Creating Your Framework

HP E3180B Semiconductor Process Evaluation Core Software
HP SPECS revision B.02.40

HP Part No. E3180-90620

Printed in Japan May 1999

Edition 1

Legal Notice

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

- **Product Warranty**

  This Hewlett-Packard system product is warranted against defects in material and workmanship for a period of one year from date of installation. During the warranty period, HP will, at its option, either repair or replace products which prove to be defective.

  Warranty service of this product will be performed at Buyer's facility at no charge within HP service travel areas. Outside HP service travel areas, warranty service will be performed at Buyer's facility only upon HP's prior agreement and Buyer shall pay HP's round trip travel expenses. In all other cases, products must be returned to a service facility designated by HP.

  For products returned to HP for warranty service, Buyer shall prepay shipping charges to HP and HP shall pay shipping charges to return the product to Buyer. However, Buyer shell pay all shipping charges, duties, and taxes for products returned to HP from another country.

  HP warrants that its software and firmware designated by HP for use with an instrument will execute its programming instructions when properly installed on that instrument. HP does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

- **Limitation of Warranty**

  The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modifications or misuse, operation outside of the environment specifications for the products, or improper site preparation or maintenance.

  No other warranty is expressed or implied. Hewlett-Packard specifically disclaims the implied warranties or merchantability and fitness for a particular purpose.

Printing History

Edition 1: May 1999

In This Manual

This manual describes common tasks and reference information for using the HP Semiconductor Process Evaluation Core System (SPECS), and consists of the following chapters:

- Chapter 1, "Structure of Framework"

  Describes the structure of framework.

- Chapter 2, "Using Outliner"

  Describes how to use Outliner. You use the Outliner to edit the framework specifications of a framework file.

- Chapter 3, "Creating Your Framework"

  Describes how to create your framework from HPSTD framework.

- Chapter 4, "Operation Panel Reference"

  Provides reference information about the widgets used to create operation panels.

- Chapter 5, "TPL Command Reference"

  Provides reference information about the TPL commands that can be used in a framework specification.

- Chapter 6, "Algorithm Library Reference"

  Provides reference information about the algorithms for framework.

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

1 Structure of Framework

Framework provides the graphical user interface to control test definition, and test execution. Framework allows you to modify the user interface and the function as you desire.

This chapter explains the structure of framework and detail of each specification, and consists of the following sections:

- “Overview”

  Explains summary of framework.

- “System Specification”

  Provides details of System Specification.

- “Tag Specification”

  Provides details of Tag Specification.

- “Panel Specification”

  Provides details of Panel Specification.

- “Action Specification”

  Provides details of Action Specification.

Overview

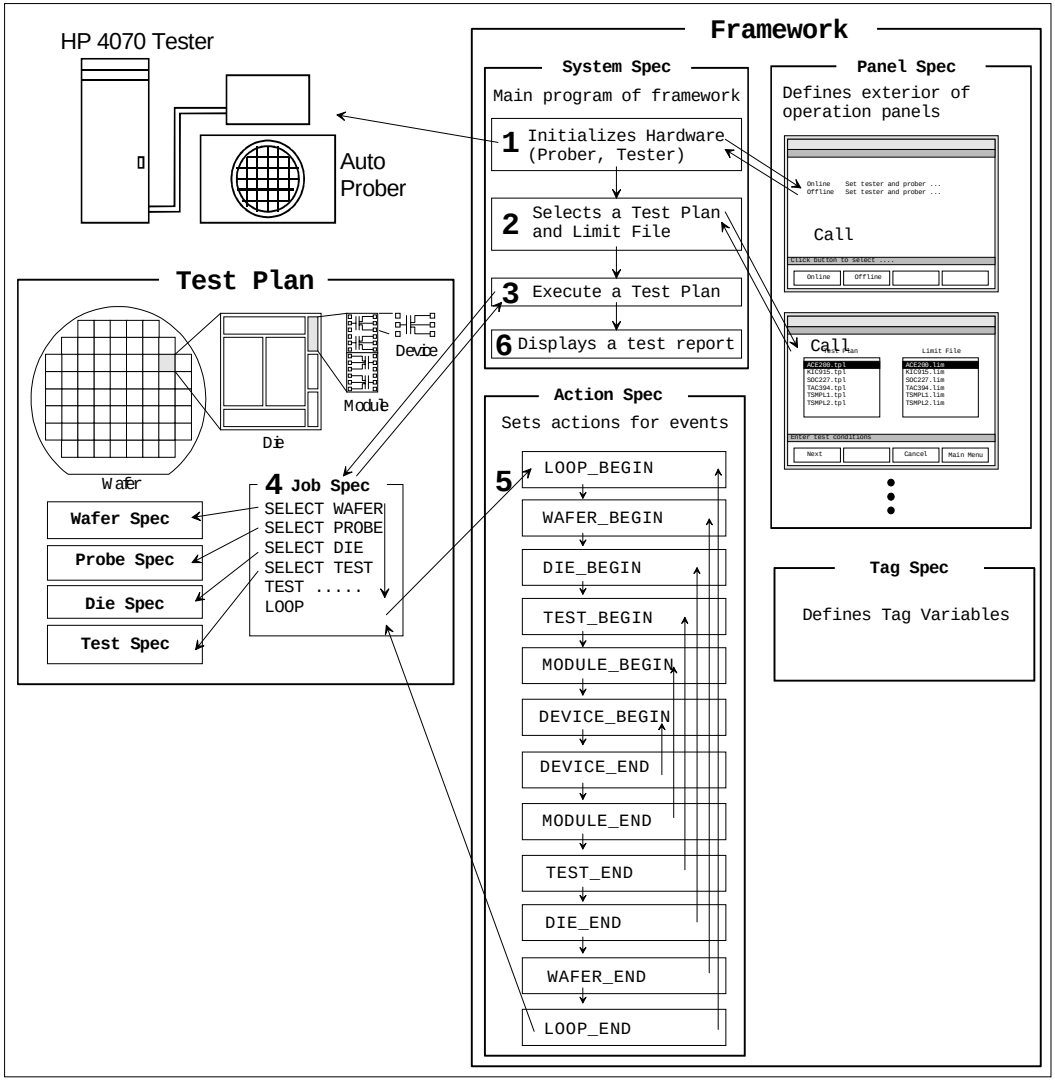
A framework consists of the following four specifications.

System	Defines overall execution flow of the framework.
Action	Defines actions to be executed for events and ACTION command.
Tag	Defines tag variables, which are used in other specifications.
Panel	Defines appearance of the operation panels that are displayed on computer screen during the framework execution.

Each specification is executed as shown below, and Figure 1-1.

1. System specification initializes the prober and tester in online or offline mode.
2. System specification calls an operation panel which selects a test plan and limit file.
3. System specification calls the test plan using RUN command.
4. RUN command moves control to the Job Specification in test plan. Job Specification selects a Wafer Specification and Probe Specification using SELECT command and assign test to dice using TEST command. At the end of Job Specification, LOOP command starts test plan execution.
5. While test plan execution, action specifications are executed when event occur.
6. When all wafer tests are completed, control is moved to System Specification. Then, test report is displayed.
7. Framework ends.

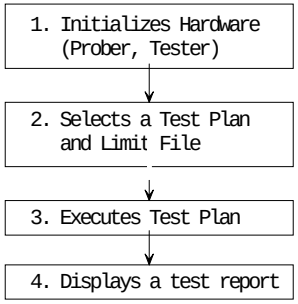
Figure 1-1 Framework execution flow



System Specification

System specification (Spec) determines overall execution flow of a framework, and is executed at beginning of framework. System Spec must be named “SYSTEM”.

Typical framework has the following execution flow:



1. Initializes hardware

   Initializes the prober and tester for online or offline mode. Also, System Spec opens an operation panel which is used to select this execution mode.

2. Selects a test plan and limit file

   Selects a test plan and limit file. Also, System Spec opens an operation panel which is used to list and select a test plan and limit file.

3. Executes a test plan.

   Executes a test plan. Also, System Spec opens an operation panel which is used to monitor test execution status.

4. Displays a test report

   Displays a test report after the test. You can select the type of test report using an operation panel.

You cannot modify a System Spec of the HPSTD framework because of the modification rule. See “To Modify Plug Framework” on page 3-10.

Parameters defined in System Specification

To define a System Spec, define the following parameters:

Branch Label (optional) Label target for GOTO command.

Command Test plan command or algorithm.

Parameter Input parameter for TPL command, or input variables for algorithm. You can use Assist menu when algorithm is entered in “Command” column.

Comment (optional) Comment for line. Maximum 64 characters.

Table 1-1 System Spec example

Branch Label	Command	Parameter	Comment
Hardware_init	ACTION	“HP_INIT_HARDWARE”	Call an Action Spec

Tag Specification

Tag specification (Spec) defines tag variables and its initial value. To enter the value to tag variables, use LET command.

HP SPECS already defined some tag variables as shown in “Variables” on page 5-7.

Tag Spec must be named SYSTEM, LOT,WAFER or DIE. Each tag spec has a different scope as shown in the following table:

Table 1-2 **Tag Spec Name and scope**

Spec Name	Scope
SYSTEM	Per framework execution. At the end of framework execution, the value of variables are stored in data file.
LOT	Per lot test execution. At the end of lot test, variables are stored in the data file and set to the initial value defined in Tag Spec.
WAFER	Per wafer test execution. At the end of wafer test, variables are stored in the data file and set to the initial value defined in Tag Spec.
DIE	Per die test execution. At the end of die test, variables are stored in the data file and set to the initial value defined in Tag Spec.

In framework, System, Action and Panel Spec use Tag variables as global variables to set the parameters for test plan commands, algorithms, and operation panel widgets.

