catch RAM (ECR) and Data Buffer Memory (DBM)*, Appendix C.

## Number of Address and Data Bits

If an ECR is used, these are the number of address and data bits after the data and address bits have been compressed. When **DataTool** validates this information, it will flag an error during data validation if the data specified on this sheet:

- Requires more than 128 Mb (or 144 Mb for the ECR) of memory.
- Illegal X or Y address, or data value.

During job validation, the specified address and data values are verified with the entries in the MTM Resource Map sheet for the job.

## Inserting a New ECR/DBM Configuration Sheet

1. Select *Worksheet* command from *Insert* menu. *Insert Worksheet* dialog opens.
2. On the I*nsert Worksheet* dialog:
   a. In the *Sheet Type* field, use to pull-down to select *ECR/DBM Configuration*.
   b. In the *Sheet Name* field, enter the appropriate name of this new sheet.
   c. Click *OK*. Another *ECR/DBM Configuration* sheet is inserted.

## Data Compression

The data compression ratios for the device addresses and the data are listed on this sheet. Data is compressed when it is acquired during redundancy analysis. Data compression creates a more compact error data image, which is useful only when redundancy analysis treats multiple data values as a unit.

The data compression values on this sheet set the **CMODE**, the ECR data compression mode; refer to Table 3-1 on page 3-108.

Table 3-1. ECR CMODE

| DUT Data Width | Data Compression | Parity Compression | Overall Compression | Address Space |
|---|---|---|---|---|
| 1 | 1:1 | N/A | 1:1 | 128 M |
| 2 | 2:2 | N/A | 2:2 | 64 M |
| | 2:1 | N/A | 2:1 | 128 M |
| 4 | 4:4 | N/A | 4:4 | 32 M |
| | 4:2 | N/A | 4:2 | 64 M |
| | 4:1 | N/A | 4:1 | 128 M |
| 8 | 8:8 | N/A | 8:8 | 16 M |
| | 8:4 | N/A | 8:4 | 32 M |
| | 8:2 | N/A | 8:2 | 64 M |
| | 8:1 | N/A | 8:1 | 128 M |
| 8+1 | 8:8 | 1:1 | 9:9 | 16 M |
| | 8:1 | 1:1 | 9:2 | 64 M |
| | 9:1 | Fold to Data Field | 9:1 | 128 M |
| 16 | 16:16 | N/A | 16:16 | 8 M |
| | 16:8 | N/A | 16:8 | 16 M |
| | 16:4 | N/A | 16:4 | 32 M |
| | 16:2 | N/A | 16:2 | 64 M |
| | 16:1 | N/A | 16:1 | 128 M |
| 16+2 | 16:16 | 2:2 | 18:18 | 8 M |
| | 16:8 | 2:1 | 18:9 | 16 M |
| | 16:2 | 2:2 | 18:4 | 32 M |
| | 16:1 | 2:1 | 18:2 | 64 M |
| | 18:1 | Fold to Data Field | 18:1 | 128 M |
| 32 | 32:32 | N/A | 32:32 | 4 M |
| | 32:16 | N/A | 32:16 | 8 M |
| | 32:8 | N/A | 32:8 | 16 M |
| | 32:4 | N/A | 32:4 | 32 M |
| | 32:2 | N/A | 32:2 | 64 M |
| | 32:1 | N/A | 32:1 | 128 M |
| 32+4 | 32:32 | 4:4 | 36:36 | 4 M |
| | 32:16 | 4:2 | 36:18 | 8 M |
| | 32:8 | 4:1 | 36:9 | 16 M |
| | 32:4 | 4:4 | 36:8 | 16 M |
| | 32:2 | 4:2 | 36:4 | 32 M |
| | 32:1 | 4:1 | 36:2 | 64 M |
| | 36:1 | Fold to Data Field | 36:1 | 128 M |

# ECR/DBM Configuration Columns

## Configuration Name

Enter the name of the ECR or DBM configuration to be defined. The required name is a string; refer to *Rules for User-Created Names* on page 3-25.

## Number of Bits (After Compression if ECR)

### Number of Bits X Address

Enter the number of X address bits (post address compression for ECR). Valid values for the *X Address*: 1 to 16. Required entry is a string.

### Number of Bits Y Address

Enter the number of Y address bits (post address compression for ECR). Valid values for *Y Address:* $((2^N - (X\text{-}Address))$ modulo 15). Required entry is a string.

### Number of Bits Data

Enter the number of data bits (post data compression for ECR). Required entry is a string. Valid values: $2^N - (X\ Address) - (Y\ Address)$. $2^N$ is ECR memory capacity.

## Data Compression

Use the drop-down list to select one of the following data compression ratios for the X- and Y-addresses and data:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1->1 | 2->1 | 4->1 | 8->1 | 9->1 | 16->1 | 18->1 | 32->1 | 36->1 |
| | 2->2 | 4->2 | 8->2 | 9->2 | 16->2 | 18->2 | 32->2 | 36->2 |
| | | 4->4 | 8->4 | 9->9 | 16->4 | 18->4 | 32->4 | 36->4 |
| | | | 8->8 | | 16->8 | 18->9 | 32->8 | 36->8 |
| | | | | | 16->16 | 18->18 | 32->16 | 36->9 |
| | | | | | | | 32->32 | 36->18 |
| | | | | | | | | 36->36 |

## Comment

Enter any optional notes or comments as a string.

# Error Sources Sheet

## Overview

This sheet specifies the ECR counter compare results that contribute to pattern rate branching.

Even most users specify only one sheet, you can create another sheet; refer to *Inserting a New Error Source Sheet* on page 3-111.

## Using the Error Sources Sheet

### Overview of ECR Counters

+ For more information about branch-on-error, refer to *Error Catch RAM (ECR) and Data Buffer Memory (DBM)*, Appendix C.

FLASH 750 has 5 ECR counters with copies for each of 2 DUTs, with a total of 10 counters:

- Total Error Counter (TEC)
- Row Error Counter (REC)
- Column Error Counter (CEC)
- BitLine Counter (BLC)
- WordLine Counter (WLC)

Each of these 10 counters has a maximum compare register. Each maximum compare generates a counter fail flag. You can conditionally enabled these 10 fail flags and use **AND**/**OR** logic on these enabled results to produce:

- any of the 10 compare results at pattern rate
- a single ECR fail flag that is routed back to the Failbus for pattern rate branching.
- cleared counters and results at pattern rate

The pattern rate control is the 3-bit SVM field from the Pattern Generator, providing 3 error source states. Each error source state includes 12 bits, defining the conditional enables for the 10 counter compares, an enable to clear all counters, and an enable for the fail registers for DUT0 and DUT1. The Error Sources sheet allows you to mix and match the fail conditions of the 10 counters for pattern branching.

### Inserting a New Error Source Sheet

1. Select *Worksheet* command from *Insert* menu. *Insert Worksheet* dialog opens.
2. On the I*nsert Worksheet* dialog:
   a. In the *Sheet Type* field, use to pull-down to select *ECR/DBM Config*.
   b. In the *Sheet Name* field, enter the appropriate name of this new sheet.
   c. Click *OK*. Another *ECR/DBM Config* sheet is inserted.

## Error Sources Columns

### Set Name

Enter the name of the ECR source set to be defined. The required name is a string; refer to *Rules for User-Created Names* on page 3-25.

### State ID

Enter in the cell or use the drop-down list to select one of the 5 ECR setups for the associated *Set Name*: *ECR_BOE0*, *ECR_BOE1*, *ECR_BOE2*, *ECR_BOE3*, or *ECR_BOE4*. This required name is a string.

### DUT0/DUT1

#### DUT0/DUT1 TEC

Enter in the cell or use the drop-down list to select the total error count (TEC) to trigger branch-on-error for the DUT. The required string is the integer 0 for disable or 1 for enable.

#### DUT0/DUT1 REC

Enter as a number the maximum row error count (REC) to trigger branch-on-error for the DUT. The required string is the integer 0 for disable or 1 for enable.

#### DUT0/DUT1 CEC

Enter in the cell or use the drop-down list to select the maximum column error count (CEC) to trigger branch-on-error for the DUT. The required string is the integer 0 for disable or 1 for enable.

#### DUT0/DUT1 WLC

Enter in the cell or use the drop-down list to select the maximum word line count (WLC) to trigger branch-on-error for the DUT. The required string is the integer 0 for disable or 1 for enable.

#### DUT0/DUT1 BLC

Enter in the cell or use the drop-down list to select the maximum bit line count (BLC) to trigger branch-on-error for the DUT. The required string is the integer 0 for disable or 1 for enable.

## Comment

Enter any optional notes or comments as a string.

# Redundancy Table Sheet

## Overview

This sheet defines the values for the redundancy table parameters that initialize the redundancy table used for redundancy analysis of a device. The redundancy table is a software structure containing device parameters that describe the device topology. These parameters are used by the redundancy software to perform the analysis.

Consequently, this sheet is required only when a test uses redundancy analysis; however, multiple *Redundancy Table* sheets are supported. Users specify the redundancy table for a test by specifying it in the *Test Instances* sheet or the *Test Instance Editor* of the functional test template.

Even most users specify only one sheet, you can create another sheet; refer to *Inserting a New Error Source Sheet* on page 3-111.

## Using the Redundancy Table Sheet

For information about redundancy analysis, refer to *Redundancy Analysis*, Chapter 11.

### Inserting a New Redundancy Table Sheet

1. Select *Worksheet* command from *Insert* menu. *Insert Worksheet* dialog opens.
2. On the I*nsert Worksheet* dialog:
   a. In the *Sheet Type* field, use to pull-down to select *Redundancy Table*.
   b. In the *Sheet Name* field, enter the appropriate name of this new sheet.
   c. Click *OK*. Another *Redundancy Table* sheet is inserted.

# Redundancy Table Columns

## Segment

Enter the segment number for all parameters on this row that apply to this segment. Enter from 0 to 255.

## Segment Row

### First Segment Row

Enter as an integer the first segment row of the address range for redundancy processing. Applies to individual segments.

### Last Segment Row

Enter as an integer the last segment row of the address range for redundancy processing. Applies to individual segments.

## Segment Column

### First Segment Column

Enter as an integer the first segment column of the address range for redundancy processing. Applies to individual segments.

### Last Segment Column

Enter as an integer the last segment column of the address range for redundancy processing. Applies to individual segments.

## Data Mask

Enter the mask of data bits replaced by the segment's redundancy element.

## Linkage

### Row Linkage

For each segment linked to another segment by row, set to the number of the segment to which it is linked. For each segment not linked to another segment by row, set to its number. Also, refer to *Linkage* on page 11-13.

### Column Linkage

For each segment linked to another segment by column, set to the number of the segment to which it is linked. For each segment not linked to another segment by column, set to its number; refer to *Linkage* on page 11-13.

## Differential

### Rows Differential

If the segments have unequal number of rows, enter the initial differential number of spare rows for the segment of the device. Applies to individual segments. Also, fill the checkbox for the *Unequal R (row)/C (column) resources per segment* flag; refer to *Flags* on page 3-117.

### Columns Differential

If the segments have unequal number of columns, enter the initial differential number of spare columns for the segment of the device. Also, fill the checkbox for the *Unequal R (row)/C (column) resources per segment* flag; refer to *Flags* on page 3-117.

## Global Row

### First Global Row

Enter the first row of the device, considering all segments.

### Last Global Row

Enter the last row of the device, considering all segments.

## Global Column

### First Global Column

Enter the first column of the device, considering all segments.

### Last Global Column

Enter the last column of the device, considering all segments.

## # of Segments

Enter the number of segments for the device.

## Row Spares

### Row Spares Number

Enter the total number of spare rows available for replacement, if any. Applies to all segments. Also, fill the checkbox for the *Spare Row Per Data* flag; refer to *Flags* on page 3-117.

### Row Spares # Used

Enter the total number of rows that must be replaced with each spare element, if any. Applies to all segments.

## Column Spares

### Column Spares Number

Enter total number of spare columns available for replacement, if any. Applies to all segments. Also, fill the checkbox for the *Spare Col Per Data* flag; refer to *Flags* on page 3-117.

### Column Spares # Used

Enter the total number of columns that must be replaced with each spare element, if any. Applies to all segments.

## Flags

These user preferences control the redundancy process and specify what results are recorded during the redundancy processing. Click the three-dot button ... in this cell to open the *Redundancy Analysis Options* window. Select a flag option by checking one or more of the following checkboxes:

| Flag | if checked | if not checked |
|---|---|---|
| Row Preference | Use rows first (if available) in assigning repairs. | Use columns first (if available) in assigning repairs. |
| Discretionary | After worst row and column are found, replace the worst, assuming both types of spares are available, regardless of the *Row Preference* selection. If worst row and worst column have same number of bad locations, replace type specified by *Row Preference.* | Row preference selection is absolute: use the specified element type (row or column) until no more are available, before using any elements of the other type except for a must-replace element, which is always replaced first. |
| Physical Rows and Columns | Record the physical rows and physical columns. After the analysis, the Bitmap Scramble RAMs determine the physical row to record. | Record the logical rows and logical columns. |
| Parity | Use segment bit masks for non-parity and parity I/O for non-parity I/Os. | Use segment bit masks for non-parity I/Os. |
| Row Results w/o Linkage | Record the row of every segment containing the bad row regardless of the row linkage. | Record the row of one segment among the linked segments if more than one linked segment contains the same bad row. |
| Column Results w/o Linkage | Record the column of every segment containing the bad column regardless of the column linkage. | Record the column of one segment among the linked segments if more than one linked segment contains the same bad column. |
| Unequal R/C Resources | Use the initial differential number of spare rows and columns for the segments for devices having unequal spares per segment. | Use the total number of spare rows and columns for devices having the same number of spares per segment. |
| Continue Analysis to End | Continue redundancy analysis despite the unrepairable segment until all segments are analyzed. | Return immediately if the redundancy analysis routine finds an unrepairable segment. |

| Flag | if checked | if not checked |
|---|---|---|
| Spare Row Per Data | Limit the recommended row repair to a single segment for linked row segments. | Recommended row is repaired at a given address in a linked row segments. |
| Spare Column Per Data | Limit the recommended column repair to a single segment for linked column segments. | Recommended column is repaired at a given address in a linked column segments. |

## *Comments*

Enter any optional notes or comments as a string.

# SDA Table Sheet

## Overview

This sheet defines the SDA (statistical defect analysis) table parameters that initialize the SDA table used for analyzing a device for defects. The SDA table is a software structure containing device parameters that describe the device defects. These parameters are used by **IG-XL** to perform the defect analysis.

Consequently, this sheet is required only when a test uses SDA; however, multiple *SDA Table* sheets are supported. To create another *SDA Table* sheet; refer to *Inserting a New SDA Table Sheet* on page 3-119. Users specify the SDA table for a test by specifying it in the *Test Instances* sheet or the *Test Instance Editor* of the functional test template.

## Using the SDA Table Sheet

### Inserting a New SDA Table Sheet

1. Select *Worksheet* command from *Insert* menu. *Insert Worksheet* dialog opens.
2. On the I*nsert Worksheet* dialog:
   a. In the *Sheet Type* field, use to pull-down to select *SDA Table*.
   b. In the *Sheet Name* field, enter the appropriate name of this new sheet.
   c. Click *OK*. Another *SDA Table* sheet is inserted.

# SDA Table Columns

## Segment

Enter the SDA segment number for all parameters on this row that apply to this segment. Enter from 0 to 255. Applies to individual segments.

## Segment Row

### First Segment Row

Enter as an integer the first segment row of the address range for SDA processing. Applies to all segments

### Last Segment Row

Enter as an integer the last segment row of the address range for SDA processing.

## Segment Column

### First Segment Column

Enter as an integer the first column segment of the address range for SDA processing. Applies to individual segments.

### Last Segment Column

Enter as an integer the last column segment of the address range for SDA processing. Applies to individual segments.

## Data Mask

Define the data mask for the segment by entering the SDA segments of the device not to be processed during SDA. If more than more segment, specify as a comma-delimited list. Applies to individual segments.

## Linkage

### Row Linkage

For each segment linked to another segment by row, set to the number of the segment to which it is linked. For each segment not linked to another segment by row, set to its number. Also, refer to *Linkage* on page 11-13.

### Column Linkage

For each segment linked to another segment by column, set to the number of the segment to which it is linked. For each segment not linked to another segment by column, set to its number; refer to *Linkage* on page 11-13.

## Minimum Row Errors

Enter the minimum number of error bits to consider a row is bad for the segment. Applies to individual segments.

## Minimum Column Errors

Enter the minimum number of error bits to consider a column is bad for the segment. Applies to individual segments.

## Minimum Rows Per Bad Segment

Enter the minimum number of bad rows to consider a segment is bad for the row. Applies to individual segments.

## Minimum Columns Per Bad Segment

Enter the minimum number of bad columns to consider a segment is bad for the column. Applies to individual segments.

## Minimum Errors Per Bad Segment

Enter the minimum number of allowable bad bits per segment. Applies to individual segments.

## Global Row

### First Global Row

Enter the first row for SDA processing, considering all segments.

### Last Global Row

Enter the last row for SDA processing, considering all segments.

## Global Column

### First Global Column

Enter the first column for SDA processing, considering all segments.

### Last Global Column

Enter the last column for SDA processing, considering all segments.

## # of Segments

Enter the number of segments for SDA processing.

## Row Spares

### Row Spares Number

Enter the total number of spare rows that must be swapped together.

### Row Spares Number # Used

Enter the total number of rows that must be replaced with each spare element, if any.

## Column Spares

### Column Spares Number

Enter the total number of spare columns that must be swapped together.

### Column Spares Number # Used

Enter the total number of columns that must be replaced with each spare element, if any.

## Flags

These user preferences control the SDA processing and specify what results are recorded during the SDA processing. Click the three-dot button ... in this cell to open the *SDA Options* window. Select a flag option by checking one or more of the following checkboxes:

| Flag | if checked | if not checked |
|---|---|---|
| Row First | Analyze rows before columns. | Analyze columns before rows. |
| Columns First | Analyze columns before rows. | Analyze rows before columns. |
| Analyze Rows | Record the row results. | Do not record the row results. |
| Analyze Columns | Record the column results. | Do not record the column results. |
| Analyze Rows Excl | Assumes *Columns First* is selected. Analyzes only those rows whose errors are not already accounted for in a bad column. | Analyzes all rows regardless of whether their errors have been accounted for in a bad column. |
| Analyze Cols Excl | Assumes *Rows First* is selected. Analyzes only those columns whose errors are not already accounted for in a bad row. | Analyzes all columns regardless of whether their errors have been accounted for in a bad row. |
| Report XY | Report only addresses of bad points. | |
| Report XYD | Reports only addresses and data of bad points. | Reports only addresses data of bad points. |
| Avg Data Value | Reports average data value by address segment. | No average data value recorded. |
| Min Data Value | Reports minimum data value by address segment. | No minimum data value recorded. |
| Maximum Data Value | Reports maximum data value by address segment. | No maximum data value recorded. |
| Sigma Value | Reports 1 sigma of data values by address segment. | No sigma value recorded. |
| Clear Rows After Analysis | Clears bad rows after analysis for speed enhancement. | Do not clear bad rows after analysis. |
| Clear Cols After Analysis | Clears bad columns after analysis for speed enhancement. | Do not clear bad columns after analysis. |

| Flag | if checked | if not checked |
|---|---|---|
| Physical Rows and Columns | Record the physical rows and physical columns. After the analysis, the Bitmap Scramble RAMs determine the physical row to record. | Record the logical rows and logical columns. |
| Analysis to End | Continue SDA despite the unrepairable segment until all segments are analyzed. | Return immediately if the SDA routine finds an unrepairable segment. |
| Spare rows Fix One Seg | Spare row fixes all I/Os in only one segment in a set of linked row segments. No meaning if segments are not linked by rows. | Spare row fixes all I/Os in all segments in a set of linked row segments. |
| Spare Cols Fix One Segment | Spare column fixes all I/Os in only one segment in a set of linked column segments. No meaning if segments are not linked by columns. | Spare column fixes all I/Os in all segments in a set of linked column segments. |
| Count Errs Once Per Row | Count errors once per row regardless of data width. | Count each fail bit of errors in a row. |
| Count Errs Once Per Col | Count errors once per column regardless of data width. | Count each fail bit of errors in a column. |

## *Comments*

Enter any optional notes or comments as a string.

# AC Specs Sheet

## Overview

This sheet defines the spec symbols used in creating formulas on other sheets. These symbols define timing values. Also, refer to *DC Specs Sheet* on page 3-133.

For the fundamentals about *AC Specs* sheets, refer to *Spec Sheets* on page 3-38.

+ Teradyne highly recommends that you use spec sheets to define spec symbols because they combine the key values used on the other sheets, thereby making it easier for you to manage your test program.

# Using the AC Specs Sheet

## Inserting and Editing a New AC Specs Sheet

1. Select the *Worksheet* command from the *Insert* menu. The *Insert Worksheet* dialog opens.
2. On the I*nsert Worksheet* dialog:
   a. In the *Sheet Type* field, use to pull-down to select *AC Specs*.
   b. In the *Sheet Name* field, enter the appropriate name of this new sheet.
   c. Click *OK*. The *Insert AC Spec Sheet* dialog opens.
3. On the *Insert AC Specs Sheet* dialog:
   a. In the *Initial Number of Categories* field, enter a number from 1 to 40; default is 3.
   b. Click *OK*.
4. A new *AC Specs* sheet is inserted. The categories are created with default names. You can rename, add, or delete categories later. Before you can edit the new sheet, it must be part of the current *Active Job*.
5. On the *Job List* sheet:
   a. Go to the row for the job that will use the new *AC Specs* sheet.
   b. From the drop-down list in the *AC Specs* column, select the new *AC Specs* sheet.
6. On the **IG-XL** *Context* toolbar, select the new *AC Specs* sheet as the *Active Job*; refer to *IG-XL Context Toolbar* on page 3-9. You can now edit the new sheet.
7. On the new *AC Specs* sheet:
   a. Enter data from the manufacturer's device specification sheets.
   b. A default *selector* will be created after you enter the first symbol name. You can also enter *Typ*, *Min*, and *Max* values for this first symbol.
   c. You can append new symbol names.
   d. To insert symbols between existing rows, refer to *Inserting and Deleting Symbols, Selectors, and Categories* on page 3-128.

## Selecting the Active Job Using an AC Specs Sheet

If the workbook has multiple jobs or a Job List sheet, you can edit an *AC Specs* or *DC Specs* sheet only if it is used in a job that is the *Active Job*. If a spec sheet is not used by the *Active Job*, the following background message is shown:

> This sheet is not used in the current job.

The *Active Job* is specified in the *AC Specs* or *DC Specs* column on the *Job List* sheet. Therefore, use the *Active Job* control on the **IG-XL** *Context* toolbar to select a job that uses the sheet you want to edit; refer to *IG-XL Context Toolbar* on page 3-9. You can now edit the spec sheet.

## Inserting and Deleting Symbols, Selectors, and Categories

**DataTool** provides special functions for inserting and deleting symbols, selectors, and categories. In most cases, you must use these functions rather than directly inserting or deleting individual rows or columns. Once you have created selectors and categories, refer to *Specifying and Configuring the Selectors and Categories* on page 3-130.

- To insert symbols, selectors, and categories:

1. On the *Insert* menu, select the *Spec Info* command. The *Insert Spec Info* dialog box appears.
2. On the *Insert Spec Info* dialog box:
   a. In the *Type* frame, select *Symbol*, *Category*, or *Selector*, as required.
   b. Enter the *Quantity*, from 1 to 16.
   c. To create the symbol below the last symbol or to create the category to the right of the last category, click *Append*. *Append* does not affect a selector.
   d. Click *OK* to apply the selections. If *Append* is not selected, a new symbol is inserted above the current symbol, or a new category is inserted to the left of the currently category.

- To append a new symbol or a new selector, type in the empty row that follows the existing data. The following two methods are the only exceptions to the rule about using the **IG-XL** functions to insert new items:
   a. For a new symbol, type the name in the *Symbol* column; default values appear in the other columns.
   b. For a new selector, type the name in the *Selector Name* column. It is inserted in the drop-down list and deleted from the row where you entered it.

- To delete symbols, selectors, and categories:

1. On the *Edit* menu, select the *Delete Spec Info* command. The *Delete Spec Info* dialog box appears.
2. On the *Delete Spec Info* dialog box:
   a. In the *Type* frame, select *Symbol*, *Category*, or *Selector*, as required.
   b. From the *Name* drop-down list, select the element to be deleted.
   c. Enter the *Quantity*, from 1 to 16.
3. Click *OK*.

## Renaming Symbols, Selectors, and Categories

By inserting a new item, the symbols, selectors, and categories are given default names. You can change these default names at any time. And, you can rename any time you have named. Note that changing a name does not change the references to that name on other sheets; you must manually update those references.

- To rename a symbol, select the cell containing its name and type in the new name. **DataTool** updates the symbol name in any formulas on this sheet that use the symbol; however, formulas on other sheets are not updated.
- To rename a selector**,** use the *Spec Selector* control on the **IG-XL** *Context* toolbar to choose the selector to rename; *IG-XL Context Toolbar* on page 3-9. Select any cell in the *Selector Name* column, and enter the new name. All cells in the *Selector Name* column are updated to the new name; however, references to the selector on the *Test Instances* sheet are not updated.
- To rename a category, select the column header cell containing the category to rename. You do not have to use the current Spec Category. Enter the new category name; however, references to the category on the *Test Instances* sheet are not updated.

  You can also prefix an environment name to the category name; refer to *Environment Name as Part of Category Name* on page 3-45. For more information about environment names, refer to *Selecting the Test by Gating* on page 3-226 and *Gate Env* on page 3-272.

## Specifying and Configuring the Selectors and Categories

After creating the selectors and categories, you define the spec context, which determines the current value of the spec symbols. The current spec context is specified by a category and a selector. They are set by the *Spec Category* and *Spec Selector* controls on the **IG-XL** *Context* toolbar; refer to *Spec Category and Spec Selector Controls* on page 3-44. If the category name is prefixed by an environment name, you also must use the *Active Environment* control; refer to *Environment Name as Part of Category Name* on page 3-45.

### Entering the Spec Values

1. In each category for each spec symbol, enter the *Typ*, *Min*, and *Max* values.
2. Select the category to enter or edit values by using the *Spec Category* control. The *Spec Selector* control is not used when you are entering spec values.
3. Enter numeric values or formulas that use spec symbols defined elsewhere on this sheet or on the *Global Specs* sheet; refer to *Global Specs Sheet* on page 3-40. Remember that an *AC Specs* or *DC Specs* sheet cannot use symbols defined on another *AC Specs* or *DC Specs* sheet because only one sheet is active

### Configuring the Spec Selector and Selecting the Selector Values

1. Use the *Spec Selector* control on the **IG-XL** *Context* toolbar to specify whether the *Typ*, *Min*, or *Max* value is the current value of the spec symbol.
2. Use the drop-down list in the *Selector Val* column of the *AC Specs* or *DC Specs* sheet to specify *Typ*, *Min*, or *Max* for each symbol.

## Displaying the Current Value of the Spec Symbols

The *Value* column displays the current value of the spec symbol, based on the current *Spec Category* and *Spec Selector* context. This column is read-only. The current values of the spec symbols are set by applying a specific selector to a specific category; refer to *Specifying and Configuring the Selectors and Categories* on page 3-130.

+ If the selector specifies a cell without data, the *Value* column displays #N/A for that spec symbol. Be aware that #N/A is not a validation error. A validation error occurs if another sheet uses a spec symbol that resolves to *#N/A*, such as a test instance trying to use a spec symbol that does not have a value.

### Excel AutoFill and Drag-and-Drop Disabled

The **Excel** features *AutoFill* and *Drag-and-Drop* have been turned off for the *AC Specs* and *DC Specs* sheets because using either of these features could cause unintended results, particularly in cells that are not visible. Also, refer to *Using the Excel Feature Versus the DataTool Feature* on page 3-26.

## AC Specs Columns

### Symbol

Enter the name of the AC specification to be defined. The symbol name in this required cell is a string; refer to the guidelines for symbol names, *Rules for User-Created Names* on page 3-25.

Other sheets access this value by prefixing the name with an underscore: *_symbol*.

To insert, rename, or delete a symbol, refer to *Inserting and Deleting Symbols, Selectors, and Categories* on page 3-128 and *Renaming Symbols, Selectors, and Categories* on page 3-129.

### Value

The current numeric value of the spec is displayed, based on the selector and category currently selected, is displayed in this required cell. You cannot write to this read-only column. To change the value displayed, use the *Spec Category* and *Spec Selector* controls on the **IG-XL** *Context* toolbar; refer to *Specifying and Configuring the Selectors and Categories* on page 3-130.

### Selector Name

The name of the currently-selected selector is displayed in this required cell. The name is a string. You use the *Spec Selector* control on the **IG-XL** *Context* toolbar to select a selector; refer to *Specifying and Configuring the Selectors and Categories* on page 3-130. All values in the column will be the same, because each selector contains a *Selector Val* setting for each spec symbol on the sheet.

To insert a selector, refer to *Inserting and Deleting Symbols, Selectors, and Categories* on page 3-128. To change the value displayed, use the *Spec Category* and *Spec Selector* controls on the **IG-XL** *Context* toolbar; refer to *Specifying and Configuring the Selectors and Categories* on page 3-130.

## Selector Val

Use the drop-down list to select which specification category column—*Typ*, *Min*, or *Max*—is used for this spec, based on the current *Selector Name*.

## Typ, Min, Max

Enter the typical, minimum, and maximum specification values for this symbol within this category. Each category has individual *Typ*, *Min*, and *Max* columns.

Before you can edit a value, use the *Spec Category* control to select the category whose values you want to edit; refer to *Specifying and Configuring the Selectors and Categories* on page 3-130. The selected category expands to show all three columns. If the category is prefixed by an environment name, use the *Active Environment* control to select the environment and the category, refer to *Environment Name as Part of Category Name* on page 3-45.

You can use a formula to enter engineering units with the value, such as *=200*mV*. The formula can also use spec symbols defined elsewhere on this sheet or on the *Global Specs* sheet; refer to *Engineering Units in Formulas* on page 3-34.

To insert, rename, configure, or delete a category or to editing the spec values, refer to *Inserting and Deleting Symbols, Selectors, and Categories* on page 3-128 and *Specifying and Configuring the Selectors and Categories* on page 3-130.

## Comment

Enter any optional notes or comments as a string.

# DC Specs Sheet

## Overview

This sheet defines the spec symbols used in creating formulas on other sheets. These symbols define pin level values. Also, refer to *AC Specs Sheet* on page 3-125.

For the fundamentals about *DC Specs* sheets, refer to *Spec Sheets* on page 3-38.

+ Teradyne recommends that you use spec sheets to define spec symbols because they combine the key values used on the other sheets, thereby making it easier for you to manage your test program.

# Using the DC Specs Sheet

## Inserting and Editing a New DC Specs Sheet

1. Select the *Worksheet* command from the *Insert* menu. The *Insert Worksheet* dialog opens.
2. On the I*nsert Worksheet* dialog:
   a. In the *Sheet Type* field, use to pull-down to select *DC Specs*.
   b. In the *Sheet Name* field, enter the appropriate name of this new sheet.
   c. Click *OK*. The *Insert DC Spec Sheet* dialog opens.
3. On the *Insert DC Spec Sheet* dialog:
   a. In the *Initial Number of Categories* field, enter a number from 1 to 40; default is 3.
   b. Click *OK*.
4. A new *DC Specs* sheet is inserted. The categories are created with default names. You can rename, add, or delete categories later.Before you can edit the new sheet, it must be part of the current *Active Job*.
5. On the *Job List* sheet:
   a. Go to the row for the job that will use the new *DC Specs* sheet.
   b. From the drop-down list in the *DC Specs* column, select the new *DC Specs* sheet.
6. On the **IG-XL** *Context* toolbar, select the new *DC Specs* sheet as the *Active Job*; refer to *IG-XL Context Toolbar* on page 3-9. You can now edit the new sheet.
7. On the new *DC Specs* sheet:
   a. Enter data from the manufacturer's device specification sheets.
   b. A default *selector* will be created after you enter the first symbol name. You can also enter *Typ*, *Min*, and *Max* values for this first symbol.
   c. You can append new symbol names.
   d. To insert symbols between existing rows, refer to *Inserting and Deleting Symbols, Selectors, and Categories* on page 3-128.

## Selecting the Active Job Using a DC Specs Sheet

If the workbook has multiple jobs or a Job List sheet, you can edit an *AC Specs* or *DC Specs* sheet only if it is used in a job that is the *Active Job*. If a spec sheet is not used by the *Active Job*, the following background message is shown: This sheet is not used in the current job.

The *Active Job* is specified in the *AC Specs* or *DC Specs* column on the *Job List* sheet. Therefore, use the *Active Job* control on the **IG-XL** *Context* toolbar to select a job that uses the sheet you want to edit; refer to *IG-XL Context Toolbar* on page 3-9. You can now edit the spec sheet.

## Inserting and Deleting Symbols, Selectors, and Categories

**DataTool** provides functions for inserting and deleting symbols, selectors, and categories. In most cases, you must use these functions rather than directly inserting or deleting individual rows or columns. After you have created selectors and categories, refer to *Specifying and Configuring the Selectors and Categories* on page 3-130.

- To insert symbols, selectors, and categories:

1. On *Insert* menu, select *Spec Info* command. *Insert Spec Info* dialog box appears.
2. On the *Insert Spec Info* dialog box:
   a. In the *Type* frame, select *Symbol*, *Category*, or *Selector*, as required.
   b. Enter the *Quantity*, from 1 to 16.
   c. To create the symbol below the last symbol or to create the category to the right of the last category, click *Append*. *Append* does not affect a selector.
   d. Click *OK* to apply the selections. If *Append* is not selected, a new symbol is inserted above the current symbol, or a new category is inserted to the left of the currently category.

- To append a new symbol or a new selector, type in the empty row that follows the existing data. The following two methods are the only exceptions to the rule about using the **IG-XL** functions to insert new items:
   a. For a new symbol, type the name in the *Symbol* column; default values appear in the other columns.
   b. For a new selector, type a name in the *Selector Name* column. The name is inserted in the drop-down list and deleted from the row where you entered it.

- To delete symbols, selectors, and categories:

1. On the *Edit* menu, select the *Delete Spec Info* command. The *Delete Spec Info* dialog box appears.
2. On the *Delete Spec Info* dialog box:
   a. In the *Type* frame, select *Symbol*, *Category*, or *Selector*, as required.
   b. From the *Name* drop-down list, select the element to be deleted.
   c. Enter the *Quantity*, from 1 to 16.
3. Click *OK*.

## Renaming Symbols, Selectors, and Categories

By inserting a new item, the symbols, selectors, and categories are given default names. You can change these default names at any time. And, you can rename any time you have named. Note that changing a name does not change the references to that name on other sheets; you must manually update those references.

- To rename an symbol, select the cell containing its name and type in the new name. **DataTool** updates the symbol name in any formulas on this sheet that use the symbol; however, formulas on other sheets are not updated.
- To rename a selector**,** use the *Spec Selector* control on the **IG-XL** *Context* toolbar to choose the selector to rename; *IG-XL Context Toolbar* on page 3-9. Select any cell in the *Selector Name* column, and enter the new name. All cells in the *Selector Name* column are updated to the new name; however, references to the selector on the *Test Instances* sheet are not updated.
- To rename a category, select the column header cell containing the category to rename. You do not have to use the current Spec Category. Enter the new category name; however, references to the category on the *Test Instances* sheet are not updated.

  You can also prefix an environment name to the category name; refer to *Environment Name as Part of Category Name* on page 3-45. For more information about environment names, refer to *Selecting the Test by Gating* on page 3-226 and *Gate Env* on page 3-272.

## Specifying and Configuring the Selectors and Categories

After creating the selectors and categories, you define the spec context, which determines the current value of the spec symbols. The current spec context is specified by a category and a selector. They are set by the *Spec Category* and *Spec Selector* controls on the **IG-XL** *Context* toolbar; refer to *Spec Category and Spec Selector Controls* on page 3-44. If the category name is prefixed by an environment name, you also must use the *Active Environment* control; refer to *Environment Name as Part of Category Name* on page 3-45.

### Entering the Spec Values

1. In each category for each spec symbol, enter the *Typ*, *Min*, and *Max* values.
2. Select the category to enter or edit values by using the *Spec Category* control. The *Spec Selector* control is not used when you are entering spec values.
3. Enter numeric values or formulas that use spec symbols defined elsewhere on this sheet or on the *Global Specs* sheet; refer to *Global Specs Sheet* on page 3-40. Remember that an *AC Specs* or *DC Specs* sheet cannot use symbols defined on another *AC Specs* or *DC Specs* sheet because only one sheet is active.

### Configuring the Spec Selector and Selecting the Selector Values

1. Use the *Spec Selector* control on the **IG-XL** *Context* toolbar to specify whether the *Typ*, *Min*, or *Max* value is the current value of the spec symbol.
2. Use the drop-down list in the *Selector Val* column of the *AC Specs* or *DC Specs* sheet to specify *Typ*, *Min*, or *Max* for each symbol.

## Displaying the Current Value of the Spec Symbols

The *Value* column displays the current value of the spec symbol, based on the current *Spec Category* and *Spec Selector* context. This column is read-only. The current values of the spec symbols are set by applying a specific selector to a specific category; refer to *Specifying and Configuring the Selectors and Categories* on page 3-130.