As default, Tag variables are saved to the data file.

Parameters defined in Tag Specification

To define a Tag Spec, define the following parameters:

Tag Name Tag variable name. Tag variable name and value are stored in data file automatically. If you put this name between { and }, the variable is not saved to data file. In maximum, you can define 1024 tag variables in a framework. Must be started with alphabet character and can have maximum 32 alphabetic, numeric and _(underline) characters.

Type Data type of Tag variable. You can use Assist menu to enter the type. Also see “Data Type” on page 5-3.

Unit (optional) Unit of Tag variable.

Value (optional) Initial value of Tag variable.

Range (optional) Value available for Tag variable. To specify the range available for the tag variable, use a tilde(~) between the lowest value and the highest value. To specify the separated values, use a comma between the values. For example,
0.1~1.5
0,1,2
”P”,”F”

Comment (optional) Comment for the line. Maximum 64 characters.

Table 1-3 Tag Spec example

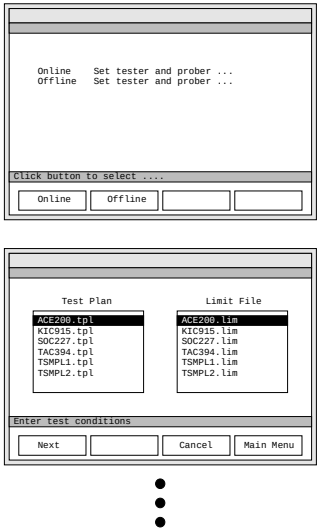
Tag Name	Type	Unit	Value	Range	Comment
{PRB_HPIB_ADRS}	INTEGER		2501		

Panel Specification

Panel specification (Spec) defines an operation panel.

In Panel Spec, you can use widget which sets a part for displaying text box, list, input field and softkey. For details of widgets, see “Operation Panel Reference” on page 4-1.

You can define a button or softkey which calls an Action Spec using ActionBtn or ActionKey widget.



Parameters defined in Panel Specification

To define a Panel Spec, define the following parameters:

Widget Name Name assigned to widget you set on operation panel. Max 32 characters. For the ActionBtn and ActionKey widget types, the name must be the event name of Action Spec assigned to the widget.

Widget Type Type of Widget. You can use Assist menu to enter widget type. See Chapter 4, “Operation Panel Reference,” on page 4-1.

Parameter Parameter for the Widget. You can use Assist menu to enter parameters.

Variable Tag Variable name for widget specified in “Widget Type” column.

Range Value available for input of widget specified in “Widget Type” column.

Message (optional) Message for widget defined in “Widget Type” column. When widget is selected, the message is displayed. Max 64 characters.

Table 1-4 Panel Spec example

Widget Name	Widget Type	Parameter	Variable	Range	Message
USER	SingleInput	Label=”User Na	USR_NAME		“Enter your na

Action Specification

Action specification (Spec) defines what action to perform when the specified event occurs.

HP SPECS executes an Action Spec in the following case:

- **At the system events**

  System events are defined by HP SPECS and each event happens during test execution. For example, an Action Spec, “WAFER_BEGIN” is executed automatically at the beginning of each wafer. For details of system events, refer to “HPSTD Framework Execution Flow” on page 3-4.

Table 1-5 Action Spec names to define actions for system events

LOOP_BEGIN	LOOP_END	WAFER_BEGIN	WAFER_END
DIE_BEGIN	DIE_END	TEST_BEGIN	TEST_END
MODULE_BEGIN	MODULE_END	DEVICE_BEGIN	DEVICE_END

- **Called by “ACTION” command**

  “ACTION” command calls another Action Spec like a sub-program. “Action” command can be used in Action or System Spec. For details of “ACTION” command, see the reference on page 5-17.

- **Button and Softkey selection in Operation panel**

  You can define an action for selection of button or softkey in operation panel. When a button or softkey is selected, Action Spec relative to the button or softkey is executed. Action Spec name to be assigned to the button or softkey as follows:

  PanelName_WidgetName

  ***PanelName*:** panel specification name

  ***WidgetName*:** widget name defined in Panel Spec widget type (must be ActionKey or ActionBtn)

  You can use underscore("_") between *PanelName* and *WidgetName*. This rule may make a confusion. For example, "A_Panel_Start" is acceptable. This indicates both “Panel_Start” of “A” panel and “Start” of “A_Panel” panel. You will find “A” or “A_Panel” Panel Spec in the framework.

You cannot modify a Action Spec named “HP_xxxxx” of HPSTD framework because of the modification rule described in “To Modify Plug Framework” on page 3-10.

Parameters defined in Action Specification

To define an Action Spec, define the following parameters:

Branch Label (optional) Label for target of GOTO command. Max 32 characters.

Command Test plan command or algorithm. For algorithm input , can use Assist menu.

You cannot define algorithms that define device terminals or device variables.

Data file does not save the output of algorithm.

Parameter Input parameter for TPL command, or input variables for algorithm. You can use Assist menu when algorithm is entered in “Command” column.

Comment (optional) Comment for line. Maximum 64 characters.

Table 1-6 Action Spec example

Branch Label	Command	Parameter	Comment
Init_tag	LET	PRB_ABORT_WAFER=0	

2 Using Outliner

Outliner is a tool to edit and executes a framework and test plan. The list below shows main features of Outliner.

- Lists specifications for framework or test plan
- Adds, deletes, copies and renames a specification
- Opens SPECS EDITOR to edit each specification
- Opens RUN PANEL to execute an editing framework
- Loads a measurement algorithm library
- Opens ALGORITHM PANEL to debug a measurement algorithm
- Shows status of parameters configured in the “sysconf” file

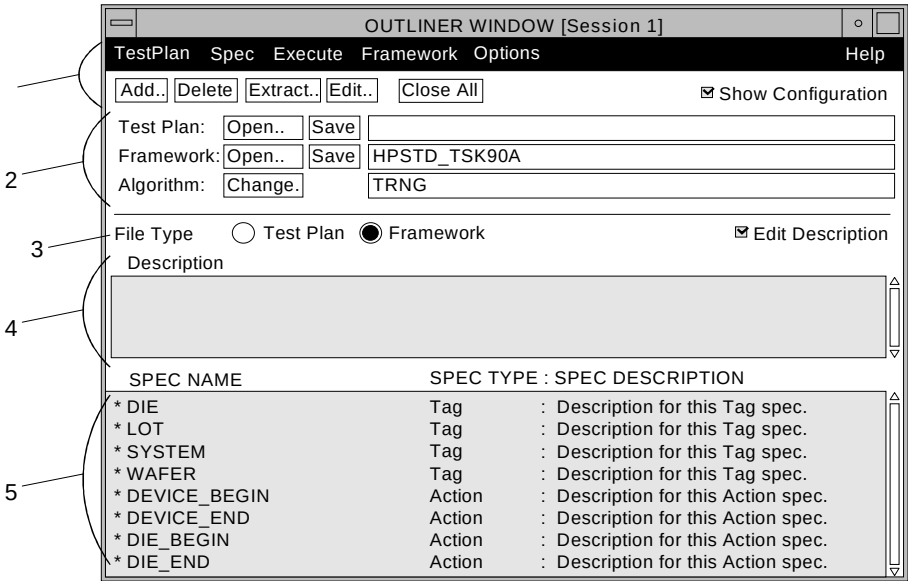
This chapter describes how to use Outliner and consists of the following sections:

- “Overview”
- “To Start Outliner”
- “To Define a New Spec”
- “To Export a Spec”
- “To Import a Spec”
- “To Extract a Spec from Other File”
- “To Copy a Spec”
- “To Rename a Spec”
- “To Find a String from Framework and Test Plan”
- “To Open a Framework”
- “To Execute a Framework”

Overview

Outliner is a tool to edit and executes a framework and test plan.

Figure 2-1 Outliner window



1. Menus and buttons

For using tools to edit or execute a framework or test plan.

2. Configuration parameters

Shows loaded framework, test plan, and measurement algorithm library name. You can hide this area using “Show Configuration” button.

3. “File Type” option menu

Selects a file type to list in “Spec list area”.

4. Description edit area

Lists description for editing framework or test plan.

5. Spec list area

Lists specifications of editing framework or test plan. Specifications for framework are marked by “*”.

TestPlan menu

New Opens a new (empty) test plan.

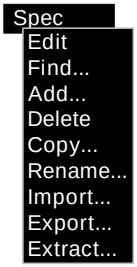
Open... Opens the “Open Test Plan” dialog, which opens a test plan file. Same as clicking the "Open..." button.

Save Saves the editing test plan to the same file. Same as clicking the "Save" button. This saves the test plan specifications, not framework specifications (marked by *).

Save As... Opens the “Save Test Plan” dialog which saves the editing test plan to another file.

Print... Opens the “Print” dialog which prints the contents of all test plan specifications currently listed in the Outliner list area.

Exit Closes Outliner window.



Spec menu

Edit... Opens the Spec Editor and opens the selected specification in it. Same as clicking the "Edit Spec" button.

Find... Opens the “Find Spec dialog” which searches a string from the all specifications in the editing framework or test plan. For details, refer to “To Find a String from Framework and Test Plan” on page 2-20.

Add... Opens the “New Spec dialog”, which can add a new specification to the opened test plan. For details, refer to “To Define a New Spec” on page 2-14.

Delete Deletes the selected specification from the opened test plan.

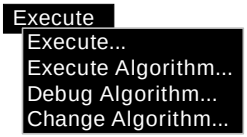
Copy... Opens the “Copy Spec” dialog which copies a selected spec in Outliner to the other specs. For details, refer to “To Copy a Spec” on page 2-18.