+ If the selector specifies a cell without data, the *Value* column displays #N/A for that spec symbol. Be aware that #N/A is not a validation error. A validation error occurs if another sheet uses a spec symbol that resolves to *#N/A*, such as a test instance trying to use a spec symbol that does not have a value.

## Excel AutoFill and Drag-and-Drop Disabled

The **Excel** features *AutoFill* and *Drag-and-Drop* have been turned off for the *AC Specs* and *DC Specs* sheets because using either of these features could cause unintended results, particularly in cells that are not visible. Also, refer to *Using the Excel Feature Versus the DataTool Feature* on page 3-26.

# DC Specs Columns

## Symbol

Enter the name of the DC specification to be defined. This required name in this required cell is a string; refer to the guidelines for symbol names, *Rules for User-Created Names* on page 3-25.

Other sheets access this value by prefixing the name with an underscore: _*symbol*.

To insert, rename, or delete a symbol, refer to *Inserting and Deleting Symbols, Selectors, and Categories* on page 3-128 and *Renaming Symbols, Selectors, and Categories* on page 3-129.

## Value

Current numeric value of the spec is displayed, based on the selector and category currently selected, is displayed in this required cell. You cannot write to this read-only column. To change the value displayed, use the *Spec Category* and *Spec Selector* controls on the **IG-XL** *Context* toolbar; refer to *Specifying and Configuring the Selectors and Categories* on page 3-130.

## Selector Name

Name of the currently-selected selector is displayed in this required cell. The required name is a string. You use the *Spec Selector* control on the **IG-XL** *Context* toolbar to select a selector; refer to *Specifying and Configuring the Selectors and Categories* on page 3-130. All the values in the column will be the same, because each selector contains a *Selector Val* setting for each spec symbol on the sheet.

To insert a selector, refer to *Inserting and Deleting Symbols, Selectors, and Categories* on page 3-128.

To change the value displayed, use the *Spec Category* and *Spec Selector* controls on the **IG-XL** *Context* toolbar; refer to *Specifying and Configuring the Selectors and Categories* on page 3-130.

## Selector Val

Use the drop-down list to select which specification category column—*Typ*, *Min*, or *Max*—is used for this spec, based on the current *Selector Name*.

### Typ, Min, Max

Enter the typical, minimum, and maximum specification values for this symbol within this category. Each category has individual *Typ*, *Min*, and *Max* columns.

Before you can edit a value, use the *Spec Category* control to select the category whose values you want to edit; refer to *Specifying and Configuring the Selectors and Categories* on page 3-130. The selected category expands to show all three columns. If the category is prefixed by an environment name, use the *Active Environment* control to select the environment and the category, refer to *Environment Name as Part of Category Name* on page 3-45.

You can use a formula to enter engineering units with the value, such as *=200*mV*. The formula can also use spec symbols defined elsewhere on this sheet or on the *Global Specs* sheet; refer to *Engineering Units in Formulas* on page 3-34.

To insert, rename, configure, or delete a category or to editing the spec values, refer to *Inserting and Deleting Symbols, Selectors, and Categories* on page 3-128 and *Specifying and Configuring the Selectors and Categories* on page 3-130.

### Comment

Enter any optional notes or comments as a string.

# Pin Levels Sheet

## Overview

This sheet defines the voltage and current levels for digital, analog, and power pins and pin groups. The values are expressed as absolute values or as formulas using symbols defined on the *DC Specs* or *Global Specs* sheet. Pin types *Input*, *Output*, *I/O*, *HVF*, and *DPS* must be defined on this sheet, while a pin or group type is defined on the Pin Map sheet.

# Using the Pin Levels Sheet

## Multiple Pin Levels Sheets

**DataTool** supports multiple *Pin Levels* sheets in a test program, but only one is active. You associate a *Pin Levels* sheet with a test by means of the *Test Instance* sheet. Each test instance specifies which *Pin Levels* sheet it is to use.

A test program workbook can contain multiple *Pin Levels* sheets, but only one is active. A *Pin Levels* sheet is associated with a test by the *Test Instances* sheet; thus, each test instance specifies a particular *Pin Levels* sheet.

## Inserting a New Pin Levels Sheet

1. Select the *Worksheet* command from the *Insert* menu. The *Insert Worksheet* dialog opens.
2. On the I*nsert Worksheet* dialog:
   a. In the *Sheet Type* field, use to pull-down to select *Pin Levels*.
   b. In the *Sheet Name* field, enter the appropriate name of this new sheet.
   c. Click *OK*. Another *Pin Levels* sheet is inserted.

## Adding Pins

The *Pin Levels* sheet is pin-type sensitive, meaning that when you add a pin or pin group to the sheet, the parameters appropriate for pins of that type are inserted by **DataTool**, one per row. For a list of the parameters inserted for each pin type, use the *Parameter* column; refer to *Parameter* on page 3-153.

If the pin type is *Unknown*, the parameters for type *I/O* are inserted.

## Pin Types Requiring Level Definitions

Every digital, analog, and power pin, either individually or as part of a group, must define a single set of parameters for the pin levels. If the level of a digital, analog, or power pin is not specified, a validation error is returned when you try to validate the test program. On the other hand, you must not define the parameters for three types of pins: *Utility*, *Gnd*, and *N/C*. If you define levels for these types of pins, an error occurs when you try to validate the test program.

The defined level parameters depend on the pin type. For example, output-only pins do not required specifications for their input voltage levels. Levels must be specified for every digital, analog, and power pin, either individually or as part of a pin group.

If the pin type is *Unknown*, the parameters for type *I/O* are inserted.

## Ranges for Parameter Values

Parameter values may be restricted by the tester hardware. Consequently, if you assign a variable to an equation, and that equation evaluates to a value outside the valid range of a parameter for existing hardware, a validation error is returned when you try to validate the test program; refer to the list of parameter definitions: *Parameter Definitions and Valid Ranges* on page 3-150.

## VIHH/VBIAS Limitations

- Only one *VIHH* level value can be specified for channels within the same *VIHH* group of a station.
- Only one *VBIAS* level value can be specified for channels within the same *VBIAS* group of a station.
- FLASH 750 provides a total of 8 *VIHH* levels and 8 *VBIAS* levels across all channels. The channels are divided into the following 8 groups and assigned the following fixed levels:

| Level | Channels | VIHH Level | VBIAS Level |
|---|---|---|---|
| 0 | 0 to 7 | Vihh<0> | Vbias<0> |
| 1 | 2 to 15 | Vihh<1> | Vbias<1> |
| 2 | 16 to 23 | Vihh<2> | Vbias<2> |
| 3 | 24 to 31 | Vihh<3> | Vbias<3> |
| 4 | 32 to 39 | Vihh<4> | Vbias<4> |
| 5 | 40 to 47 | Vihh<5> | Vbias<5> |
| 6 | 48 to 55 | Vihh<6> | Vbias<6> |
| 7 | 56 to 63 | Vihh<7> | Vbias<7> |

## Displaying the Pin Levels Data

By default, the pin levels for each pin or group are collapsed into a single row. The rows can be expanded and collapsed; refer to *Collapsing and Expanding Groups* on page 3-35.

By filtering the *Pin Levels*, you identify all pins and groups belonging to a given parameter; refer to *Filtering Data in a Column* on page 3-37.

## Defining Parameter Values with Spec Symbols in Formulas

You can enter parameter values as formulas that use the spec symbols defined on the *DC Specs* sheet and *Global Specs* sheet. Once you have defined these symbols, the *Spec Category* and *Spec Selector controls* on the **IG-XL** *Context* toolbar are active. The current value of the spec symbols is determined by the spec context selected by these controls; refer to *Spec Category and Spec Selector Controls* on page 3-44.

**DataTool** assigns some pin level parameters, based on the specific spec symbols on the *Global Specs* sheet. You can change these values by editing the Global Specs sheet, but this should not be necessary.

## Power-Up and Power-Down Sequencing

### Overview

Power sequencing, which is the timing of the programmed levels applied to the device pins, is important because a DUT can be damaged if the voltage levels applied to the signal pins exceed the voltage levels applied to its power pins, most likely during powering up or powering down a DUT. For instance, if a test program applies a lower-than-normal voltage level to a power pin before it applies a voltage level for a signal pin, the power pin voltage would temporarily be less than the signal pin voltage, which could possibly damage the DUT.

To reduce the chance of damaging a DUT when powering up or powering down a DUT, **IG-XL** uses an algorithm to control the power-up and -down sequence. This algorithm uses the same method for both sequences. The only difference between these two sequences is that in the voltages in the power-down sequence are moving in the opposite direction from the power-up sequence; consequently, the power-down sequence is the reverse of the power-up sequence.

### Specifying the Power Sequence

The *Seq* column on this sheet specifies the sequence in which pins are powered up and down. To control the sequence, enter a number greater than 0 in the Seq column for a pin or pin group. Note that the value 0 is reserved; refer to *Sequencing of Power Pins* on page 3-146 and *Guidelines for Assigning Sequence Numbers* on page 3-147.

The order of processing the pins is not the same as the power sequence. Processing a pin means determining the pin's place in the power sequence. The actual place in the sequence depends on whether the voltage is increasing or decreasing; refer to *Order of Pin Processing* on page 3-146.

### Order of Pin Processing

A sequence value is assigned only for the following types of pins: *Power*, *I/O,* and *Input*; other pin types are ignored. The order of processing the pins:

1. All pins with a *non-blank* sequence number are processed before all pins with a *blank* sequence column.
2. Pins with a *non-blank* sequence number are processed numerically, from the lowest to highest number.
3. If two pins have the same sequence number, they are processed in the order they appear on the sheet, from top to bottom.
4. Pins with a *blank* sequence column are processed in the order they appear on the sheet, from top to bottom.

### Sequencing of Power Pins

The sequencing of power pins is unique because their position in the power sequence is critical:

- A power pin with a *non-blank* sequence number is processed like any other pin with a sequence number.
- If the column for a power pin is *blank*, **DataTool** assigns it the sequence number 0, which is lower than the lowest valid user input. Thus, the power pins are powered up or down first if their voltage is increasing or last if their voltage is decreasing; refer to *Sequencing for Increasing or Decreasing Voltage* on page 3-147.

### Sequencing for Increasing or Decreasing Voltage

The power sequence algorithm ensures:

1. All pins with increasing voltage are powered up before pins with decreasing voltage.
2. Pins with lower sequence numbers are closer to the head of the sequence if their voltage is increasing, and closer to the end if their voltage is decreasing:

   If the pin voltage increases:

   a. Pin is placed in the first half of the sequence, behind any pins already assigned to the head of the sequence.

   b. If a power pin, it is powered up before all other pins.

   If the pin voltage decreases:

   a. Pin is placed in the second half of the sequence, ahead of any pins already assigned to the end of the sequence.

   b. If a power pin, its present voltage is maintained while the voltages for other pins are changed. The power pin voltage is changed only after all other pins have been changed.

   Pins with lower sequence numbers are closer to the head of the sequence if their voltage is increasing, and closer to the end if their voltage is decreasing.

### Guidelines for Assigning Sequence Numbers

For example, if pin *p1* must always have a higher voltage than *p2*, you should assign *p1* a lower sequence number than *p2*, meaning that *p1* is processed before *p2*, with the following results:

- If *p1* voltage is increasing, it is powered up before *p2*. If *p2* were powered up first, its new voltage might temporarily exceed the old voltage of *p1*, possibly damaging the DUT.
- If *p1* voltage is decreasing, it is powered up after *p2*. If *p1* were powered up first, its lowered voltage might temporarily be less than old voltage of *p2*.

### Default Assigning of Sequence Numbers

If you do not enter a sequence number for a pin, **DataTool** assigns a number to ensure that the power pins are properly powered up and down in relation to the other pins:

- Power pins are assigned the sequence number 0, meaning they are processed first. They are closer to the head of the sequence if their voltage increases, and last if it decreases; refer to *Sequencing for Increasing or Decreasing Voltage* on page 3-147.
- All other pins are processed in the order in which they appear on the sheet.

### Optimizing Test Program Execution and Power Sequencing

**IG-XL** optimizes the execution of a test program by downloading the pin levels from the test program prior to execution. However, if you assign a non-blank sequence number to an *Input* or *I/O* pin affects the execution of a test program because the pin levels for these two types cannot be optimized. Equally important, even if you assign sequence numbers to the power pins without assigning sequence numbers to the *Input* or *I/O* pins, **IG-XL** can optimize the execution of a test program.

Consequently, if you require optimum execution of a test program and can accept the default assignment of sequence numbers, Teradyne recommends that you do not enter any values in the *Seq* column except for power pins. You should assign sequence numbers to *Input* or *I/O* pins only when necessary.

### Power-Up Sequence Example

Assume 5 pins, *p1* through *p5*. If you assign them the listed sequence numbers, including two blanks, they are processed in the following order:

| *Pin* | *Sequence* | *Order processed* |
|---|---|---|
| *p1* | *3* | *p3* (Seq=1) |
| *p2* | blank | *p5* (Seq=2) |
| *p3* | *1* | *p1* (Seq=3) |
| *p4* | blank | *p2* (blank) |
| *p5* | *2* | *p4* (blank) |

Assume the pin voltages are increasing and decreasing as listed:

Increasing: *p1*, *p2*, *p3*

Decreasing: *p4*, *p5*

The pins are processed in the following order, with the sequencing result listed in the *Sequence* column (/ separates the first and second halves of the power-up sequence):

| *Pin* | *Voltage* | *Position* | *Sequence* |
|---|---|---|---|
| p3 | ↑ | first half, first in sequence | p3 / |
| p5 | ↓ | second half, last in sequence | p3 / p5 |
| p1 | ↑ | first half, behind p3 | p3, p1 / p5 |
| p2 | ↑ | first half, behind p1 | p3, p1, p2, / p5 |
| p4 | ↓ | second half, before p5 | p3, p1, p2 / p4, p5 |

The power-up sequence: *p3*, *p1*, *p2*, *p4*, *p5*

+ The two sequences differ only in that the voltages in the power-down sequence are moving in the opposite direction from the power-up sequence; consequently, the power-down sequence is the reverse of the power-up sequence.

## Pin Levels and Ganged Power Supplies

Because ganged power supplies are defined as a group, a pin level is assigned to a group, not to individual pins in a group; refer to *Ganged Power Supplies* on page 3-87.

## Parameter Definitions and Valid Ranges

### Overview

**DataTool** inserts different combinations of parameters in the *Parameter* column of this sheet when you enter a pin or group name for a particular pin type. Each parameter is defined by a symbol in the *Parameter* column, and the symbols shown depend on the pin type. Entered values are checked during validation; an out-of-range value causes an error. For a list of the default parameters inserted for each type, refer to *Default Parameters for Input, Output, and I/O Pin Types* on page 3-150.

### Default Parameters for Input, Output, and I/O Pin Types

| | |
|---|---|
| *VCL* | Clamp voltage limit on low-going output; range: -2 to +5 V. |
| *VCH* | Clamp voltage limit on high-going output; range: 0 to +7 V. |
| *VIL* | Voltage representing a logic zero applied to the DUT (*VDriveLo*); range: -1 to +7 V. |
| *VIH* | Voltage representing a logic one condition applied to the DUT (*VDriveHi*); range: -1 to +7 V. VIH must be >= (*VDriveLo* + 0.1). |
| *VOL* | Testing threshold for logic zero. DUT output voltage below this value interpreted as a logic 0 ( *VCompareLo*); range: -1.0 to +7.0 V. |
| *VOH* | Testing threshold for logic one. DUT output voltage above this value interpreted as a logic one *(VCompareHi)*; range: -1.3 to +7.3 V. |
| *IOL* | Current sunk by a DUT output in the logic zero state (*ISource,* below *VT*); range: 20 µA to +25 mA. Also, refer to *Programming iLoads Near Zero* on page 3-151. |
| *IOH* | Current sourced by a DUT output in the logic one state, ( *ISink;* above *VT*; range: -20 µA to -25 mA Also, refer to *Programming iLoads Near Zero* on page 3-151. |
| *VT* | Threshold voltage below which *IOL* is valid and above which *IOH* is valid; range: -1 to +7 V. |

| | |
|---|---|
| *VBIAS* | 0 to +15V (applies to *I/O* and *Input* pins) |
| *PCVOH** | -1 to +7 V |
| *PCVOL** | -1 to +7 V |
| *PCIOH** | 0 to -25mA |
| *PCIOL** | 0 to +25mA |

* Parametric error catch compare values apply to *I/O* and *Output* pins.

In addition, the appropriate global specs listed below are also inserted, with default values, for *Input*, *Output*, and *I/O* pins.

### *Programming iLoads Near Zero*

If *iLoad*s are programmed near 0, oscillations could occur during testing; thus, use the following algorithms to prevent oscillations when programming the *iLoad*s. Note that the absolute value of *IOH* is used.

| | |
|---|---|
| *IOL* and *IOH* >= 50 µa | Program *IOL* and *IOH* to the specified values; enable *iLoad*s. |
| *IOL* and *IOH* < 50 µa | Program *IOL* and *IOH* to 1 mA; disable *iLoad*s. |
| *IOL* or *IOH* < 50 µa | Program the *iLoad* that is < 50µa to 50 ua; program the *iLoad* that is >= 50 µa to the specified value; enable *iLoad*s. |
| *VBIAS* | Program the *iLoad to* 0 to +15V (applies to *I/O* and *Input* pins); refer to *VIHH/VBIAS Limitations* on page 3-144. |

### Default Parameters for Global specs on Global Specs Sheet

| | |
|---|---|
| *VCH* | Clamp voltage limit on high-going outputs; range: 0 to +7 V, with 0.6V diode drop; *VCH* >= *VCL*; default: 7 V. |
| *VCL* | Clamp voltage limit on low-going outputs; range: -2 to +5 V, with 0.6V diode drop, *VCH* >= *VCL*; default: -2 V. |
| *VIHH* | Programming voltage, high for high-voltage pins; range: 0 to +16 V; default: 0 to +15 V. All channels have *VIHH* capability; refer to *VIHH/VBIAS Limitations* on page 3-144 and *High-Voltage Mode* on page 3-173. |
| *VHVFORCE* | High-voltage force level; range: +10 to +25 V; (applies to *I/O* and *Input* pins) |

### Default Parameters for DPS Pin Type

| | |
|---|---|
| *Vps* | Supply voltage applied to DUT; range: -2 to +10 V. |
| *Isc* | Short-circuit current limit for the DPS; range: 0 to 1 A. |
| *Tdelay* | Delay time after supply voltage applied; range: 0 to 500 ms. By programming *Tdelay,* you can wait some amount of time after applying device power before applying the logic levels. In the power-down sequence, *Tdelay* is applied after removing device power. |
| *Vrreset* | DPS ratchet, reset level; range -15 to +15 V |
| *Vrstep* | DPS ratchet, increment level; range -15 to +15 V |
| *Vrmin* | DPS ratchet, minimum level; range -15 to +15 V |
| *Vrmax* | DPS ratchet, maximum level; range -15 to +15 V |

# Pin Levels Columns

## Pin/Group

Enter the name of a pin or pin group whose levels are defined. The required name is a string. Valid values are the names of a pin group, as specified on the Pin Map sheet.

## Seq

Enter an optional value in this column that specifies the sequence of the levels applied to the DUT. You can enter a numeric value for each pin or group or leave the column blank; **DataTool** resolves the entry to an integer greater than 0. This value is meaningful only for pins or pin groups whose type is *Power*, *Input*, or *I/O*; other types are ignored. For more information, refer to *Power-Up and Power-Down Sequencing* on page 3-145.

## Parameter

Displays the parameter type for the specified pin or pin group in the *Pin/Group* column. The required name is a string. Even though **DataTool** inserts default parameters each pin type, you can select a parameter from a drop-down list:

| *Device Pin Type* | *Parameters* |
|---|---|
| Digital Input | *VIH*, *VIL*, *VIHH*, *Vbias* |
| Digital Output | *VOH*, *VOL*, *VT*, *IOH*, *IOL*, *VCH*, *VCL*, *PCVOH*, *PCVOL*, *PCIOH*, *PCIOL* |
| Digital I/O | *VIL*, *VIH*, *VOL*, *VOH*, *VT*, *IOL*, *IOH*, *VCH*, *VCL*, *PCVOH*, *PCVOL*, *PCIOH*, *PCIOL*, *VIHH*, *Vbias* |
| Analog | *VLevel*, *VPpmu* |
| Device Power | *Vps*, *Isc*, *Tdelay*, *Vrreset*, *Vrstep*, *Vrmin*, *Vrmax* |

For more information about the parameters, refer to *Parameter Definitions and Valid Ranges* on page 3-150.

## Value

Enter the value for the specified parameter type, as either a numeric value or a formula. The required name is a numeric formula. For a list of the valid values for the different parameters, refer to *Parameter Definitions and Valid Ranges* on page 3-150.

Default values are inserted for the parameters that have global spec symbols defined on the Global Specs sheet; refer to *Default Parameters for Global specs on Global Specs Sheet* on page 3-152. These parameters are *VCH*, *VCL*, *VIHH*, *Iph*, and *Tpr*. You can edit the values on the *Global Specs* sheet, but this should not be necessary.

Teradyne recommends entering values as formulas that uses spec symbols; refer to *Creating Formulas* on page 3-32. You can use the **Excel** *IF* function to test the value of a spec symbol before you use it; refer to *Using the Excel IF Function* on page 3-33.

## Comment

Enter any optional notes or comments as a string.

# Scramble Program Sheet

## Overview

This sheet specifies the algorithms for address and data topological scrambling, ECR scrambling, and topological data inversion. Each algorithm consists of one or more equations that assign input bits to output bits.

From a hardware point of view, this sheet:

- Specifies the contents of the MTM Address Topological RAM, ECR Scramble RAM, and the Data Topological RAM, based on the scrambling algorithm.
- Specifies data to configure the XYZ Address Crossover, which maps the ADG outputs for the X, Y, and Z addresses, 16 bits each, into the X and Y addresses, 32 bits total, for the Address Topological Scramble RAM. The programming of the XYZ Address Crossover is indirectly specified by references to the X, Y, and Z address bits in the algorithm equation.
- Specifies data to configure the input multiplexers to the Data Topological Invert RAM. The programming of this RAM is indirectly specified by references to the X, Y, and Z address bits in the algorithm equation.



## Using the Scramble Program Sheet

### Overview

This sheet is not required because scrambling and topological data inversion are not required. A job can specify multiple algorithms; refer to *Inserting a New Scramble Program Sheet* on page 3-156.

If the test instance does not use a topological inversion algorithm, the RAM is loaded with a default set of values to pass through all data. A default set is required because the topological inversion RAM may not be removed from the path.

### Address Compression and Scrambling

Address scrambling or compression during error capture and redundancy analysis requires that the Topological Scan/Compression RAMs be loaded with four maps:

- one for each of the two banks of the X Scramble RAM
- one for each of the two banks of the Y Scramble RAM.

## Algorithms Restrictions and Rules

- Rules for algorithm equations must be followed; refer to *Algorithm Equation Rules* on page 3-157.
- Scramble algorithms: up to 16 different input bits may be used per algorithm. Any input bit may be used multiple times, but no more than 16 different bits may be used in the algorithm equations.
- Topological data inversion algorithms: up to 8 different input bits may be used per algorithm. Any input bit may be used multiple times, but no more than 8 different bits may be used in the algorithm equations.
- Rows of an algorithm must all be of the same type: scramble or topological data inversion.
- If the test instance does not use a topological inversion algorithm, the topological inversion RAM is loaded with a default set of values to pass through all data. A default set is required because this RAM may not be removed from the path.

## Displaying the Algorithm Data

Each algorithm, which contains a row for each equation, can be expanded and collapsed; refer to *Collapsing and Expanding Groups* on page 3-35.

## Inserting a New Scramble Program Sheet

1. Select *Worksheet* command from *Insert* menu. *Insert Worksheet* dialog opens.
2. On the I*nsert Worksheet* dialog:
   a. In the *Sheet Type* field, use to pull-down to select *Scramble Program*.
   b. In the *Sheet Name* field, enter the appropriate name of this new sheet.
   c. Click *OK*. Another *Scramble Program* sheet is inserted.

## Algorithm Equation Rules

### Overview

Use the following rules for forming equations in the *Equation* column. Most of the examples apply to scramble equations. Also, refer to *Rules for Topological Inversion Equations* on page 3-159.

### Equation Syntax

Output bit(s) = Input bit(s)

where the output bits are designated as *O*[*i*] and the input bits as *X*[*i*] and *Y*[*i*].

### Algorithm Syntax Elements

| *Element* | *Definition* |
|---|---|
| *O* | Output bit |
| *X, Y* | Input bit |
| [*i*] | Bit index; *i* range is 15...0 |
| [*i:j*] | Bit range, *i* to *j*; *i* and *j* range is 15...0. A range of addresses may be in either ascending or descending order: for example, *[4:0]* and *[0:4]* are both legal. Within an equation, however, the order must be consistent. For example:<br>*O[4:0]=X[5:1]* is legal<br>*O[4:0]=X[1:5]* is illegal |
| &, \|, ^ | **AND**, **OR**, **XOR** operators |
| ~ | **NOT** operator |
| (,) | Open and close parentheses |
| = | Assignment |

### Valid Equations

- Valid assignment statements, which substitute *Y* for *X*:

| *Syntax* | *Definition* |
|---|---|
| *O[0]=X[1]* | Individual output from individual input. |
| *O[7:3]=X[9:5]* | Contiguous range of outputs from contiguous range of inputs. Ranges must be same size. |
| *O[15:0]=X[15:0]* | Syntax for linear fill. |
| *O[0]=0* | Force output to 0 (or 1). |
| *O[15:14]=1* | Force range of outputs to all 1's or 0's. |

- Valid statements using logical operators, which substitute *X* or *Y* for each other:

+ Association and precedence rules for logical operators are the same as for the C language.

| *Syntax* | *Definition* |
|---|---|
| *O[0]=~X[1]* | Individual output from individual input, inverted. |
| *O[7:3]=~Y[9:5]* | Contiguous range of outputs from contiguous range of inputs, inverted. |
| *O[0]=X[0]^X[4]* | Individual output is the **XOR** of 2 inputs, same for the \| and ^ operators. |
| *O[0]=X[0]&Y[1]^Y[8]* | Complex expression. By default association, this expression is equivalent to *O[0]=(X[0]&Y[1])^Y[8]*. |
| *O[0]=X[0]&(Y[1]^Y[8])* | Complex expression, with parentheses overriding the default association. |
| *O[0]=~X[0]&X[1]^X[8]* | Complex expression. By default associative and precedence, this expression is equivalent to *O[0]=((~X[0])&X[1])^X[8].* |
| *O[0]=~((X[0]&X[1])^X[8])* | Complex expression; expression inside parentheses overrides the default association and precedence. |

### Rules for Topological Inversion Equations

Because the output of the topological inversion RAM is a single bit, you do not have to specify the output in the equation. The output
*O[0]=*
may be included or omitted.

Because of the topological inversion RAM size, a maximum of 8 input addresses may be specified.

| *Syntax* | *Definition* |
|---|---|
| *O[0]=X[4]* | Individual input determines the topo output. |
| *O[0]=X[4]^(Y[4]&Y[8])* | Combination of X and Y inputs determines the topo output. |
| *X[0]&X[1]\|Y[0]&Y[1]^ X[2]&X[3]\|Y[2]&Y[3]* | Maximum of 8 inputs determines the topo output. |
| *X[4]\|x[5]^(~(Y[4]\|Y[5]))* | Overriding normal precedence using parentheses. |

# Scramble Program Columns

## Algorithm

Enter the name of the defined algorithm. An algorithm consists of one or more equations that program a RAM. The required name is a string; also, refer to *Rules for User-Created Names* on page 3-25.

If you enter any additional equations in the *Equation* column, the algorithm name is displayed in grey in this column. Furthermore, you can enter to a new name in this column to identify a new algorithm.

## Type

Enter the type of RAM that is programmed by the algorithm equations. The required name is a string. From the drop-down menu, select one of the following types: *XATopo*, *YATopo*, *ECR X Scramble*, *ECR Y Scramble*, or *DTopo*. All equations in a given set must be of the same type; thus, if you change one type is a set, all types in the set must be changed also.

## Equation

On each row, enter an equation for this algorithm. The required name is a string. Each algorithm is described by one or more equations; refer to *Algorithm Equation Rules* on page 3-157. To add an equation to the algorithm, enter the equation on the next line in this column; **DataTool** will insert the algorithm name and type.

Maximum number of equations for scramble RAMs = 16.

Maximum number of equations for Data Topological RAM = 8.

## Comment

Enter any optional notes or comments as a string.

# Data Generator Sheet

## Overview

This sheet defines the data sets used with the Data Generator. You specify the named collections of the 32 data sets available for the Data Generator. For each data set, you may specify the opcodes for the two Logic Function Generators and the inputs to the two Data Generators.

+ For your convenience, the inputs to the two Logic Function Generators are also specified on this sheet, but they are actually independent of the data sets or the data generator.

Each job has one *Data Generator* sheet; however, you can create other *Data Generator* sheets suitable for specific jobs; refer to *Inserting a New Data Generator Sheet* on page 3-163.

## Using the Data Generator Sheet

### Defined Data Set Required for 2-Bit Data Generator

#### Introduction

This sheet is not required because the 2-Bit Data Generator is not required for DUT testing. However, if you try to use the 2-Bit Data Generator in a test program without first defining at least one data set on this sheet, a validation error occurs. **DataTool** does not verify the data sets as you fill in this sheet; therefore, you must be sure that any data set used in a pattern is defined on this sheet.

### *Overview of 2-Bit Data Generator*

The 2-Bit Data Generator produces data that is derived algorithmically from the generated addresses.

Each of the two bits has a designated source. One source is each bit's Logic Function Generator, designated as *LFG0* and *LFG1*. Each LFG accepts as input one specified bit from the X and Y addresses, on which the designated logical operation is performed.

Up to four data groups can be stored, each containing 8 data sets. Each set contains one set of instructions for deriving the 2-bit data. Within the pattern file, MTM opcodes can specify a data group and data set on a per-vector basis.

Each data set contains the following:

- *Opcode*—For each LFG, a logical operation is performed on the two specified bits from the X and Y addresses. The bits themselves are not part of the data set, but are constant throughout the collection of data sets.
- *Source*—specifies the source of each bit for the 2-bit data generator:

1. Output of the specified LFG, *LFG0* or *LFG1*, for either bit. The complement can also be selected.
2. *Coincidence Detector* generates a 1 if the X and Y addresses are equal, or a 0 if they are not equal. The complement can also be selected. Only enabled bits of the addresses are compared; a *Coincidence Enable Mask* is set outside of the data set.
3. 1—Logic 1.
4. 0—Logic 0.

## *Data Generator Restrictions and Rules*

- Each data group/data set pair is unique: for example, data group 0, data set 0 may not be defined twice.
- If one column on a row is filled in, the entire row must be filled in.
- In a named collection of data sets, the various LFG inputs must be the same for each set. For example, *LFG0 X In* must be the same for each of the sets in a collection, but it may be different from *LFG1 X In* in the same collection or *LFG0 X In* in a different collection or both *LFG1 X In* and *LFG0 X In.*

### Displaying the Setup Collection

Each setup collection, which contains a row for each data set, can be expanded and collapsed; refer to *Collapsing and Expanding Groups* on page 3-35.

### Inserting a New Data Generator Sheet

1. Select the *Worksheet* command from the *Insert* menu. The *Insert Worksheet* dialog opens.
2. On the I*nsert Worksheet* dialog:
   a. In the *Sheet Type* field, use to pull-down to select *Data Generator*.
   b. In the *Sheet Name* field, enter the appropriate name of this new sheet.
   c. Click *OK*. Another *Data Generator* sheet is inserted.

## Data Generator Columns

### Setup Name

Enter the name for this collection of data sets. A setup collection consists of the specifications for each data setup. The required name is a string; also, refer to *Rules for User-Created Names* on page 3-25.

If you enter additional setup information, the setup name is displayed in grey appears in this column. Furthermore, you can enter to a new name in this column to identify a new algorithm.

### DGroup

Enter the number of the data group containing the data set programmed on this row or select one of the following data group numbers from the drop-down list: *0, 1, 2,* or *3*. Data groups with the same number do not need to be contiguous. The required number is an integer.

### DSet

Enter the number of the data set programmed on this row or select one of the following data set numbers from the drop-down list: *0, 1, 2, 3, 4, 5, 6,* or *7*. Data groups support up to 8 sets. The required number is an integer.

## Source Selection

### DGen 0 and DGen 1

Select from the drop-down list or enter the input source for each of the data generators. Required entry is a string. These columns must be filled in if a data group and data set have been selected.

*DGen 0* selects one of the following sources for the first of the two bits, while *Dgen 1* selects the source for the second bit:

- *Logic 0*
- *Logic 1*
- *LFG0*—output of *Logic Function Generator 1*
- *LFG1*—output of *Logic Function Generator 2*
- *Coincidence*—output of *Coincidence Detector*. A 1 is generated if the X and Y addresses are equal; 0 if they are not equal. Only the enabled bits of the addresses are compared; a *Coincidence Enable Mask* is set outside of the data set.
- *~LFG0*—LFG0 output, inverted.
- *~LFG1*—LFG1 output, inverted.
- *~Coincidence*—output of *Coincidence Detector*, inverted. A 0 is generated if the X and Y addresses are equal; 1 if they are not equal. Only the enabled bits of the addresses are compared; a *Coincidence Enable Mask* is set outside of the data set.

## LFG0 and LFG1

+ The cells in this group are enabled only if *LFG0/LFG1* or *~LFG0/LFG1* is specified in the *Source Selection Dgen 0* or *Dgen 1* column for this row; thus, the string entries in the cells are required only if *LFG 0* or *LFG 1* is used.

### LFG0/LFG1 X In and Y In

Select from the drop-down list or enter the X and Y addresses for the appropriate logic function generator: *X0* to *X15* and *Y0* to *Y15*. Each generator requires two input bits: *X In* and *Y In*.

These values are not specific to a data set, but are the same for all data sets in a setup collection; consequently, changing a value on one row of a collection changes it for all other rows of the collection as well. Thus, once these are filled in for a named collection of data sets, they must be the same for all members of the collection.

### LFG0/LFG1 Opcode

For each input bit, select from the drop-down list or enter the opcode for the desired logical operation to perform on the input bits of the appropriate logical function generator:

- *AND*
- *NAND*
- *OR*
- *NOR*
- *XOR*
- *XNOR*

The opcode is specific to a data set defined on this row, while the X and Y bits are the same for all data sets in a setup collection; thus, changing this value on one row of a collection does not change it for all other rows of the collection as well.

## Comment

Enter any optional notes or comments as a string.

# Frame Select Sheet

## Overview

This sheet defines named combinations of frame select bits that control the Frame Select RAM on the Memory Test Module. This RAM is used to serialize the inputs of multiplexed devices.

Each job has one *Frame Select* sheet; however, you can create other *Frame Select* sheets suitable for specific jobs; refer to *Inserting a New Frame Select Sheet* on page 3-167.

## Using the Frame Select Sheet

### Key Concept: Frames

Frames in a test program are used to serialize inputs for muxed input devices. These devices are controlled by a *Frame Select RAM* on the Memory Test Module (MTM). This RAM provides up to 8 frames or formats for the multiplexed devices.

A frame in the Frame Select RAM consists of 16 output bits of data and address values, where each bit is any 32-bit address or data bit. Data bits are output from the DBM/ADG XOR, while the address bits are from the Address Topological Scramble RAM.

The first set of frame is the *Pass Thru* frame, which is the mapping of the lower 16 data bits to the frame output bits. When the FLASH750 hardware is initialized, these 16 bits are loaded into the Frame Select RAM. If you try to modify or delete the *Pass Thru* frame on the sheet, warning and error messages appear.

A frame is selected OTF based on the bits in the Frame Select field of the pattern vector, giving a maximum of 8 frame. Also, refer to *Frame Select Opcodes* on page 6-84, *Pattern Language Reference*, Chapter 6.

## Default Frame Set

**DataTool** creates a default frame set with following parameters:

- Frame identification: *FrameID = 0*
- Frame outputs *FS0* to *FS15* are all data bits that correspond to the frame outputs: data bit 0 is assigned to *FS0*, data bit 1 is assigned to *FS1*, and so on.

## Inserting a New Frame Select Sheet

1. Select the *Worksheet* command from the *Insert* menu. The *Insert Worksheet* dialog opens.
2. On the I*nsert Worksheet* dialog:
   a. In the *Sheet Type* field, use to pull-down to select *Frame Select.*
   b. In the *Sheet Name* field, enter the appropriate name of this new sheet.
   c. Click *OK*. Another *Frame Select* sheet is inserted.

# Frame Select Columns

## Frame Set Name

Enter the name for this collection of frame select bits. The required name is a string; also, refer to *Rules for User-Created Names* on page 3-25.

## Frame ID

Select from the drop-down list or enter the frame identification number for the defined frame set. The required number is a integer from 0 to 7. This number specifies which frame in the Frame Select RAM this frame is to be loaded into. Also, is used OTF to select 1 of 8 frames for the pattern vector.

## Frame Outputs

### FS0 to FS15

Select from the drop-down list or enter the source for each of the 16 frame output bits, FS0 to FS 15:

- X address: *X0* to *X15*
- Y address: *Y0* to *Y15*
- Data bit: *D0* to *D15*

Field can be left blank.

## Comment

Enter any optional notes or comments as a string.

# Time Sets (Basic) Sheet

## Overview

### Purpose of Time Sets (Basic) Sheet

This sheet defines named combinations of timing values and data formats used by particular pins or pin groups when running a test pattern. It defines up to 32 sets of timing edges for the DUT. To define more than 32 sets, use the separate Edge Sets sheets and Time Sets sheets.

### Key Concept: Time Sets

Each time set represents a particular timing scheme, such as *Read Cycle*, *Write Cycle*, or *I/O Read Cycle*. Thus, a time set describes how the DUT pins behave within each cycle type. Time sets are commonly identified by their symbolic name: *ContTest* or *HighSpeed*; alternatively, you can enter integer values.



## Using the Time Sets (Basic) Sheet

### Overview

Timing sheets can be complex because of the number of possible entries. For instance, each FLASH 750 pin can be individually programmed to have as many as 32 edge sets in any burst, each with 5 timing values in normal mode or 6 timing values in *Extended* mode. To support this complexity, **DataTool** has an *Edge Sets* sheet to define all edge sets (data formats plus timing edges), and a separate *Time Sets* sheet to assign the named edge sets to pins and pin groups. On the other hand, most test programs do not require this level of complexity:

- For 32 or fewer sets of timing edges, use this sheet to both define the edge sets and to assign them to the pins. In many applications it can specify all test timing.
- For more that 32 sets, you must use the separate Edge Sets sheets and Time Sets sheets.

### Inserting a New Time Sets (Basic) Sheet

1. Select the *Worksheet* command from the *Insert* menu. The *Insert Worksheet* dialog opens.

2. On the *Insert Worksheet* dialog:
   a. In the *Sheet Type* field, use to pull-down to select *Time Sets(Basic)*.
   b. In the *Sheet Name* field, enter the appropriate name of this new sheet.
   c. Click *OK*. The *Insert Time Sheet* dialog opens.
3. On the *Insert Timing Sheet* dialog:
   a. In the *Tester Timing Mode* field, select *Normal Timing (100 MHz)* or *Extended Timing (50 MHz)*.
   b. Click *OK*.
4. A new *Time Sheets (Basic)* sheet is inserted.

## Entering the Time Set Data

The data entered on this sheet is usually taken from the manufacturer's specification sheet for the DUT. Because the DUT specifications are often based on other specification values, Teradyne recommends:

1. Defining the base specifications as spec symbols on the AC Specs sheet; refer to *Fundamentals of AC Specs and DC Specs Sheets* on page 3-41.
2. On the *Time Sets (Basic)* sheet, enter timing values as formulas that use the spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

## Defining Timing Values with Spec Symbols in Formulas

Timing values can be entered as formulas that use spec symbols defined on the *AC Specs* sheet and *Global Specs* sheet. Once you have defined these spec symbols, the *Spec Category* and *Spec Selector controls* on the **IG-XL** *Context* toolbar are active. The current value of the spec symbols is determined by the spec context selected by these controls; refer to *Spec Category and Spec Selector Controls* on page 3-44.

## Displaying the Time Set Data

Each time set definition, which contains a row for each pin or group, can be expanded and collapsed; refer to *Collapsing and Expanding Groups* on page 3-35.

The *Pin/Group Name* column can be filtered to identify all timing values for a given pin or group; refer to *Filtering Data in a Column* on page 3-37.

## Requirements and Restrictions

### Time Set and Edge Set

- Every digital pin, either individually or as part of a pin group, must be mentioned once per time set.
- No more than 8 combinations of data source and data format may be used per channel within a given burst.
- Assigning edges to pins type *Gnd*, *Unknown*, or *Utility* causes a validation error.
- Time set 0 is reserved for Teradyne use.
- All time sets in a burst must have the same timing mode: either *Normal* or *Extended*.

### Edge Placement

Even if the timing sheets are validated, the tester hardware may not always deliver the defined timing and formatting because the behavior of the tester hardware depends on the particular sequence of pattern data applied to the DUT. Consequently, certain time set definitions may be valid with certain data sequences and invalid with others.

## Selecting the Hardware Mode

### Overview

The tester has several special hardware capabilities supported by **DataTool**.

### Multiple DUT Power Supplies in Parallel

Multiple supplies (DPS) may be *ganged* or wired in parallel (up to 8) to provide more current than is available from a single supply; refer to *Ganged Power Supplies* on page 3-87.

### Multiplex (Mux) Mode

Multiplex (mux) mode allows pairs of adjacent channels to be connected together to provide signals at twice the normal clock rate; refer to *Multiplexed Pins* on page 3-85.

### SCIO Mode (io_midband or io_valid)

In this hardware mode, the channel driver and comparator operate simultaneously within the same cycle, meaning that drive and compare are in the same cycle. Two types of SCIO are provided:

1. *io_midband* mode—comparisons to *High*, *Low*, neither *High* nor *Low*, and no compare.
2. *io_valid* mode—comparisons to *High*, *Low*, either *High* or *Low*, and no compare.

In both cases, these high and low logic levels are defined by the *VOL* and *VOH* parameters on the Pin Levels sheet.

Requirements:

- Timing sheet, Edge Sets or *Time Sets (Basic)* must be in *Extended* mode.
- Timing sheet must set the mode of the pin by selecting *io_valid* or *io_midband* in the *Pin/Group Setup* column.
- Pattern file must contain a pin_setup statement specifying the pins that are to be in *io_valid* or *io_midband* mode; refer to *Vector Data in SCIO and Mux Modes* on page 6-46, *Pattern Language Reference*, Chapter 6.

### Frequency Counter Mode

This hardware mode allows a channel comparator to count the number of state transitions on the pin it is wired to.

Requirements:

- Timing sheet, *Edge Sets* or *Time Sets (Basic)* must be in *Extended* mode.
- Timing sheet must set the mode of the pin that counts the frequency by selecting *freq_counter* in the *Pin/Group Setup* column.
- Timing sheet must set the *Compare Mode* column of the pin to *Window*.
- Pattern file must contain a **pin_setup freq_counter** statement specifying the pins or channels in the frequency counter mode. The pattern uses the symbolic data characters 0 and X to open and close the frequency counter window; refer to *Frequency Counter Data* on page 6-50, *Pattern Language Reference*, Chapter 6.

### High-Voltage Mode

In this mode a channel driver can produce higher-than-normal voltages, making it a 3-level driver capable of low, high, and very high logic levels. The higher-than-normal voltages are used when programming programmable devices.

Requirements:

- Pin Levels sheet must specify values for *VIHH* (programming voltage), *Tpr* (voltage rise time), and *Iph* (current limit) for the selected pin.
- Timing sheet, *Time Sets* or *Time Sets (Basic)*, must set the high-voltage pin by selecting *high_voltage* in the *Pin/Group Setup* column.
- Timing sheet, *Time Sets (Basic)* or *Edge Sets*, should program the *D3* (Drive Off) edge to 0 ns, which is the beginning of the cycle. The *D3* edge controls entry into the high voltage state; therefore, *D3* edge should be set to the beginning of the cycle, which is an offset of 0 ns.
- Pattern file must contain a **pin_setup high_voltage** statement specifying the pins or channels in the high voltage mode. The data codes for a high-voltage pin are 0, 1, and 2, where 2 indicates the very high voltage; refer to *High-Voltage Data* on page 6-49, *Pattern Language Reference*, Chapter 6.
- All channels have high-voltage capability.

### *Multi-Clock Generator Mode*

- Overview

In the multi-clock generator (*mcg*) mode, the tester applies clock waveforms with periods at a sub-multiple of the timing specified in the *Period* column of the timing sheet. This mode is used for testing devices whose clock input operates several times faster than the other signals on the device.

Because CPP is specified for individual time sets, the CPP value can be changed dynamically (on-the-fly) during a burst.

- Setting up the *mcg* Mode

1. Timing sheet must be in the *Extended* mode.
2. *Cycle CPP* column on the timing sheet must specify the number of clocks per tester cycle for the pin. It must be an integer from 1 to 1000.
3. *Pin/Group Setup* column on the timing sheet must specify the *mcg* mode.
4. Timing sheet value for the *Drive Data* column must be between 0 and *Period*/*Cycle CPP*.
5. *Data Return* value must not be greater than the *Period/Cycle CPP*.
6. All generated clock waveforms share the *Data Src* and *Data Fmt* specified for the pin or group.

- Restrictions

   a. *Cycle Period/CPP* must be equal to or greater than 20 ns, which is the minimum period in the *Extended* mode.

   b. Resulting waveforms must be within the specification for the minimum pulse width: 3.3 ns, maximum; minimum pulse width at 50% point, output pulse height 3.0 V.

   c. Receive data cannot be used on pins in the *MCG* mode.

## Debugging Vectors with DataTool

### Overview

In most cases, the easiest and most powerful way to debug vectors is to use **PatternTool**; however, you can use **DataTool** for some low-level debugging.

You specify a data format and data source for each pin or pin group on the *Time Sets (Basic)* sheet or the *Edge Sets* sheet. The *Data Src* and *Data Fmt* columns offer many possible combinations of values; however, only a few of the combinations are useful for ordinary device testing; refer to *Data* on page 3-184. Most of the combinations are used for low-level debugging of the test program.

### Low-Level Debugging

For low-level debugging, the other *Data Src* and *Data Fmt* values can be useful:

- If the polarity of a device pin is inverted from what is found in the .pat file vectors, you can invert the *.pat* file data by setting *Data Src* to *PATNOT*.
- To force a constant 1 or 0 on an input pin for drive cycles only, set *Data Fmt* to *NR* and set *Data Src* to *PATHI* for a constant logic 1 or *PATLO* for constant logic 0. If you force a constant 1 or 0 on a bi-directional pins, they will still have their outputs compared during read cycles.
- To force a constant level of 1 or 0 on a pin for a read or write cycle, *Data Fmt* to *NR* and set *Data Src* to *ALLHI* for a constant logic 1 or *ALLLO* for a constant logic 0.
- To create clock vectors for vectors that do not include data for the device clock pin, set *Data Src* to *ALLHI* and set *Data Fmt* to *RL* for a positive clock pulse or *RH* for a negative clock pulse. The *ALLHI* and *ALLLO* source values apply the data level (1 or 0) to both read and write cycles, which is the desired action for a free-running clock.
- To create during debugging a new time set that is a copy of a regular time set but differs by having a pin/group, set *Data Fmt* to *STAY* format and then change the vector to use the new debug time set. The *STAY* format causes the pin/group to keep the logic level of the previous vector.

## Mapping the Hardware Drive and Compare Formats

### Overview

**DataTool** allows you to specify drive and compare formats. **DataTool** maps these formats to certain hardware formats. You do not have to understand how the formats are mapped in order to create a test program because **DataTool** assigns these format; however, the information in this section can help you understand how to display and change these formats.

### Displaying and Changing Hardware Formats

You can use the **Digital Channel Display** to display and change the hardware drive format for a specific channel. You start this application through the **IG-XL Debug Display Manager**.

### Mapping of Hardware Drive Formats

The *Edge Set* and *Time Set (Basic)* sheets define the hardware drive format:

- *Data Src*—specifies the source of the logic value.
- *Data Fmt*—specifies what format is applied to the data.

Certain combinations of **DataTool** values may not correspond to any hardware format. These cases are displayed with a (*2*) after the hardware format name, and the combination of **DataTool** values is mapped to the corresponding hardware format displayed with a (*1*) after the format name. For example, the values *PATHI* and *SBC* map to *Drive Clock Hi (2)*, which is not an existing hardware format; thus, **DataTool** maps these values to *Drive Clock Hi (1)*; refer to the following listing.

In some cases, two different combinations of **DataTool** values may map to the same hardware format. Both are displayed in the *Data Fmt* column, with the second value in enclosed in square brackets. Furthermore, if you enter the bracketed *Data Fmt* value along with the *Data Src* value in that line, **DataTool** will change the bracketed *Data Fmt* value to the unbracketed value. For example, for *Drive Hi (1)*, *Data Src* is *PATHI*, but *Data Fmt* could be *NR* or *RH*. If you enter *RH* for *Data Fmt* and then *PATHI* for *Data Src*, **DataTool** will change *Data Fmt* to *NR; r*efer to the following table. Consequently, a given hardware format will map to a single *Data Src-Data Fmt* combination, which ensures that any hardware readback is really what is shown on the **DataTool** sheet.

The following list shows the mapping between each hardware drive format and the software values you entered on the **DataTool** sheets:

| *Hardware Format* | *Data Src* | *Data Fmt* |
|---|---|---|

| | | |
|---|---|---|
| *Return to Zero* | *PAT* | *RL* |
| *Ret to Zero Cmplmnt* | *PATNOT* | *RH* |
| *Return to One* | *PAT* | *RH* |
| *Ret to One Cmplmnt* | *PATNOT* | *RL* |
| *Non Ret to Zero* | *PAT* | *NR* |
| *Non Ret to Zero Cmplmnt* | *PATNOT* | *NR* |
| *NRZ Return Off* | *PAT* | *ROFF* |
| *NRZC Return Off* | *PATNOT* | *ROFF* |
| *Cmplmnt Surround* | *PAT* | *SBC* |
| *Cmplmnt Surr Cmplmnt* | *PATNOT* | *SBC* |
| *Drive Clock Hi* (1) | *PATHI* | *RL* |
| *Drive Clock Low* (1) | *PATLO* | *RH* |
| *Drive Clock Hi* (2) | *PATHI* | *SBC*, [*SBL*] |
| *Drive Clock Low* (2) | *PATLO* | *SBC*, [*SBH*] |
| *Drive Hi* (1) | *PATHI* | *NR*, [*RH*] |
| *Drive Lo* (1) | *PATLO* | *NR*, [*RL*] |
| *Drive Hi Return Off* | *PATHI* | *ROFF* |
| *Drive Lo Return Off* | *PATLO* | *ROFF* |
| *Drive Hi* (2) | *PATHI* | *SBH* |
| *Drive Lo* (2) | *PATLO* | *SBL* |
| *Clock Hi* (1) | *ALLHI* | *RL* |
| *Clock Lo* (1) | *ALLLO* | *RH* |
| *Clock Hi* (2) | *ALLHI* | *SBC*, [*SBL*] |
| *Clock Lo* (2) | *ALLLO* | *SBC*, [*SBH*] |
| *Force Hi* (1) | *ALLHI* | *NR*, [*RH*] |
| *Force Lo* (1) | *ALLLO* | *NR*, [*RL*] |
| *Force Hi Return Off* | *ALLHI* | *ROFF* |
| *Force Lo Return Off* | *ALLLO* | *ROFF* |
| *Force Hi* (2) | *ALLHI* | *SBH* |
| *Force Lo* (2) | *ALLLO* | *SBL* |
| *One Surround* | *PAT* | *SBH* |
| *One Surr Cmplmnt* | *PATNOT* | *SBL* |

| | | |
|---|---|---|
| *Zero Surround* | *PAT* | *SBL* |
| *Zero Surr Cmplmnt* | *PATNOT* | *SBH* |
| *Stay* | Do not care | *STAY* |

### *Mapping of Hardware Compare Formats*

The *Edge Sets* and *Time Sets (Basic)* sheets also define the comparison formats. The *Compare Mode* options *Edge* and *Window* both assume that the pattern data is used; they therefore both map to the hardware format *Compare Pattern*. The other option *Off* disables comparison, and therefore maps to the hardware format *Compare Off*. The other hardware compare formats listed below cannot be selected from **DataTool**.

| *Hardware Formats* | *Compare Mode* |
|---|---|
| *Compare Pattern* | *Edge* or *Window* |
| *Receive Hi* | not available |
| *Receive Lo* | not available |
| *Receive Midband* | not available |
| *Receive Valid* | not available |
| *Receive Mask* | not available |
| *Compare Hi* | not available |
| *Compare Lo* | not available |
| *Compare Midband* | not available |
| *Compare Valid* | not available |
| *Compare Mask* | not available |
| *Compare Off* | *Off* |

# Using the Graphical Timing/Formats Display

### Overview

**DataTool** can display shows a graphical representation of the edges defined for each pin or pin group of an edge set or time set. You can edit the timing of the edges by dragging them on the display: the timing of the changed edges listed in the columns of the timing sheet is also changed.

### Invoking the Graphical Timing/Formats Display

Before you invoke the display:

- Be sure that the current spec context can evaluate all the spec symbols used on the *Time Sets (Basic)*, *Time Sets*, or *Edge Sets* sheet. Use the *Spec Category* and *Spec Selector* controls to set the spec context; refer to *Spec Category and Spec Selector Controls* on page 3-44.
- The sheet must be displaying the result values. If the sheet is in formula mode, the buttons are not visible.

To invoke the display, click the three-dot button ...:

- On a *Time Sets (Basic)* sheet, the button is in the *Time Set* column. The display shows the edges for each pin or pin group defined in the time set.
- On a *Time Sets* sheet, the button is in the *Time Set* column. For each pin or group specified in the time set, the display shows the edges in the specified edge set.
- On an *Edge Sets* sheet, the button is in the *Pin/Group* column. The display shows the edges for all the edge sets defined for that pin.

### Parts of the Graphical Timing/Formats Display

The display consists of one row for each set of edges. The pin name or pin edge name appears on the left side; time intervals are marked across the top.

Use the scroll bars to view parts of the display not currently visible. Zoom buttons are also available. The zoom buttons above the vertical scroll bar affect the height of the edge displays, while the zoom buttons to the left of the horizontal scroll bar affect the width of the time interval units.

### Displaying and Interpreting the Edge Values

- Put the cursor over an edge to display the value or formula that defines the edge on the sheet.
- A vertical line on the right side of the display represents the end of the period; position the cursor at this point to display the value of the period.
- A red edge indicates an edge whose value causes a violation. Position the cursor over the red edge to display the edge value causing the violation.

### Pattern Data Controls

The *Pattern Data* controls at the bottom of the display let you see the effect of applying different data values to the edges. D?, D0, D1, D-, DX, RL, RH, RV, and RM.

Changing how the data is displayed does not change the edge values.

### Changing the edges

To change an edge, grab it with the cursor and drag it. You can also change the period length by dragging the vertical line that represent the period. When you drag an edge, you change its value. The change is simultaneously applied to the corresponding column of the timing sheet, and the current formula for the edge is displayed; the evaluation of the formula appears under the edge.

\+ Changes are applied immediately

When you exit from the display window, the changed value remains. If the original value was a formula or a value with engineering units, represented as a formula, the changed value will consist of the original formula plus or minus the amount of the change.

# Time Sets (Basic)Field

## Timing Mode

This field displays the current mode of the sheet: *Normal* or *Extended*. When you insert a new sheet, you are asked whether the timing mode should be *Normal* or *Extended*. The required entry is a string: either *Normal* or *Extended.*

You can edit this field to change the timing mode. Changing the mode causes certain columns to be enabled or disabled, depending on which columns are valid for the new mode. In addition, the values available on drop-down lists will change. If an existing value becomes invalid in the new mode, you will need to change the value manually. If you do not make the change, the invalid value is detected at validation.

For the effect of timing mode on the pattern file, refer to *Pattern Modes* on page 6-16, *Pattern Language Reference,* Chapter 6.

# Time Sets (Basic) Columns

## Time Set

Enter the name for this collection of data sets. A setup collection consists of the specifications for each data setup. The required name is a string; also, refer to *Rules for User-Created Names* on page 3-25.

You can also enter an integer to refer directly to a hardware location of a time set because **DataTool** maps symbolic names to hardware locations; however, you cannot mix numbers and names on the same *Time Sets (Basic)* sheet.

### Graphical Editor for Timing/Format

Clicking the three-dot button ... in this cell open the graphical timing/format display, which shows the edges that this time set has defined for all pins. The edges can be edited; refer to *Using the Graphical Timing/Formats Display* on page 3-179.

## Cycle

### Cycle Period

Enter the clock period used by this time set. This required entry of each time set is a numeric formula:

| | |
|---|---|
| Period, minimum | 10 ns |
| Period, maximum | 10.0 ms |
| Maximum edge range: (smaller of) | 4 * (tester cycle) - 20 ns or 10.24 microseconds. |

You can enter the value as a formula that uses spec symbols; refer to *Spec Sheets* on page 3-38. A formula may use engineering units; refer to *Engineering Units in Formulas* on page 3-34.

### Cycle CPP

Enter the Clocks-per-Period (*CPP*) factor for this time set if the *Pin/Group Setup* column specifies the *mcg* mode. The *CPP* specifies the number of times per period the pin timing is triggered. This entry is an integer; default is 1. The default value is used if this column is blank.

For more information about the *mcg* mode, refer to *Multi-Clock Generator Mode* on page 3-174.

## Pin/Group

### Pin/Group Name

Select from the drop-down list or enter on successive column rows the name of each pin or group to apply the time set timing to. The required name is a string. Valid values are the pins or pin groups specified on the Pin Map sheet.

Assigning edges to pins of type *Gnd*, *Unknown*, or *Utility* causes a validation error.

You can use a drop-down filter in the column header to display all time sets assigned to a particular pin or group; refer to *Filtering Data in a Column* on page 3-37.

### Pin/Group Setup

Select from the drop-down list or enter the hardware mode to be applied to the pin or group for all time sets on this sheet. The name in this optional field is a string. If you select a value for one pin or group, it is applied to all other occurrences of the pin or group on this sheet.

+ You must also insert the necessary constructs in the pattern file for the selected hardware mode.

The drop-down list displays only those values valid for the current mode:

- *Normal*: *i/o* and *high_voltage*
- *Extended*: *i/o*, *high_voltage, freq_counter, mux, io_midband, io_valid,* and *mcg*

| *Setup* | *Meaning* |
|---|---|
| *i/o* | Default—no special hardware mode. The pin is a regular digital drive/compare pin. If left blank, the default is *i/o*. |
| *high_voltage* | Pins use the high-voltage features; refer to *High-Voltage Mode* on page 3-173. |
| *freq_counter* | Pins measure frequency; refer to *Frequency Counter Mode* on page 3-173. |
| *mux* | Pairs of channels operate in multiplex (*mux*) mode; refer to *Multiplex (Mux) Mode* on page 3-172. |
| *io_midband* | Pins operate in Single Cycle I/O Midband mode; refer to *SCIO Mode (io_midband or io_valid)* on page 3-172 |
| *io_valid* | Pins operate in Single Cycle I/O Valid mode; refer to *SCIO Mode (io_midband or io_valid)* on page 3-172. |
| *mcg* | Pins use multi-clock generator; refer to *Multi-Clock Generator Mode* on page 3-174. |

## Data

### Data Src

Select from the drop-down list or enter the source of the logic value to verify or to apply to the pin or group on this sheet. This required value is a string:

| | |
|---|---|
| *PAT* | Pin data is sourced from the pattern file (.*pat*). This value is most often used. |

+ In most cases, *Src* is set to *PAT*. The other values are generally used for low-level debugging; refer to *Debugging Vectors with DataTool* on page 3-175.

| | |
|---|---|
| *PATNOT* | Pattern is negated. Drive data is the complement of the pattern file data; receive data is not affected. This is a quick method of inverting the drive values without regenerating the test patterns. |
| PATHI | Pattern High. Logic high is substituted for the pattern data. |
| *PATLO* | Pattern Low. Logic low is substituted for the pattern data. |
| *ALLHI* | All High. Logic high is substituted for all pattern data; cycle type, drive and compare, are ignored. |
| *ALLLO* | All Low. Logic low is substituted for all pattern data; cycle type, drive and compare, are ignored. |

The combination of *Data Src* and *Data Fmt* values determines the hardware drive format to be used; refer to *Mapping of Hardware Drive Formats* on page 3-176.

### Data Fmt

Select from the drop-down list or enter the formatting to apply to the pin or group on this sheet. This required value is a string. The value specified depends on the waveforms required by the pin.

In most cases, *Fmt* should be set one of these: *RL*, *RH*, *NR*, *SBC*, *SBL*, or *SBH*. Most of the following values are standard except for *ROFF*. The specific format depends on the nature of the waveforms that the pin or group requires:

| | |
|---|---|
| *RH* | Return-to-High |
| *RL* | Return-to-Low |
| *ROFF* | Return-to-Off. This *NR* (Non-Return) format unconditionally forces the driver to turn off (go tri-state) at the D3 (Drive Off) event, regardless of the data in the next cycle. Usually, the D3 event goes to tri-state only if the subsequent cycle is a receive cycle. This format is available only in the extended mode. One application is to allow multiplexed pins to drive the DUT during part of a cycle, while allowing the DUT to drive during the remainder of the cycle. |
| *NR* | Non-Return |
| *SBC* | Surround-by-Complement. |
| *SBH* | Surround-by-High. |
| *SBL* | Surround-by-Low. |
| *STAY* | Hold the pin state. Used for debugging. |

**DataTool** may change the source/format combination you entered because it does not identify an existing hardware format; therefore, **DataTool** changes the format to an appropriate value; refer to *Mapping of Hardware Drive Formats* on page 3-176:

| *Src* Value | *Fmt* entered: | *Fmt* changed to: |
|---|---|---|
| *PATHI* | *RH* | *NR* |
| | *SBL* | *SBC* |
| *PATLO* | *RL* | *NR* |
| | *SBH* | *SBC* |
| *ALLHI* | *RH* | *NR* |
| | *SBL* | *SBC* |
| *ALLLO* | *RL* | *NR* |
| | *SBH* | *SBC* |

## *Drive*

### *Drive On*

Enter the delay of the cycle start from the beginning of a pattern period (*D0*). For surround formats, this is the time the surround value is first applied. This required value is a numeric formula. Leaving the cell blank disables the edge or entering the string *Disable* suppresses the edge.

If you enter a number, you can use a formula to include engineering units; refer to *Engineering Units in Formulas* on page 3-34. Or, you can enter the value as a formula that uses spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

### *Drive Data*

Enter the time at which the data is applied (*D1*). The time is defined with respect to the beginning of the pattern period (*D0*). This required value is a numeric formula. Leaving the cell blank disables the edge or entering the string *Disable* suppresses the edge.

You can use a formula to include engineering units; refer to *Engineering Units in Formulas* on page 3-34. Or, you can enter the value as a formula that uses spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

### Drive Return

Enter the time that the data returns (*D2*) if a return format (*Fmt*) has been selected or that the surround data is reapplied if a surround format (*Fmt*) has been selected. The time is defined with respect to the beginning of the pattern period (*D0*). This value is a numeric formula and is required for formats other than *NR*.

You can use a formula to include engineering units; refer to *Engineering Units in Formulas* on page 3-34. Or, you can enter the value as a formula that uses spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

### Drive Off

Enter the time the tester switches off the driver when executing a receive cycle (*D3*). The time is defined with respect to the beginning of a pattern period. This required value is a numeric formula. Leaving the cell blank disables the edge or entering the string *Disable* suppresses the edge.

The *D3* definition depends on the mode:

- *normal*: edge is part of the receive cycle. Because the edge occurs in the receive cycle, you might program it at the beginning of the cycle, before the receive edge.
- *extended*: edge is part of the drive cycle and does not occur during receive cycles. Because the edge occurs in the drive cycle, you might program it near the end of the cycle, after the other drive edges.

You can use a formula to include engineering units; refer to *Engineering Units in Formulas* on page 3-34. Or, you can enter the value as a formula that uses spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

## Compare

### Compare Mode

Select from the drop-down list or enter the comparison mode to be used by the pin or group for all time sets defined on this sheet. This required value is a string:

| | |
|---|---|
| *Edge* | Perform an instantaneous value test at the time specified by *Open*. |
| *Windows* | Verify the state of the logic signal during the time period specified by the *Open* and *Close* values. This mode is available only in the *Extended* timing mode; refer to *Timing Mode* on page 3-181. |
| *Off* | Disregard the pattern data and perform no test. This mode is assigned by **DataTool** for *Input* type pins. Also, select this mode to disable testing of a device pin with many failures. |

If you set a value for one pin or group, **DataTool** applies it to all other occurrences of the pin or group on this sheet.

You can use a formula to include engineering units; refer to *Engineering Units in Formulas* on page 3-34. Or, you can enter the value as a formula that uses spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

### Compare Open

Enter when the output is tested (*R0*), defined with respect to the beginning of the pattern period (*D0*). This required value is a numeric formula. Leaving the cell blank disables the edge; entering the *Disable* string suppresses the edge.

Minimum compare width (*Close* minus *Open*) is 2 ns.

You can use a formula to include engineering units; refer to *Engineering Units in Formulas* on page 3-34. Or, you can enter the value as a formula that uses spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

### Compare Close

Enter the time an extended (window) test of the device output ends (*R1*). The time is defined with respect to the beginning of the pattern period (*D0*). This optional value is a numeric formula, and is available only in the *extended* mode. Leaving the cell blank disables the edge or entering the string *Disable* suppresses the edge.

Minimum compare width (*Close* minus *Open*) is 2 ns.

You can use a formula to include engineering units; refer to *Engineering Units in Formulas* on page 3-34. Or, you can enter the value as a formula that uses spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

## Comment

Enter any optional notes or comments as a string.

# Time Sets Sheet

## Overview

### Purpose of Time Sets Sheet

This sheet defines up to 255 time sets, each specifying a unique assignment of edge sets to pins that can be used on a vector-by-vector basis. Use this sheet if the test program requires more than 32 edge sets for all pins in a burst; compare this sheet to the Time Sets (Basic) sheet.

### Key Concept: Time Sets Sheet and Time Sets (Basic) Sheet

You may have to use a *Time Sets* sheet rather than a *Time Sets (Basic)* sheet because the tester hardware permits each pin to select from a maximum of 32 different edge sets during a burst. Use the following guidelines in selecting which sheet to use:

- If the total number of edge sets during a burst exceeds 32, you must define the named edge sets on a *Edge Sets* sheet and assign these edge sets to the DUT pins on a *Time Sets* sheet.
- As long as the total number of edge sets used by all device pins in a test program is 32 or fewer, Teradyne recommends that you use the *Time Sets (Basic)* sheet to simplify your test program. A *Time Sets (Basic)* sheet is sufficient, and you do not need a *Time Sets/Edges Set* sheet pair.

## Using the Time Sets Sheet

### Overview

Timing sheets can be complex because of the number of possible entries. For instance, each FLASH 750 pin can be individually programmed to have as many as 32 edge sets in any burst, each with 5 timing values (in normal mode) or 6 timing values (in *Extended* mode). To support this complexity, **DataTool** has an Edge Sets sheet to define all edge sets (data formats plus timing edges), and a separate *Time Sets* sheet to assign the named edge sets to pins and pin groups.

## Inserting a New Time Sets Sheet

1. Select the *Worksheet* command from the *Insert* menu. The *Insert Worksheet* dialog opens.
2. On the I*nsert Worksheet* dialog:
   a. In the *Sheet Type* field, use to pull-down to select *Time Sets*.
   b. In the *Sheet Name* field, enter the appropriate name of this new sheet.
   c. Click *OK*. The *Insert Timing Sheet* dialog opens.
3. On the *Insert Time Sheet* dialog:
   a. In the *Tester Timing Mode* field, select *Normal Timing (100 MHz)* or *Extended Timing (50 MHz)*.
   b. Click *OK*.
4. A new *Time Sheets* sheet is inserted.

## Defining the Edge Set Data Before the Time Set Data

The data in the *Time Sets* sheet is dependent on the edge set names defined on the *Edge Sets* sheet and pin/group names defined in the *Pin List* . Timing values are usually dependent on symbols defined on one or more *AC Specs*, *DC Specs*, or *Global Specs* sheets. Therefore, Teradyne recommends that you create these sheets in the following order:

1. On the Edge Sets sheet, define all timing for each pin.
2. On the *Time Sets* sheet, specify the groupings of those pin timings into named time sets by assigning an edge set to each pin.

+ Since you may have to create a *Time Sets* sheet instead of a *Time Sets (Basic)* sheet after determining that the test program requires more than 32 edge sets, you can cut and paste data from the *Time Sets (Basic)* sheet to the new *Time Sets* and *Edge Sets* sheets before deleting the *Time Sets (Basic)* sheet.

## Entering the Time Set Data

The data entered on this sheet is usually taken from the manufacturer's specification sheet for the DUT. Because the DUT specifications are often based on other specification values, Teradyne recommends:

1. Defining the base specifications as spec symbols on the AC Specs sheet; refer to *Fundamentals of AC Specs and DC Specs Sheets* on page 3-41.
2. On the *Time Sets (Basic)* sheet, enter timing values as formulas that use the spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

## Defining Timing Values with Spec Symbols in Formulas

Timing values can be entered as formulas that use spec symbols defined on the *AC Specs* sheet and *Global Specs* sheet. Once you have defined these spec symbols, the *Spec Category* and *Spec Selector controls* on the **IG-XL** *Context* toolbar are active. The current value of the spec symbols is determined by the spec context selected by these controls; refer to *Spec Category and Spec Selector Controls* on page 3-44.

## Displaying the Time Set Data

Each time set definition, which contains a row for each pin or group, can be expanded and collapsed; refer to *Collapsing and Expanding Groups* on page 3-35.

The *Pin/Group Name* column can be filtered to identify all timing values for a given pin or group; refer to *Filtering Data in a Column* on page 3-37.

## Requirements and Restrictions

- Every digital pin, either individually or as part of a pin group, must be mentioned once per time set.
- No more than 255 time sets can be defined by a *Time Sets* sheet.
- Assigning edges to pins type *Gnd*, *Unknown*, or *Utility* causes a validation error.
- Time set 0 is reserved for Teradyne use.
- All time sets in a burst must have the same timing mode: either *Normal* or *Extended*.

# Time Sets Field

## Timing Mode

This field displays the current mode of the sheet: *Normal* or *Extended*. When you insert a new sheet, you are asked whether the timing mode should be *Normal* or *Extended*. The required entry is a string: either *Normal* or *Extended.*

You can edit this field to change the timing mode. Changing the mode causes certain columns to be enabled or disabled, depending on which columns are valid for the new mode. In addition, the values available on drop-down lists will change. If an existing value becomes invalid in the new mode, you will need to change the value manually. If you do not make the change, the invalid value is detected at validation.

For the effect of timing mode on the pattern file, refer to *Pattern Modes* on page 6-16, *Pattern Language Reference,* Chapter 6.

# Time Sets Columns

## Time Set

Enter the name for this collection of data sets. A setup collection consists of the specifications for each data setup. The required name is a string; also, refer to *Rules for User-Created Names* on page 3-25.

You can also enter an integer to refer directly to a hardware location of a time set because **DataTool** maps symbolic names to hardware locations; however, you cannot mix numbers and names on the same *Time Sets (Basic)* sheet.

### Graphical Editor for Timing/Format

Clicking the three-dot button ... in this cell open the graphical timing/format display, which shows the edges that this time set has defined for all pins. The edges can be edited; refer to *Using the Graphical Timing/Formats Display* on page 3-179.

## Cycle

### Cycle Period

Enter the clock period used by this time set. This required entry of each time set is a numeric formula. The minimum and maximum clock period values differ for each timing mode:

| | |
|---|---|
| Period, minimum | 10 ns |
| Period, maximum | 10.0 ms |
| Maximum edge range: (smaller of) | 4 * (tester cycle) - 20 ns or 10.24 microseconds. |

You can enter the value as a formula that uses spec symbols; refer to *Spec Sheets* on page 3-38. A formula may use engineering units; refer to *Engineering Units in Formulas* on page 3-34.

### Cycle CPP

Enter the Clocks-per-Period (*CPP*) factor for this time set if the *Pin/Group Setup* column specifies the *mcg* mode. The *CPP* specifies the number of times per period the pin timing is triggered. This entry is an integer; default is 1. The default value is used if this column is blank.

For more information about the *mcg* mode, refer to *Multi-Clock Generator Mode* on page 3-174.

## Pin/Group

### Pin/Group Name

Select from the drop-down list or enter on successive column rows the name of each pin or group to apply the time set timing to. The required name is a string. Valid values are the pins or pin groups specified on the Pin Map sheet.

Assigning edges to pins of type *Gnd*, *Unknown*, or *Utility* causes a validation error.

You can use a drop-down filter in the column header to display all time sets assigned to a particular pin or group.

### Pin/Group Setup

Select from the drop-down list or enter the hardware mode to be applied to the pin or group for all time sets on this sheet. The name in this optional field is a string. If you select a value for one pin or group, it is applied to all other occurrences of the pin or group on this sheet.

+ You must also insert the necessary constructs in the pattern file for the selected hardware mode.

The drop-down list displays only those values valid for the current mode:

- *Normal*: *i/o* and *high_voltage*
- *Extended*: *i/o*, *high_voltage, freq_counter, mux, io_midband, io_valid,* and *mcg*

| *Setup* | *Meaning* |
|---|---|
| *i/o* | Default—no special hardware mode. The pin is a regular digital drive/compare pin. If left blank, the default is *i/o*. |
| *high_voltage* | Pins use the high-voltage features; refer to *High-Voltage Mode* on page 3-173. |
| *freq_counter* | Pins measure frequency; refer to *Frequency Counter Mode* on page 3-173. |
| *mux* | Pairs of channels operate in multiplex (*mux*) mode; refer to *Multiplex (Mux) Mode* on page 3-172. |
| *io_midband* | Pins operate in Single Cycle I/O Midband mode; refer to *SCIO Mode (io_midband or io_valid)* on page 3-172 |
| *io_valid* | Pins operate in Single Cycle I/O Valid mode; refer to *SCIO Mode (io_midband or io_valid)* on page 3-172. |
| *mcg* | Pins use multi-clock generator; refer to *Multi-Clock Generator Mode* on page 3-174. |

## Edge Set

Select from the drop-down list or enter the name of the edge set defined on the Edge Sets sheet that the pin or pin group will use in this time set. Each pin or pin group can select one of its 32 defined edge sets.This required value is a string:

## Comment

Enter any optional notes or comments as a string.