Rename... Opens the “Copy Spec” dialog. This item deletes the selected spec in Outliner after copying a spec. For details, refer to “To Rename a Spec” on page 2-19.

Import... Opens the “Import Spec” dialog, which imports a spec in to the editing test plan. For details, refer to “To Import a Spec” on page 2-16.

Export... Opens the “Export Spec” dialog, which exports a selected spec in Outliner to a separate file. Fore details, refer to “To Export a Spec” on page 2-15.

Extract... Opens the "Extract Spec” dialog which extracts a spec defined in the other framework or test plan and imports it to the editing framework or test plan. For details, refer to “To Extract a Spec from Other File” on page 2-17.



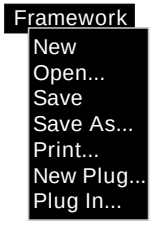
Execute menu

Execute... Opens RUN PANEL to execute the editing framework. Refer to “To Execute a Framework” on page 2-21.

Execute Algorithm... Opens Algorithm Panel or HP BASIC/UX to execute an algorithm without executing a framework.

Debug Algorithm... Starts xdb program for debugging the C algorithm. (This item is effective when the Algorithm type is C.)

Change Algorithm... Opens the “Change Algorithm” dialog, which loads another measurement algorithm library.



Framework menu

New Opens a new (empty) framework.

Open... Opens the “Open Framework” dialog which opens a framework file. Refer to “To Open a Framework” on page 2-9.

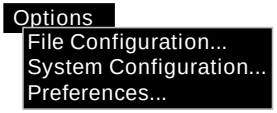
Save Saves the editing framework to the same file.

Save As... Opens the “Save Framework” dialog which saves the editing framework to another file. Refer to “To Save a Framework” on page 2-22.

Print... Opens the “Print” dialog which prints the all specifications of editing framework.

New Plug... Opens the “New Framework (Plug)” dialog which creates a new plug framework. Refer to “If You Create a New Plug Framework” on page 3-9.

Plug In... Opens the “Plugin Framework” dialog which generates your framework by combining the plug framework to HPSTD framework. Refer to “To Generate New Framework” on page 3-16.



Options menu

File Configuration... Opens the “File Configuration” dialog which shows test plan, framework and algorithm library names loaded to the Outliner. Refer to “To View File Configurations” on page 2-10

System Configuration... Opens the “System Configuration” dialog which shows the parameters configured in the “sysconf” file. Refer to “To View Parameters in the “sysconf” File” on page 2-11.

Preferences.... Opens the “Preferences” dialog to set preferences of the Outliner. Refer to “To Configure Outliner Window” on page 2-12.

Help menu

Contents... Opens the contents for all help topics.

Outliner... Opens the help of Outliner window.

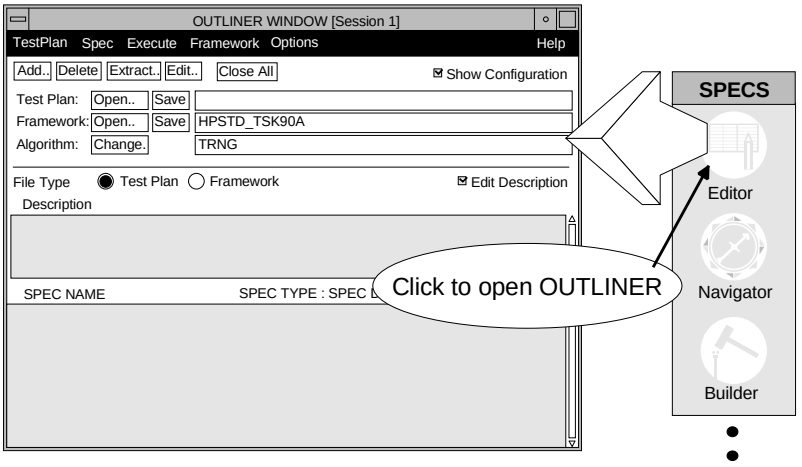
TPL Command... Opens the help of TPL Command.

To Start Outliner

You can start Outliner from the following steps:

- **SPECS subpanel of HP CDE Front Panel**

  Click on the “Editor” icon in the “SPECS” sub-panel of HP CDE Front Panel.



- **HP CDE File Manager**

  Double-click a framework file icon. *Or* drag a framework file icon and drop it on “Editor” button of Front Panel.

- **Terminal emulator window**

  Type the following after a command-line prompt:

  $ tpledit [*TestPlan*] &

  where *TestPlan* is a name to be loaded. Also, the following options can be set:

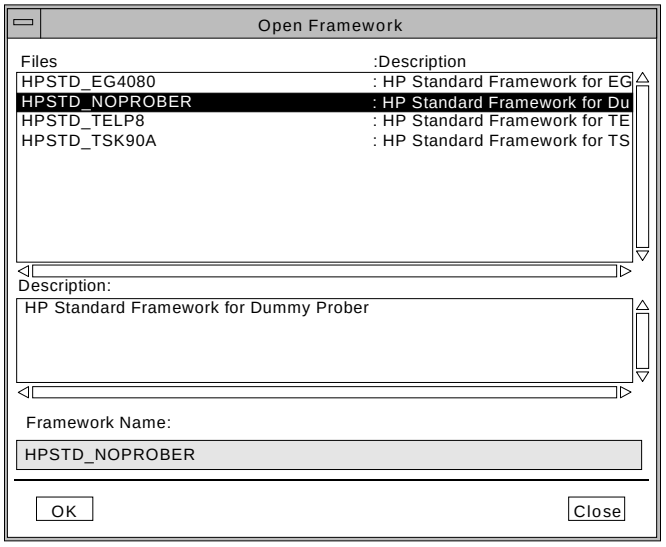
Table 2-1 Options for tpledit command

Options	Description
-e	open and execute the specified test plan.
-f *framework*	open test plan with specified framework.
-a *library*	select the specified algorithm library for test plan.
-F	open test plan without framework

To Open a Framework

Outliner loads a framework set in sysconf file. To open other framework, do as follows.

1. Select the “Framework:Open...” menu in Outliner to open the “Open Framework” dialog.

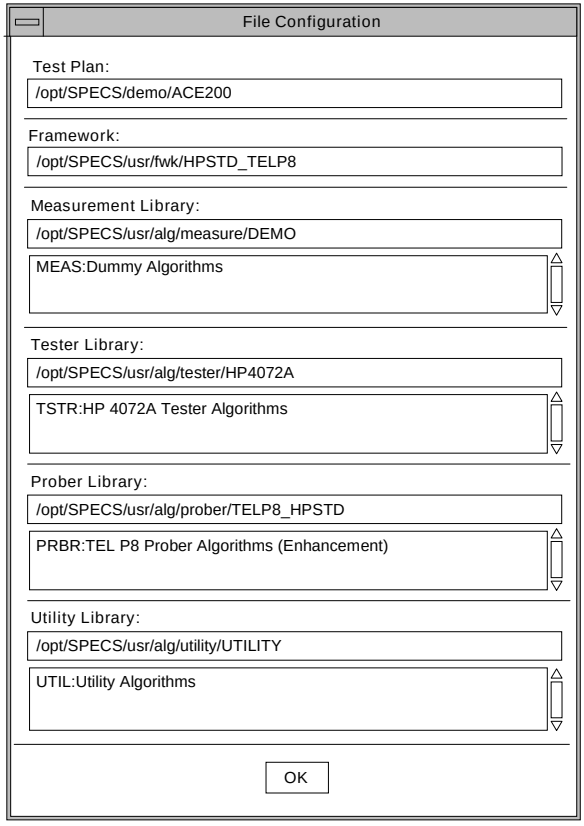


2. Select a framework file in the “Files” list in the dialog.
3. Click on the “OK” button in the dialog.

To View File Configurations

You can view test plan, framework and algorithm library names loaded to the Outliner.

1. Select the “Options:File Configuration...” menu in Outliner to open the “File Configuration” dialog.



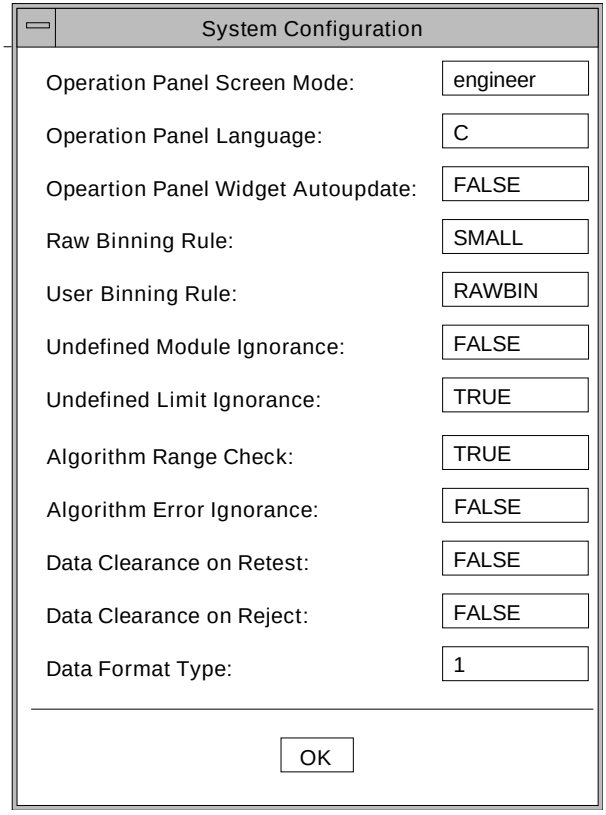
2. Click on the “OK” button to close the dialog.

To View Parameters in the “sysconf” File

You can view parameters in the “sysconf” file as follows.

1. Select the “Options:System Configuration...” menu in Outliner to open the “System Configuration” dialog.

   For details of each parameter, refer to “HP SPECS Installation Guide”.

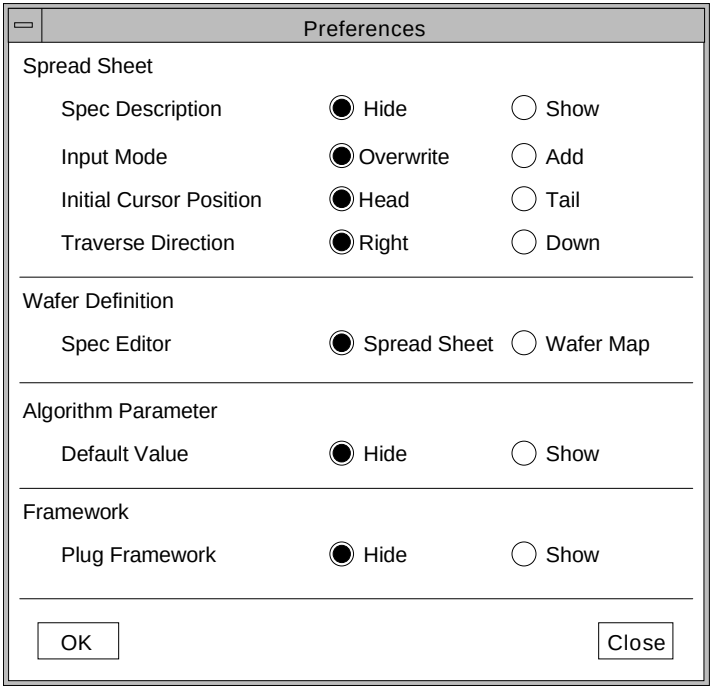


2. Click on the “OK” button to close the dialog.

To Configure Outliner Window

You can Configure Outliner Window as follows.

1. Select the “Options:Preferences...” menu in Outliner to open the “Preferences” dialog.



2. Set preferences in the dialog.

Table 2-2 Configuration Parameters in “Preferences” dialog.

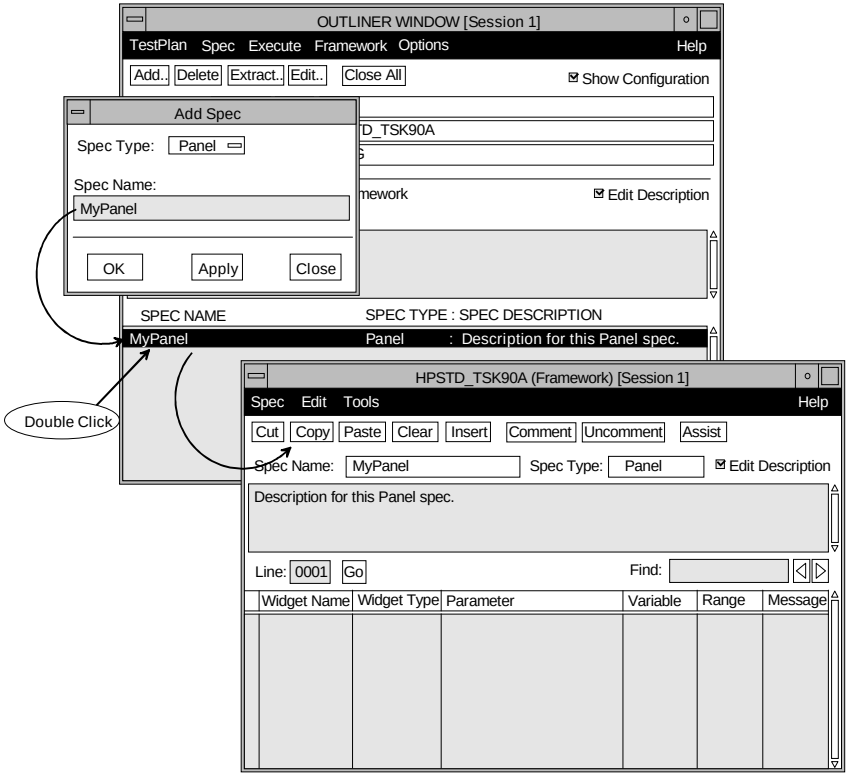
Spread Sheet	
Spec Description	Sets whether hides or shows a “Description” area when Spec Editor opens.
Focus Column Input	Sets whether overwrites or appends inputs in Spec Editor.
Initial Cursor Position	Sets cursor position in column of Spec Editor when a column is selected.
Focus Traverse	Sets direction of cursor movement when “Enter” or “Tab” is entered in Spec Editor.
Wafer Definition	
Spec Editor	Sets whether Wafer Map Editor or Spec Editor is opened to edit a Wafer Spec.
Algorithm Command and Parameter	
Default Value	Sets whether hides or shows the default value in Spec Editor.
Framework	
Plug Framework	Sets whether enables or disables the framework plug-in function in Outliner.

3. Click on the “OK” button in the dialog.

   These values of configuration parameters except “Plug Framework” are stored in “~/.specspref” file. So this configurations are effective at the next use of Outliner.

To Define a New Spec

1. Select the “Spec:New Spec...” menu in Outliner to open the dialog.
2. Select a spec type using the “Spec Type” option menu in the dialog.
3. Type in the new spec name in the “Spec Name” field.
4. Click on the “OK” button. Outliner list the added spec.
5. Double-click the name of new spec on list in Outliner. SPEC EDITOR opens for the new spec.



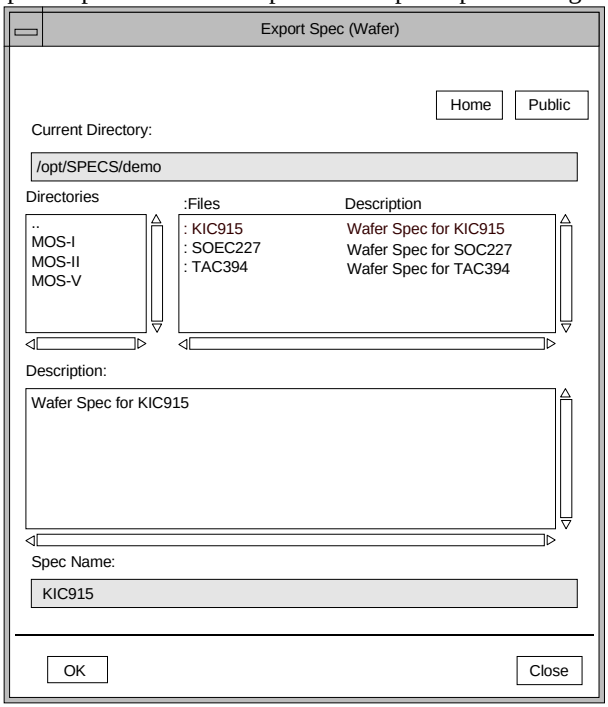
To Export a Spec

A test plan or framework file consists of multiple specifications, which are listed in the Outliner list area. You can export (copy) a specification to a separate file.

If a specification is exported to a separate file, you can import the specification to another test plan or framework. For importing, refer to “To Import a Spec” on page 2-16.

The following describes how to export a specification of a test plan or a framework to a separate file:

1. Select specification to export by clicking on the desired specification in the list area of the Outliner window.
2. Select the “Spec:Export...” menu to open the “Export Spec” dialog.



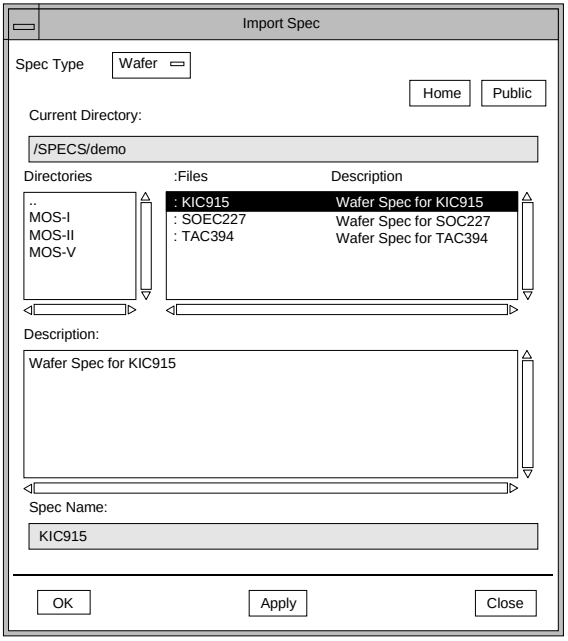
3. Change to desired directory.
4. Enter file name to which you want to save the specification.
5. Choose the OK button. The specification is saved to the file.

To Import a Spec

A spec of a test plan or a framework can be exported to a separate file, as described in “To Export a Spec” on page 2-15.

This section describes how to import a specification (that was exported to a separate file) to the opened test plan or framework.

1. Select the “Spec:Import...” menu to open the “Import Spec” dialog.



2. Select the specification type by using the Type option menu. Only the selected type of specifications will be displayed in Files list area.
3. Change to the desired directory by double-clicking in Directories list area.
4. Click to select the desired specification file in Files list area.
5. Choose the OK button. The specification is imported to opened test plan or framework.