# Test Instances Sheet

## Overview

### Purpose of the Test Instances Sheet

This sheet defines the tests applied to the DUT, including parameters for each test.

### Key Concepts: Test Instance and Test Program

#### Test Instance

A test instance is a test template with an applied set of data values for a particular test. This set of parameters is associated with a particular test instance by assigning a name to the test instance. On the *Test Instances* sheet, you create a test instance named *MyFct*. You then specify its parameters, including the timing and levels sheets, what patterns to apply to the DUT, and so on. Parameter values are entered directly on the *Test Instances* sheet or indirectly by using the Instance Editor.

A test instance can be based on a user-written function that performs an action needed in the test program; refer to *Routine Reference*, Appendix G.

#### Test Program

A *test program* consists of a number of test instances. The order of how these test instances are executed is specified by the Flow Table sheet.

# Using the Test Instances Sheets

## User-Written Functions

### Overview

Functions written by the user can perform needed test procedures within the test flow. A function can be an **Excel** macro, a **Visual Basic** (**ActiveX**) *DLL*, or an **OLE** (C++) *DLL*.

### Requirements for User-Written Functions

- Any user-written procedure that is not a template must be specified as a *Test Procedure Type* of *Other*.
- Any function whose *Test Procedure Type* is *Other* must:

1. Use the following calling sequence, whether it is an **Excel** macro or a DLL:

   **Function** *function-name()* **As Long**

   ...

   *function-name* = **TL_SUCCESS**

   **End Function**
2. Return one of these pre-defined macros: either **TL_SUCCESS** or **TL_FAILURE**.

### Arguments

User-written functions do not take arguments as part of their call: you must use the *Arg* columns to pass arguments to the function. These values are stored during validation. The templates retrieve at runtime the arguments for the sheet parameters.

The following **VB-Test** method retrieves an array of argument values from the *Arg* columns of the currently executing test instance:

TheExec.DataManager.GetArgumentList

However, this method does not retrieve arguments in the *DC Specs, AC Specs,* or *Sheet Parameters* columns. To retrieve this information, use the following method:

TheExec.DataManager.GetInstanceContext

### Example

The following function sets the global address register with a pattern name and a label in the pattern. The **PatGen** method **SetGlobalAddr** accepts a pattern name and a label as its parameters. The function assumes that the pattern is entered as *Arg0* and the label as *Arg1*:

```
Public Function PatGlobal() As Long

Dim ArgStr() As String

Dim ArgCnt as Long

Call TheExec.DataManager.GetArgumentList(ArgStr, ArgCnt)

Call TheHdw.Digital.PatGen.SetGlobalAddr(ArgStr(0),ArgStr(1))

PatGlobal = TL_SUCCESS

End Function
```

### Troubleshooting

If a user-written function returns an error:

- Make sure that the declaration is correct and uses the calling sequence listed in *Requirements for User-Written Functions* on page 3-198.
- Make sure that the entire module in which the function occurs compiles correctly. An error in another part of the module can cause an error, even if the specific function contains no errors.

## Parameter Values and Arg Columns

Most test instances require parameter values. The number and type of parameters for a given test instance depend on the type of test instance. The common parameters are entered into the following columns on the *Test Instances* sheet: *Category* and *Selector* for both *AC Specs* and *DC Specs*, *Sheet Parameters, Time Sets, Edge Sets, Pin Levels,* and the *MTM Parameters—Resource Map*, *Frame Set*, *XATopo*, *YATopo, DTopo*, *Data Gen*, and *Pin Funcs*. For each column, a drop-down list displays the currently active categories, selectors, or sheets.

All other, template-specific, parameters are entered in the *Arg* columns. A button is provided to let you hide and expose these other columns. Because you will more than likely use the instance editor to enter and edit these parameter values, the *Arg* columns are usually hidden. The instance editor will display all parameters, whether or not the columns are hidden. You can, however, expose and directly edit these columns when a user-written template has no instance editor, as well as for any other user-written functions that take arguments.

## Test Groups

The *Test Instances* sheet supports test groups. A test group is a set of test instances that have the same name but that can have different values for their parameters. You create a test group by entering the same test name on subsequent rows. Once created, the group can be expanded and collapsed to view or hide individual test members; refer to *Collapsing and Expanding Groups* on page 3-35. The *Test Name* of a group is in bold.

### Formulas Using Spec Symbols

For timing and level values, you can enter formulas that use spec symbols defined on the *AC Specs*, *DC Specs*, or *Global Specs* sheet; refer to *Spec Sheets* on page 3-38. Each test instance specifies the AC and DC categories and selectors that define the spec context, which in turn, determines the value of spec symbols used by the test instance; refer to *Categories and Selectors in a Test Instance* on page 3-43.

When you are using a instance editor, the spec context evaluates the formulas that use spec symbols. The instance editors provide read-only areas that display the value that the formulas evaluate in the current spec context.

On the other hand, if you edit the sheet columns without the instance editor, the spec symbols are evaluated in the spec context currently set by the *Spec Category* and *Spec Selector* controls on the *IG-XL Context* toolbar; refer to *Spec Category and Spec Selector Controls* on page 3-44. You can set the spec category to the test instance name and view the values the spec symbols have for that instance.

## Using the Test Instance Editor

### Overview

An instance editor allows you to enter parameter values for a test template. For user-created templates, you must create an instance editor for the template.

An instance editor is a dialog box that clearly labels each parameter and, where appropriate, provides a drop-down of valid values, or other windows for entering the parameter value. The editor also validates the entered values you try to apply to the test instance.

You bring up the instance editor by clicking on the button in the *Test Name* column.

# Test Instances Columns

## Test Name

Enter the name of a test to be defined. The required name is a string; also, refer to *Rules for User-Created Names* on page 3-25. All instance names must be unique within a *Test Instances* sheet.

A test name in bold is a test group; refer to *Test Groups* on page 3-200.

Clicking three-dot button ... in this cell opens the test instance editor for the test; refer to *Using the Test Instance Editor* on page 3-201.

## Test Procedure

### Test ProcedureType

Select from the drop-down list or enter type of test procedure:

- *IG-XL Template*—a standard Teradyne-supplied template. Not currently supported.
- *Template*—a template written by a user. To be recognized as a template, a procedure must contain certain expected functions, such as **Prebody()**, **Body()**, and **PostBody()**.
- *Other*—any user-written procedure that does not meet the template requirements. These can be called as functions in the test flow, between the execution of test instances; refer to *User-Written Functions* on page 3-198.

The required cell data is a string.

### *Test Procedure Name*

Select from the drop-down list or enter in the cell the name of the test program upon which this test instance is based. Note that this name is different than the *Test Name*, which identifies the test instance.

The required cell data is a string. You must enter a name in a specified format, so **IG-XL** can find the procedure and then execute it during the test flow. The entered name depends on the type of procedure and how the user-written functions are called. If the *Test Procedure Type* is *Template* or *Other*, the *Name* format you enter here depends on the value of *Test Procedure Called As*:

| *Type* | *Called As* | Use *Name* format |
|---|---|---|
| *Template* | **Excel** macro | *[wkbook.xls!]templatename* |
| | **VB** DLL | *project.templatename* |
| | OLE DLL | *templatename* |
| *Other* | **Excel** macro | *[wkbook.xls!]function* |
| | **VB** DLL | *project.class.function* |
| | OLE DLL | *module.function* |

- Requirements for procedure called as **Excel** macro:

  If you specify the workbook as *wkbook*, it must include the *.xls* extension; however, if *wkbook* is omitted, the default is the current workbook.

  If the macro is in another workbook, *wkbook.xls* must be specified, and the workbook must be loaded with the workbook containing the test program.

- Requirements for procedure called as DLL: the name must be registered in order to be called.

- Requirements for macro add-ins:

  Template can be part of a user-created add-in that contains a library of templates. In this case, the *Test Procedure Type* is *Template* and *Test Procedure Called As* is *Excel Macro*. You must load the add-in by using the *Add-Ins* command in the *Tools* menu.

  Also, you must use the following *Name* format for a template in a library:

add-inname.xla!*templatename*

- Changing the test procedure type

  If you try to change the procedure type after defining a test instance, you are prompted to confirm that you want to change the test procedure type because the existing arguments are likely invalid for the new type. When you change the type, **IG-XL** clears the argument columns and re-initializes these columns to the defaults for the new type.

- Viewing the code

  To view the code, refer to *Using the Test Instance Editor* on page 3-201.

### *Test Procedure Called As*

Select from the drop-down list or enter in the cell how the test instance or procedure is called:

| | |
|---|---|
| **Excel** *Macro* | **VBA** macro is called from within **Excel**. If the test procedure is called as an **Excel** macro, the interactive execution lets you use **Visual Basic** for debugging. |
| **VB** *DLL* | **ActiveX** project typically created in **Visual Basic**. Not currently supported. |
| *OLE DLL* | Typically created in C++. |

The compiled DLL may have a performance advantage, but the **Excel** macro is far easier to debug, because it supports the **Visual Basic** debugging tools.

The type of test instance can affect your selection:

- User-written templates or functions can be **Excel** macros, **Visual Basic** (**ActiveX**) DLLs, or OLE (C++) DLLs. If called as **Excel** macros, they can be debugged using **Visual Basic**.

The required cell data is a string.

## DC Specs

### DC Specs Category

Select from the drop-down list or enter in the cell a category of DC specifications to be used for the pin levels. The valid values are listed on the drop-down list, which contains only the categories defined on the *DC Specs* sheet specified by the current active job. The selected spec category evaluates any DC spec symbols referenced by the test instance; refer to *Spec Category and Spec Selector Controls* on page 3-44. The required cell data is a string.

### DC Specs Selector

Select from the drop-down list or enter in the cell the selectors of the DC specifications to be used for the pin levels. The valid values are listed on the drop-down list, which contains only the selectors defined on the *DC Specs* sheet specified by the current active job. The selected spec selector evaluates any DC spec symbols referenced by the test instance; refer to *Spec Category and Spec Selector Controls* on page 3-44. The required cell data is a string.

## AC Specs

### AC Specs Category

Select from the drop-down list or enter in the cell a category of AC specifications to be used for the pin levels. The valid values are listed on the drop-down list, which contains only the categories defined on the *AC Specs* sheet specified by the current active job. The selected spec category evaluates any AC spec symbols referenced by the test instance; refer to *Spec Category and Spec Selector Controls* on page 3-44. The required cell data is a string.

### AC Specs Selector

Select from the drop-down list or enter in the cell the selectors of the AC specifications to be used for the pin levels. The valid values are listed on the drop-down list, which contains only the selectors defined on the *AC Specs* sheet specified by the current active job. The selected spec selector evaluates any AC spec symbols referenced by the test instance; refer to *Spec Category and Spec Selector Controls* on page 3-44. The required cell data is a string.

## Sheet Parameters

### Sheet Parameters Time Sets

Select from the drop-down list or enter in the cell the name of the *Time Sets* or *Time Sets (Basic)* sheet to be used by the test instance. The optional cell data is a string. The valid selections are listed on the drop-down list.

### Sheet Parameters Edge Sets

Select from the drop-down list or enter in the cell the name of the *Edge Sets* sheet to be used by the test instance. Cell data is a string; it is required only if the *Time Sets* column specifies a *Time Sets* sheet, not a *Time Set (Basic)* sheet. Thus, if the *Time Sets* column lists the name of a *Time Sets (Basic)* sheet, this column is disabled. Note that if the *Time Sets* column is empty, this column must be also empty.

### Sheet Parameters Pin Levels

Select from the drop-down list or enter in the cell the name of the *Pin Levels* sheet to be used by the test instance. The DC levels specified on this sheet are applied to the DUT. This optional cell data is a string. Valid selections are on the drop-down list.

## Overlay

Select from the drop-down list or enter in the cell the name of an adjust defined on the current *Characterization* sheet, whose spec value is applied to this test instance. An adjusted value of a spec overlays the spec of the same name in the spec definition for this test instance. The optional cell data is a string.

An overlay is specified for an individual test instance or for an individual member of a test group. The adjusted value is propagated to any equations using the spec; refer to *Running an Adjust* on page 3-329. The adjusted values from one device are available to the next, as long as the program is running, and the values are not overwritten by a another setup.

Valid overlay selections, which are *Adjust Spec Names* defined on the current *Characterization* sheet, are on the drop-down list. These selections include the names of all characterization setups with at least one adjust row are displayed. A multi-row setup can contain other kinds of setups, but it must contain at least one adjust row.

If the specified overlay refers to a setup not yet executed, the test instance is executed with the non-adjusted values, without any warning to the user.

## MTM Parameters

### Overview

Parameters for all MTM tests are entered in these columns.

### MTM Resource Map

Select from the drop-down list or enter in the cell the *Map Name* on the *MTM Resource Map* sheet to be used by the test instance. The MTM resources specified on this sheet are applied to the DUT. The optional cell data is a string. The valid selections are listed on the drop-down list.

### Frame Set

Select from the drop-down list or enter in the cell the *Frame Set Name* on the *Frame Select* sheet to be used by the test instance. The frame set specified on this sheet are applied to the DUT. The optional cell data is a string. The valid selections are listed on the drop-down list.

### XATopo

Select from the drop-down list or enter in the cell the algorithm on the *Scramble Program* sheet to be used by the test instance. The scrambled X addresses specified on this sheet are applied to the DUT. The optional cell data is a string. The valid selections are listed on the drop-down list.

### YATopo

Select from the drop-down list or enter in the cell the algorithm on the *Scramble Program* sheet to be used by the test instance. The scrambled Y addresses specified on this sheet are applied to the DUT. The optional cell data is a string. The valid selections are listed on the drop-down list.

### DTopo

Select from the drop-down list or enter in the cell the algorithm on the *Scramble Program* sheet to be used by the test instance. The scrambled data specified on this sheet is applied to the DUT. The optional cell data is a string. The valid selections are listed on the drop-down list.

### Data Gen

Select from the drop-down list or enter in the cell the *Setup Name* on the *Data Generator* sheet to be used by the test instance. The data generator setup specified on this sheet is applied to the DUT. The optional cell data is a string. The valid selections are listed on the drop-down list.

### Pin Funcs

Select from the drop-down list or enter in the cell the *Name* on the *Pin Function Set* sheet to be used by the test instance. The pin function specified on this sheet is applied to the DUT. The optional cell data is a string. The valid selections are listed on the drop-down list.

## Other Parameters (Arg0 and up)

Parameters for all tests that use the ECR and DBM resources are entered in these columns. Because the number of type of template-specific parameters in these columns vary with the type of test instance or vary row to row, the headings of these columns are labeled *Arg0*, *Arg1*, and so on. Consequently, the Test Instance Editor should be used to edit values for these columns. For this reason, the columns are hidden; however, you can manually edit these values. Teradyne recommends that you do not manually edit these values.

These columns can be exposed or hidden by clicking on the plus (+) or minus (-) sign above the columns; refer to *Hiding and Exposing Columns* on page 3-36.

By default, the parameters are identified as *Arg1* through *Argn*.

## Comment

Enter any optional notes or comments as a string.

# Pin Function Sets Sheet

## Overview

This optional sheet associates special functional groups (*VIHH* level or *VBIAS* level) of pins into named groups.

Each job has one *Pin Function Sets* sheet; however, you can create other *Pin Function Sets* sheets suitable for specific jobs; refer to *Inserting a New Pin Functions Sets Sheet* on page 3-210. Alternatively, you can copy an existing *Pin Functions* sheet and then edit it.

# Using the Pin Function Sets Sheet

## Inserting a New Pin Functions Sets Sheet

1. Select the *Worksheet* command from the *Insert* menu. The *Insert Worksheet* dialog opens.
2. On the I*nsert Worksheet* dialog:
   a. In the *Sheet Type* field, use to pull-down to select *Pin Function Sets*.
   b. In the *Sheet Name* field, enter the appropriate name of this new sheet.
   c. Click *OK*. Another *Pin Function Sets* sheet is inserted.

## Using the Select Pin Functions Pins Dialog

### Overview

The S*elect Pin Functions Pins Dialog* is titled for a particular group of pins, such as *Select Pin Functions Pins [VIHH Enable Group 3]* or *Select Pin Functions Pins [VBIAS Enable Group 1]*.

### Available Pins Window and Selected Pins Window and Controls

This dialog has two windows: *Available Pins* and *Selected Pins*. The *Available Pins* window shows the name of the pins on the *Pin Map* sheet. To select a pin, move it from the *Available Pins* window to the *Selected Pins* window. Use the following controls:

- > and < controls move highlighted items between the windows.
- << and >> controls move all items from one window to the other window.
- To move items up or down in the *Selected Pins* window, select one or more items and then use the ^ and v buttons to move the items.

After the required pins are in the *Selected Pins* window, click OK. A row is inserted on the *Pin Function Sets* sheet for each selected pin if the row does not already exist.

A pin cannot be enabled in one or more *VIHH* enable pin columns and the *VBIAS* pin columns; however, a pin may be enabled in more than one *VIHH* enable pins column.

# Pin Function Sets Columns

## Name

Select from the drop-down list or enter in the cell the name of a group of pins to be defined. The required name is a string; also, refer to *Rules for User-Created Names* on page 3-25.

## Pin Name

Enter in the cell the name of the DUT pin to be enabled to *VIHH* or *VBIAS*. The required name is a string; also, refer to *Rules for User-Created Names* on page 3-25. This name should match the one of the entries on the *Channel Map* and *Pin Map* sheets. This column will contain the list of pins you have selected in the *Select Pin Functions Pin* dialog; refer to *Using the Select Pin Functions Pins Dialog* on page 3-210.

## VIHH Enable Pins (Group 0, Group 1, Group 2, Group 3)

Use the dialog box or enter in the cell the pins to be controlled by the 4 group of OTF *VIHH* enable controls. If the entry value is enabled, that pin is enabled for the specific *VIHH* group. Clicking three-dot button ... in the first row opens a dialog box for selecting the pin function pins for a particular group. The three-dot button appears only for the first pin listed for the group; refer to *Using the Select Pin Functions Pins Dialog* on page 3-210.

## VBIAS Pins

Use the dialog box or enter in the cell the *VBIAS* pin names for each group, *Group 0*, *Group 1*, *Group 2*, and *Group 3*. Clicking three-dot button ... in the first row opens a dialog box for selecting the *VBIAS* pins for a part pin group. The three-dot button appears only for the first pin listed for the group; *Using the Select Pin Functions Pins Dialog* on page 3-210.

## Comment

Enter any optional notes or comments as a string.

# Flow Table Sheet

## Overview

This sheet lets you create, debug, and maintain the flow portion of an **IG-XL** test program. You can edit the flow control and binning portion of a test program and execute the new flow without compiling any source code or reloading a running test program.

Using the *Flow Table* sheet, you can:

- execute instances of test templates, including execution of opcodes
- process the test results by setting pass/fail, sort, and bin values for devices and process errors and retest
- create conditional flows based on pass/fail results of previous instances or the state of devices or groups of devices
- assume test results for specific sites, for help in debugging the flow
- specify test numbers and names for datalogging and printing

## Key Concepts

### Simple and Complex Flows

A flow table is simple or complex:

- A simple flow table supports a simple test sequence, consisting of a list of tests to be executed sequentially, plus one or two steps for assigning bin and sort values. Even though the *Flow Table* sheet may seem to offer many features and options, most users require only a simple flow table; refer to *Simple Flow Table* on page 3-215.
- A complex flow table execute tests conditionally, based on the results of previous tests or on the current execution context.

# Using the Flow Table Sheet

## Overview of Flow Table Programming

Using the *Flow Table* sheet, you can

a. execute instances of test templates

b. process the test results, setting pass/fail, sort, and bin values for devices

c. create conditional flows based on pass/fail results of previous instances or the state of devices or groups of devices

d. assume test results for specific sites for debugging the flow

e. specify test numbers and names for datalogging

- Parallel testing

  The *Flow Table* sheet supports parallel (multi-site) testing, including testing that is conditional on the states of some of the devices. The *Flow Table* implicitly supports parallel (multi-site) testing, without requiring you to explicitly program each device because the **IG-XL** manages the multi-site testing and binning and the active site list.

- *Flow Table* steps

  Each row of the *Flow Table* sheet is a *step* in the test program. The *Command Opcode* column defines the step action. The *Test* opcode executes the specified test instance. Opcodes can also contain other instructions, such as **goto** or **stop**.

  The other columns provide additional information for the execution of the step. For example, a *Test* step may contain instructions for result processing; a **goto** step may specify the conditions under which the **goto** should be executed. Different opcodes require or permit different sets of columns.

  When an opcode is entered, any other columns legal for that opcode are enabled for entering values; columns that cannot be used with the opcode are disabled.

  Each step can have an enable word. If the word is enabled, or if the *Enable* column is blank, the step is executed. These words are enabled as a runtime option or from within the flow table. If the word is not enabled, it is skipped.

  For more information, refer to *Executing the Opcode* on page 3-218.

- Test selection by gating

You can select which tests or other steps are included in the executed flow, depending on the execution context specified before running the program by using the gating columns.

Gating columns specify the context in which the step is executed: for a specific job, for specific part numbers, or in a specific environment, such as temperature. A blank means no gating, meaning the step is always executed. Also, refer to *Selecting the Test by Gating* on page 3-226.

Before running the program, you set the execution context, which consists of the active job, active part, and active environment. If a step has a non-blank value for one or more gating columns, the step is executed only if the values matches the execution context.

- Flow control

  The following columns control the execution flow based on the results of previous steps: *Group Specifier*, *Group Sense*, *Group Condition*, *Group Name*, *Device Sense*, *Device Condition*, and *Device Name*.

  A group qualifier bases execution of a group of devices based on a group criteria. A device qualifier defines the set of devices that the opcode will affect; thus, if no devices satisfy the device qualifier, the opcode is not executed.

  Default execution of the *Flow Table* execution is a sequence of steps; however, the *Flow Table* sheet allows the test program to control the flow of execution.

  Most opcodes can be conditioned, meaning a step is executed only if a specified condition is met:

  a. A *group qualifier* specifies the condition, usually based on the results of previous test execution, such as all devices failed a specified test; refer to *Group Specifiers* on page 3-262.

  b. Device opcodes can accept a *device qualifier*, which specifies the set of devices affected by the operation. The device qualifier is usually based on the results of previous test execution, such as all devices that passed the last instance executed. If no devices meet the condition set by the device qualifier, the step is not executed. Also, refer to *Device Qualifier Values* on page 3-265.

- Result processing

  The *Flow Table* sheet provides a means for processing test results. For each device, a set of state variables can be maintained:

  a. pass/fail result

  b. sort number

  c. bin number

  d. logic state variable (whose meaning is user-determined)

  You can control how these variables are set, and use their current values for conditional execution. When execution of the flow table stops, the state variables are written out.

  In addition, you can define flags whose state depends on whether a device passes or fails a test instance. Execution of subsequent steps can be made conditional on the value of these user-defined flags. Also, refer to *Processing Results* on page 3-231.

- Assume capability

  This feature instructs **IG-XL** to assume that a test instances passes or fails for the specified sites. It is used for testing and debugging conditional flows and binning schemes; refer to *Assume Capability* on page 3-243.

- Datalogging and printer output

  You can set up test numbers and test names in the Flow Table that are included with the datalogged test result information. In addition, a set of opcodes is provided for printing; refer to *Datalogging and Printer Output* on page 3-229.

- Sorting the flow

  The order in which test instances are executed in a flow can affect the performance of the test program.

## Simple Flow Table

### Overview

The *Flow Table* offers many features and options; however, most test programs require a *Flow Table* that is simple and straightforward, meaning that you use only a subset of the columns. The necessary columns are visible when the sheet is collapsed; in fact, only some of the visible columns are required.

### Basic Elements of a Simple Flow Table

- Order of execution

  Columns used: *Command Opcode* and *Command Parameter.*

  The *Flow Table* must order the execution of the test instances. By default, the rows of a table are executed sequentially; thus, you enter the tests in the order in which you want them executed. For each row you enter *Test* in the *Opcode* column, and the name of the test instance in the *Parameter* column.

+ This simple flow table does not use conditional execution of opcodes for flow control; refer to *Flow control* on page 3-214. Instead, each row is executed in sequence.

- Test name and number

  Columns used: *TName* and *TNum.*

  These columns are not required, but they are useful for datalogging. You can enter a test name and base test number for each test instance.

- Result processing

  Columns used: *Bin Number Pass/Fail*, *Sort Number Pass/Fail*, and *Result.*

  This table must specify how results are assigned to the tested devices. For each device, it sets a pass/fail result, a bin number, and a sort number.

- Error processing

  Columns used: *Command Opcode*, *Bin Number Fail*, *Sort Number Pass/Fail*, and *Result*.

  This table can specify a bin number and an optional sort number that is assigned to the device when a run-time error occurs during testing. In this way, the device can be retested.

- Setting the result

  As a device is tested, **IG-XL** maintains a pass/fail result variable for it. However, the variable is not set by **IG-XL**; you must use the *Result* column to set it. For each test, you can specify: (1) devices passing the test have their result variable set to *Pass*, (2) failing devices are set to *Fail*, or both, or neither. Once the result of a device is set to *Pass* or *Fail*, it is removed from the active device list, and no longer tested.

  Usually this variable is set to *Fail* for each test. If the device fails the test and its *Result* is set to *Fail*; it is removed from the active device list. On the other hand, if the device passes the test, you usually do not want to have its *Result* set, because you want to continue testing the device. By setting *Result* to *Pass*, the device would be removed from the active device list. As a final step, you use the **set-device** opcode to set the *Result* variable for all remaining devices because these devices have not failed any tests, their *Result* variable is set to *Pass*.

- Setting the bin and sort numbers

  This table can also assign each device a bin number and a sort number, based on the pass/fail result of the device and, if it failed, on which test it failed. Each row has a *Bin Number Pass* and a *Bin Number Fail* column and two columns for the sort numbers. If these columns are filled with values, a device that passes the test has the *Bin Number Pass* value assigned to it, and a device that fails the test gets the *Bin Number Fail* value.

  Usually *Bin Number Fail* and *Sort Number Fail* values are assigned for each test instance. If a device fails this test, it is assigned a bin and sort number. As a final step, use the **set-device** opcode to assign *1* as the *Bin Number Pass* and *Sort Number Pass* value. Because devices have not failed any tests, they are assigned this value, which is a passing device.

### Sample Flow Table

| Opcode | Parameter | Command TName | TNum | Bin # Pass | Bin # Fail | Sort # Pass | Sort # Fail | Result |
|---|---|---|---|---|---|---|---|---|
| set-error-bin | | | | | 99 | | 99 | Fail |
| Test | ff_vccmin | vmin | 100 | | 3 | | 51 | Fail |
| Test | ff_vccnom | vnom | 200 | | 3 | | 52 | Fail |
| Test | dyn_icc | dicc | 300 | | 4 | | 61 | Fail |
| … | | | | | | | | |
| set-device | | | | 1 | | 1 | | |
| stop | | | | | | | | |

## Executing the Opcode

### Introduction

Each table step specifies an action, such as execute an instance, go to another step, or stop. The action is specified by an opcode. The following columns specify the opcode, plus any label for the step and any enable flow word.

*Label* — Target of any **goto** for this step; refer to *Label* on page 3-219.

*Enable* — Enables or disables this step. An enable word can be specified at runtime. Steps containing the enable word or no enable word is executed; refer to *Enable* on page 3-219.

*Command* — Action specified by the **Opcode**. The command opcode may take a **Parameter,** which specifies the object on which the action is to be performed; refer to *Command Opcode column* on page 3-221 and *Opcode operands* on page 3-221.

The most common opcode is *Test*, which uses its parameter as the name of the executed test instance.

By default, the flow table steps are executed sequentially. You have options for controlling what steps are executed:

a. Selecting which steps are to be executed based on gating, refer to *Selecting the Test by Gating* on page 3-226.

b. Conditioning the execution of a step, refer to *Flow control* on page 3-214.

### Label

- Label requirements

  The *Label* column creates a label that identifies this step. It has the same requirements as any user-defined name in an **IG-XL** workbook:

  a. Label is a string of non-blank, printable characters.

  b. Labels are case-insensitive.

  c. Each label must be unique, although a label can be the same as any non-label name used in the program: for example, it can be the same as a test instance name.

- Using labels

  a. Label is the target of the following opcodes: **goto**, **goto-on-all-done**, and **goto-on-all-lastfail**; refer to *Flow Control Opcodes* on page 3-249.

  b. Label is part of a group qualifier or device qualifier as the parameter for the step-fail or step-pass condition, specifying devices that pass or failed the test identified by the label; refer to *Group Conditions* on page 3-263 and *Device Qualifier Values* on page 3-265.

  c. Label is simply a visual marker in the flow table.

### Enable

- Static and dynamic flow control

  The enable flow-word is a static form of flow control, meaning that the flow-word must be enabled or disabled before the program is run; refer to *Flow control* on page 3-214. For dynamic flow control, where execution is conditioned on the results of a previous step, use group qualifiers and device qualifiers; refer to *Group Specifiers* on page 3-262 and *Device Qualifier Values* on page 3-265.

  Requirement: An enabled step must also satisfy the conditions set up by the group qualifier and device qualifier before it is actually executed. Enabling the step permits the evaluation of any qualifiers in the step.

- Enabling an enable word

  An enable word is a user-defined character string that is enabled in one of two ways:

  a. At runtime set the *Run Option* for enabling the flow word; refer to *Enable Words* on page 3-58. Enable words must be set before test execution begins; they cannot be changed during program execution, such as when the test program is stopped at a breakpoint.

  b. From within the flow table, by executing the **enable-flow-word** opcode prior to this step. Note that the **disable-flow-word** opcode can disable the flow word; refer to *Administrative Opcodes* on page 3-245.

- Using Enable

  The *Enable* column conditions the execution of each individual step:

  a. If the *Enable* column is blank, the step is executed.

  b. If an enable word is in the *Enable* column, the step is executed only if the word has been previously enabled.

  For example, you can enter the string *cont* in each step for continuity testing. At runtime, you can decide if you want to execute the continuity portion of the flow. If you do, enable

### *Opcode*

- *Command Opcode* column

  The *Command Opcode* column defines the action for the current flow step. The *Test* opcode executes the named test instance. Other opcodes control the flow, set state variables or flags, or print messages.

  You must specify an opcode for each step of the flow table. Use the **nop** opcode for a step that is not executed, such as a label definition or comment.

- Opcode operands

  Some opcodes require an operand. For example, *goto* requires a label to jump to, and *Test* requires a test instance name to execute. Operands are entered in the *Command Parameter* column. If you enter an opcode that requires a parameter, the *Command Parameter* column is enabled; otherwise, the *Command Parameter* column is disabled.

- Conditionally controlling the flow

  When you enter an opcode into the *Command Opcode* column, the columns that conditionally control the flow are enabled or disabled if the opcode supports that kind of flow control. For example, if the opcode accepts a device qualifier, those columns are enabled; if not, they are disabled.

- Opcodes in other columns

  Some opcodes may use values that you enter into other columns. For example, the **defaults** opcode sets the default test, bin, and sort numbers based on the values in the *Tnum*, *Bin Number Fail*, and *Sort Number Fail* columns. If the opcode you enter can use other columns, those columns are enabled; otherwise, they are disabled.

## Flow Table Controls

### Overview

The default flow table is a sequential execution of steps. The *Flow Table* sheet provides a way to dynamically control how the test program is executed. Most opcodes can be conditioned, so the step is executed only if a specified condition is met:

- A group qualifier bases execution of a group of devices based on a group criteria; refer to *Group Qualifier* on page 3-222.
- A device qualifier defines the set of devices that the opcode will affect; thus, if no devices satisfy the device qualifier, the opcode is not executed; refer to *Device Qualifier* on page 3-224.

### Group Qualifier

A group qualifier specifies both a group of devices (*all devices* or *all active device*) and a condition that the group of devices must meet (*failed the last test* or *have had their bin set*) before the step is executed.

- Evaluating the group qualifier
    a. If the group qualifier columns are not blank, the group qualifier is evaluated. If the condition evaluates to **True**, the opcode is executed (assuming that any flow word is enabled and any device qualifier is satisfied). If it evaluates to **False**, the step is skipped.
    b. If the group qualifier columns are blank, the default group qualifier is *always*, and the step is executed unconditionally.
    c. Several opcodes cannot use a group qualifier, such as those for administering the flow table, setting defaults, or enabling or disabling flow words. All other opcodes can be conditioned with a group qualifier so execution depends on the previous results.

- Group qualifier columns

*Group Specifier* — Defines the group of devices for satisfying the *Group Condition*, such as *all* (all devices), or *any-active* (at least one device in the active site list). Several *Group Specifier* values define conditions that are not device-specific: *always*, *logging*, and *not-logging*; thus, they do not take a *Group Condition* value.

*Group Sense* — Can be blank or *not* string. The *not* string negates the *Group Condition*.

Note that the *Group Condition* may use three-state logic: *True*/*False*/*Clear* or *Pass*/*Fail*/*Clear*. In these cases, the *not* of a value resolves to two states, not one. For example, *Not True* resolves to both *False* and *Clear*, and *Not Clear* resolves to both *True* and *False*. In contrast, the conditions *bin-set* and *sort-set* use a two-state logic: a bin or sort number is either set or not set.

*Group Condition* — Defines the condition that must be satisfied by the devices defined by the *Group Specifier*. For example, *bin-set* (devices have had a bin value set).

*Group Name* — Specifies any parameter required by the *Group Condition*. For example, the *flag-true* condition requires a flag name.

+ Note that not all combinations of values make sense as a group qualifier. For example, *all-active fail* defines a null set: a device is removed from the active site list once its pass/fail result is set; therefore, an active device cannot have its result value set. Many of these illogical combinations cause validation errors.

- Examples

| Group Spec | Group Sense | Group Cond | Group Name | Note |
|---|---|---|---|---|
| all | | sort-set | | If all devices have had their Sort Number set. |
| | any | not | sort-set | If at least 1 device has not had its Sort Number set. |
| all-active | | flag-true | Cont | If all devices still on the active site list have had a flag named *Cont* set to *True*. |

### Device Qualifier

Opcodes that operate on devices or on the state variables associated with specific devices accept a device qualifier that specifies which devices are to be operated on. A device is included in the device list for the current flow step if it meet all conditions in the device qualifier. For example, the device qualifier specifies a certain condition, such as *have not had their sort number set* or *have a certain flag set to True*. Once a device satisfies the device qualifier, the opcode is executed. In contrast, opcodes that do not operate on specific devices, such as those that jump to a label or initialize state variables, do not accept a device qualifier.

- Dynamic flow control

  The device qualifier provides dynamic flow control by evaluating the results of previous steps. In contrast, the enable flow word provides static flow control, meaning that the flow word must be enabled before execution begins; refer to *Enable* on page 3-219.

- Default device qualifier

  The default device qualifier is *not done*, which defines a list of the devices that have not yet had their pass/fail result state variable set, and are therefore still on the active site list.

- *Test* opcode

  For the *Test* opcode, a valid device list is a subset of the active site list, which executes a test only on those devices not marked as *done* by setting their pass/fail result state variable. A device qualifier can execute a test on a subset of the devices still on the active site list. If the device qualifier columns are blank, the test is executed on the devices that are not *done*.

- State variables of devices

  The other opcodes that accept a device qualifier will set or modify the state variables of the devices. The device qualifier for these opcodes include all devices, even those that have had their pass/fail result set.

- Evaluating the device qualifier columns

  a. If the device qualifier columns are not blank, the device qualifier is evaluated. If the condition evaluates to **True**, the device is included in the device list, and the opcode is executed.

  b. If the device qualifier defines no devices, the step is not executed, and the flow passes to the next step.

- Device qualifier columns

*Device Sense* — Can be blank or *not* string. The *not* string negates the *Device Condition*.

Note that the *Device Condition* may use three-state logic: *True*/*False*/*Clear* or *Pass*/*Fail*/*Clear*. In these cases, the *not* of a value resolves to two states, not one. For example, *Not True* resolves to both *False* and *Clear,* and *Not Clear* resolves to both *True* and *False*. In contrast, the conditions *bin-set* and *sort-set* use a two-state logic: a bin or sort number is either set or not set.

*Device Condition* — Defines the condition that must be satisfied by the devices to be included in the device list. For example, *bin-set* (devices have had a bin value set).

*Group Name* — Specifies any parameter required by the *Device Condition*. For example, the *step-fail* condition requires a label.

- Examples

| Device Sense | Device Cond | Device Name | Note |
|---|---|---|---|
| | bin-set | | All devices that have had their Bin Number set. |
| not | bin-set | | All devices that have not had their Bin Number set. |
| | flag-true | Cont | All devices that have had a flag named *Cont* set to *True*. |

## Selecting the Test by Gating

### Overview

You can build a flow table by including more than one version of the same test, each with slight variations; thus, you could choose different versions of a test to run for different times, jobs, parts, or test conditions. By using the gating feature, you can specify the context in which the step is executed: for a specific job, for specific part numbers, or in a specific environment (temperature).

The execution context is set by selecting the active job, active part, and active environment:

- If a step has a non-blank value for one or more gating columns, the step is executed only if the values matches the execution context.
- If a set is blank, it means no gating: that is, always execute the step.