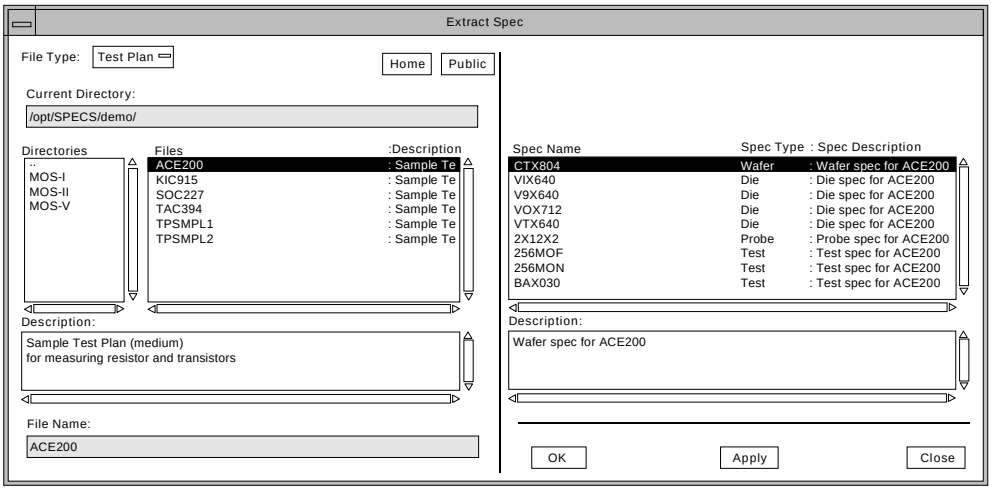
Be aware that a framework specification is imported as a test plan specification, that is, not marked by asterisk (*).

To Extract a Spec from Other File

A spec in the other test plan can be extracted to an opened test plan.

This section describes how to extract a specification (that was in other test plan file) to an opened test plan.

1. Select the “Spec:Extract...” menu to open the “Extract Spec” dialog.

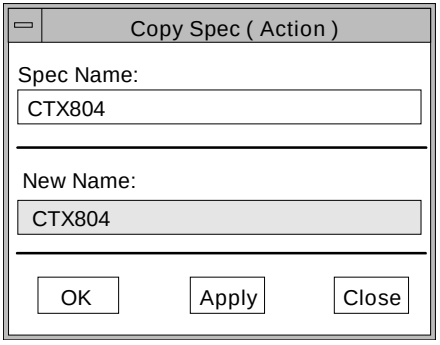


2. Select the file type, “Framework” or “Test Plan” for a target file to extract a spec.
3. Enter the directory name in the “Current Directory” field, or double-click the directory name listed in the “Directories” list to select the directory.
4. Enter the file name to extract a spec in the “File Name” field, or click the file name listed in the “Files” list to select the target file to extract a spec.
5. Click on the "List Test Plan Contents" button for listing the specifications in the selected file.
6. Click on the specification name for adding to the editing file.
7. Click on the “OK” button. The specification is added to file.

To Copy a Spec

Do as follows to copy a spec to another spec.

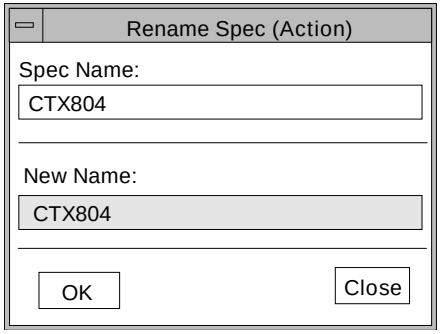
1. Select a spec to copy in Outliner.
2. Select the “Spec:Copy...” menu to pen the “Copy Spec” dialog.
3. Enter a spec name to be copied in the “New Name” field. Then, click on the “OK” button.



To Rename a Spec

This function deletes the selected spec in Outliner after copying a spec. Do as follows to rename a spec to another spec.

1. Select a spec to rename in Outliner.
2. Select the “Spec:Rename...” menu to pen the “Copy Spec” dialog.
3. Enter a spec name to be renamed in the “New Name” field. Then, click on the “OK” button.

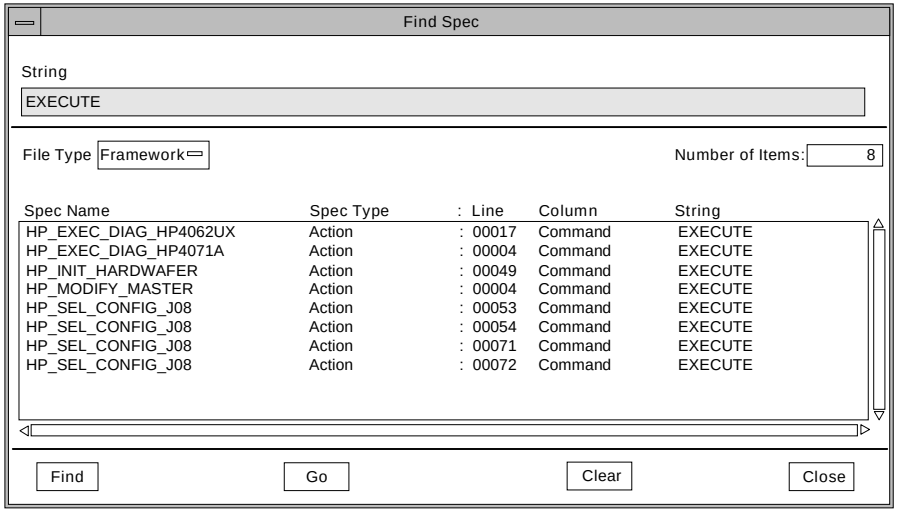


To Find a String from Framework and Test Plan

Select the “Spec:Find...” menu to open the “Find Spec” dialog. In the dialog, do as follows to find a string from the editing framework and test plan.

1. Enter a string to the “Find Name” field.
2. Select a target, “Framework” or “Test Plan” using the “File Type” option menu.
3. Click on the “Find” button. Then the all specifications which include a string entered in “Step 1” are listed in the dialog.

   Select a spec name from the list and click on the “Go” button to open the spec using SPEC EDITOR. Or, double-click a spec name.

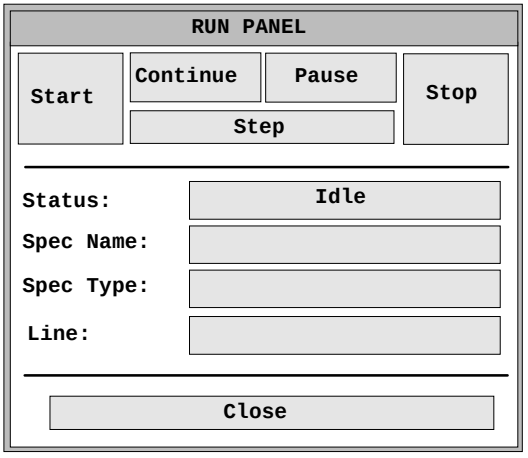


To Execute a Framework

Outliner loads a framework set in sysconf file. To open other framework, do as follows.

1. Select the “Execute:Execute...” menu in Outliner to open the Run Panel

Figure 2-2 Run Panel



The following functions are assigned to each button:

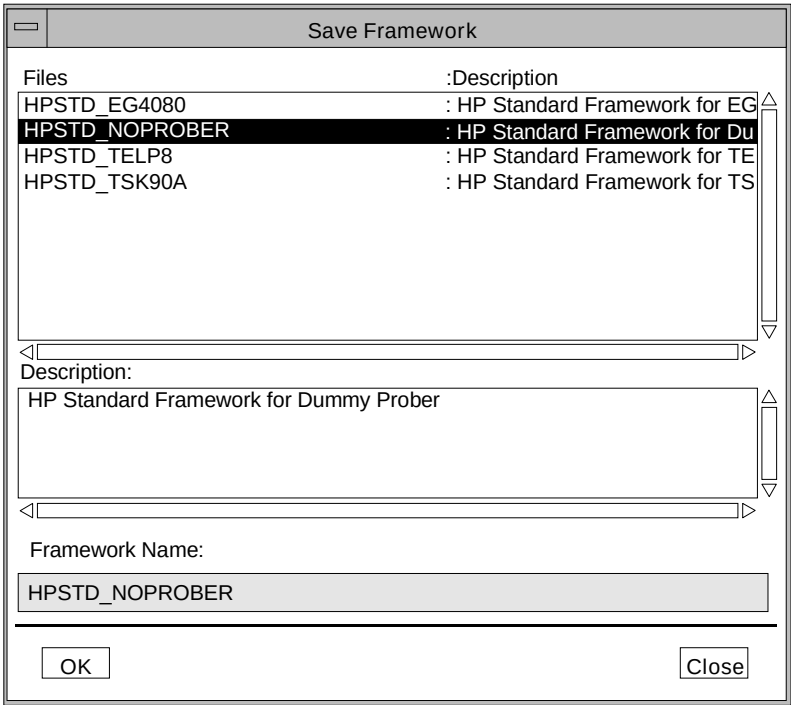
Start button	Starts the framework.
Continue button	Continues the paused framework.
Pause button	Pauses the running framework.
Stop button	Stops the running or paused framework.
Step button	Executes one line of paused framework.
Status: field	Displays the test status.
Spec Name field	Displays present specification name.
Spec Type field	Displays present specification type.
Line field	Displays present line in the specification.
Close button	Closes the Run Panel.

2. Click on the “Start” button to execute the framework.

To Save a Framework

Do as follows:

1. Select the “Framework:Save As...” menu in Outliner to open the “Save Framework” dialog.



2. Enter a file name in “Framework Name” field in the dialog.
3. Click on the “OK” button in the dialog.

3 Creating Your Framework

This chapter describes how to modify HPSTD framework, and consists of the following sections:

- **“Framework Plug-in Function”**

  Framework plug-in function is a tool to modify HPSTD framework for HP SPECS rev.B.02.40 or later. This section explains the framework plug-in function.

- **“HPSTD Framework Execution Flow”**

  Explains execution flow of HPSTD framework.

- **“To Create Your Framework”**

  Explains how to create your framework using framework plug-in function.

- “Modification Examples”

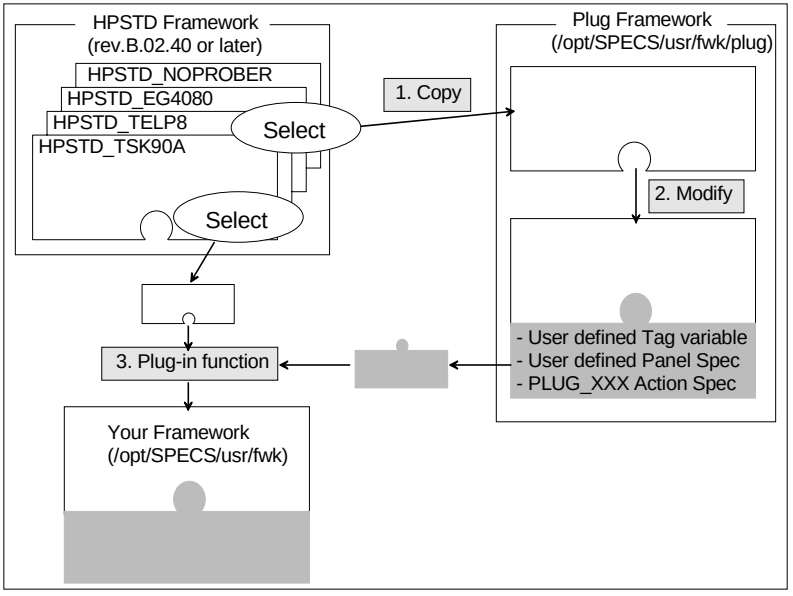
  Explains how to modify HPSTD framework and shows the following modification examples.

  - **“To Change Label in Operation Panel”**
  - **“To Add New Choices to Menu Panels”**
  - **“To Run Desired Test Report Commands”**
  - **“To Add a New Panel”**
  - **“To Replace an Existing Panel with New Panel”**

Framework Plug-in Function

Framework plug-in function is a tool to modify HPSTD framework for HP SPECS rev.B.02.40 or later. You can create your framework as shown below.

1. Select and copy an HPSTD framework as plug framework.
2. Add modifications you desire to the plug framework.
3. Select an HPSTD framework and a plug framework you use, and generate your framework using the plug-in function.



To create your framework, the framework plug-in function uses the following frameworks:

HPSTD Framework Standard frameworks are used as a template of your framework. You can use HPSTD framework without modification. See “Framework Operation Guide”.

Plug Framework Standard Framework to be modified, or Framework with modifications you did. The file must be stored in the “/opt/SPECS/usr/fwk/plug” directory. You need special instruction to open the plug framework. See “To Open Plug Framework” on page 3-8.

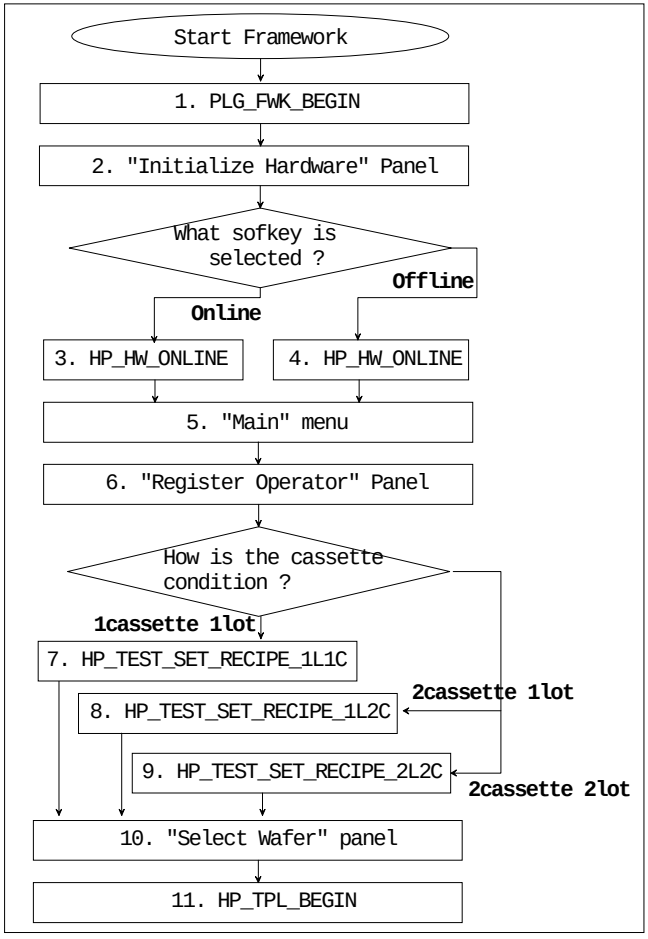
HPSTD Framework Execution Flow

This section shows an execution flow of HPSTD framework.

Before Executing a Test Plan

Figure 3-1 is a summarized execution flow of HPSTD framework before executing a test plan.

Figure 3-1 Flow Chart of HPSTD Framework (Before executing a test plan)



Each step in Figure 3-1 are as shown below:

1. Action Spec executed at the beginning of the framework.
2. Selects an initialization mode for tester and prober.
3. Action Spec to be executed when “ONLINE” is selected at “Initialize Hardware” panel.
4. Action Spec executed when “OFFLINE” is selected at “Initialize Hardware” panel.
5. Calls the “Register Operator” panel, “Utility” menu or “Set System Setting” menu.
6. Sets a name of operator.
7. Action Spec to set test conditions for 1 cassette 1 lot.
8. Action Spec to set test conditions for 2 cassette 1 lot.
9. Action Spec to set test conditions for 2 cassette 2 lot.
10. Selects or unselects wafers in cassette.
11. Action Spec executed at the beginning of the test plan execution.

While Executing a Test Plan

Figure 3-2 is a summarized execution flow of HPSTD framework while executing a test plan. Table 3-1 lists descriptions for each step of the flow chart shown in Figure 3-2.

Figure 3-2 Flow Chart of HPSTD Framework (While executing a test plan)

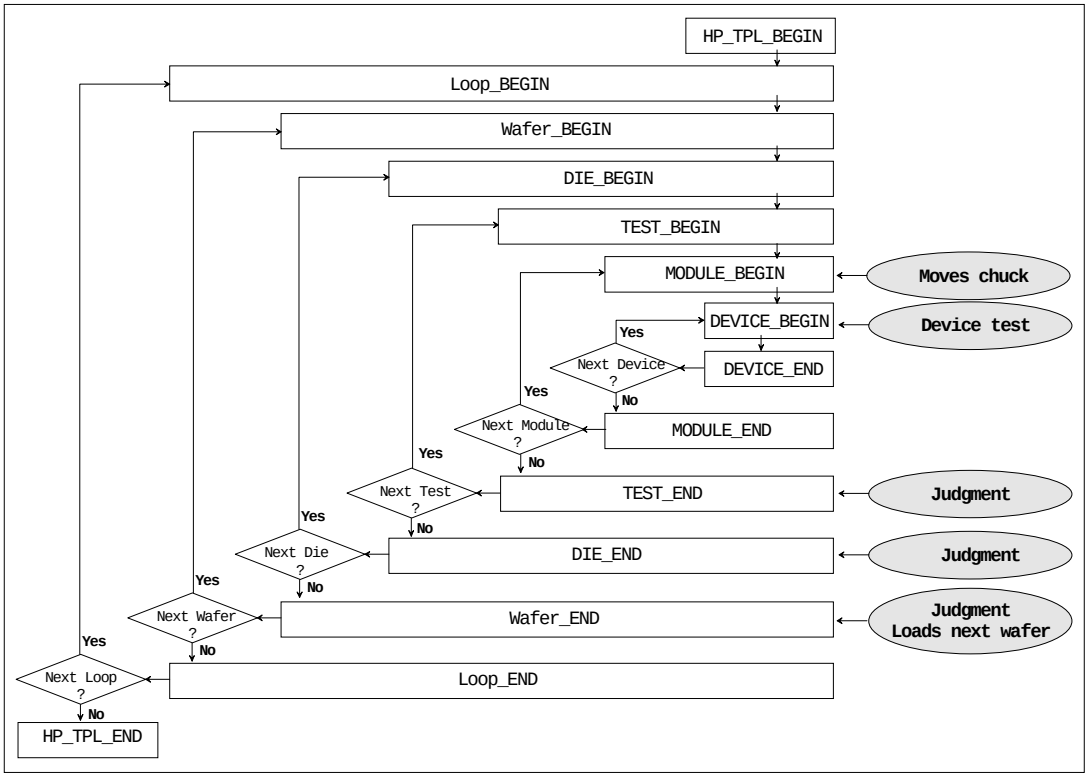


Table 3-1 lists descriptions for each step of the flow chart shown in Figure 3-2.

Table 3-1 Descriptions for each step

Step of flow chart	Description	Typical Action
HP_TPL_BEGIN	Action Spec executed at the beginning of test plan execution	Checks if the specified data file name is effective
LOOP_BEGIN	Action Spec executed at the beginning of each loop	Selects dice and modules. Initializes the prober.
WAFER_BEGIN	Action Spec executed at the beginning of each wafer	Updates the “Test Execution” panel.
DIE_BEGIN	Action Spec executed at the beginning of each die	Updates the result of die judgment
TEST_BEGIN	Action Spec executed at the beginning of test spec	Measures the current device
MODULE_BEGIN	Action Spec executed at the beginning of each module	Moves the chuck of prober
DEVICE_BEGIN	Action Spec executed at the beginning of each device	No action
DEVICE_END	Action Spec executed at the end of each device	No action
MODULE_END	Action Spec executed at the end of each module	Judges the current die and wafer
TEST_END	Action Spec executed at the end of test spec.	No action
DIE_END	Action Spec executed at the end of each die	Judges the current die and wafer
WAFER_END	Action Spec executed at the end of each wafer	Judges the current wafer. Loads next wafer.
LOOP_END	Action Spec executed at the end of each loop	Judges the current lot.
HP_TPL_END	Action Spec executed at the end of test plan execution	Stores the measurement data into the data file.