### Gate Columns

Each of the three gating columns refers to one aspect of the execution context: active job, part, and environment:

| | |
|---|---|
| *Gate Job* | Gates on the active job. Valid values are the job names defined on the *Job List* sheet. |
| *Gate Part* | Gates on the active part. Unlike the other gate columns, valid part numbers are not defined elsewhere in the workbook; any string entered in this column becomes a valid value; also, refer to *Gate Part on page 3-271*. |
| *Gate Env* | Gates on the active environment. Valid values are taken from the prefixes to the category names on the *AC Specs* or *DC Specs* sheets. For example, if you name a *DC Specs* category *Hot::10Mhz*, the string *Hot* is a valid environment value. |

### Specifying a Gate

You specify a gate by entering one or more valid values in the appropriate *Gate* column. For more than one value in a cell, separate them with commas. At runtime, this step is executed only if one of the values matches the execution context. For example, if you set *Gate Part* to *HL101, HL102, HL103*, the step is executed only if the active part is one of those three values.

You can prefix a gating value with an exclamation point (*!*), meaning the step is executed only if the values do not match the execution context. For example, if you set *Gate Part* to*!HL101, !HL102, !HL103,* the step is executed only if the active part is not one of those three values. Either all values in a single cell must have exclamation points, or none can have exclamation points; you cannot mix the two options within a single cell.

Other guidelines:

- If a column is blank, the execution of the step is not gated; the step is always performed.
- If more than one of the gate columns have values, all columns with values must match the execution context for the executed step.

### Set up of the Execution Context

The execution context consists of the active job, active part, and active environment. When the program is executed in the engineering or offline mode, with the **IG-XL** workbook visible, you set the context by using the appropriate boxes on the **IG-XL** *Context* toolbar; *Setting the Execution Context* on page 3-54. The values that appear in the drop-down lists for each box are as follows:

| | |
|---|---|
| *Active Job* | Jobs defined on the *Job List* sheet. The values on the other two drop-down lists depend on the active job; therefore, you should select the job first. |
| *Active Part* | Strings entered in the *Gate Job* column (minus any prefixed !) of the *Flow Table* sheet specified by the current active job. |
| *Active Environment* | Strings prefixing the category names on the *AC Specs* and *DC Specs* sheets specified by the current active job. |

For the production mode, the contents of the drop-down list for all three controls can be made available through the operator interface so the operator can select the job, part, and environment for the device to be tested.

+ Setting the execution context includes selecting the *Active Channel Map*; however, it has affect on test selection by gating; refer to *Setting the Execution Context* on page 3-54.

Once you have set the execution context, you can run the program.

## Datalogging and Printer Output

### Overview

From the Flow Table, you can set up test numbers and test names that are included with the datalogged test result information. In addition, a set of opcodes is provided for printing.

The following columns set up the test names and *numbers* to be datalogged: *TName*, *TNum*; refer to *Test Name* on page 3-229 and *Test Number* on page 3-230.

Use the **IG-XL DataCollect Setup** to enable datalogging and to specify what information to datalog for each test execution.

+ The *Group Specifiers* **logging** and **not-logging** make flow control decisions based on whether datalogging is enabled; refer to *Group Qualifier* on page 3-222.

### Test Name

The test name is specified in the *Tname* column.

If a step includes a *Test* opcode, enter a name to identify the test for datalogging. If the column is blank, **DataTool** creates a datalog name based on the test instance name. There is no default test name.

### Test Number

The *Tnum* column sets the base test number used by a test instance. A test instance performs any number of individual tests. You can enter one of the following in the *Tnum* column:

- nothing—If you enter nothing, **DataTool** increments the test number from its current value. The initial value of *Tnum* is set to zero.
- integer constant—If you enter an integer, the next test executed has an the integer for the test number. If you enter an expression, the test number is set by algorithm.
- flow expression
- flow variable

The *Tnum* column is enabled only on steps with the *Test* opcode. If a step with the *Tnum* value is not executed because the step is skipped, the flow word is not enabled, or the group qualifier or device qualifier is not satisfied, any operation in the *Tnum* column is not performed.

### Printing

The opcodes for printing, **print**, **logprint**, and **error-print**, accept a literal text string as their operand. If datalogging is enabled, the opcodes print the string to the datalog output destination; refer to *Print Opcodes* on page 3-251

## Processing Results

### Overview

The *Flow Table* sheet provides a means for processing test results and defining flags based on the results.

### State Variables

For each device, **IG-XL** maintains a set of state variables:

- pass/fail result
- sort number
- bin number
- logic state variable, user-defined
- User-defined flags, whose state depends on the results of a test instance.

You can control how these variables are set, and you can use their current values for conditional execution of subsequent steps. When the test flow stops executing, the state variables are written out. to the executive.

### Columns for Results Processing

The following columns assign values to the state variables (that the flow controller maintains to record) recorded for the results of testing each device:

- *Bin Number Pass, Bin Number Fail*
- *Sort Number Pass, Sort Number Fail*
- *Result*
- *Flag Pass*, *Flag Fail*
- *State*

### Result Processing Opcodes

Results can be processed on the same step as the *Test* opcode. Based on the results of the test instance executed in the step, you can set any of the state variables or flags. Or, you can process results later in the flow when the following opcodes set the state variables and flags: **set-device**, **set-device-new**, and **modify**; refer to *Result Processing Opcodes* on page 3-254.

### Active Site List

**IG-XL** maintains an *active site list*. Initially, it is equivalent to the *starting site list*, with the sites enabled by the *Run Options Dialog* on page 3-56. By default, all sites are enabled when the test program is validated.

An important concept is the difference between passing or failing a test instance and setting the *Pass/Fail* Result variable in the *Result* column. If this variable is set to *Pass* or *Fail*, the tested site of the device is removed from the active site list. Setting this variable for a device marks it as completed, meaning no further testing. Thus, if you set the *Result* column to *Fail* for a device that fails a test instance, this method is usually acceptable because it is removed from the active site list, meaning it is not tested further. On the other hand, you do not usually set the *Result* column to *Pass* for a device that passes a test instance because it would be removed from the active site list. Instead, you can process results at the end of the flow by using the **set-device** opcode to set the *Pass/Fail Result* variable to *Pass* for any devices remaining on the active site list; refer to *set-device* on page 3-255.

### Flow Control

**IG-XL** offers on-the-fly control of the test flow because it can be based on the current value of any of the state variables or user flags.

### Simple Result Processing

Refer *Sample Flow Table* on page 3-218 for an example of simple result processing.

### Additional Processing

In addition to setting state variables, you can specify an error bin and a retest bin:

- error bin—the device is assigned to the error bin by a runtime error.
- retest bin—any device assigned to the retest bin is retested.

## Bin and Sort Numbers

### Overview

**IG-XL** maintains a bin number and sort number for each device:

- Bin number—actual hardware bin the device is placed in.
- Sort number—virtual or software bin that supports additional sorting of the devices in this bin.

Properties and Guidelines:

- Changes to the *Bin Number* and *Sort Number* variables do not effect the subsequent flow.
- Setting one of these variables does not remove a device from the active site list.
- Group specifiers and device specifiers can limit testing for devices that have their bin or sort number set or not set.
- Final values for these variables are written to **IG-XL** when the flow terminates and are displayed on the Output window.

### Bin and Sort Number Columns

Use these columns to assign Bin and Sort Numbers to a device: *Bin Number Pass, Sort Number Pass, Bin Number Fail,* and *Sort Number Fail.*

- If the result is *Pass*, the *Bin Number* variable is set to the value of *Bin Number Pass*, and the *Sort Number* variable is set to the value of *Sort Number Pass*.
- If the result is *Fail*, the *Bin Number Fail* and *Sort Number Fail* columns are used. If a *Fail* column is blank, the *Bin Number* or *Sort Number* variable is not changed.

The bin and sort columns accept the following values:

- Blank (the current value is not affected)
- An integer
- flow expression, flow variable, or site variable.

  Because flow expressions and flow and site variables are evaluated at runtime, a sort or bin value may be out-of-range. In this case, a default value (0) is used for both sort and bin, and a warning is issued.

### Opcodes for Setting Bin and Sort Numbers

You can set *Bin Number* and *Sort Number* variables in the same step as the *Test* opcode, or you can set them later in the flow with the **set-device, set-device-new,** opcode or **modify** opcode. The different opcodes set the *Bin Number* and *Sort Number* variables differently:

- In a step with the Test opcode, **IG-XL** uses the result of the test instance executed in this step to determine whether the *Pass* columns or the *Fail* columns set the *Bin Number* and *Sort Number* variables. If the device passed, it uses *Bin Number Pass* and *Sort Number Pass* are used; if it failed, it uses *Bin Number Fail* and *Sort Number Fail*.

+ The step does not have to set the *Pass/Fail Result* variable in he *Result* column; even if this variable is not set, the *Bin Number* and *Sort Number* variables can be set.

- In a step with the **set-device**, **set-device-new**, or **modify** opcode, **IG-XL** uses the current value of the *Pass/Fail Result* state variable to determine whether to use the *Pass* or the *Fail* columns; refer to *Result Processing Opcodes* on page 3-254.

With these opcodes, you can also use the *Result* column to set the *Pass/Fail Result* variable; in this case, the *Pass/Fail Result* variable is set first, and the *Bin Number* and *Sort Number* variables are set according the *Pass*/*Fail Result* variable that has just been set.

### Clearing the Variables and Setting Defaults

To clear the *Bin Number* and *Sort Number* variable, use the **modify** opcode and enter *–1* in the *Bin Number Fail* or *Sort Number Fail* column.

You can use the **defaults** opcode to set the default values for the *Bin Number* and *Sort Number* variables. Enter the default values in the *Bin Number Fail* and *Sort Number Fail* columns.

## Pass and Fail Flags

### Overview

In addition to the **IG-XL** state variable, user-defined flags are supported. By defining a flag, a distinct flag of that name is maintained for each device.

Once you have created and set flags, you can use either a group qualifier or a device qualifier to test for devices with a specific flag name set to *True*, *False*, or *Clear*.

### Opcodes for Setting the Pass/Fail Flags

In a step with a Test opcode, enter names in the *Flag Pass* and *Flag Fail* columns:

- If the test instance passes, the flag in *Flag Pass* is set to *True*. It also sets any flag named in *Flag Fail* to *False*.
- If the test instance fails, the flag in *Flag Fail* is set to *True*. It also sets any flag named in *Flag Pass* to *False*.
- If a flag name does not already exist, the **IG-XL** creates it and sets it to *True* or *False*.
- If a flag named in a test instance step already exists, its value is changed only for those devices that were tested in that step, which are the devices specified by the device qualifier. If a device was not tested, its flag is not changed.
- An empty *Flag Pass* or *Flag Fail* column in a flow step means that no flag is affected by the pass/fail result of the instance executed in that step.

In addition to letting **IG-XL** set the flags based on test results, you can explicitly set the value of a flag with the following opcodes: **flag-true**, **flag-false**, and **flag-clear**; refer to *Result Processing Opcodes* on page 3-254. Use a device qualifier with these opcodes to specify what devices are to have the named flag set.

### Creating and Initializing the Pass/Fail Flags

Flags may be created in any flow step that has the *Flag Pass* and *Flag Fail* columns enabled. For example, the **defaults** opcode provides a useful place to define flags set in later steps. A newly-created flag has the value *Clear*. Note that it is not necessary to create a flag before setting it.

You can initialize all flags for all devices with these opcodes: **flag-true-all**, **flag-false-all**, and **flag-clear-all**; refer to *Variable Initialization Opcodes* on page 3-252.

## Pass/Fail Result

### Overview

**IG-XL** maintains a *Pass/Fail Result* state variable for each device. The variable can have one of three values: *Pass*, *Fail*, or *None*. If this variable is set to either *Pass* or *Fail*, the device is removed from the active site list once it passes or fails the test instance, and is not tested further; also, refer to *Active Site List* on page 3-232.

You can use the state of the *Pass/Fail Result* variable and a group qualifier or a device qualifier to test for devices in the test program flow with a specific variable setting:

- Group qualifier can test for devices that are active or not active.
- Device qualifier can test for devices that have completed testing.

### Setting the Pass/Fail Result

You use the *Result* column in a step with a **Test** opcode to command **IG-XL** to set the *Pass*/*Fail Result* variable of the device to be the same as the pass/fail result of the test instance execution. *Result* accepts the following values:

| | |
|---|---|
| *Pass* | Set *Pass*/*Fail Result* to *Pass* if it passes this test. |
| *Fail* | Set *Pass*/*Fail Result* to *Fail* if it fails this test. |
| *All* | Set *Pass*/*Fail Result* to *Pass* if it passes the test and to *Fail* if it fails the test. |
| *None* | Do not change *Pass*/*Fail Result* (default). |

For example, if the *Result* column specifies *Fail* and the device passes the test instance, the *Pass/Fail Result* variable of the device is not changed.

### Using Other Opcodes

You can also use other opcodes to change the *Pass/Fail Result* variable of the devices specified by the device qualifier:

- **set-device** and **set-device-new** opcodes set the *Pass/Fail Result*, *Bin Number*, and *Sort Number* variables for the devices in the device list; refer to *set-device* on page 3-255 and *set-device-new* on page 3-257.
- **modify** opcode sets the *Pass/Fail Result* variable independently of the *Bin Number* and *Sort Number* variables; refer to *modify* on page 3-258.

Use a device qualifier with these opcodes to specify what devices will have their *Pass/Fail Result* variable changed; refer to *Result Processing Opcodes* on page 3-231.

### Usage

Typically, as shown in the simple flow table, the *Pass/Fail Result* variable is set to *Fail* for each test. Thus, if a device fails a test, its *Pass/Fail Result* variable is set to *Fail*, and the device is removed from the active site list. On the other hand, if it passes, testing continues.

As a final step, when **set-device** is used with the *Pass/Fail Result* variable set to *Pass*, Teradyne recommends the *Result* column set to *Pass* or left blank. Any devices still on the active site list have passed all tests; consequently, **IG-XL** sets their *Pass/Fail Result* variable to *Pass*. If the *Result* column is blank, refer to *set-device* on page 3-255 and *set-device-new* on page 3-257.

### Fast Binning

Sometimes when a device passes a test, you may want to set its *Pass/Fail Result* to *Pass* immediately rather than continue testing the device, and then remove it from the active site list and assign it to a bin, rather than continue testing the device. This strategy is *fast binning*. Alternatively, you can set *Pass/Fail Result* to *All* if you want either a *Pass* or *Fail* to be assigned as the *Result*.

## State Variable

### Overview

**IG-XL** maintains a logical *State* variable for each device. It has one of three values: *True*, *False*, and *Clear*, which is specified in the *State* column. The user defines the meaning of this variable. You can explicitly set its value and then test for the value later.

You can use either a group qualifier or a device qualifier to test for devices whose *State* variable is set to **True**, **False**, or **Clear**.

### Initializing and Setting the State Variable

To initialize this variable for all devices, use one of the following opcodes: **state-true-all**, **state-false-all**, or **state-clear-all**; also, refer to *Variable Initialization Opcodes* on page 3-252.

To set this variable for specific devices, use the modify opcode. This opcode accepts a device qualifier, which specifies the devices whose *State* variable is set. You can set the *State* column to *True*, *False*, or *Clear*, which sets the *State* variable of the devices specified by the device qualifier

With the **modify** opcode, you can also set the *State* column to *And*, *Or*, or *Xor*. These specified logic operations are performed on the current *State* value and the value derived by testing whether the device is in the device-list as specified by the device qualifier for this step). For details, see the description of the modify opcode.

## Error Processing

### Overview

The pass/fail status of the part cannot be determined when a part has a runtime failure. Consequently, **IG-XL** can assign the part to a specific bin, so **IG-XL** can specially processes it later.

### Processing for Runtime Error

If a runtime error occurs while a test is being executed, the flow controller takes these actions:

- If the *Result* column value for the test is *Fail* or *All*, the *Pass/Fail Result* variable for all sites under test is set to *Fail*.
- For a **set-error-bin** opcode executed in this flow, the *Bin* variable for the sites is set to the error bin number; if **set-error-bin** also specified the optional sort number, the site *Sort* variable is also set.
- If a set-error-bin has not been executed, the *Bin* and *Sort* variables are set according to the *Fail Bin* and *Fail Sort* columns for the test.
- If the *Result* column value for the test is *Pass* or is blank, no errors are processed.

Errors are processed only for the sites that are under test when the error occurs. These sites are defined as the *selected sites*. In some cases, even in parallel test, a single site or only a few sites may be selected at a given time in the test execution. In these cases, only the sites actually under test processes any errors.

### *set-error-bin Opcode*

This opcode sets the bin number and the optional sort number assigned to sites that are under tested when a runtime error occurs.

On the same line as the **set-error-bin** opcode, enter the bin number in the *Bin Number Fail* column; you can also enter an optional sort number in the *Sort Number Fail* column.

Requirements:

- Only one **set-error-bin** opcode in a flow table.
- Opcode must occur before any *Test* opcode.
- Bin and sort numbers must be constants. Flow expressions or variables are not permitted.
- Error bin cannot be the same as the retest bin; refer to *Retest Processing* on page 3-241.

## Retest Processing

### Overview

In certain cases, a test result states that a device should be retested, instead of immediately assigning it a pass/fail status. For example, if a fail result is due to poor contact between the device and the interface board, the device should be reseated and then retested. The retest, however, should not corrupt the data already collected: the original test and the retest should be counted as a single test, and the retested device should not increment the device count twice.

You can designate a bin as the retest bin, meaning that devices assigned to this bin are retested, and the device counts and datalogged results are adjusted accordingly.

### set-retest-bin Opcode

A site that should be retested is assigned a bin number and the optional sort number by the **set-retest-bin** opcode. The numbers are entered in the *Bin Number Fail* and *Sort Number Fail* columns.

This opcode has the following requirements and restrictions:

- Only one **set-retest-bin** opcode in a *Flow Table*.
- This opcode must occur before any **Test** opcode.
- Bin and sort numbers must be constants; no flow expressions or variables.
- Retest bin cannot be the same as the error bin; refer to *Error Processing* on page 3-239.

### *Retest Sequence*

- The **set-retest-bin** opcode designates a retest bin.
- When a job completes, if a site has been assigned to the retest bin, **IG-XL** informs the handler or prober that the next run at that site is a retest.
- If the handler or prober does not support retest, the site is binned out; no error or warning is issued. The next test at the site will be a regular test of the next device, not a retest of the current device.
- If the handler or prober does support retest, the device remains at that site, and is retested.
- When a device is retested, the data collection software discards the summary counts from the previous run and uses the retest counts instead. The device ID is not incremented from the previous ID.
- If data is written to an STDF file, the retest data sets flags, which signify that the retest STDF data should replace the previous STDF data. Counters are also incremented to track the retests per site.

## Assume Capability

### Overview

The assume capability is for debugging conditional flows and binning schemes. By assuming that the specified sites pass or fail the test instances in a flow, you can observe how the flow reacts.

### Enabling the Assume Capability

- These columns are enabled only for those steps with a **Test** or **characterize** opcode; refer to *Test Execution Opcodes* on page 3-253.
- On the Run Options window, the *Assume* button must be checked.

### Debug Assume and Debug Sites Columns

- *Debug Assume*—sets the assumed result for the specified site: *Pass*, *Fail*, or *None*. *None* means no result is assumed. Each step can have a separate assume action; default is *None*. An empty *Assume* column means *None*.
- *Debug Sites*—specifies a list of sites to apply the assumed condition. It is a series of site numbers, each separated by white space. The reserved word **all** in the *Site* column means *all active sites* or all devices whose pass/fail result has not been set, not all sites.
- A blank column defaults to 0—site 0.

# Flow Table Opcodes

## Overview

Flow table opcodes are keywords that determine the action at each flow step. Opcodes are entered in the *Command Opcode* column. Valid opcodes are available from a drop-down list. When you select an opcode, the valid columns for an opcode are enabled, while invalid columns are disabled.

Some opcodes accept an operand, which are entered in the *Command Parameter* column. If an opcode takes a parameter, selecting the opcode enables the column. In the descriptions of opcodes, if an opcode takes a *Command Parameter* value, it is given in italics after the opcode.

Some of these opcodes can be conditioned with a group qualifier or device qualifier, or both.

## Administrative Opcodes

### Overview

The following opcodes administer the flow table, such as setting defaults and enabling flow words.

- Accept group qualifier: No
- Accept device qualifier: No

#### defaults

Specifies a default value for the test number and for the sort and bin numbers for all devices. On the same line as the **defaults** opcode, enter the default values in the appropriate columns: (1) default bin number in the *Bin Number Fail* column, (2) default Sort Number in the *Sort Number Fail* column, and (3) default test number in the *Tnum* column.

This opcode can create flags, even though flags can be created on any line. Enter a flag name in the *Flag Pass* or *Flag Fail* column. A newly-created flag is initialized to *clear* (neither *pass* nor *fail*) for all devices.

A single **defaults** can set any combination of values. You can also leave columns blank.

**defaults** is often the first entry in a flow table, but may appear anywhere.

#### reset

Clears all state information accumulated since the flow began by returning each cleared item to its value before the flow started. Because it clears each device pass/fail result, it also restores the original active site list: all sites are active.

You do not have to reset at the start of a flow; the **IG-XL** executes a reset before it starts executing the flow.

**reset** clears the following state information:

- *Pass/Fail Result* variable
- *Sort Number* variable
- *Bin Number* variable
- Logical *State* variable
- User-created flags

### enable-flow-word *enable-word*

Enables the execution of subsequent rows with the specified *enable-word* in the *Enable* column; refer to *Enable* on page 3-219.

### disable-flow-word *enable-word*

Disables the execution of subsequent rows with the specified *enable-word* in the *Enable* column; refer to *Enable* on page 3-219.

### set-error-bin

Sets the bin number for the error bin. Requirements are listed in *set-error-bin Opcode* on page 3-240.

### set-retest-bin

Sets the optional sort number for the retest bin. Requirements are listed in *set-retest-bin Opcode* on page 3-241.

### nop

**IG-XL** does nothing in this step and the flow proceeds to the next step.

## Flow Variable Opcodes

### Overview

The *Flow Table* supports integer variables that allow communication between the flow table and the executing test instance code. These variables can set the bin and sort numbers and test numbers by the following two methods:

- Flow table opcodes assign values to these variables; the variables can also be tested in group qualifiers, which control the flow based on the current value of a variable.
- Functions called from the test instance code can read and set variable values.

**IG-XL** provides two kinds of integer variables:

- Site variables are specific to a site; refer to *Site Variable Opcodes* on page 3-260. Because creating a site variable creates a variable for all sites, it, therefore, does not accept a device qualifier. In contrast, you can set the variables for specific sites; thus, an existing site variable accepts a device qualifier.
- Flow variables are global to the entire job. These opcodes create, assign values to, and delete flow variables, which communicate global information between the flow table and the execute test instance code.
- Accept group qualifier: No
- Accept device qualifier: No

In the following three opcodes, the *variable* is a string naming the variable, and *value* is a constant integer value. Use a blank to separate *variable* and *value*.

#### create-integer *variable*

Creates the named flow *variable*.

#### assign-integer *variable value*

Assigns the named integer *value* to the named integer flow *variable*.

#### delete-integer *variable*

Deletes the named *variable*.

### Restrictions

Execution of these opcodes cannot be conditioned with a group or device qualifier.

In the flow table, you can use a flow variable in the *Bin Number Pass* or *Fail* column and *Sort Number Pass* or *Fail* column, as well as in the *Tnum* column.

### Group Qualifiers

Flow variables can be used in a group qualifier to condition a flow step by entering them in the *Group Condition* column, with the appropriate parameter in the *Group Name* column:

**variable-exists *variable***—**True** if the variable exists.

**variable-equals *variable value***—**True** if the *variable* equals the specified integer *value*.

**variable-and *variable value***—**True** if the bitwise **AND** of *variable* and *value* is non-zero.

These *Group Condition values* do not accept a *Group Specifier value*. Most group qualifiers include a group specifier: the group of devices for the condition to be true, for example, all or any-active. In contrast, the group qualifiers for flow variables consist only of a group condition and a group name; they do not include a group specifier, because the flow variables are global, not specific to individual devices.

For the group qualifier **variable-and variable *value***, *value* is a constant integer bit-mask that determines if a given bit in the *variable* is turned on or off. Value is first bitwise **AND**'ed with variable. The result is then treated logically: if any 1's leak through the mask, the group specifier is true and the step is executed; otherwise, the group specifier is false, and the step is not executed.

## Flow Control Opcodes

### Overview

These opcodes control the flow of execution. They are usually used with group qualifiers, so that the flow can change depending on the results of previous test execution. They do not operate on devices; thus, they do not take a device qualifier.

- Accept group qualifier: Yes
- Accept device qualifier: No

### stop

Terminates a flow table:

1. An instance result is set for each device that was active when the flow began.

   If the flow set the device *Pass/Fail Result* variable to *Pass*, the instance result for that device is *Pass*; otherwise, the instance result for the device is *Fail*.
2. If the flow set the *Sort Number* variable, a device sort value is written for each device; otherwise, the device sort value is not altered, default is 0.
3. If the flow set the *Bin Number* variable, a device bin value is written for each device. otherwise, the device bin value is not altered, default is 0.
4. If you use **stop**, Teradyne recommends that you use the **defaults** opcode to establish default *Sort Number* and *Bin Number* values.
5. If a **stop** does not explicitly end the flow execution, a **stop** is implicitly performed when the final step in a flow is executed.

### goto *label*

Continues execution of the flow at the flow step whose label is specified in the *Command Parameter* column.

Restrictions:

- **goto** cannot branch to itself, because an infinite loop would result.
- **goto** cannot branch backward, because this could cause an infinite loop; thus, **goto** supports forward-only branching.

**goto-on-all-done *label***

Jumps to the label specified in the *Command Parameter* column after setting the pass/fail result state variable for each device, meaning that testing is done. The **goto-on-all-done** condition is cleared. This opcode usually appears near the top of the flow. Once it is executed, the flow table condition is checked after every subsequent step.

**goto-on-all-lastfail *label***

Jumps to the *label* when all devices fail the previous instance. This opcode usually appears near the top of the flow. Once it is executed, the flow table condition is checked after every subsequent step.

If all devices in the test instance's device list have failed the test, the flow branches to the label specified in the *Command Parameter* column, and the **goto-on-all-last fail** condition is cleared.

**skip**

Does not execute the **next** step in the flow. Note that **skip** is not **nop**: it does not skip the current step.

## Print Opcodes

### Overview

These opcodes print the specified string. They accept a literal text string as their operand, which is specified in the *Command Parameter* column. The string may not contain format conversion characters.

- Accept group qualifier: Yes
- Accept device qualifier: No

### Enabling Print Opcodes

The behavior of these opcodes depends on whether datalogging is enabled:

a. If datalogging is enabled, the print opcodes print the string to the specified output destination.

b. If datalogging is not enabled, the print and error-print opcodes print the string to the Test Program Output window, but only if **DataTool** is running in the *debug* mode. If this mode is not selected, these opcodes do not print.

To enable datalogging and to set up the printing options and destinations, refer to *Enabling Datalogging* on page 4-7, *DataCollect Setup*, Chapter 4.

**print** ***string***

If datalogging is enabled, the *string* is printed to the datalog output destination; otherwise, it is printed to the *Test Program Output* window when the tester is in the *debug* mode.

**logprint** ***string***

When datalogging is enabled, the *string* is printed to the datalog output destination; otherwise, it is not printed.

**error-print** ***string***

When datalogging is enabled, the *string* is printed to the datalog output destination; otherwise, it is printed to the *Test Program Output* window when the tester is in the *debug* mode.

\+ Printing an error message has no side effects: state of devices or the execution of the test program is not affected.

## Variable Initialization Opcodes

### Overview

These opcodes initialize a user-created flag or the *State* variable for all devices; refer to *Pass and Fail Flags* on page 3-235 and *State Variable* on page 3-238.

- Accept group qualifier: Yes
- Accept device qualifier: No

Because these opcodes operate on the flags and variables for all devices, they do not accept a device qualifier.

### Flag Opcodes

If a flag has not previously been created, **DataTool** will created it and the specified operation will be executed. Flags can be created in any flow step by specifying the desired flag name in either the *Flag Pass* or *Flag Fail* column.

**flag-true-all flag**

Sets the named *flag* to *True* for all devices.

**flag-false-all flag**

Sets the named *flag* to *False* for all devices.

**flag-clear-all flag**

Sets the named *flag* to *Clear* for all devices.

**state-true-all**

Sets the logical *State* variable to *True* for all devices.

**state-false-all**

Sets the logical *State* variable to *False* for all devices.

**state-clear-all**

Sets the logical *State* variable to *Clear* for all devices.

## Test Execution Opcodes

### Overview

These opcodes cause a test instance to be executed or characterized.

- Accept group qualifier: Yes
- Accept device qualifier: Yes

### Test *instance-name*

Executes the named test instance or test group. Enter the *instance-name*, which is defined on the *Test Instances sheet*, in the *Command Parameter* column.

Remember that devices are tested by flow tables executing instances of templates that can be executed more than once in a flow. Furthermore, you can set other *Flow Table* columns so they are executed along with the test instance:

- Result processing: assign values to columns: *Bin Number Pass* or *Bin Number Fail*, *Sort Number Pass* or *Sort Number Fail*, *Flag Pass* or *Flag Fail*, and *Result*.
- Datalogging: assign values to *Tname* and *Tnum*.
- Debugging: assign values to *Debug Assume* and *Debug Sites*.

### characterize instance-name *setup-name*

Characterizes the specified test instance by using the configuration in the specified setup. The *instance-name* is defined on the *Test Instances* sheet, while the *setup name* is defined on the *Characterization* sheet. Both are entered in the *Command Parameter* column, separated by a blank.

The statements about the **Test** opcode apply as well to the characterization opcode. One restriction is that the *instance-name* cannot be a test group; it must be the name of an individual test instance.

## Result Processing Opcodes

### Overview

These opcodes process the results; refer to *Processing Results* on page 3-231.

- Accept group qualifier: Yes
- Accept device qualifier: Yes

Because result processing is based on the results of the previous test execution, the opcodes accept a group qualifier. Furthermore, because these opcodes set the state variables of specific devices, they take a device qualifier.

### Common Restrictions and Limitations

- **set-device** and **set-device-new** set the *Pass/Fail Result*, the *Bin Number*, and the *Sort Number* as a unit. To set or modify these individually or to set or modify the *State* variable, use **modify.**
- **set-device** does not overwrite existing values; **set-device-new** overwrites any existing values.
- In general, **set-device** and **set-device-new** are usually at the end of the flow table, assigning the final result, bin, and sort values. **modify** is usually earlier in the flow, modifying specific values as required by immediate circumstances.
- **flag-true, flag-false,** and **flag-clear** set the values of specific flags. The other result processing opcodes do not affect flag values.

### *set-device*

Sets the *Pass/Fail Result*, *Bin Number*, and *Sort Number* variables for the devices in the device-list, provided the *Pass/Fail Result* variable has not already been set. Also, refer to *Usage* on page 3-255.

- Usage

  In general, **set-device** is usually at the end of the flow table, assigning the final result, bin, and sort values; thus, it delays setting these values until some or all of the tests have been run. You can use several methods of processing results:

  a. You can use a device qualifier to specify what devices are affected by **set-device**. In this way, you can execute a series of **set-device** statements, each with different device conditions, creating a device sorting algorithm separate from the test execution section of the flow table. The simplest flow uses the default device qualifier, which is all remaining active sites; refer to *Device Qualifier Values* on page 3-265.

  b. Another method: each *Test* opcode sets its *Result* column to *Fail*. Consequently, if a device fails the test, its *Pass/Fail Result* is set to *Fail*, and it is removed from the active site list. At the end of the flow, insert a **set-device** opcode with the default device qualifier and the *Result* column set to *Pass* or left blank, which is recommended; refer to *set-device with a blank Result Column* on page 3-257. This opcode causes all remaining devices—those passing all tests—to have their *Pass/Fail Result* set to *Pass*.

- Process

**set-device** examines *Pass/Fail Result* variable for each device in the device list:

a. Leave the *Result* column blank, refer to *set-device with a blank Result Column* on page 3-257.

b. If the device's *Pass/Fail Result* variable has already been set to either *Pass* or *Fail,* no action is taken.

c. If the device *Pass/Fail Result* variable has not already been set, the opcode sets the *Pass/Fail Result, Bin Number*, and *Sort Number* variables based on the value in the *Result* column:

If *Result* is *Pass*:

1. Set *Pass/Fail Result* variable to *Pass.*
2. Set *Bin Number* variable to the value of the *Bin Number Pass* column.
3. Set *Sort Number* variable to the value of the *Sort Number Pass* column.

If *Result* is *Fail*:

1. Set *Pass/Fail Result* variable to *Fail.*
2. Set *Bin Number* variable to the value of the *Bin Number Fail* column.
3. Set *Sort Number* variable to the value of the *Sort Number Fail* column.

- **set-device** with a blank *Result* Column

  Test time may significantly be reduced by using the **set-device** opcode with the *Result* column left blank. In this case, the *Result* defaults to *Pass*, with one important difference:

  a. If the *Result* column has *Pass* or *Fail*, the opcode cycles through the sites, shutting them down serially.

  b. If the *Result* column is blank, the opcode sets the result, bin, and sort variables to *Pass* values, but it does not shut down the sites.

  This feature is used at the end of a standard flow, where each *Test* opcode sets any failing devices to *Fail*, and where a final **set-device** opcode sets all remaining devices to *Pass*. If the **set-device** has a blank *Result*, the remaining devices are set to *Pass*, but the sites are not shut down until the flow terminates. At that time, the sites can be shut down in parallel, using **download**. The saving in test time can be significant; thus, Teradyne recommends this method. Also, refer to *set-device Opcode with Blank Result* on page 3-70.

***set-device-new***

Sets the *Pass/Fail Result*, *Bin Number*, and *Sort Number* variables for the devices in the device list, even if the result value has already been set.

This opcode is as **set-device** except that it is not conditional on the absence of an existing *Pass/Fail Result* value. It will perform the specified operations on all devices in the device list whether or not the *Pass*/*Fail Result* variable is already set.

**flag-true *flag***

Sets the specified user-defined *flag* to *True*.

**flag-false *flag***

Sets the specified user-defined *flag* to *False*.

**flag-clear *flag***

Sets the specified user-defined *flag* to or *Clear*.

**modify**

Change the *Pass/Fail Result*, *Bin Number*, *Sort Number*, and *State* variable for the devices in the device-list.

You modify the these variables by entering the new value in the appropriate columns of this row. You clear a value by entering a special keyword or value. If you leave a column blank, the value is not changed.

- *Pass/Fail Result* variable

  Set the *Result* column to *Pass*, *Fail*, or *None*, which clears the variable. If the existing value is *Fail*, setting it to *Pass* or *None* causes a warning to be issued.

- *Bin Number* variable

  Enter the new values in the *Bin Number Pass* and *Bin Number Fail* columns:

  a. If the current *Pass*/*Fail Result* variable is *Pass* or *None*, the *Bin Number* variable is set to the value of the *Bin Number Pass* column.

  b. If the current *Pass*/*Fail Result* variable is *Fail*, the *Bin Number* variable is set to the value of the *Bin Number Fail* column.

  c. If the row has a non-blank *Result* column, the *Pass*/*Fail Result* variable is set first and then the *Bin Number* variable is set based on the new *Pass*/*Fail Result* value.

+ Out-of-range values default to 0, and a warning is issued.

  d. To clear the *Bin Number* variable, set the *Bin Number Fail* column to -1.

- *Sort Number* variable

  Enter the new values in the *Sort Number Pass* and *Sort Number Fail* columns:

  a. If the current *Pass/Fail Result* variable is *Pass* or *None*, the *Sort Number* variable is set to the value of the *Sort Number Pass* column.

  b. If the current *Pass/Fail Result* variable is *Fail*, the *Sort Number* variable is set to the value of the *Sort Number Fail* column.

  c. If row has a non-blank *Result* column, *Pass*/*Fail Result* variable is set first, the *Sort Number* variable is set based on the new *Pass*/*Fail Result* value.

  d. To clear the *Sort Number* variable, set the *Sort Number Fail* column to -1.

- Logic *State* variable

  To change the logic *State* variable, set the *State* column to *True*, *False*, or *Clear*.

  You can enter values in the *State* column that perform an **AND**, **OR**, or **XOR** operation on the current *State* variable. The logic state is determined by whether the device is in the device-list as specified by the device qualifier for this step.

  The following truth tables show the relationship of the *and*, *or*, and *xor* values for the *State* column; *unchanged* means the current *State* variable is not changed:

| | *Current State* | *In device list?* | *New State* |
|---|---|---|---|
| *and* | *T*/*F*/*C* | *T* | unchanged |
| | v | *F* | *F* |
| | *F*/*C* | *F* | unchanged |
| *or* | *T*/*F*/*C* | *T* | *T* |
| | *T*/*F*/*C* | *F* | unchanged |
| *xor* | *T* | *T* | *F* |
| | *T*/*F*/*C* | *F* | unchanged |
| | *F* | *T* | *T* |
| | *C* | *T* | *C* |

## Site Variable Opcodes

### Overview

The *Flow Table* supports integer variables that are passed between the flow table and the executing test instance code. These variables can set the bin and sort numbers and test numbers by the following two methods:

- Flow table opcodes assign values to these variables; the variables can also be tested in group qualifiers, which control the flow based on the current value of a variable.
- Functions called from the test instance code can read and set these variable values.

**IG-XL** provides two kinds of integer variables:

- Flow variables are global to entire job; refer to *Flow Variable Opcodes* on page 3-247.
- Site variables are specific to a site. These opcodes create and assign values to site variables that pass site-specific information between the *Flow Table* and the executing test instance code. Each site has a separate site variable; thus, a site variable value is specific to an individual site, as compared to a flow variable, which is global to the entire job; refer to *Flow Variable Opcodes* on page 3-247.

  Because creating a site variable creates a variable for all sites, it, therefore, does not accept a device qualifier. In contrast, you can set the variables for specific sites; thus, an existing site variable accepts a device qualifier.

#### create-site-var *variable*

Creates the named site string *variable*.

- Accepts group qualifier: Yes
- Accepts device qualifier: No

**assign-site-var *variable value***

Assigns the constant integer *value* to the site string *variable*. Use a blank to separate *variable* and *value*.

- Accepts group qualifier: Yes
- Accepts device qualifier: Yes

  You use the device qualifier to assign a value to the site variable for only those devices that meet certain criteria. Note that you cannot reference a site variable by site number, because the flow table is not aware of site numbers.

- Group Qualifiers for Flow Variable Opcodes

  Site variables can be used in a group qualifier to condition a flow step by entering them in the *Group Condition* column, with the appropriate parameter in the *Group Name* column:

  **site-var> *variable value***—True if the *variable* is greater than the *value*.

  **site-var= *variable value***—True if the *variable* equals the *value*.

  **site-var< *variable value***—True if the *variable* is less than the *value*.

  These *Group Condition* values can be combined with any *Group Specifier* value; refer to *Group Specifiers* on page 3-262. For example:

  **all-active site-var> sitevar1 10**—True if all active devices have a sitevar1 value greater than 10.

## Group Specifiers

### Overview

Most of the *Group Specifier* values are always used with a *Group Condition* value. These *Group Specifier* values specify what devices must meet the group condition.

The *Group Specifier* values include unique specifiers for *all devices* and *all devices in the active site list*:

- *all devices*—all sites active when testing began.
- *all devices in the active site list*—Once the *Pass/Fail Result* variable for a device is set to *Pass* or *Fail*, it is removed from the active site list. Consequently, the active site list contains those devices not completely tested.

### Group Specifier Values

| *Specifier* | *Group Condition must be true for …* |
|---|---|
| **all** | All devices |
| **all-active** | All devices in the active site list |
| **any** | At least one device |
| **any-active** | At least one device in the active site list |
| **none** | None of the devices |
| **none-active** | None of the devices in the active site list |

Several *Group Specifier* values do not take a *Group Condition* value, because they test for general conditions:

| *Specifier* | *Condition tested for* |
|---|---|
| **always** | Always true (default group qualifier) |
| **logging** | True when datalogging is turned on. |
| **not-logging** | True when datalogging is turned off. |

Use **DataCollect Setup** to control datalogging; refer to *Enabling Datalogging* on page 4-7, *DataCollect Setup*, Chapter 4.

## Group Conditions

### Overview

*Group Condition* values state the condition that must be met by the devices specified by the *Group Specifier*; refer to ***Group Condition Values*** on page 3-264**.** Some *Group Condition* values require a parameter value that is entered in the *Group Name* column. If you enter a *Group Condition* value that requires a parameter, the *Group Name* column will be enabled.

Negate any *Group Condition* value by specifying **not** in the *Group Sense* column.

### Group Condition Values

| *Condition* | *True if the specified devices have …* |
|---|---|
| **pass** | Their *Pass/Fail Result* variable set to *Pass*. |
| **fail** | Their *Pass/Fail Result* variable set to *Fail*. |
| **done** | Completed testing: if its *Pass*/*Fail Result* variable has been set to either *Pass* or *Fail*. |
| **sort-set** | Had their *Sort Number* variable set. |
| **bin-set** | Had a *Bin Number* variable set. |
| **last-pass** | Passed the last instance executed. |
| **last-fail** | Failed the last instance executed. |
| **step-pass *label*** | Passed the flow step labeled *label*. |
| **step-fail *label*** | Failed the flow step labeled *label*. |

For **step-pass** and **step-fail**, if *label* does not identify a test instance that has already been executed, the group qualifier will be *False*.

| | |
|---|---|
| **flag-true *flag*** | Specified *flag* set to *True*. |
| **flag-false *flag*** | Specified *flag* set to *False*. |
| **flag-clear *flag*** | Specified *flag* set to *Clear*. |
| **state-true** | *State* variable set to *True*. |
| **state-false** | *State* variable set to *False*. |
| **state-clear** | *State* variable set to *Clear*. |
| **site-var> *var int*** | Site variable *var* greater than *int*. |
| **site-var= *var int*** | Site variable *var* equal to *int*. |
| **site-var< *var int*** | Site variable *var* less than *int*. |
| **variable-and *var int*** | Flow variable *var* whose **AND** with *int* is non-zero. |
| **variable-exists *var*** | *F*low variable *var* that exists. |
| **variable-equal *var int*** | *F*low variable *var* equal to *int*. |

## Device Qualifier Values

### Overview

A device qualifier consists of a *Device Condition* value, which specifies the condition that devices must meet to be included in the device list for a step.

Some *Device Condition* values require a parameter value, which is entered in the *Device Name* column. If you enter a *Device Condition* value that requires a parameter, the *Device Name* column will be enabled.

A *Device Condition* value can be negated by a **not** in the *Device Sense* column.

| *Condition* | *Devices specified* |
|---|---|
| **all** | All devices. |
| **pass** | Devices with *Pass/Fail Result* variable set to *Pass*. |
| **fail** | Devices with *Pass/Fail Result* variable set to *Fail*. |
| **done** | Completed testing: if its *Pass*/*Fail Result* variable has been set to either *Pass* or *Fail*. |
| **not done** | Testing not completed; (by inserting **not** in the *Device Sense* column) device *Pass/Fail Result* variable has not been set. It is neither *Pass* nor *Fail*. The active site list consists of the devices that have not completed testing. |
| **sort-set** | Devices with *Sort Number* variable set. |
| **bin-set** | Devices with *Bin Number* variable set. |
| **last-pass** | Devices that passed the last instance executed (regardless of whether a *Pass*/*Fail Result* variable was set). |
| **last-fail** | Devices that failed the last instance executed regardless of whether a *Pass*/*Fail Result* variable was set. |
| **step-pass *label*** | True for devices that passed the flow step labeled *label*. |

| | |
|---|---|
| **step-fail *label*** | True for devices that failed the flow step labeled *label*.<br>For step-pass and step-fail, if *label* does not identify a test instance that has already been executed, the device qualifier is *False*. |
| **flag-true *flag*** | Devices for *flag* set to *True*. |
| **flag-false *flag*** | Devices for *flag* set to *False*. |
| **flag-clear *flag*** | Devices for *flag* is *Clear*. |
| **state-true** | Devices for *State* variable set to *True*. |
| **state-false** | Devices for *State* variable set to *False*. |
| **state-clear** | Devices for *State* variable set to *Clear*. |
| **site-var> *var int*** | Devices whose site variable *var* is greater than *int*. |
| **site-var= *var* int** | Devices whose site variable *var* equal to *int*. |
| **site-var< *var int*** | Devices whose site variable *var* is less than *int*. |

## Flow Expressions

### Overview

Flow expressions on the *Flow Sheet* allow algorithms to determine test numbers, sort values, and bin values. They usually make the test results easier to understand.

### Flow Expression Operators

Flow expressions use a set of integer operators. Each operator is used with an integer operand.

| *Operator* | *Meaning* | *Value =* |
|---|---|---|
| + | Add | *present-value + operand* |
| - | Subtract | *present-value – operand* |
| > | Round up | *(( (present-value / operand) + 1) * operand)* |
| < | Round down | *(( (present-value / operand) – 1) * operand)* |

The syntax operator and operand cannot be separated by white space.

### Rounding

Rounding is usually associated with test numbers with other flow concepts, such as speed grading.

When rounding, the new value should be the next integer multiple of the operand, starting at the present value. For example, level corners might be given values in the 100's, and timing corners values in the 1000's.

Therefore, to round up, the present value must first be divided by the operand, and any remainder discarded. The result is the current increment of the base (operand value). One is then added to the base, and the result is multiplied by the base (operand). The new value is the next highest increment of the base value.

### Usage

Flow expressions can be used in the following columns of the *Flow Table* sheet:

- *Tnum*
- *Bin Number Pass*
- *Sort Number Pass*
- *Bin Number Fail*
- *Sort Number Fail*

+ These columns can also accept an integer constant instead of a flow expression.

## Parallel Testing

The *Flow Table* sheet is specifically designed to support parallel testing. All of the concepts associated with parallel test are fully supported, such as the concept of a test flow conditioned by the states of some of the devices.

This sheet implicitly manages parallel (multi-site) testing, without requiring explicit programming. You can program the sheet with minimal regard to the fact that testing may be taking place at multiple sites with differing pass/fail profiles. **IG-XL** handles this multi-site testing and binning, and also manages the active site list (as described in the next item).

With the assume capability you assign specific results to different sites, to help in debugging multi-site execution.

## Flow Expressions and Flow/Site Variables

Certain columns that take a numeric value also let you enter a flow expression to calculate a value. For example, a bin number may be set to 'current value plus two'. This is done by entering +2 in the bin field. Entering 2 in the bin field sets the bin number to 2.

For certain numeric columns, you can also use a flow or site variable. You use opcodes to create and assign values to these variables, which also permit communication with the executing test instance.

## Executing General Purpose Macros During the Flow

You may call general purpose routines to perform a calculation or other action by creating a custom user test on the *Test Instance* sheet (refer to *User-Written Functions* on page 3-198) and executing it in the *Flow Table* as if it were a standard test.

## Displaying Columns

A simple test flow does not use all columns; thus, you can hide and expose the less frequently used columns by clicking on the plus (+) or minus (-) sign above the columns; refer to *Hiding and Exposing Columns* on page 3-36.

## Sorting the Flow

The order in which test instances are executed in a flow can affect performance; refer to *Optimizing Test Programs* on page 3-70.

## Multiple Flow Tables

A workbook can contain multiple *Flow Table* sheets. In this case, the workbook must contain a Job List sheet to specify which *Flow Table* to use for a particular job;

# Flow Table Columns

## Label

Enter a name of a row label, which can be the target of the **goto** opcodes. The optional name is a string

## Enable

Enter an optional string that, if enabled, will permit the execution of the opcode on this row:

- If the *Enable* column is blank, the step is always executed.
- If this column contains a string, it is treated as a flow word, and the opcode is executed only if the flow word has been previously enabled.
- Flow word may be enabled at runtime as a Run Option, or from within the *Flow Table* by executing the **enable-flow-word** opcode on a prior row.

## Gate

### Gate Job

Enter at least one job context defined on the Job List sheet in which this step should be executed. If you enter more than one item, separate them with commas. The optional name is a string:

- If this column is blank, the step is executed for all jobs.
- If it is not blank, the step is executed only if the current *Active Job* context matches a job specified here. Set the job context by using the Active Job control on the **IG-XL** *Context* toolbar as part of the execution context. Options for specifying the job context for a step:
    a. Enter one or more job names, each with an exclamation point prefixed, such as *!WAFER*. The step is executed only if the current *Active Job* context is not among the job names listed here.
    b. You cannot mix these two options within a single cell. Either all of the job names must have *!*, or none of them can.

### *Gate Part*

Enter at least one part context defined on the Job List sheet in which this step should be executed. If you enter more than one item, separate them with commas. The optional name is a string:

- If this column is blank, the step is executed for all parts.
- If it is not blank, the step is executed only if the current Active Part context matches a part specified here. Set the job context by using the Active Part control on the **IG-XL** *Context* toolbar as part of the execution context. Options for specifying the job context for a step:

    a. Enter one or more part names, each with an exclamation point prefixed, such as *!PARTxx*. The step is executed only if the current *Active Part* context is not among the part names listed here. The Active Part control will list all strings (minus any *!*) entered in the *Gate Part* column.

    b. You cannot mix these two options within a single cell. Either all of the part names must have *!*, or none of them can.

To ensure the drop-down list is set up correctly, Teradyne recommends listing all part names in a single step. Find a step to be executed for all parts, preferably the first line of the *Flow Table*: because you are listing all part names here, the step must be executed for all parts. For example: a **nop** step that defines the part names. In the *Gate Part* column for that step, enter a complete, comma-separated list of all part names used for gating, which ensures all part names are in the drop-down list.

### Gate Env

Enter at least one environment context defined on the AC Specs or DC Specs sheet in which this step should be executed. If you enter more than one item, separate them with commas. The optional name is a string:

- If this column is blank, the step is executed for all jobs.
- If it is not blank, the step is executed only if the current Active Environment context matches an environment specified here. Set the job context by using the Active Environment control on the **IG-XL** *Context* toolbar as part of the execution context. Options for specifying the environment context for a step:

    a. Enter one or more environment names, each with an exclamation point prefixed, such as *!80C*. The step is executed only if the current *Active Environment* context is not among the environment names listed here.

    b. You cannot mix these two options within a single cell. Either all of the environment names must have *!*, or none of them can.

## Command

### Command Opcode

Enter in the cell or use the drop-down list to select one of the command opcodes to be executed. The required name is a string.

Some opcodes require a value in the *Command Parameter* column. If you select an opcode that requires a parameter, the *Parameter* column is enabled.

Valid values are listed on the drop-down list. This list has two parts. To view the part of the drop-down list not displayed, scroll to the end of the list and click on the keyword *Next* or *Previous*.

If the selected opcode can be conditioned based on a group qualifier, the *Group* columns are enabled. If the opcode accepts a device qualifier to specify the devices to operate on, the *Device* columns are enabled.

These opcodes cannot be conditioned with a group qualifier, and do not accept a device qualifier:

| *Opcode* | *Parameter* | *Function* |
|---|---|---|
| **defaults** | | Set defaults for bin, sort, and test number |
| **nop** | | Do nothing |
| **reset** | | Clear result, sort, bin, state, and flags |
| **enable-flow-word** | *word* | Enable any step containing *word* |
| **disable-flow-word** | *word* | Disable any step containing *word* |
| **set-error-bin** | | Set bin for run-time error |
| **set-retest-bin** | | Set bin for retest |
| **create-integer** | *var* | Create a flow variable *var* |
| **assign-integer** | *var int* | Assign integer *int* to a flow variable *var* |
| **delete-integer** | *var* | Delete a flow variable *var* |

These opcodes can be conditioned with a group qualifier, but do not accept a device qualifier:

| *Opcode* | *Parameter* | *Function* |
|---|---|---|
| **goto** | *label* | Jump to *label* |
| **goto-on-all-done** | *label* | Jump when all devices done |
| **goto-on-all-lastfail** | *label* | Jump when all fail last test |
| **skip** | | Skip test on next row |
| **stop** | | Terminate the *Flow Table* |
| **flag-clear-all** | *flag* | Initialize *flag* to *Clear* |
| **flag-false-all** | *flag* | Initialize *flag* to *False* |
| **flag-true-all** | *flag* | Initialize *flag* to *True* |
| **state-clear-all** | | Initialize logic state to *Clear* |
| **state-false-all** | | Initialize logic state to *False* |
| **state-true-all** | | Initialize logic state to *True* |
| **create-site var** | *var* | Create a site variable *var* |
| **print** | *text* | Print the *text* |
| **logprint** | *text* | Print the *text* |
| **error-print** | *text* | Print the *text* |

These opcodes accept a group qualifier as a conditioner and a device qualifier:

| *Opcode* | *Parameter* | *Function* |
|---|---|---|
| **Test** | *test* | Execute a test instance |
| **characterize** | *test setup* | Characterize *test* using *setup* |
| **set-device** | | Write results for devices |
| **set-device-new** | | Overwrite results, if needed |
| **flag-clear** | *flag* | Clear *flag* for these devices |
| **flag-false** | *flag* | Set *flag False* for these devices |
| **flag-true** | *flag* | Set *flag True* for these devices |
| **modify** | | Change bin, sort, result, state |
| **assign-site var** | *var int* | Assign integer *int* to site variable *var* |

### *Command Parameter*

Enter the operand that the opcode in this step requires. Certain command opcodes require a parameters; refer to *Command Opcode* on page 3-272. For example, the *Test* opcode requires the name of a test instance to be performed.

If the opcode selected in this step accepts an operand, the *Command Parameter* column is enabled. If the opcode does not take an operand, the *Command Parameter* column is disabled.

A *Command Parameter* is a flag, a label, a text string, a flow word, a test instance name, or a variable. The description of the *Command Opcode* column states whether each opcode requires and parameter and, if so, what kind of parameter.

Sometimes the *Command Parameter* consists of two elements: for example, assign-integer takes a variable and a value to be assigned to the variable, and characterize takes a test instance name and a characterization setup name. In these cases, you must separate the two elements with a blank.

If the *Command Parameter* is a test name, a bold test name indicates a test group; refer to *Test Name* on page 3-202.

## TName

Enter a name to identify the test for datalogging. The optional name is a string.

If the column is left blank, **DataTool** creates a datalog name from the test instance name specified in the *Command Parameter* column.

## TNum

Enter the base test number to be used by a test instance. The optional name is an integer, flow expression, or a flow variable.

If the column is blank, **DataTool** increments the test number from its current value.

Initial value of *Tnum* is set to zero.

## Bin Number

### Bin Number Pass

Enter the bin number to assign to devices that pass this test. This optional number can be entered as an integer, a flow expression, or a flow or site variable.

Use the **set-device**, **set-device-new**, or **modify** opcode to set the bin pass number.

### Bin Number Fail

Enter the bin number to assign to devices that fail this test. This optional number can be entered as an integer, a flow expression, or a flow or site variable.

Use the **set-device**, **set-device-new**, or **modify** opcode to set the bin pass number.

## Sort Number

### Sort Number Pass

Enter the sort number to assign to devices that pass this test. This optional number can be entered as an integer, a flow expression, or a flow or site variable.

Use the **set-device**, **set-device-new**, or **modify** opcode to set the bin pass number.

### Sort Number Fail

Enter the sort number to assign to devices that fail this test. This optional number can be entered as an integer, a flow expression, or a flow or site variable.

Use the **set-device**, **set-device-new**, or **modify** opcode to set the bin pass number.

## Result

Enter in the cell or select from the drop-down list how the *Pass/Fail Result* value of the device is set. This optional value is a string. For more information, refer to *Active Site List* on page 3-232 and *Pass/Fail Result* on page 3-236. You can also use the **set-device**, **set-device-new**, or **modify** opcode to set the *Pass/Fail Result* value.

Options for setting the result of the device:

| *Value* | *Meaning* |
|---|---|
| *Pass* | Pass if it passes this test. |
| *Fail* | Fail if it fails this test. |
| *All* | Pass or fail according to this test result. |
| *None* | Do not set the result state; default. |

To improve the performance of the test program, you can leave this column blank when a **set-device** opcode is at the end of the program flow; refer to *set-device Opcode with Blank Result* on page 3-70.

## Flag

### Flag Pass

Enter the name of a flag variable to set to *True* if the device passes the test; if it fails, the flag is set to *False*. This optional value is a string.

If the flag does not already exist, it is created and set to *True* or *False*, depending on the test result.

Group qualifiers and device qualifiers can check the flag value to control the execution flow.

### Flag Fail

Enter the name of a flag variable to set to *True* if the device fails the test; if it passes, the flag is set to *False*. This optional value is a string.

If the flag does not already exist, it is created and set to *True* or *False*, depending on the test result.

Group qualifiers and device qualifiers can check the flag value to control the execution flow.

## State

Enter in the cell or select from the drop-down list a value to which the logical *State* variable will be set. This optional, user-defined, string variable has values: *True*, *False*, and *Clear*. For more information, refer to *State Variable* on page 3-238.

By using the **modify** opcode, you can use the following values: *True*, *False*, *Clear*, *And*, *Or*, and *Xor*.

## Group

### Group Specifier

Enter in the cell or select from the drop-down list the group of devices that must satisfy the condition specified in the *Group Condition* column for the opcode to be executed. This optional value is a string. If the column is left blank, the default is the opcode is always executed (unconditional).

This column and the *Group Condition* columns are part of the group qualifier, which conditions execution of the step; refer to *Group Specifiers* on page 3-262.

This optional value is a string. Use the following values to define what devices must satisfy the group condition:

| *Specifier* | *Condition must be* **True** *for:* |
|---|---|
| *all* | All devices in the site list |
| *all-active* | All devices in the active site list |
| *any* | At least one device in the site list |
| *any-active* | At least one device in the active site list |
| *none* | No device in the site list |
| *none-active* | No device in the active site list |

In addition, the following *Group Specifier* values do not take a *Group Condition* value—they specify a condition that is not dependent on the state of devices:

| *Specifier* | *Condition* |
|---|---|
| *always* | Always **True**; default. |
| *logging* | **True** if datalogging is enabled |
| *not-logging* | **True** if datalogging is not enabled |

### Group Sense

Use this column to invert the selection in the *Group Condition* column by selecting *not* from the drop-down list or by entering *not* in the cell. This string value is optional; you can leave the cell blank.

## Group

### Group Condition

Enter in the cell or select from the drop-down list the condition that the devices specified in the *Group Specifier* column must satisfy for the opcode to be executed; also, refer to *Group Conditions* on page 3-263.

This optional cell value is a string; however, some values require a value in the column *Group Name* on page 3-281. You can invert the value by selecting *not* in the column *Group Sense* on page 3-279. Select from the following values, and also enter any required entry in the *Group Name* column:

| *Condition* | *Group Name* | *Condition is **True** if* |
|---|---|---|
| **bin-set** | | Device bin value has been set |
| **done** | | Device result has been set |
| **fail** | | Devices result is set to *fail* |
| **flag-clear** | *flag* | Devices *flag* is clear |
| **flag-false** | *flag* | Device *flag* is set to *false* |
| **flag-true** | *flag* | Device *flag* is set to *true* |
| **last-fail** | | Device failed the last test |
| **last-pass** | | Device passed the last test |
| **pass** | | Device result is set to *pass* |
| **site-var>** | *var int* | Device site variable *var* is > *int* |
| **site-var=** | *var int* | Device site variable *var* is = *int* |
| **site-var<** | *var int* | Device site variable *var* is < *int* |
| **sort-set** | | Device sort value has been set |
| **state-clear** | | Device logic *State* is clear |
| **state-false** | | Device logic *State* is *false* |
| **state-true** | | Device logic *State* is *true* |
| **step-fail** | *label* | Device failed test at step *label* |
| **step-pass** | *label* | Devices passed test at step *label* |
| **variable-and** | *var int* | *F*low variable *var* **AND** *int* is non-zero |
| **variable-exists** | *var* | Flow variable *var* exists |
| **variable-equal** | *var int* | Flow variable *var* equals *int* |

### Group Name

Enter the value required, if any, by the *Group Condition* for this step. This cell value is a string, such as a flag, a label, a variable, or a variable value. For the kind of *Group Name* value required by a specific *Group Condition* value, some refer to *Group Condition* on page 3-280. One type of *Group Name* consists of a variable name and an integer value for the variable. In this case, use a blank to separate the variable from the value. *Device Qualifier Values* on page 3-265.

If the *Group Name* column is disabled, the *Group Condition* for this step does not require a value.

## Device

### Device Sense

Use this column to invert the selection in the *Device Condition* column by selecting *not* from the drop-down list or by entering *not* in the cell. This string value is optional; you can leave the cell blank.

### Device Condition

Enter in the cell or select from the drop-down list the condition that the devices must satisfy for the opcode to be executed.

This optional cell value is a string; however, some values require a value in the *Device Name* column; refer to *Device Name* on page 3-283. The value can be inverted by selecting *not* in the *Device Sense* column. On the other hand, if the column is left blank, the default device qualifier (device sense plus device condition) is *not done*, which identifies a device whose result value has not been set; also, refer to *Device Qualifier Values* on page 3-265. Select from the following values, and also enter any required entry in the *Device Name* column:

| *Condition* | *Device Name* | *Devices Specified* |
|---|---|---|
| *all* | | All devices |
| *bin-set* | | Bin value has been set |
| *done* | | *Result* value has been set |
| *fail* | | *Result* value is set to *fail* |
| *flag-clear* | *flag* | *flag* is *clear* |
| *flag-false* | *flag* | Whose *flag* is *false* |
| *flag-true* | *flag* | Whose *flag* is *true* |
| *last-fail* | | Devices that failed the last instance |
| *last-pass* | | Devices that passed the last instance |
| *pass* | | Whose result value is set to *pass* |
| *site-var>* | *var int* | Whose site variable *var* is > *int* |
| *site-var=* | *var int* | Whose site variable *var* is = *int* |
| *site-var<* | *var int* | Whose site variable *var* is < *int* |
| *sort-set* | | Sort value has been set |
| *state-clear* | | Whose logic *State* is *clear* |
| *state-false* | | Whose logic *State* is *false* |
| *state-true* | | Whose logic *State* is *true* |
| *step-fail* | *label* | Failed the test at step *label* |
| *step-pass* | *label* | *P*assed the test at step *label* |

### Device Name

Enter the value required, if any, by the *Device Condition* for this step. This cell value is a string, such as a flag, a label, a variable, or a variable value. For the kind of *Device Name* value required by a specific *Group Condition* value, refer to *Device Condition* on page 3-282. For instance, one type of *Device Name* consists of a variable name and an integer value for the variable. In this case, use a blank to separate the variable from the value; refer to *Device Qualifier Values* on page 3-265. If the *Device Name* column is disabled, the *Device Condition* for this step does not require a value.

## Debug

### Debug Assume

Enter in the cell or select from the drop-down list one of the following instructions that force a test condition: *Pass*, *Fail*, or *None*. These optional values are strings.

When debugging the *Flow Table*, use *Debug Assume* and *Debug Sites* columns to force test conditions. The selected state are applied to the sites in the *Sites* column. For more information about this feature, refer to *Assume Capability* on page 3-243.

Before running a test program that uses this feature, you must enable *Assume* on the *Run Options* dialog box; refer to *Assume* on page 3-57.

### Debug Sites

Enter an optional list of sites that are forced to accept the pass/fail result listed in the column *Debug Assume* on page 3-283.

Guidelines and Requirements:

- String list must be space-delimited integers.
- Reserved word **all** means all active sites.
- Blank defaults to 0, which is site 0.
- Before running a test program that uses this feature, you must enable *Assume* on the *Run Options* dialog box; refer to *Assume* on page 3-57.

## Comment

Enter any optional notes or comments as a string.

# Global Specs Sheet

## Overview

This sheet defines symbols for global values, which are often specific to the tester. These symbols are used by other sheets in creating formulas.

## Using the Global Specs Sheet

For information about using global specifications, refer to *Spec Sheets* on page 3-38 and *Global Specs Sheet* on page 3-40.

# Global Specs Columns

## Symbol

Enter in the cell the name of the global symbol to be defined. The name in this required cell is a string.

When you create a *Global Specs* sheet, **DataTool** provides a set of default symbols that are listed in this column, along with the values appropriate for the tester.

## Job

Enter in the cell or select from the drop-down list the job that will use this symbol value. Job names defined on the Job List sheet are on the drop-down list. If this column is left blank the symbol value is valid for all jobs.

The name in this optional cell is a string.

## Value

Enter the numeric value or formula for the symbol. The formula can use other spec symbols defined on this sheet; however, it cannot use symbols defined on either the AC Specs or DC Specs sheet. Other sheets access this value as _<*symbol*>.

The data in this required cell is numeric.

## Comment

Enter any optional notes or comments as a string.

# Edge Sets Sheet

## Overview

### Purpose of Edge Sets Sheet

This sheet defines named combinations or sets of timing values and data formats used by particular pins or pin groups when running a test pattern. Use this sheet and the Time Sets sheet to define more than 32 sets.

### Key Concept: Edge Sets

An edge set represents a particular timing behavior experienced or performed by each pin or pin group. Even though this sheet can define any number of edge sets, a given test program can use no more than 32 sets of timing edges per pin/group in a single burst.

Edge sets are commonly identified by their symbolic name: *ContTest* or *HighSpeed*; alternatively, you can enter integer values.

## Using the Edge Sets Sheet

### Overview

Timing sheets can be complex because of the number of possible entries. For instance, each FLASH 750 pin can be individually programmed to have as many as 32 edge sets in any burst, each with 5 timing values in the *Normal* mode or 6 timing values in the *Extended* mode. To support this complexity, **DataTool** has an *Edge Sets* sheet to define all edge sets (data formats plus timing edges), and a separate *Time Sets* sheet to assign the named edge sets to pins and pin groups. On the other hand, most test programs do not require this level of complexity:

- For 32 or fewer sets of timing edges, use this sheet to both define the edge sets and to assign them to the pins. In many applications it can specify all test timing.
- For more that 32 sets, you must use separate *Edge Sets* and Time Sets sheets.

## Inserting a New Edge Sets Sheet

1. Select the *Worksheet* command from the *Insert* menu. The *Insert Worksheet* dialog opens.
2. On the I*nsert Worksheet* dialog:
   a. In the *Sheet Type* field, use to pull-down to select *Edge Sets*.
   b. In the *Sheet Name* field, enter the appropriate name of this new sheet.
   c. Click *OK*. The *Insert Timing Sheet* dialog opens.
3. On the *Insert Timing Sheet* dialog:
   a. In the *Tester Timing Mode* field, select *Normal Timing (100 MHz)* or *Extended Timing (50 MHz)*.
   b. Click *OK*.
4. A new *Edge Set* sheet is inserted.

## Entering the Edge Sets Data

The data entered on this sheet is usually taken from the manufacturer's specification sheet for the DUT. Because the DUT specifications are often based on other specification values, Teradyne recommends:

1. Defining the base specifications as spec symbols on the AC Specs sheet; refer to *Fundamentals of AC Specs and DC Specs Sheets* on page 3-41.
2. On the *EdgeSets* sheet, enter timing values as formulas that use the spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

## Defining Timing Values with Spec Symbols in Formulas

Timing values can be entered as formulas that use spec symbols defined on the *AC Specs* sheet and *Global Specs* sheet. Once you have defined these spec symbols, the *Spec Category* and *Spec Selector controls* on the **IG-XL** *Context* toolbar are active. The current value of the spec symbols is determined by the spec context selected by these controls; refer to *Spec Category and Spec Selector Controls* on page 3-44.

## Defining the Edge Sets Data Before the Time Sets Data

The data in the *Time Sets* sheet is dependent on the edge set names defined on the *Edge Sets* sheet and pin/group names defined on the *Pin List* sheet. These timing values are usually dependent on symbols defined on one or more *AC Specs*, *DC Specs*, or *Global Specs* sheets. Consequently, Teradyne recommends that you create these sheets in the following order:

1. On the *Edge Sets* sheet, define all timing for each pin.
2. On the Time Sets sheet, specify the groupings of those pin timings into named time sets by assigning an edge set to each pin.

+ Since you may have to create a *Time Sets* sheet instead of a *Time Sets (Basic)* sheet after determining that the test program requires more than 32 edge sets, you can cut and paste data from the *Time Sets (Basic)* sheet to the new *Time Sets* and *Edge Sets* sheets before deleting the *Time Sets (Basic)* sheet.

## Displaying the Edge Sets Data

Each time set definition, which contains a row for each pin or group, can be expanded and collapsed; refer to *Collapsing and Expanding Groups* on page 3-35.

The *Pin/Group Name* column can be filtered to identify all timing values for a given pin or group; refer to *Filtering Data in a Column* on page 3-37.

## Requirements and Restrictions

- Maximum number of edge set per pin: 32
- All edge sets in a burst must have same timing mode: either *Normal* or *Extended*.
- No more than 8 combinations of data source and data format may be used per channel within a given burst.
- Assigning edges to pins type *Gnd*, *Unknown*, or *Utility* causes a validation error.
- Every digital pin, either individually or as part of a pin group, must be mentioned once per edge set.
- Edge set 0 is reserved for internal Teradyne use.

# Edge Sets Field

## Timing Mode

This field displays the current mode of the sheet: *Normal* or *Extended*. When you insert a new sheet, you are asked whether the timing mode should be *Normal* or *Extended*. The required entry is a string: either *Normal* or *Extended.*

You can edit this field to change the timing mode. Changing the mode causes certain columns to be enabled or disabled, depending on which columns are valid for the new mode. In addition, the values available on drop-down lists will change. If an existing value becomes invalid in the new mode, you will need to change the value manually. If you do not make the change, the invalid value is detected at validation.

For the effect of timing mode on the pattern file, refer to *Pattern Modes* on page 6-16, *Pattern Language Reference,* Chapter 6.

# Edge Sets Columns

## Pin/Group

Enter the name of the defined pin or group. The required name is a string. Valid values are the pins or pin groups specified on the Pin Map sheet. You define all edge sets for each pin or group on consecutive rows.

Assigning edges to pins of type *Gnd*, *Unknown*, or *Utility* causes a validation error.

Expand or collapse each pin or pin group to view or hide all edge set definitions for that pin or group; refer to *Collapsing and Expanding Groups* on page 3-35.

### Graphical Editor for Timing/Format

Clicking the three-dot button ... in this cell open the graphical timing/format display, which shows the edges that this edge set has defined for all pins. Edges can be edited; refer to *Using the Graphical Timing/Formats Display* on page 3-179.

## Edge Set

Enter the name for this collection of edge sets. A setup collection consists of the specifications for each data setup. The required name is a string; refer to *Rules for User-Created Names* on page 3-25.

Defined names must be unique for each pin/group; thus, it may not be repeated within a given pin or group. You can reuse names across different pins or groups; however, Teradyne recommends that you reuse edge set names with other edge set names, which reduces possible confusion.

You can use a drop-down filter in the column header to display all time sets assigned to a particular pin or group; refer to *Filtering Data in a Column* on page 3-37.

For more information about the *mcg* mode, refer to *Multi-Clock Generator Mode* on page 3-174.

## Pin/Group

Select from the drop-down list or enter on successive column rows the name of each pin or group to apply the time set timing to. The required name is a string. Valid values are the pins or pin groups specified on the Pin Map sheet.

Assigning edges to pins of type *Gnd*, *Unknown*, or *Utility* causes a validation error.

You can use a drop-down filter in the column header to display all time sets assigned to a particular pin or group.

## Data

### Data Src

Select from the drop-down list or enter the source of the logic value to verify or to apply to the pin or group on this sheet. This required value is a string:

| | |
|---|---|
| *PAT* | Pin data is sourced from the pattern file (*.pat*). This value is most often used. |

+ In most cases, *Src* is set to *PAT*. The other values are generally used for low-level debugging; refer to *Debugging Vectors with DataTool* on page 3-175.

| | |
|---|---|
| *PATNOT* | Pattern is negated. Drive data is the complement of the pattern file data; receive data is not affected. This is a quick method of inverting the drive values without regenerating the test patterns. |
| PATHI | Pattern High. Logic high is substituted for the pattern data. |
| *PATLO* | Pattern Low. Logic low is substituted for the pattern data. |
| *ALLHI* | All High. Logic high is substituted for all pattern data; cycle type, drive and compare, are ignored. |
| *ALLLO* | All Low. Logic low is substituted for all pattern data; cycle type, drive and compare, are ignored. |

The combination of *Data Src* and *Data Fmt* values determines the hardware drive format to be used; refer to *Mapping of Hardware Drive Formats* on page 3-176.

### *Data Fmt*

Select from the drop-down list or enter the formatting to apply to the pin or group on this sheet. This required value is a string. The value specified depends on the waveforms required by the pin.

In most cases, *Fmt* should be set one of these: *RL*, *RH*, *NR*, *SBC*, *SBL*, or *SBH*. Most of the following values are standard except for *ROFF.* The specific format depends on the nature of the waveforms that the pin or group requires:

| | |
|---|---|
| *RH* | Return-to-High |
| *RL* | Return-to-Low |
| *ROFF* | Return-to-Off. This *NR* (Non-Return) format unconditionally forces the driver to turn off (go tri-state) at the D3 (Drive Off) event, regardless of the data in the next cycle. Usually, the D3 event goes to tri-state only if the subsequent cycle is a receive cycle. This format is available only in the extended mode. One application is to allow multiplexed pins to drive the DUT during part of a cycle, while allowing the DUT to drive during the remainder of the cycle. |
| *NR* | Non-Return |
| *SBC* | Surround-by-Complement. |
| *SBH* | Surround-by-High. |
| *SBL* | Surround-by-Low. |
| *STAY* | Hold the pin state. Used for debugging. |

**DataTool** may change the source/format combination you entered because it does not identify an existing hardware format; therefore, **DataTool** changes the format to an appropriate value; refer to *Mapping of Hardware Drive Formats* on page 3-176:

| *Src* Value | *Fmt* entered: | *Fmt* changed to: |
|---|---|---|
| *PATHI* | *RH* | *NR* |
| | *SBL* | *SBC* |
| *PATLO* | *RL* | *NR* |
| | *SBH* | *SBC* |
| *ALLHI* | *RH* | *NR* |
| | *SBL* | *SBC* |
| *ALLLO* | *RL* | *NR* |
| | *SBH* | *SBC* |

## Drive

### Drive On

Enter the delay of the cycle start from the beginning of a pattern period (*D0*). For surround formats, this is the time the surround value is first applied. This required value is a numeric formula. Leaving the cell blank disables the edge or entering the string *Disable* suppresses the edge.

If you enter a number, you can use a formula to include engineering units; refer to *Engineering Units in Formulas* on page 3-34. Or, you can enter the value as a formula that uses spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

### Drive Data

Enter when the data is applied (*D1*). Time is defined with respect to the beginning of the pattern period (*D0*). This required value is a numeric formula. Leaving the cell blank disables the edge or entering the string *Disable* suppresses the edge.