To Create Your Framework

You can modify HPSTD framework and create your framework using the framework plug-in function, as shown below:

1. Open a plug framework

   See “To Open Plug Framework” on page 3-8

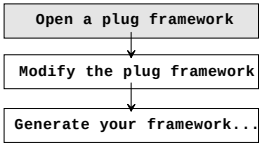
2. Modify the plug framework

   See “To Modify Plug Framework” on page 3-10.

3. Create your framework using plug-in function

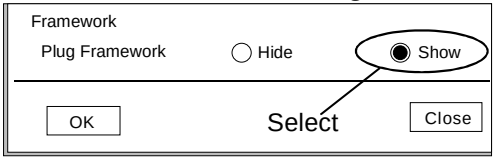
   See “To Generate New Framework” on page 3-16.

To Open Plug Framework

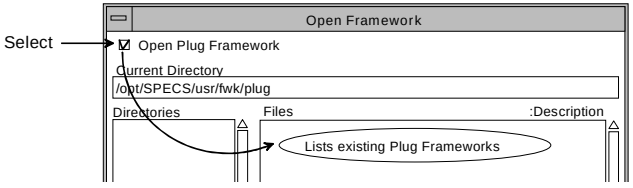


Open a Plug framework as shown below:

1. Start OUTLINER.
2. Select the “Options:Preferences..” menu to open the “Preferences” dialog.
3. Select the “Show” radio button for “Plug framework”, then click “OK”.



4. Select the “Framework:Open...” menu to open the “Open Framework” dialog.



5. In the dialog, select and open a plug framework as shown below.

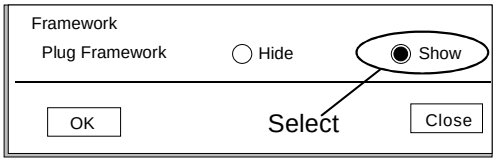
   a. Click on the “Open Framework As Plug” button in the dialog to list the existing plug frameworks stored in the “/opt/SPECS/usr/fwk/plug” directory.

   b. Select a file from the “Files” list. Then, click “OK”.

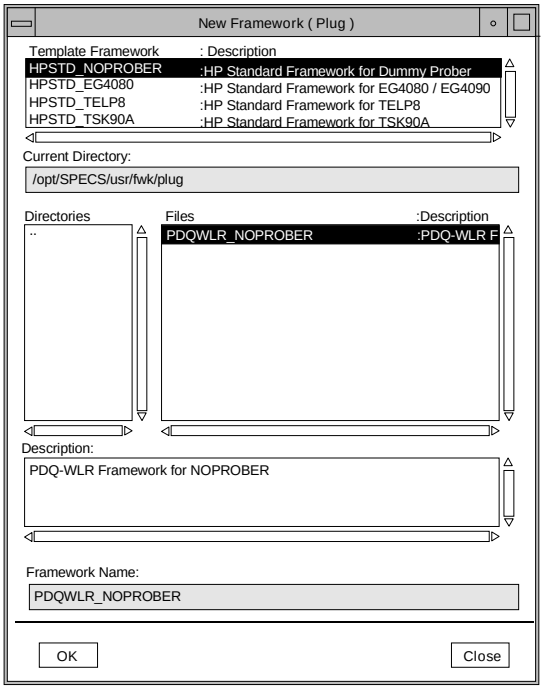
If You Create a New Plug Framework

You can create a new plug framework as shown below:

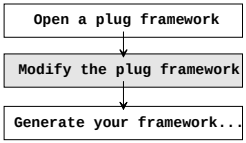
1. Start OUTLINER.
2. Select the “Options:Preferences..” menu to open the “Preferences “ dialog.
3. Select the “Show” radio button for “Plug framework”, then click on the “OK” button.



4. Select the “Framework:New Plug...” menu to open the “New Framework (plug)“ dialog.



5. Create a new plug framework from HPSTD framework as follows:
   a. Select a HPSTD framework from the “Template Framework” list.
   b. Enter a plug framework name in the “Framework Name” field. Then click “OK”.



To Modify Plug Framework

You can add and modify specifications in HPSTD framework. Table 3-2 shows modification rule of HPSTD framework.

Table 3-2 **Specifications You Can Add or Modify**

Modification	System Spec	Tag Spec	Panel Spec	Action Spec
Defining a new spec	N.A.[a]	N.A.	OK	OK
Modifying specifications	N.A.	OK	N.A.	OK

a.Not Applicable

See the typical cases to modify the plug framework in “Modification Examples” on page 3-18 which shows the following examples:

- “To Change Label in Operation Panel”
- “To Add New Choices to Menu Panels”
- “To Run Desired Test Report Commands”
- “To Add a New Panel”
- “To Replace an Existing Panel with New Panel”

NOTE

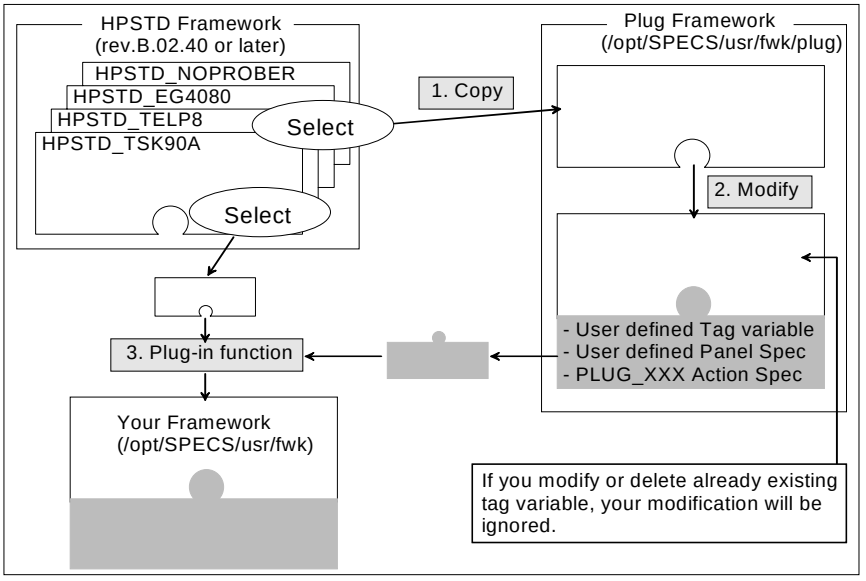
Create a backup file before you modify plug framework

If you modify an existing plug framework, create a backup file as shown below:

1. Open a plug framework. See “To Open Plug Framework” on page 3-8.
2. Select “Framework:Save As...” menu to open the “Save Framework (plug)” dialog.
3. Enter a new name for opened plug framework in the “Framework Name” field.Then, click “OK”.

To Modify Tag Specification

You can only define new tag variables in SYSTEM, LOT, WAFER or DIE Tag Spec. If you delete or modify already existing tag variable, the framework plug-in function ignores your modification.



NOTE

Limitation of prefix

You cannot use the prefix as shown below.
DEV_ , DIR_ , DSP_ , ENV_ ,
FIL , JDG_ , MNU_ , PNL_ ,
PRB_ , RPT_ , RTN_ , SPECS_ ,
SYS_ , TIM_ , TMP_, TST_ ,
UIF_ , UPD_ ,

Framework plug-in function ignores Tag variables which has prefix listed above.

To Modify Action Specification

You can modify only the action specifications named “PLG_xxxx”. Table 3-3 and Table 3-4 list specifications allowed to be modified in HPSTD framework.

CAUTION Do not modify action specifications named “HP_xxxx”

You can modify only action specifications named “PLG_xxxx”.

Table 3-3 **Specifications you can modify (before test plan execution)**

Action Spec Name	Description
PLG_FWK_BEGIN	Called at System Spec after the “Copyright” panel.
PLG_INIT_HW_ONLINE	Called at HP_INIT_HW_ONLINE.
PLG_INIT_HW_OFFLINE	Called at HP_INIT_HW_ONLINE.
PLG_GET_RECIPE_DOUBLE	Called at HP_TEST_SET_RECIPE_1L2C if test plan is not loaded yet. This Action Spec gets test conditions using “master.dat” file.
PLG_GET_RECIPE_SINGLE	Called at HP_TEST_SET_RECIPE_1L1C or HP_TEST_SET_RECIPE_2L2C if test plan is not loaded yet. This Action Spec gets test conditions using “master.dat” file.
PLG_SET_RECIPE_DOUBLE	Called at HP_TEST_SET_RECIPE_1L2C if test plan is not loaded yet. This Action Spec shows and selects test conditions.
PLG_SET_RECIPE_SINGLE	Called at HP_TEST_SET_RECIPE_1L1C or HP_TEST_SET_RECIPE_2L2C if test plan is not loaded yet. This Action Spec shows and selects test conditions.
PLG_SET_TAGINFO_DOUBLE	Called at HP_TEST_SET_RECIPE_1L2C if test plan is already loaded. This Action Spec shows and selects test conditions.
PLG_SET_TAGINFO_SINGLE	Called at HP_TEST_SET_RECIPE_1L1C or HP_TEST_SET_RECIPE_2L2C if test plan is already loaded. This Action Spec shows and selects test conditions.
PLG_NAME_DATAFILE	Called at HP_TEST_SET_RECIPE_1L2C, HP_TEST_SET_RECIPE_1L1C or HP_TEST_SET_RECIPE_2L2C if test plan is already loaded. This Action Spec names data file name and data directory.