You can use a formula to include engineering units; refer to *Engineering Units in Formulas* on page 3-34. Or, you can enter the value as a formula that uses spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

### Drive Return

Enter the time that the data returns (*D2*) if a return format (*Fmt*) has been selected or that the surround data is reapplied if a surround format (*Fmt*) has been selected. The time is defined with respect to the beginning of the pattern period (*D0*). This value is a numeric formula and is required for formats other than *NR*.

You can use a formula to include engineering units; refer to *Engineering Units in Formulas* on page 3-34. Or, you can enter the value as a formula that uses spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

### Drive Off

Enter the time the tester switches off the driver when executing a receive cycle (*D3*). The time is defined with respect to the beginning of a pattern period. This required value is a numeric formula. Leaving the cell blank disables the edge or entering the string *Disable* suppresses the edge.

The *D3* definition depends on the mode:

- *normal*: edge is part of the receive cycle. Because the edge occurs in the receive cycle, you might program it at the beginning of the cycle, before the receive edge.
- *extended*: edge is part of the drive cycle and does not occur during receive cycles. Because the edge occurs in the drive cycle, you might program it near the end of the cycle, after the other drive edges.

You can use a formula to include engineering units; refer to *Engineering Units in Formulas* on page 3-34. Or, you can enter the value as a formula that uses spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

## Compare

### Compare Mode

Select from the drop-down list or enter the comparison mode to be used by the pin or group for all time sets defined on this sheet; *Mapping of Hardware Drive Formats* on page 3-176. This required value is a string:

| | |
|---|---|
| *Edge* | Perform an instantaneous value test at the time specified by *Open*. |
| *Windows* | Verify the state of the logic signal during the time period specified by the *Open* and *Close* values. This mode is available only in the *Extended* timing mode; refer to *Timing Mode* on page 3-181. |
| *Off* | Disregard the pattern data and perform no test. This mode is assigned by **DataTool** for *Input* type pins. Also, select this mode to disable testing of a device pin with many failures. |

If you set a value for one pin or group, **DataTool** applies it to all other occurrences of the pin or group on this sheet.

You can use a formula to include engineering units; refer to *Engineering Units in Formulas* on page 3-34. Or, you can enter the value as a formula that uses spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

### Compare Open

Enter when the output is tested (*R0*). Time is defined with respect to the beginning of the pattern period (*D0*). This required value is a numeric formula. Leaving the cell blank disables the edge or entering the string *Disable* suppresses the edge.

Minimum compare width (*Close* minus *Open*) is 2 ns.

You can use a formula to include engineering units; refer to *Engineering Units in Formulas* on page 3-34. Or, you can enter the value as a formula that uses spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

### Compare Close

Enter the time an extended (window) test of the device output ends (*R1*). The time is defined with respect to the beginning of the pattern period (*D0*). This optional value is a numeric formula, and is available only in the *extended* mode. Leaving the cell blank disables the edge or entering the string *Disable* suppresses the edge.

Minimum compare width (*Close* minus *Open*) is 2 ns.

You can use a formula to include engineering units; refer to *Engineering Units in Formulas* on page 3-34. Or, you can enter the value as a formula that uses spec symbols; refer to *Choosing a Spec Symbol* on page 3-39.

## Comment

Enter any optional notes or comments as a string.

# Job List Sheet

## Overview

This sheet specifies combinations of other sheets in a test program workbook. Each sheet combination defines one job. Also, this sheet can manage several job versions within a workbook.

## Using the Job List Sheet

### Key Concept: Jobs

**DataTool** uses the concept of jobs, which are test programs that are variations of each other. For example, two jobs may be identical except for their pin maps. On the one hand, you could create two nearly identical workbooks with only different *Pin Map* sheets, but this method would cause additional effort in developing and maintaining the program. Alternatively, **DataTool** lets you combine both programs in the same workbook. You define each program as a named job on the *Job List* sheet. The jobs would share all sheets except for the *Pin Map*: on the Job *List sheet*, in the *Pin Map* column, you specify the appropriate *Pin Map* for each job.

A workbook may have only one *Job List* sheet. If it does not have a *Job List* sheet, a workbook consists of a single job.

## Key Concept: Active Job

By defining multiple jobs on the *Job List* sheet, one of the jobs is the *Active Job*. Before validating and running the test program, you select the *Active Job* by using the *Active Job* control on the **IG-XL** *Context Toolbar*; refer to *Execution Context Controls* on page 3-55.

The concept of the *Active Job* is important for validating the test program because a test program can be validated only for a particular configuration of sheets, not all possible sheet combinations. For example, if a workbook has multiple *Pin Map* sheets, it is not meaningful or useful to validate all the test instances against all *Pin Maps*. Instead, the test program is validated within the context of the single *Active Job*, which associates one *Pin Map* sheet with one *Test Instances* sheet; thus, only the errors this particular combination are significant.

## Relationship to Other Sheets

For each *Job List* column, the names of all sheets of that type currently in the workbook are shown in the drop-down list. You can select a name from the drop-down list. Consequently, the sheet must exist before you can select it for a job on the *Job List* sheet.

In addition to the sheets listed on the *Job List* sheet, the Global Specs sheet also supports job-specific spec values.

# Job List Columns

## Job Name

Select from the drop-down list or enter the name of the job to be defined. This required value is a string; refer to the *Rules for User-Created Names* on page 3-25. The name of the currently *Active Job* is displayed in bold.

## Pin Map

Select from the drop-down list or enter the name of the *Pin Map* sheet used in this job. This required value is a string. Valid values are the current *Pin Map* sheets shown in the drop-down list.

## Test Instances

Select from the drop-down list or enter the name of the *Test Instances* sheet used in this job. This required value is a string. Valid values are the current *Test Instances* sheets shown in the drop-down list.

## Flow Table

Select from the drop-down list or enter the name of the *Flow Table* sheet used in this job. This required value is a string. Valid values are the current *Flow Table* sheets shown in the drop-down list.

## AC Specs

Select from the drop-down list or enter the name of the *AC Specs* sheet used in this job. This optional value is a string. Valid values are the current *AC Specs* sheets shown in the drop-down list.

## DC Specs

Select from the drop-down list or enter the name of the *DC Specs* sheet used in this job. This optional value is a string. Valid values are the current *DC Specs* sheets shown in the drop-down list.

## Pattern Sets

Select from the drop-down list or enter the name of the *Pattern Set* sheet used in this job. This optional value is a string. Valid values are the current *Pattern Set* sheets shown in the drop-down list.

## Pattern Groups

Select from the drop-down list or enter the name of the *Pattern Group* sheet used in this job. This optional value is a string. Valid values are the current *Pattern Group* sheets shown in the drop-down list.

## Bin Table

Select from the drop-down list or enter the name of the *Bin Table* sheet used in this job. This optional value is a string. Valid values are the current *Bin Table* sheets shown in the drop-down list.

## Characterization

Select from the drop-down list or enter the name of the *Characterization* sheet used in this job. This optional value is a string. Valid values are the current *Characterization* sheets shown in the drop-down list.

## Comment

Enter any optional notes or comments as a string.

# Pattern Groups Sheet

## Overview

This sheet associates pattern files into named groups, which are treated as a contiguous burst of vectors.

## Using the Pattern Groups Sheet

### Overview

Use this sheet to precisely control which patterns are loaded contiguously into the tester memory and executed as a single burst. Pattern groups defined on this sheets are traded as a single burst of vectors.

### Pattern File Requirement

If a pattern file is specified without a pathname, it must be in the same directory as the test program workbook (*.xls* file). Any relative pathname must be relative to this directory; otherwise, the pattern file must have an absolute pathname.

### Unique Pattern Group Names

Pattern group names defined on this sheet, pattern set names defined on the *Pattern Sets* sheet, and pattern filenames (minus the *.pat* extension) must be unique. Thus, if a pattern file is named *leakage.pat*, you cannot define a pattern set or pattern group named *leakage*. Or, if a pattern group is named *leakage*, you cannot define a pattern set named *leakage*, even if the pattern set does not contain the pattern group.

# Pattern Groups Columns

## Group Name

Enter the name of the pattern group to be defined.

The required name is a string; refer to *Rules for User-Created Names* on page 3-25. Also, a pattern group name cannot be the same as a pattern set name or as a pattern filename; refer to *Unique Pattern Group Names* on page 3-301. You can refer to the group names from the Pattern Set sheet or a test instance; they are displayed in bold to differentiate them from simple pattern names.

## Pattern File

Enter on successive rows the names of the pattern files to be concatenated into a group. They are loaded and executed in the order in which they are entered on this sheet. The required string is an absolute or relative filename with a *.pat* extension. If you enter a relative filename, its pathname is referenced to the folder containing the **IG-XL** workbook file; refer to *Pattern File Requirement* on page 3-301.

## Comment

Enter any optional notes or comments as a string.

# Pattern Set Sheet

## Overview

This sheet creates named lists of pattern files and groups to be executed or used by a test instance. Pattern groups are created on the Pattern Groups sheet.

## Using the Pattern Set Sheet

### Overview

This sheet provides options for how a test instance refers to pattern files. Pattern sets can use the same pattern file, but modify the execution options. For example, a single pattern file may be called from different test instances, each at a different start label. Thus, a test instance can specify different pattern set names that refers to the same pattern file, but with a different start label.

By using this sheet information, the pattern loader identifies all patterns used in a test and loads them into tester memory. Each *.pat* files in a pattern set is loaded and treated as a separate burst, while any pattern group in the pattern set is executed as a single burst.

### Pattern File and Pattern Group Requirements

- Named pattern groups are defined on the Pattern Groups sheet.
- Named start locations must exist in the named pattern file.
- If a pattern file is specified without a pathname, it must be in the same directory as the test program workbook (*.xls* file). Any relative pathname must be relative to this directory; otherwise, the pattern file must have an absolute pathname.

### Unique Pattern Set Names

Pattern group names defined on the *Pattern Group* sheet, pattern set names defined on this sheet, and pattern filenames (minus the *.pat* extension) must be unique. Thus, if a pattern file is named *leakage.pat*, you cannot define a pattern set or pattern group named *leakage*. or if a pattern group is named *leakage*, you cannot define a pattern set *leakage*, even if the pattern set does not contain the pattern group.

## Pattern Set Columns

### Pattern Set

Enter the name of the pattern set to be defined. The required name is a string; refer to *Rules for User-Created Names* on page 3-25. Also, a pattern set name cannot be the same as a pattern group name or as a pattern filename; refer to *Unique Pattern Group Names* on page 3-301.

### File/Group Name

Enter on successive rows the names of the individual pattern files or pattern groups defined on the Pattern Groups sheet that belong to this pattern set. Use an absolute or relative filename with a *.pat* extension. If you enter a relative filename, its pathname is referenced to the folder containing the **IG-XL** workbook file; refer to *Pattern File and Pattern Group Requirements* on page 3-303.

The required name is a string; they are displayed in bold to differentiate them from simple pattern names.

### Start Label

Enter the global label of the vector in the pattern file or pattern group that starts the pattern execution. This optional label is a string. If this column is blank, the pattern is executed from its first location.

Entering the keyword **subr** in this column means that the pattern file contains only subroutines; the pattern file is loaded but not executed. The subroutines can then be called from other pattern files.

## Stop Label

Enter the global label of the vector in the pattern file or pattern group at which the pattern execution stops. This optional label is a string. If this column is blank, the pattern is executed until it halts.

## Comment

Enter any optional notes or comments as a string.

# Characterization Sheet

## Overview

This sheet defines the setups that characterize a test instance. A setup consists of a number of rows, each of which contains information about a parameter to be varied.

## Using the Characterization Sheet

### Multiple Characterization Sheets

A workbook can contain multiple *Characterization* sheets. If the workbook has more than one *Characterization* sheet, a Job List sheet must specify which *Characterization* sheet is used for a particular job.

### Multi-Row Setups

A setup supports more than one row. The execution of a multi-row setup depends on the combination of modes in the setup. A mode defined in the *Mode* column is either *X Shmoo*, *Y Shmoo*, *Z Shmoo*, *Adjust*, *Measure*, or *Margin*.

In a *Shmoo* setup, all *Shmoo* rows—both the primary and tracking parameters—are used in each test execution. On the different types of *Shmoo* parameters, see running a Shmoo. The order of the primary X, Y, and Z axes is not significant, although a Y requires and X, and a Z requires an X and Y. A *Shmoo* supports rows for *Adjust* or *Measure*.

In a setup containing multiple *Adjusts* or *Margins*, each row is independent of the other rows. For example, if a setup has three *Adjust* rows, the first *Adjust* is performed, followed by the second, and finally the third is executed. These three separate operations are combined for convenience in a single setup; consequently, a single row of the *Flow Table* can execute the three *Adjusts* against a test instance.

+ A *Margin* setup cannot contains rows of any other mode. If a setup contains a *Margin* row, any other rows must also be *Margin*.

If a *Schmoo* includes *Adjusts* or *Measures*, the operations have the following order:

- *X*, *Y*, and *Z* axis parameter values are applied.
- Perform any *Adjusts*, in the order defined in the setup. Propagate the results to specs as defined.
- Perform any *Measures*, in the order defined in the setup.
- If *Measures* were performed, move to the next point. If no *Measures* are performed, either *Retest* or *Reburst* the test instance as defined.

## Using Interpose Functions

Interpose functions can be called during the execution of the characterization:

| *Function* | *When called* |
|---|---|
| **Start** | Before beginning the characterization |
| **PrePoint** | Before performing the test at each point |
| **PostPoint** | After performing the test at each point |
| **End** | Completion of characterization |

Any mode can call **Start** or **End** interpose functions. Only *Shmoo* and *Margin* can call **PrePoint** and **PostPoint** functions: because *Adjust* and *Measure* do not generate points, these functions would be meaningless.

In a *Shmoo* setup, only one set of interpose functions can be called. These can be specified on the first *Shmoo* row in the setup. The **PrePoint** and **PostPoint** interpose functions will be executed on each point generated for the X axis: X is the fast axis, changing on every point of the *Shmoo*.

Characterization interpose functions are declared as **Public** functions. Each function accepts a set of comma-separated arguments.

Characterization interpose functions can access and manipulate the characterization results. **IG-XL** creates an **RtaDataObj** for each setup on the active *Characterization* sheet, and fills the data object when the setup is executed. **RtaDataObj** provides properties and methods for accessing the characterization results. Typically, the **End** interpose function manipulates the characterization results just completed.

## Device Characterization

### Overview

Device characterization investigates the behavior of the DUT under various input conditions. It is used both during test program debug and during production testing to discover what combinations of input parameters cause the DUT to fail a test and what other combinations can cause it to pass. Exploring these pass/fail regions produces useful information about the manufacturing process and the appropriateness of the chosen test limits.

Although the following operations differ in many ways, they all modify or adjust the tester setup from what is defined in the **DataTool** worksheets. In addition, these changes are made at runtime, as opposed to when the program was created:

- *Shmoo* (1-, 2-, and 3-dimensional)
- *Margin*
- *Measure*
- *Adjust*, which is an edge-find operation that is more than the standard device characterization: it actually propagate changes to the tester setup.

### Characterization Features

- You can characterize from the *Flow Table*, as part of the regular testing or characterize interactively while stopped at a breakpoint in the **IG-XL** debugger.
- Adjustments can be made to hardware parameters and parameters on the *Pin Levels*, *Edge Set*, and *Time Set* sheets.
- Adjustments to raw AC or DC spec values. **IG-XL** determines the affect of changing a spec value, modifying any dependent hardware values as needed.
- Characterization results are viewed graphically or sent to a file or to a **DataTool** worksheet.
- Characterization setups define what parameters are varied, by what amounts, according to what algorithm, and so on. The setups are stored with the test program on the *Characterization* sheet. They can be called from the *Flow Table*, as part of a test program, or from a *Characterization Editor* for interactive debugging; the editor can also create or modify setups on the fly; refer to *Characterization Editor* on page 3-327.

### Characterization and Test Instances

Characterization is always performed on a user-selected *Test Instance*. The instance is any kind of test, including functional (digital) tests and parametric tests and even macros. Characterization does not interfere with or require any particular setup of the tester. It is controlled like any other test activity because it is like any other test; refer to Figure 3-1 on page 3-309.

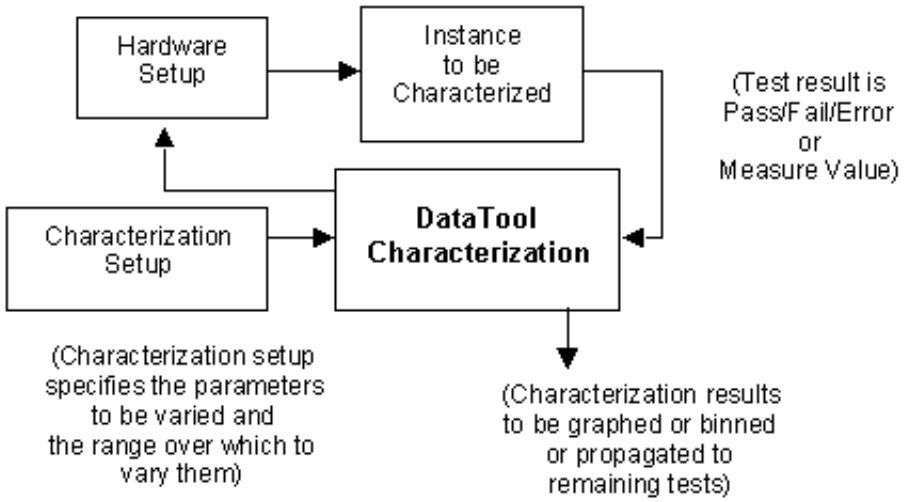



Figure 3-1. Simplified Characterization Flowchart

### Relationship to DataTool Sheets

The information for defining and running a characterization is on several sheets:

- *Characterization* sheet—defines the setup to be applied, including the parameters to be varied and the ranges over which they are to vary.
- *Test Instances* sheet—defines the test to be characterized, including the nominal timing and levels. Timing or levels or both are modified during characterization.

  In the *Adjust* mode, subsequent test instances on the *Test Instances* sheet can specify which adjusted value is applied to their base definitions.
- *Flow Table* sheet—contains a characterization opcode that specifies the *Test Instance* to characterize and the setup to apply to it.

## Characterization Modes

### Modes

- *Shmoo*—1-D, 2-D, or 3-D—Varies one, two, or three primary parameters or specs (and any number of tracking parameters or specs for each axis) over a range. For more information about Schmoo, refer to *Running a Schmoo* on page 3-313.
- *Margin*—Varies a primary parameter or spec over a range while recording *Pass, Fail,* or *Error* information. It is identical to a 1-D shmoo except for no tracking.
- *Adjust*—Varies a hardware parameter or spec over a range until the transition point between *Pass* and *Fail* is located, and optionally sets a spec to the final result. A *backoff* value can be applied to the result, ensuring a safety margin. By using a spec, the result can be propagated and overlaid onto subsequent test instances that can use the new adjusted value; refer to *Running an Adjust* on page 3-329.

  In this mode, the tester is in the *just passing* state if no spec receives the final result; however, if a spec is specified, the result is applied to the spec and the tester state is restored.

  An *Adjust* is also known as an *edge find*.
- *Measure*—Measures one or more parameters, finds the *Pass*/*Fail* transition point, and returns the measured value. *Measure* uses the same method as *Adjust*, but restores the tester state when the measurement has been made and does not set a spec with the result.
- *Shmoo* with *Measure* and *Adjust*—Similar to the standard *Shmoo* except that it performs one or more *Measure*s at each point instead of rebursting or retesting. In addition, an *Adjust* can be performed before the *Measure*s, to leave the tester in an adjusted state. A typical application shmoos $V_{DD}$, adjusts certain other parameters, and then measures the maximum operating frequency; refer to *Running a Schmoo* on page 3-313.

### Comparison of Characterization Modes

| Feature | Shmoo | Margin | Adjust | Measure |
|---|---|---|---|---|
| Tracking parameters | Yes | No | No | No |
| Locates pass/fail transition | No | No | Yes | Yes |
| Datalogs | Each point | Each point | At end | At end |
| Restores state | N/A | N/A | Yes (note) | Yes |
| Overlay tester setup | N/A | N/A | Yes | No |
| Backoff | N/A | N/A | Yes | No |

+ *Adjust* restores the tester state if a spec is named to receive the final result. If no spec is named, the tester remains in the *just passing* state.

## Running Characterizations

### Interactive Characterization

The typical interactive operation is to create a *Shmoo*; refer to *Running a Schmoo* on page 3-313; however, you can also run other characterizations interactively:

1. Run to a breakpoint on the *Flow Table* sheet; refer to *Retesting and Rebursting, and Setting Breakpoints* on page 3-312.
2. At the desired test, press the  button on the **IG-XL** toolbar to open the *Characterization Editor*. You can use it to select, edit, or create a characterization setup; refer to *Characterization Editor* on page 3-327.
3. Run the characterization by pressing the *Run* button on the editor. The editor lets you specify the output destinations. For a Shmoo, the logical choice is the *Shmoo Data Display* window, which lets you interactively query and re-shmoo the data points.

To examine the results of the characterization other than Schmoo, send the output to a worksheet in the job workbook or an ASCII output file.

### Characterization During Production Testing

Characterization during production testing allows you to check or verify the process or device operation over a range of inputs. The *Flow Table* supports a characterization opcode that specifies the test instance to be characterized and the characterization setup. The setup can specify a range of parameter values under which the test instance is expected to pass.

If the *Flow Table Result* column is set to *Fail*, devices with one or more failing points within the characterization will be binned out.

For more about running a characterization from a test program, refer to *Running an Adjust* on page 3-329.

### Retesting and Rebursting, and Setting Breakpoints

For each row of a setup, you can specify in the *Point Generation Test Method* column, you specify what part of the characterized test to execute repeatedly to generate the characterization results:

- *Retest*—reexecutes **Body** section of the current template. This method is applicable to all types of tests.
- *Reburst*—reruns current test pattern to be rerun. Used mainly in functional tests.

Both of these methods execute the **PreBody**, **Body**, and **PostBody**, with the following differences:

- With *Retest*, the **PreBody** is executed, then the entire **Body** is repeatedly executed until the desired result is reached. At that point, the **PostBody** is executed.
- With *Reburst*, the **PreBody** and **Body** are executed. At that point, the pattern is repeatedly reburst until the desired result is reached. At that point, the **PostBody** is executed.

Note the significance for setting a breakpoint when running interactively. When you run a to a breakpoint and then invoke the *Characterization Editor* to run a characterization, execution always starts at the **PreBody** of the test instance. Even if you run to a breakpoint and then step to before the **Body** (as indicated by the yellow flow arrow) prior to invoking the editor, the characterization will still start execution from the **PreBody**.

## Running a Schmoo

### Overview

A Shmoo varies one or more parameters over a range while recording *Pass/Fail/Error* information. It varies up to three primary parameters, which are displayed on the X axis, Y axis, and Z axis. Any primary parameter supports any number of secondary parameters that track the variation on the primary parameter.

Shmoos are run (1) within a test program, with results written to a worksheet or a file or (2) interactively from a breakpoint in the flow. The schmoo interface is a *Characterization Editor,* which lets you edit the characterization setup, and view and manipulate the displayed results.

### X, Y and Z Axis

A Shmoo setup defines the primary parameters that correspond to X, Y, and Z axis display values. The X axis is the *fast* axis, changing on every point of the Shmoo. The Y axis changes more slowly than the X axis, and the Z axis changes more slowly than Y. Parameters on to the Y and Z axis have the longest settling time.

### Tracking Parameters

The first row in a characterization setup whose mode is *X Shmoo*, *Y Shmoo*, or *Z Shmoo* is the *primary parameters* for that axis. Subsequent X, Y, or Z parameters are the *tracking parameters*, which vary in step with the primary parameter. These parameters are updated before the test instance is re-executed. For example, if the X axis has three parameters, they are updated before the test instance is executed.

A tracking parameter always uses the *Linear* algorithm, even if the primary parameter uses *List*. You specify the range over which the tracking parameter is varied. It uses the same number of steps as the primary parameter.

Tracking parameters are helpful when one parameter must move with another. For example, the output voltage comparator must track the power supply voltage. Thus, if the power supply voltage is varied over several steps for a Shmoo, the output parameter must vary by the same steps. Likewise, if the comparator voltage level has a spec-based definition using the power supply, then it will vary properly for the Shmoo. However, if the two definitions are independent, the Shmoo setup must explicitly define a tracking parameter: the power supply voltage is the primary parameter for that axis, and the output comparator voltage is defined as a tracking parameter on the same axis.

### Characterizing a Parametric Test

Typically, a Shmoo characterizes a functional test by rebursting a pattern under varying conditions. Also, **IG-XL** can characterize a parametric test; refer to *Schmooing a Parametric Test* on page 3-316.

### Standard Shmoos and Shmoos with Measures and Adjusts

- Standard Schmoo

  A standard Shmoo varies one to three parameters over a range to determine the effect of a particular set of conditions. For each point in a standard Shmoo, a pattern is rebursted or a test is rerun to determine whether the set of conditions results in a pass or a fail. At each point, a pass, fail, or error is recorded.

- Schmoo with *Measures*

  At each point determined by the Shmoo axes, this Schmoo performs one or more *Measure*s with the current set of conditions. Each *Measure* finds the pass/fail transition point and returns the measured value. If no transition point is found, the *Measure* returns a fail; a pass means that a transition point was found. If a setup contains multiple *Measure*s, each *Measure* is executed independently, and the result of each is recorded.

- Schmoo with *Adjust* and *Measure*

  This Shmoo performs an *Adjust* followed by a *Measure*. At each point, if the *Adjust* finds a pass/fail transition; a *backoff* value can be applied, leaving the tester in a *just passing* state. An *Adjust* restores the tester state only if it adjusts a spec. Then the *Measure* is performed, returning a measured value under the current Shmoo conditions and the *just passing* state.

- Order of Schmoo Operations with *Adjusts* or *Measures*

1. Apply the X, Y, and Z axis parameter values, setting up the current conditions.
2. Perform any *Adjusts*, in the order they are defined in the setup. Propagate the results to specs, if desired.
3. Perform any *Measures*, in the order they are defined in the setup.
4. If *Measures* were performed, move to the next X, Y, and Z point. If no *Measures* are performed, either reburst the pattern or rerun the test, as specified; then move to the next X. Y, and Z point.
5. As an indication of the final point: at each point, there will be either a retest/reburst (Shmoo), or one or more *Measure* operations, but not both.

## Running a Shmoo Interactively

1. Insert a *Characterization* sheet and define a Shmoo setup. For each displayed axis, define the parameter to be varied, their ranges, the number of steps, and other required information. You can define any number of tracking parameters for any of the primary parameters.
2. Optionally, you can define multiple setups, and select the one to use at runtime. Be aware that at runtime you can create a setup interactively.
3. Select the *Flow Table* sheet and set a breakpoint on the test instance to be characterized. The row must contain either the **Test** or **characterize** opcode; refer to *Retesting and Rebursting, and Setting Breakpoints* on page 3-312.
4. Start running the program in *Debug* mode.
5. When the breakpoint is reached, invoke the *Characterization Editor* by pressing the  button on the **IG-XL** toolbar; refer to the *Characterization Editor* on page 3-327.
6. Using the *Characterization Editor*:
   a. Select a setup from the those defined on the *Characterization* sheet. You can edit the selected setup.
   b. After completing the setup, press *Run*.
   c. The editor shows the resulting Shmoo in the *Shmoo Data Display* window. You can manipulate the data, querying individual points, or re-shmoo of one or more points; refer to the *Shmoo Data Display*.
   d. The Shmoo output can be written to a worksheet in the job workbook or to an ASCII file.
   e. After a Shmoo terminates, the tester remains in the state of the last performed Shmoo operation. The **PreBody** for the next test template resets all tester levels properly.

## Schmooing a Parametric Test

### Overview

You can also characterize a parametric test, such as BPMU, PPMU, and DPS tests. The procedure is almost the same as for a functional test, with the differences noted.

A parametric test generates a measured value for each pin; the measured values are compared to the test limits to determine the pass/fail results of the test. Both the pass/fail result and the measured values are datalogged. **IG-XL** displays these measured results as part of the Shmoo output. Also, you can access and manipulate the measured values.

### Creating the Characterization Setup

Create a characterization setup that contains a single X Shmoo row. Set *Parameter to Vary Type* to *Level* and *Parameter to Vary Name* to *Vps*. Specify the VPS range to be varied, the number of steps, and other parameters.

The *Point Generation Test Method* must be set to *Retest*, not *Reburst*. Parametric tests require this setup

To manipulate the parametric values datalogged by the test, create an **End** interpose function; refer to *Manipulating the Measured Results* on page 3-317. To call the interpose function, enter its name in the *Interpose Functions End* column.

### Outputting the measured results

A parametric test datalogs both the pass/fail result and measured parametric values. **IG-XL** displays these measured values as part of its standard output:

- If the Shmoo is run interactively, the *Shmoo Data Display* window displays the results. When you place the cursor over a point of the Shmoo plot, the tool tip or flyover displays the measured values after the test result. If many values are measured, the list may be truncated. The displayed measured values are not associated with specific pins. For this reason, the display summarizes the trend of measured values. To get specific results, use the programmatic access.
- If the characterization setup directs output to a file or a worksheet, the measured values will be included in the output.

### Manipulating the Measured Results

As the parametric test datalogs the pass/fail result and the measured values, the datalogged measured values are stored in a *LogVals* collection. Each member of the collection is an *RtaLogVal* object. Each *RtaLogVal* object is indexed by the name of the pin for the measured result.

# Characterization Columns

## Setup Name

Enter the name of the characterization setup to be defined. The required name is a string; refer to *Rules for User-Created Names* on page 3-25.

If a setup uses multiple rows, the *Setup Name* is the same for each row of the setup.

Click the three-dot button ... in this cell to open the *Characterization Editor* window; refer to *Invoking the Editor* on page 3-339.

## Row Name

Enter the program reference name of the data object for this row, which is enabled only for *Adjust* and *Measure*. This optional name is a string; refer to *Rules for User-Created Names* on page 3-25.

## Mode

Select from the drop-down list or enter in the text field the characterization mode for this row. This required value is a string. Valid values are in the drop-down list:

| | |
|---|---|
| *X Shmoo* | Defines an X-axis parameter for a *Shmoo* |
| *Y Shmoo* | Defines a Y-axis parameter for a *Shmoo* |
| *Z Shmoo* | Defines a Z-axis parameter for a *Shmoo* |
| *Adjust* | Defines an *Adjust* parameter |
| *Adjust From* | References a spec whose value was adjusted by the specified setup. |
| *Margin* | Defines a *Margin* parameter |
| *Measure* | Defines a *Measure* parameter |
| *None* | Defines row as a temporary **nop**. |

A single setup can contain multiple rows; refer to *Multi-Row Setups* on page 3-306.

## Parameter to Vary

### Parameter to Vary Type

Select from the drop-down list or enter in the text field the type of parameter to vary. The required name is a string. Valid values are shown in the drop-down list:

| | |
|---|---|
| *Level* | DC Level defined on a *Pin Levels* sheet. |
| *Edge* | timing parameter defined on a *Time Set* or *Edge Set* sheet. |
| *Period* | period value defined on a *Time Set* sheet. |
| *Spec* | Name of an *AC, DC,* or *Global* spec. |

*Type* restriction: A *Level* or *Edge* parameter can be varied on a pin only if the pin is already programmed to a state compatible with the parameter being varied. For example, you can vary the *Close* edge of a pin if the pin is already in *window* mode and has a *Close* edge programmed; however, if the pin is in *edge* mode (with an *Open* edge but no *Close* edge), trying to vary the *Close* edge will not change the pin from *edge* mode to *window* mode. Similarly, if an edge has the value *Disabled*, trying to vary the edge will not enable the edge.

### Parameter to Vary Name

Select from the drop-down list or enter in the text field the name of parameter—*Level*, *Edge*, or *Spec*—to vary. The required name is a string; however, it is optional if specified *Type* is *Period*. Valid values may be shown in the drop-down list, depending on the specified value in the *Parameter to Vary Type* column:

| *Specified Type* | *Drop-down list displays:* |
|---|---|
| *Edge* | Available timing edges. |
| *Level* | Available pin levels. |
| *Spec* | Enter the name of the spec to be varied. Do not include a leading underscore with the name. (is drop-down listing valid?) |
| *Period* | Do not enter a name. |

*Edge* and *Level* values are restricted, refer to *Parameter to Vary Type* on page 3-319.

## Range

### Range From

Enter the starting value for the parameter range. This cell value is **double**; it is required for *Schmoo* and *Margin* if the *Point Generation Algorithm* value is *Linear* or if *Mode* is *Adjust* or *Measure*; however, if algorithm is *List*, this column is ignored.

In the *Adjust* and *Measure* modes, the *Range From* and *Range To* values are the test limits, which means that if the indicated transition does not occur within the specified range, the test is datalogged as a *Fail*.

The *Range From* value must be less than the *Range To* value except for certain conditions in the *Adjust* and *Measure* modes. In these two modes, if the *Point Generation Algorithm* is *Pos Linear* or *Neg Linear*, (drop-down list shows just *Linear* and *List*), the *Range To* value can be less than the *Range From* value. This additional feature is called *reversed limits*, which can search backwards over the search range. Furthermore, if the limits are reversed, the *Range Step Size* value must be negative.

### Range To

Enter the final value for the parameter range. This cell value is **double**; it is required for *Schmoo* and *Margin* if the *Point Generation Algorithm* value is *Linear* or if *Mode* is *Adjust* or *Measure*; however, if the algorithm is *List*, this column is ignored.

In the *Adjust* and *Measure* modes, the *Range From* and *Range To* values are the test limits, which means that if the indicated transition does not occur within the specified range, the test is datalogged as a *Fail*.

The *Range To* value must be greater than the *Range From* value except for certain conditions in the *Adjust* and *Measure* modes. In these two modes, if the *Point Generation Algorithm* is *Pos Linear* or *Neg Linear*, (drop-down list shows just *Linear* and *List*), the *Range To* value can be less than the *Range From* value. This additional feature is called *reversed limits*, which can search backwards over the search range. Furthermore, if the limits are reversed, the *Range Step Size* value must be negative.

### *Range Steps*

Enter the number of steps for the parameter range. This optional value is an integer.

This column is enabled only for the *Schmoo* and *Margin* modes. The *Adjust* and *Measure* modes and the schmoo tracking parameters ignore this column. Also, the shmoo primary parameters ignore this column if the *Point Generation Algorithm* is *List*, not *Linear*.

Instead of specifying the number of *Range Steps*, you can enter the *Range Step Size*, and **DataTool** will calculate the number of points. The number of shmoo points equals the *Range Steps* + 1. For example, a shmoo plot from 0 volts to 1 volt in 10 steps will contain 11 points. On the other hand, if you specify both *Range Step Size* and *Range Steps*, *Range Steps* is used, and *Range Step Size* is ignored.

### *Range Step Size*

Enter the size of the range steps for the parameter to be varied. This cell value is **double**. It is optional for the *Schmoo* and *Margin* modes, and required for the *Adjust* and *Measure* modes.

Shmoo tracking parameters ignore this column. Also, the shmoo primary parameters ignore this column if the *Point Generation Algorithm* is *List*, not *Linear*.

For the *Shmoo* and *Margin* modes:

1. If you do not enter this value, **DataTool** calculates this value from the *Range From*, *Range To*, and *Range Steps* values.
2. If you specify both *Range Step Size* and *Range Steps*, the *Range Steps* value is used, and *Range Step Size* value is ignored.
3. Enter the resolution to be achieved. For example, to find a time delay to within 1 ns, enter 1E-9.

Under certain circumstances, the *Adjust* or *Measure Range To* value can be less than the *Range From* value. In this case, the *Range Step Size* must be negative; refer to *Range From* on page 3-320.

## Point Generation

### Point Generation Algorithm

Enter the name of the algorithm to be used for this characterization mode. The required name is a string. Valid algorithms, which depend on the specified *Mode*, are shown in the drop-down list:

- *Auto*—uses a variation of binary search even if the measured parameter drifts. This method requires that the *Range To* value be greater than the *Range From* value.
- *Pos*—uses the *Auto* method, but finds only a *Fail-to-Pass* transition. This method is identical to *Auto*, but specifies that the more positive the value, the more the DUT is likely to pass. It is faster than the *Auto* method by one test.
- *Neg*—uses the *Auto* method, but finds only a *Pass-to-Fail* transition. This method is identical to *Auto*, but specifies that the more negative the value, the more the DUT is likely to pass. It is faster than the *Auto* method by one test.
- *Pos Binary*—pure binary search method; finds only a *Fail-to-Pass* transition.
- *Neg Binary*—pure binary search method; finds only a *Pass-to-Fail* transition.
- *Pos Linear*—uses a linear method to find the first passing point in the search range; if the first point in the range passes, this method returns a *Stuck at Pass* error code.
- *Neg Linear*—uses a linear method to find the first failing point in the search range; if the first point in the range fails, this method returns a *Stuck at Fail* error code.

These linear methods, if used with big steps, can be the fastest way to find an edge, if accuracy is not important.

The pure and modified binary searches assume a single transition point; these searches stop as soon as they find a transition. If a range has more than one transition point, use a linear search to find them. For example, if a range contains both a fail-to-pass and a pass-to-fail transition, you can use *Pos Linear* to find the first and *Neg Linear* to find the second.

For *Shmoo* or *Margin*, select one algorithm for generating the schmoo points:

- *Linear*—sweeps from the *Range From* value to the *Range To* value in the number of steps calculated from *Range Steps* and *Range Step Size* values.
- *List* —Uses a specific set of values in a comma-separated list in the *Arguments* column for generating the schmoo points. Values in the *Range From*, *Range To*, *Range Steps*, and *Range Step Size* are ignored.

  A shmoo primary parameter can use either *Linear* or *List*, while a shmoo tracking parameter must use *Linear* even if the primary parameter uses *List*. For this reason, the column is disabled for tracking parameters.

For *Adjust* or *Measure*, select a method for locating the *Pass/Fail* point. The basic algorithms are pure binary search, modified binary search (*Auto*), and linear search. An algorithm may find any *Pass*/*Fail* transition point, only a *Fail-to-Pass* transition point (*Pos*), or only a *Pass-to-Fail* transition point (*Neg*). If a *Pos* or *Neg* search does not find a transition in the specified direction, the *Adjust* or *Measure* operation fails; the searches not designated *Pos* or *Neg* can find a transition in either direction.

### *Point Generation Test Method*

Select either re-execution of the test body or re-running of the current pattern. The required value is a string. Select one of the valid choices from the drop-down list:

- *Retest*—causes the **Body** section of the current template to be re-executed. This method is applicable to all types of tests.
- *Reburst*—causes the current test pattern to be rerun, which is applicable primarily to functional tests.

### Point Generation Arguments

Specify the additional arguments for point values in the *Shmoo* and *Margin* modes or the backoff value in the *Adjust* modes. This cell value is numeric. It is required for *Schmoo* and *Margin* modes if specified algorithm is *List*; optional for *Adjust* mode.

- For the *Shmoo* and *Margin* modes, if the *Point Generation Algorithm* value is *List*, enter a comma-separated list of point values for this axis. These values replace the *Range From*, *Range To*, *Range Steps*, and *Range Step Size* values. On the other hand, if the algorithm is *Linear*, any arguments are ignored.

  The point values must be in ascending order. If the list begins with a negative number, **Excel** will not accept the input, unless you enter a leading space before the negative sign; the leading space causes **Excel** to interpret the list properly.

- For the *Adjust* mode, optionally enter a backoff value that is applied to the final result. The backoff value is added to the result: a negative backoff value makes the result smaller; refer to *Running an Adjust* on page 3-329.

- For the *Measure* mode, the column is disabled.

## Apply To

### Apply To Pins/Groups

Enter a comma-separated list of the pins or pin group names whose parameter will be varied. Valid values are the pins and pin group names defined on the *Pin Map* sheet, which are *Level* and *Edge* types of pins or pin groups. If this column is blank, the default is all pins. You can use this column to restrict the parameter variation to a subset of pins. This optional cell value is a string.

### Apply To Time Sets

Enter a comma-separated list of the time set names whose parameter will be varied. This optional cell value is a string. It is used for *Edge* and *Period* types of time sets. Valid values are the time set names defined on the *Time Set* or *Time Set* (*Basic*) sheet. If this column is blank, the default is all time sets. You can use this column to restrict the parameter variation to a subset of time sets.

You can also enter a list of time set names, each prefixed with an exclamation point, such as *!SngClk* or *!DblClk*, which means to vary all time sets except the ones listed. If one name in the list has an exclamation point, all must have exclamation points; you cannot mix names with and without exclamation points.

## Adjust

### Adjust Spec Name

Enter the name of the levels or timing spec that will be set to the result of the *Adjust* operation or the name of the adjusted spec to be retrieved from another setup; refer to *Running an Adjust* on page 3-329. When entering a spec name, it must not begin with a leading underscore. Value spec names are defined on a spec sheet. This optional cell value is a string.

The function of this column depends specified type of *Adjust* mode:

- *Adjust* mode

  In this mode this row defines an *Adjust* operation, and this column specifies the name of the spec that receives the result of the *Adjust*. Once an *Adjust* is complete, the named spec is set to the result.

  The adjusted spec is typically referenced by a *Pin Levels*, *Timing Set*, or *Timing Set (Basic)* sheet as an overlay. In this way, the results of the *Adjust* are sent to the rest of the test program.

  If a spec is specified, the *Adjust* restores the tester state. On the other hand, if no spec is specified, the tester remains in the *just passing* state.

- *Adjust From* mode

  In this mode this row references a spec whose adjusted value has been set by an *Adjust* in the setup specified in the *Adjust From Setup* column.

  This feature lets you combine specs whose adjusted values come from different setups that have been applied to different test instances. This new setup can then be overlaid onto a later test instance.

### Adjust From Setup

Enter the name of the setup where the spec named in the *Adjust Spec Name* column has its adjusted value. This optional cell is a string. This column is enabled only if the *Mode* column specifies the *Adjust From* value. Valid spec names are defined on a spec sheet. The specified setup must be executed in the flow before the setup defined on this row, meaning that if *SetupA* has an *Adjust From Setup* value of *SetupB*, then *SetupB* must be executed in the flow before *SetupA*.

## Output

### Output File

Enter the filename to write the characterization output to. This optional cell is a string. If the filename already exists, it is overwritten.

### Output Sheet

Enter the name of the workbook worksheet to write the characterization output to. This optional cell is a string. If the sheet already exists, it is cleared and filled with the new data. You can specify the sheet name in the form *Workbook!Sheet*, where *Workbook* is the name of another workbook currently loaded in **DataTool**.

## Interpose Functions

### Interpose Functions Start

Enter the name of an interpose function to be called before beginning the characterization. This optional cell is a string.

### Interpose Functions PrePoint

Enter the name of an interpose function to be called before beginning the test at each point. This optional cell is a string.

### Interpose Functions PostPoint

Enter the name of an interpose function to be called after the test at each point. This optional cell is a string.

### Interpose Functions End

Enter the name of an interpose function to be called at the end of characterization. This optional cell is a string.

### Interpose Functions Args

The *Start*, *PrePoint*, *PostPoint*, and *End* columns each have an adjacent Arg column. that accepts a comma-separated list of arguments for the particular interpose function; refer to *Using Interpose Functions* on page 3-307.

## Comment

Enter any optional notes or comments as a string.

# Characterization Editor

## Schmoo Data Display Window

### Overview

This window, which allows you to view and manipulate the Schmoo data, appears when the *Shmoo Data Display* box on the *Characterization Editor* is checked and after the *Run* button is pressed.

### X, Y, and Z Axis

- One-dimensional Shmoo (X axis only): display is a line of cells along the X axis.
- Two-dimensional Shmoo (X and Y axes): display is a grid of square cells.
- Three-dimensional Shmoo (X, Y, and Z axes), display is like the two-dimensional Shmoo, but includes a slider bar beneath the display, which is the Z axis. Moving the slider bar displays the different values of the Z axis.

### Site Buttons

**IG-XL** configures this schmoo display to have a *Site n* button for each site used, plus one *Site Merge* button. Press the individual *Site* buttons to see the results for that site; press *Site Merge* to see the results from all sites merged. These two displays provide slightly different information; refer to *Displayed Information* on page 3-327.

### Displayed Information

- Test result

  Each cell shows the parameter results by using a letter and a color:

  P—green = Pass

  F—red = Fail

  E—gray = Error

  No letter—white = No test result or unable to test

  On the *Site Merge* display, cells with both passing and failing sites mix green and red to indicate the relative number of passing and failing sites. In addition, these cells display a percentage of the sites that passed the test at this cell.

- Tool tip point

  1. On the display of an *individual site*, moving the cursor over a data point, which is a cell of the display, causes a tool tip box (flyover) to appear. It shows:

     In square brackets, the values for the X, Y, and Z parameters used at this point in the Shmoo.

     Test result generated by executing the test instance with these parameters— Pass/Fail; or an Error message if executing the test caused a runtime error.

     Measured value or values if the Shmoo setup also contains one or more *Measures* or *Adjusts*, or if the characterized test is a parametric test.

  2. On the *Site Merge* display, the tool tip displays only the X, Y, and Z values.

- Identity

  The cells at the bottom of the widow display the following names:

  Characterization setup

  Test instance

  AC category and selector

  DC category and selector

### *Reshmooing a Cell*

To reshmoo an individual cell, double-click on it. The point is reshmooed with the appropriate X, Y, and Z coordinates, and the result are displayed.

Reshmooing is most useful to determine the repeatability of a particular result of cells along the pass/fail line.

### Zoom Button

To zoom, select a rectangle of cells by clicking on one corner of the rectangle, drag it to the opposite corner, and press the *Zoom* button. The window disappear and then reappears with a Shmoo of the new area. The rectangle selects the new *From* and *To* values for the X and Y axes. The new Shmoo uses the same number of steps on each axis as the previous Shmoo.

Zooming changes the *From* and *To* values in the *Characterization Editor*. After returning to the editor, press *Apply* to save the new *Characterization* values.

### Editor Button

After you have finished viewing the schmoo, press the *Editor* button to return to the *Characterization Editor*. Clicking on the *x* in the title bar also returns to the editor.

## Running an Adjust

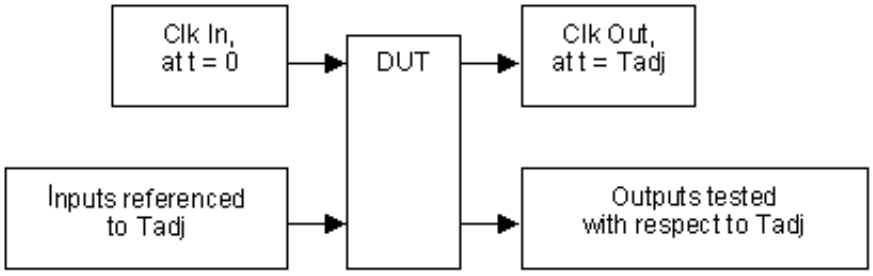
### Overview

An *Adjust* varies a hardware parameter or spec over a range until the transition point between pass and fail is located. It is also known as an *edge find.*

*Adjust* is like *Measure*. An *Adjust* determines a value and then propagate it to other sheets as a spec; in contrast, *Measure* determines the value without propagating it.

An *Adjust* is used in two ways in a test program:

- Test a propagation delay or input voltage threshold.
- Locate an output edge of the DUT and adjust the tester setup based on the actual location of that edge. As shown below, use this method when the DUT timing is referenced to an output event that cannot be determined before runtime.




Once *Tadj* is measured, an AC spec propagates its value to timing sheets, causing the edge and window placements for the device to be recalculated.

### Parts of an Adjust

An *Adjust* uses elements in the *Characterization* sheet, *Test Instances* sheet, and *Flow Table* sheet:

- *Characterization* sheet defines a characterization setup for the *Adjust*. The setup specifies the parameter to be varied, the range to be varied, and the algorithm to find the pass/fail transition point: either a variation on binary search or a linear step method. The setup also specifies in the *Adjust Spec Name* column, a levels or timing spec to receive the parameter value found by the *Adjust*. By assuming this spec will be used in the definition of other levels or timing values, the value found by the *Adjust* is propagated to the rest of the test program.
- *Test Instances* sheet defines the tests included in the flow. The test instance used for the *Adjust* is usually a functional test that runs a pattern, although parametric tests can be used. In addition, the *Test Instances* sheet can specify any subsequent test on which the adjusted value is overlaid.
- On the *Flow Table* sheet, the characterization opcode identifies the test instance used for the *Adjust*, plus the characterization applied to it. This opcode causes the test to be repeatedly performed at runtime, while the setup conditions are varied. The process is complete when the test just barely passes. If the adjusted value is not applied to a spec, the *Adjust* leaves the tester in the *just passing* state. If the adjusted value is applied to a spec, the tester hardware state is restored.
- The parameter value in force at the end of the *Adjust* can be applied to subsequent test instances in the flow. If the setup has specified a backoff value specified in the *Point Generation Arguments* column, it is applied to the result so the test passing has a certain safety margin. If a subsequent test instance has an *Overlay* column that specifies the *Adjust* setup, the adjusted value is overlaid on top of that base timing or levels definition.

  In addition, the *Test Instances* sheet defines the subsequent test instances using the propagated value found by the *Adjust*. For these tests, the *Overlay* column specifies the name of an *Adjust* setup. Before the test is executed, the adjusted value of the spec is overlaid on top of the spec of the same name the spec definitions of this test instance and propagated to this test instance.

  As long as the test program is running, the adjusted values generated by a setup remain available from one device to the next, unless the values are overwritten by a subsequent execution of the same setup. This feature can improve test times; refer to *Optimizing Timing Adjusts* on page 3-333.

### Contexts of the Setup and the Overlaid Test

Each test instance is executed within the context that the instance has specified: an AC context, consisting of the timesets sheet, and AC category and selector, and a DC context, consisting of the levels sheet, and DC category and selector. When you are running an *Adjust*, the test characterized by a setup has a context, and the test overlaid with the adjusted value also has a context, perhaps a different one.

**IG-XL** can run a setup in one context and apply the adjusted value to a test with a different context. When a setup is run, **IG-XL** finds all tests that will be overlaid with the adjusted value, and it creates an overlay specifically for the context of each test. Use this feature to run a slow speed test that gathers delay or setup/hold values that are not dependent on clock rate. These values can be overlaid on tests that are run at different speeds.

### Adjust Example

To determine an output delay and use its value in subsequent tests:

1. Insert a *Characterization* sheet and define a setup, *MyAdjust*, of mode *Adjust*.
2. Define the sweep parameter as type *Edge*, name *Open*. This is the detector *Open Edge*.
3. Set a range to perform the *Adjust*.
4. Set *Apply To Pins/Groups* column to the name of the tested pin.
5. Set *Apply To Spec Name* column to *Tadj*.
6. *Adjust* will locate the output event by varying the *Open Edge* of the pin until the test just passes. The spec *Tadj* receive the *Open Edge* value in effect when the *Adjust* operation end.
7. On the *Test Instances* sheet, define an instance, *MyAdjustTest*, for the *Adjust* to use.
8. On the *Test Instances* sheet, set the *Overlay* column to *MyAdjust* for subsequent tests using the *Tadj* spec. On all *Timing* and *Levels* sheets for these subsequent tests, make sure the equations depend on *Tadj*.
9. On the *Flow Table*, insert a line with the characterize opcode and a parameter value of *MyAdjustTest MyAdjust*, which performs the *MyAdjust Adjust* operation on the *MyAdjustTest* test instance.

### Adjust and Measure Failures

If an *Adjust* or a *Measure* fails, the following error messages appear:

Stuck at Pass
Stuck at Fail

An *Adjust* or *Measure* locates the point of pass/fail transition. If the test pass or fails at all points between *From* and *To*, there is no transition, and the operation fails. Stuck at Pass means that the test passed at all points; Stuck at Fail means it failed at all points.

In addition, the *Pos Linear* method remains at pass if the first point in the search range is a pass, and *Neg Linear* remains at fail if the first point is a fail; refer to *Pos Linear and Neg Linear Algorithms for Adjusts and Methods* on page 3-336.

Busy Adjusting

*Adjust* or *Measure* has been stopped, but the adjust process is not complete because:

- A linear search has been specified and the maximum iteration count of 100 tests has been reached.
- An *Adjust*/*Measure* was started but all sites are inactive, perhaps because failed tests caused parts to be binned out, meaning no device for the *Adjust*/*Measure* operation; refer to *Troubleshooting an Adjust or Measure* on page 3-334.

### Adjust From Mode—a Collection of Setups

The standard *Adjust* consists of a single setup with one or more *Adjust* operations that characterize a single test instance. The result of each *Adjust* is written to a spec, whose value can be overlaid on a later test instance.

You can also set up a spec whose adjusted values are from different setups that have been applied to different test instances. For example, *SetupA* used with *TestB* may adjust *SpecC*, and *Setup1* used with *Test2* may adjust *Spec3*. You may want to overlay both the adjusted specs *SpecC* and *Spec3* on *TestX*; however, only one setup can be overlaid on a test instance.

The solution is to use a setup with rows whose *Mode* column specifies the mode *Adjust From*. This mode does not perform an *Adjust*; instead, the value of the spec named in *Adjust Spec Name* is used. This value has been adjusted by the previous execution of the setup named in *Adjust From Setup*. In this example, a *SetupZ* has two rows: one specifying *SpecC* as adjusted by *SetupA*, and the other specifying *Spec3* as adjusted by *Setup1*. *SetupZ* could be used as the argument to a characterize opcode on the *Flow Table*. Then *SetupZ* could be overlaid on a subsequent test instance, meaning that the test instance will use the adjusted values of *SpecC* and *Spec3*.

A setup containing *Adjust From* rows can also contain *Adjust* or other operations.

### *Optimizing Timing Adjusts*

A timing *Adjust* can be performed on every device under test, but throughput is optimized by performing the *Adjust* only if the device is operating at a point that is sufficiently different to require an adjustment. Because the overlay values generated by an *Adjust* are available for as long as the program is executing, the adjusted values generated by characterizing one device can be used for subsequent devices.

The timing adjustment is optimized by inserting a new verification test before the test performs the *Adjust*:. The verification test, whose definition on the *Test Instances* sheet has an *Overlay* column that references the setup for the Adjust test. No overlay value is used on the first execution of the test program. On the first execution, the verification test uses the timing values loaded into hardware. On subsequent executions, if an *Adjust* has been performed, the verification test uses the overlay value. The test verifies the device timing is satisfactory for the rest of the test program:

- If the verification test passes, the *Adjust* test is not executed, and the existing overlay value is used.
- If the verification test fails, the *Adjust* test is executed, and a new overlay value is generated.

*Adjust* test, whose entry in the *Flow Table* has a group or device qualifier that executes only this test if the verification test fails; thus, the time-consuming timing Adjust is executed only when necessary. Also, refer to *Optimizing Test Programs* on page 3-70.

## Troubleshooting an Adjust or Measure

### Debugging an Adjust or Measure

The following list gives techniques can be used to debug Adjusts and Measures (*edge searches*) in **IG-XL**.

1. Test instance to be characterized must never fail due to marginal timing or intermittent synchronization. To verify a test can be used as a basis of an *Adjust* or *Measure*:
   a. Create a standalone test of the instance.
   b. Use the debugger to break at the test.
   c. Open the pattern debugger and loop on the test until it fails. If it fails, use standard debugging techniques to find the problem.
2. Modify the setup to call a custom **End** interpose function you have created. Set a breakpoint in the interpose function and execute the test program in the **IG-XL** debugger. At a breakpoint, use the regular debug displays to examine the state of the hardware at the completion of the *Adjust* or *Measure*.
3. Like the last technique, create a **PostPoint** interpose function called by the characterized instance. Use it to examine the state of the tester while measuring the value.
4. Value of a measured or adjusted result should not depend on the method used. Try duplicating the setup row and modifying it for another debugging method. For example, you may have performed an *Adjust* using the *Auto* method. Add a second *Adjust* to the same setup and try using the *Pos Linear* or *Neg Linear* method. Add a third *Adjust* and try the *Pos Linear* or *Neg Linear* method. All methods should return the same answer. If the results differ, consider:
   a. Drifting or unstable device behavior
   b. Non-repeatable test instance; or y
   c. Multiple events occurring in the search window.

### Reasons for Failing Adjust or Measure or Incorrect Results

- DUT requires synchronization, but the pattern does not synchronize, or the synchronization is not stable. This causes unrepeatable burst behavior, confusing the *Adjust* algorithm. The *Adjust* will usually find an answer, but it will not be stable, and it will not be correct.
- Setup does not focus its actions on a pinlist or timeset list or both because it is applying values to all pins in all time sets, creating illegal conditions for the DUT.
- Search range is too broad, encompassing multiple pin events. *Adjust* algorithm assumes a single transition point in the search range.
- If the setup is modifying an *Edge Open* or *Close* time, the pins being *Applied To* must be in the *Compare Edge* or *Window* mode. If the compare mode is *Off*, the *Open*/*Close* timing changes have no effect.
- For multiple *Adjust*s or *Measure*s in different timing sets in the same pattern, the pattern must pass when it is run as a standalone. If it does not pass a standalone run, the *Adjust* cannot influence the burst pass/fail result when the parameter is varied. The result is a *stuck at* condition.

## Optimizing an Adjust or Measure

- Use *Pos* or *Neg Binary* methods rather than *Auto*, *Pos*, or *Neg*.
- Use *Linear* methods with large step sizes.
- Narrow the search range or increase the desired resolution or do both.
- Conditionally perform an *Adjust.*
- Overlays remain after a single device is tested. Add an overlaid test in your *Flow Table* to verify that the current values of the overlay are adequate. If the test fails, repeat the *Adjust* to create new values for the overlay.

## Pos Linear and Neg Linear Algorithms for Adjusts and Methods

### Overview

A characterization setup requires a value for the *Point Generation Algorithm* column to specify the method to search for the pass/fail transition. For an *Adjust* or *Measure*, select of one the two options:

- *Pos Linear*—finds the first passing point in the search range; if the first point in the range passes. This method returns a Stuck at Pass error code.
- *Neg Linear*— finds the first failing point in the search range; if the first point in the range fails. This method returns a Stuck at Fail error code.

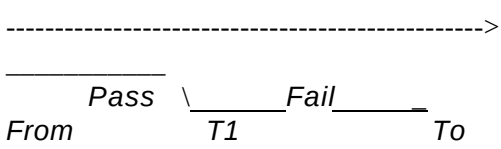
### Examples

The following examples show how these search methods work under specific conditions:

- Each shows a search range defined by the *Range From* and *Range To* values.
- If the Range *From* value is greater than the *RangeTo* value, the search uses *reversed limits*— backwards search from the higher value (*From*) to the lower value (*To*). This search is unique to the *Pos Linear* and *Neg Linear* methods.
- In each example, the search range contains a pass/fail transition. A range without a transition fails a search, producing a Stuck at Pass or Stuck at Fail error.
- In each example, the pass/fail transition occurs at time *T1*. If the search succeeds, the search stops at *T1+* (point greater than *T1*) or *T1-* (point less than *T1*).
- In each example pair, one search of the pair fails because the first point in the range is a pass for *Pos Linear* or a fail for *Neg Linear*. The search stops at the first point, *From*, and returns a Stuck error.
- The examples include the *Reason* and *ReasonText* values for each search. These are properties of the *Adjuster* object in **VB-Test**. The *Reason* gives the search result as an enumerated type value; *ReasonText* gives the result as a string. In addition, the *Reason* value is appended to the *MeasVals* collection of the *RtaDataObj*, if the value is not *adjReasonOK*.

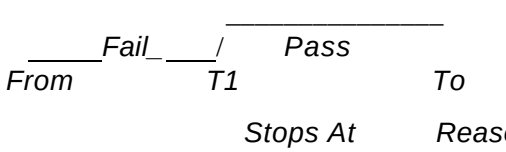
- Forward searches

In these searches, From is less than To, and the search is forward through the range.




| | Stops At | Reason | ReasonText |
|---|---|---|---|
| Pos Linear | From | adjReasonAllPass | Stuck at Pass |
| Neg Linear | T1- | adjReasonOK | Success |

The first point is *Pass*, so *Pos Linear* stops at *From* and returns Stuck at Pass. *Neg Linear* successfully finds the *Pass*/*Fail* transition, and returns the last passing point.

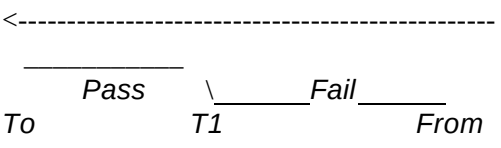



| | Stops At | Reason | ReasonText |
|---|---|---|---|
| Pos Linear | T1+ | adjReasonOK | Success |
| Neg Linear | From | adjReasonAllFail | Stuck at Fail |

*Pos Linear* successfully finds the *Fail*/*Pass* transition, and returns the first passing point.The first point is *Fail*, so *Neg Linear* stops at *From* and returns Stuck at Fail.

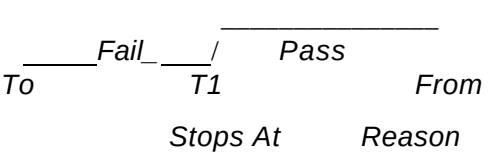
- Reversed limits searches

  In these searches, From is greater than To, and the search is backwards, from the higher limit to the lower. Note that the positions of From and To are reversed from the previous examples; the search direction is now from right to left.




| | Stops At | Reason | ReasonText |
|---|---|---|---|
| Pos Linear | T1- | adjReasonOK | Success |
| Neg Linear | From | adjReasonAllFail | Stuck at Fail |

  *Pos Linear* successfully finds the *Fail*/*Pass* transition and returns the first passing point. The first point is *Fail*, so *Neg Linear* stops at *From* and returns Stuck at Fail.




| | Stops At | Reason | ReasonText |
|---|---|---|---|
| Pos Linear | From | adjReasonAllPass | Stuck at Pass |
| Neg Linear | T1+ | adjReasonOK | Success |

  The first point is *Pass*, so *Pos Linear* stops at *From* and returns Stuck at Pass. *Neg Linear* successfully finds the *Pass*/*Fail* transition, and returns the last passing point.

## Using the Characterization Editor

### Overview

The *Characterization Editor* edits an existing characterization setup or creates one. Alternatively, you can directly edit the column of the *Characterization* sheet.

### Invoking the Editor

Invoke this editor from the *Characterization* sheet by pressing the three-dot button ... in the *Setup Name* column  .

You can also open up the editor during *interactive characterization*. On the *Flow Table*, set a breakpoint at a **Test** or **characterize** opcode and run the program in *Debug* mode; refer to *Test Execution Opcodes* on page 3-253. When the breakpoint is reached, invoke the *Characterization Editor* by pressing the  button on the **DataTool** toolbar or the *DUT Characterization Run* command from the **IG-XL** menu; refer to *Running a Shmoo Interactively* on page 3-315.

### Creating a Setup

Stop the test program at a breakpoint and open the *Characterization Editor* to create a setup. In the *Setup* field, set the value to *(New)*. A setup is created with a default name, such as *New1*. After you click *Apply*, the setup is saved to the current *Characterization* sheet. You can rename the setup.

### Creating, Viewing, and Deleting Rows

Any axis can have more than one row for any axis. First row is the primary parameter; the rest are tracking parameters; refer to *Running a Schmoo* on page 3-313.

- To create a row in a setup:

1. Note the selected boxes in the *Mode* field.

   If no rows have been assigned to the desired mode, check the appropriate box, which adds a tab for that mode. The default tabs are *X-axis*, *Y-axis*, and *Z-axis*, which appear for *Schmoo* checkbox.

2. Select the desired tab. Click *Add*. Fill in the fields.

   You must *Add* a row before entering the field values. When you select a mode, a tab is added for the mode if no rows for that mode exist in the setup. You cannot edit the tab, however, until you have click *Add* to add a row of that mode.

- To view or delete an existing row of a setup:

1. Select the tab for the mode of the row to view.

   For example, the label ***m* of *n*** means *1 of 1 or 2 of 3*, which is the number of rows of that mode in the setup, as well as the number of the current row; refer to the figure below.


   Use the < and > buttons to cycle through the rows.

2. You can edit the row you are viewing.

- To delete the row, click *Del*.

## Characterization Editor Controls

### Run, Apply, and Cancel Buttons

To run the characterization after setting it up, click *Run*.

To save changes you have made, press *Apply*; changes are made to the setups on the *Characterization* sheet.

To exit from the editor without saving any changes, press *Cancel*.

### Setup Field

Use this drop-down list to select the setup to edit or run. To create a new setup, select **(New)** at the top of the list.

By invoking the editor from the *Characterization* sheet, the name of the current setup is displayed in the *Setup* field; however, the control is disabled. You cannot change to a different setup.

### Mode Field

This field displays the modes of the rows currently in the setup. To add rows of a new mode to the setup, check the appropriate box. The added tabs contain the fields for the mode.

You must add a row before entering values for the new mode. After adding a mode, you can edit the tab for the mode; refer to *Creating, Viewing, and Deleting Rows* on page 3-340.

If you deselect a mode, the rows for that mode disappear. Or, if you reselect the mode, the rows reappear. To delete the rows for a mode permanently, click the *Del* button and then click *Apply*; Once the rows are deleted, you can deselect the mode.

This field has two mutually-exclusive group: one for *Margin*; another for all modes other than *Margin* (*Schmoo*, *Adjust*, *Measure*, and *Adjust From*); thus, the setup mode is either *Margin* or non-*Margin*. In non-*Margin* mode, to combine *Shmoo*, *Adjust*, and *Measure*: check the box for one or more of these modes.

For more information, refer to *Characterization Modes* on page 3-310.

If the *Margin* radio button is selected, checking any non-*Margin* box clears any *Margin* setups; if non-*Margin* boxes are checked, selecting the *Margin* radio button clears any non-*Margin* setups. You are asked to confirm this change.

### Output Field

- *Shmoo Data Display*—the *Shmoo Data Display* window will open when the characterization is run, which lets you query and reshmoo specific points; refer to *Schmoo Data Display Window* on page 3-327. Used for Shmoos only.
- *Worksheet* —specify the worksheet that receives the characterization data; refer to *Output Sheet* on page 3-326.
- *File*—specify the full path of the file that receives the characterization data or click the *Browse* button to open the standard Windows browser to select a file or folder for the output file; refer to *Output File* on page 3-326.

### Shmoo and Margin Tabs

+ Most of the editor fields on these tabs are similiar to columns of the *Characterization* sheet; thus, for more information about an item, refer to the reference for the appropriate *Characterization* column.

- Radio Buttons in *Mode* Field

  By checking the *Shmoo* box in the *Mode* field, tabs for *X-axis*, *Y-axis*, and *Z-axis* appear. The existence of a tab does not mean that the axis rows have been defined. Along with *Schmoo*, you can also check the *Adjust*, *Measure* or *Adjust From* boxes, or all four.

  By checking the *Margin* radio button in the *Mode* field, the *X-axis* tab appears. No other tabs are used with *Margin*; all rows of a *Margin* setup must be of the *Margin* mode. A *Margin* is similar to a one-dimensional (*X-axis*) Shmoo, except that tracking parameters are not permitted. If a *Margin* setup has more than one row, each row is executed independently as a separate *Margin* operation.

For more information, refer to *Characterization Modes* on page 3-310, *Mode Field* on page 3-341, and *Mode* on page 3-318.

- *Type* Field

  In the text field or from the drop-down list, select the type of parameter to vary: *Level*, *Edge*, *Period*, or *Spec*. For more information, refer to *Parameter to Vary Type* on page 3-319.

+ X-axis is used for both the X-axis of a *Shmoo* and for a *Margin*.

- *Name* Field

  In the text field or from the drop-down list, select the name of the parameter (level, edge, or spec) to vary. Valid values are listed in the drop-down list. For more information, refer to *Parameter to Vary Name* on page 3-319.

- *Algorithm* Field

  In the text field or from the drop-down list, select, the algorithm to generate the pass/fail points for the *Shmoo* or *Margin*:

  1. *Linear*—Sweep the range specified in the *From* and *To* fields in *Step Size* steps.
  2. *List*—Uses a set of comma-separated values specified as *Point List*

+ For more information, refer to *Point Generation Algorithm* on page 3-322.

- *Point List* Field

  If the *Algorithm* is *List*, enter a comma-separated list of points swept by the algorithm; refer to *Point Generation Arguments* on page 3-324.

- *From*, *To*, *Steps*, and *Step Size* Fields

  If the *Algorithm* is *Linear*, enter the values that define the steps in the sweep: *From* and *To* fields are required, and either *Steps* or *Step Size* is required: either of these two can be calculated from the other. For more information, refer to *Range From* on page 3-320, *Range To* on page 3-320, *Range Steps* on page 3-321, and *Range Step Size* on page 3-321.

- *Apply To* Field

  1. *Pins/Groups*

     Enter a comma-separated list of the pins whose parameter will be varied. Leave this field blank to vary all pins. This field is enabled if *Type* is *Edge* or *Level*. For more information, refer to *Apply To Pins/Groups* on page 3-324.

  2. *Time Sets*

     Enter a comma-separated list of the time sets whose parameter will be varied. Leaving this field blank means all time sets are varied. This field is enabled if *Type* is *Spec*, *Edge* or *Period*; refer to *Apply To Time Sets* on page 3-324.

- *Interpose* Field

  Two fields are provided. In the first field, enter the names of the user-created functions to be called at specific points in the characterization. These fields are enabled only on the *X-axis* tab; refer to *Interpose Functions* on page 3-326.

  For each function, second input box accepts comma-separated list of arguments:

  1. **Start**—Function called before beginning the characterization.
  2. **PrePoint**—Function called before performing the test at each point.
  3. **PostPoint**—Function called after performing the test at each point.
  4. **End**—Function called at the end of the characterization.

- *Test Method* Field

  In the text field or from the drop-down list, select, the characterization method:

  1. *Retest*—Re-execute the test body.
  2. *Reburst*—Rerun the current pattern.

+ Also, refer to *Point Generation Test Method* on page 3-323.

### *Adjust and Measure Tabs*

+ Most of the editor fields on these tabs are similiar to columns of the *Characterization* sheet; thus, for more information about an item, refer to the reference for the appropriate *Characterization* column.

- Radio Buttons in *Mode* Field

  By checking the *Measur7e* or *Adjust* box in the *Mode* field, a *Measure* or *Adjust* tab appears. The existence of a tab does not mean that the axis rows have been defined. Along with *Adjust*, you can also check the *Schmoo*, *Measure* or *Adjust From* boxes, or all four.

+ For more information, refer to *Characterization Modes* on page 3-310, *Mode* on page 3-318, and *Mode Field* on page 3-341.

- *Type* Field

  In the text field or from the drop-down list, select the type of parameter to vary: *Level*, *Edge*, *Period*, or *Spec*. For more information, refer to *Parameter to Vary Type* on page 3-319.

- *Name* Field

  In the text field or from the drop-down list, select the name of the parameter (*Level*, *Edge*, or *Spec*) to vary. Valid values are listed in the drop-down list. For more information, refer to *Parameter to Vary Name* on page 3-319.

- *Row Name* Field

  Enter a name by which this row can be referenced when accessing results from **VB-Test**. Enabled only with *Adjust* and *Measure*. This field not is currently supported. For more information, refer to *Row Name* on page 3-318.

- *Algorithm* Field

  In the text field or from the drop-down list, select, the algorithm to generate the pass/fails points for the *Adjust* or *Measure*: *Auto, Pos, Neg, Pos Binary, Neg Binary, Pos Linear*, and *Neg Linear. Auto*, *Pos*, *Neg*, and *Pos Binary* and *Neg Binary* are versions of binary search, while *Pos Linear* and *Neg Linear* are linear step algorithms; refer to *Point Generation Algorithm* on page 3-322.

- *Backoff* Field

  This field is enabled only for the *Adjust* mode. Enter the *backoff* or safety margin value to be applied to the adjusted value; refer to *Point Generation Arguments* on page 3-324 and *Running an Adjust* on page 3-329.

- *From*, *To*, *Steps*, and *Step Size* Fields

  Enter the values that define the steps in the sweep: *From* and *To* fields are required, and either *Steps* or *Step Size* is required: either of these two can be calculated from the other; refer to *Range From* on page 3-320, *Range To* on page 3-320, *Range Steps* on page 3-321, and *Range Step Size* on page 3-321.

- *Apply To* Field

  1. *Pins/Groups*

     Enter a comma-separated list of the pins whose parameter will be varied. Leaving this field blank means all pins are varied. This field is enabled if *Type* is *Edge* or *Level*; refer to *Apply To Pins/Groups* on page 3-324.

  2. *Time Sets*

     Enter a comma-separated list of the time sets whose parameter will be varied. Leaving this field blank means all time sets are varied. This field is enabled if *Type* is *Spec*, *Edge* or *Period*; refer to *Apply To Time Sets* on page 3-324.

     Enter a comma-separated list of the time sets whose parameter will be varied. Leaving this field blank means all time sets are varied. This field is enabled if *Type* is *Spec*, *Edge* or *Period*; refer to *Apply To Time Sets* on page 3-324.

- *Name* in Adjust Field

  Enter the name of the levels or timing spec that receives the results of the *Adjust*. For more information, refer to *Running an Adjust* on page 3-329 and *Adjust Spec Name* on page 3-325.

- *Interpose* Field

  Two fields are provided. In the first field, enter the names of the user-created functions to be called at specific points in the characterization. These fields are enabled only on the *X-axis* tab; refer to *Interpose Functions* on page 3-326.

  For each function, second input box accepts a comma-separated list of arguments:

  1. **Start**—Function called before beginning the characterization.
  2. **PrePoint**—Function called before performing the test at each point.
  3. **PostPoint**—Function called after performing the test at each point.
  4. **End**—Function called at the end of the characterization.

### *Adjust From Tab*

+ Most of the editor fields on this tab is similiar to columns of the *Characterization* sheet; thus, for more information about an item, refer to the reference for the appropriate *Characterization* column.

- Radio Buttons in *Mode* Field

  By checking the *Adjust From* box in the *Mode* field, a *Adjust From* tab appears. The existence of a tab does not mean that the axis rows have been defined. Along with *Adjust From*, you can also check the *Schmoo*, *Measure* or *Adjust* boxes, or all four.

+ For more information, refer to *Characterization Modes* on page 3-310, *Mode* on page 3-318, and *Mode Field* on page 3-341.

- *Spec Name in Adjust Field*

  Enter the name of the levels or timing spec whose value will be taken from the specified setup. For more information, refer to *Running an Adjust* on page 3-329 and *Adjust Spec Name* on page 3-325.

- *From Setup in Adjust Field*

  Enter name of the setup to retrieve the adjusted spec value from; refer to *Running an Adjust* on page 3-329 and *Adjust From Setup* on page 3-325.