Figure 3-3 Panels and Actions before executing a Test Plan (before test plan execution)

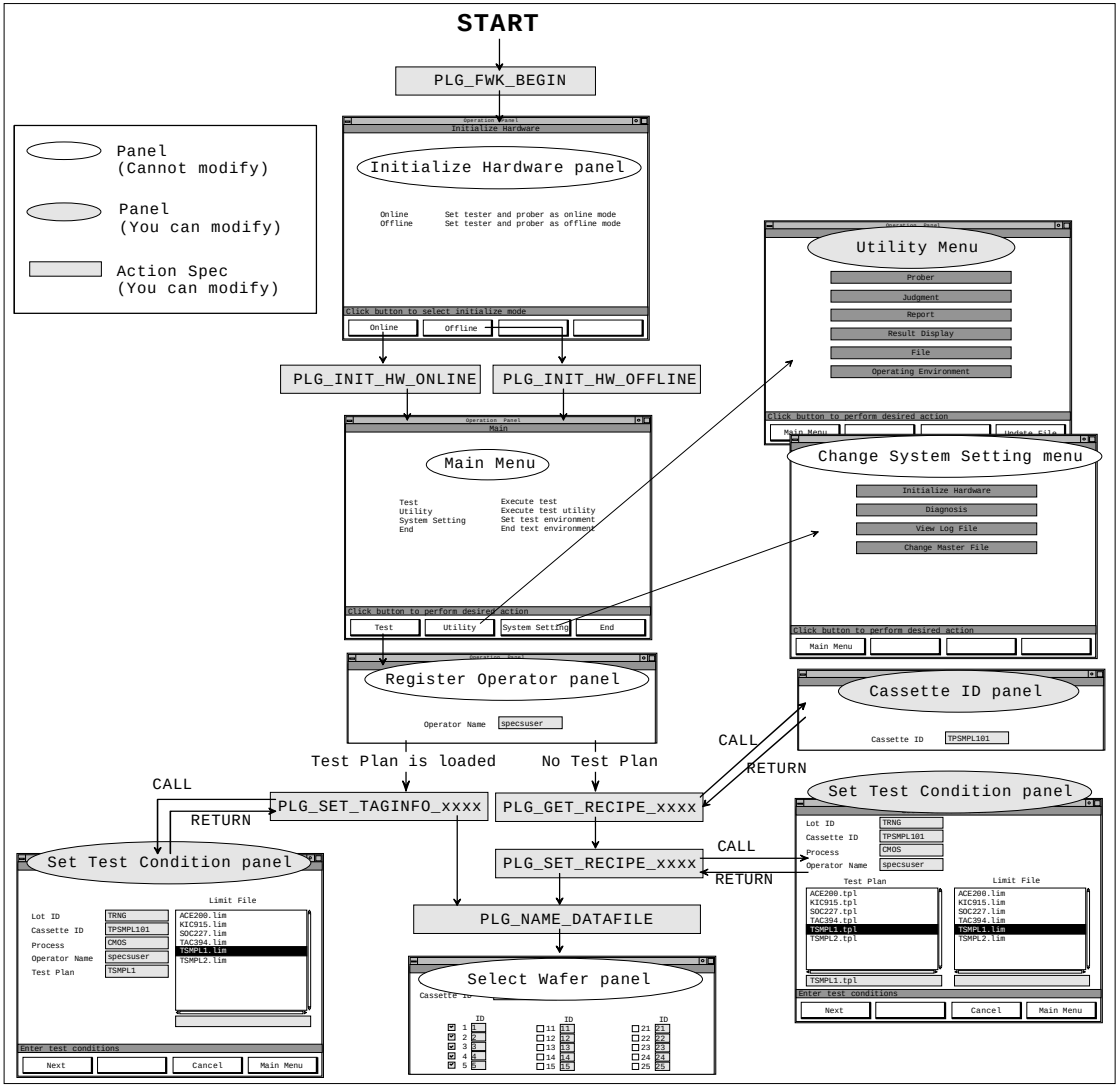
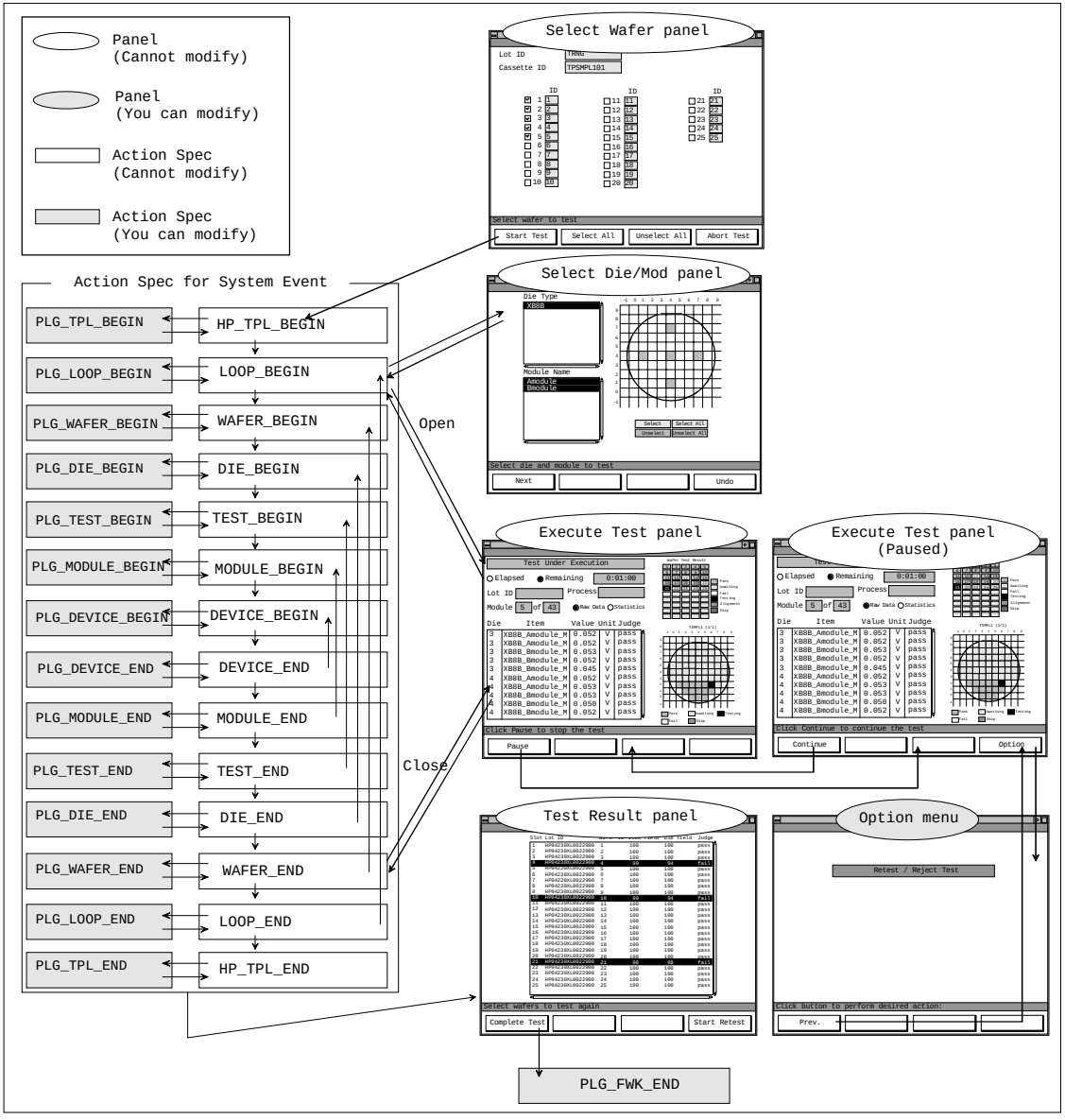


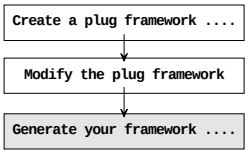
Table 3-4 **Specifications you can modify (while test plan execution)**

Action Spec Name	Description
PLG_TPL_BEGIN	Called at the beginning of HP_TPL_BEGIN.
PLG_LOOP_BEGIN	Called at the beginning of LOOP_BEGIN.
PLG_WAFER_BEGIN	Called at the beginning of WAFER_BEGIN.
PLG_DIE_BEGIN	Called at the beginning of DIE_BEGIN.
PLG_TEST_BEGIN	Called at the beginning of TEST_BEGIN.
PLG_MODULE_BEGIN	Called at the beginning of MODULE_BEGIN.
PLG_DEVICE_BEGIN	Called at the beginning of DEVICE_BEGIN.
PLG_DEVICE_END	Called at the beginning of DEVICE_END.
PLG_MODULE_END	Called at the beginning of MODULE_END.
PLG_TEST_END	Called at the beginning of TEST_END.
PLG_DIE_END	Called at the beginning of DIE_END.
PLG_WAFER_END	Called at the beginning of WAFER_END.
PLG_LOOP_END	Called at the beginning of LOOP_END.
PLG_TPL_END	Called at the beginning of TPL_END.
PLG_FWK_END	Called at the end of framework execution.

Figure 3-4 Panels and Actions while executing a Test Plan (while test plan execution)

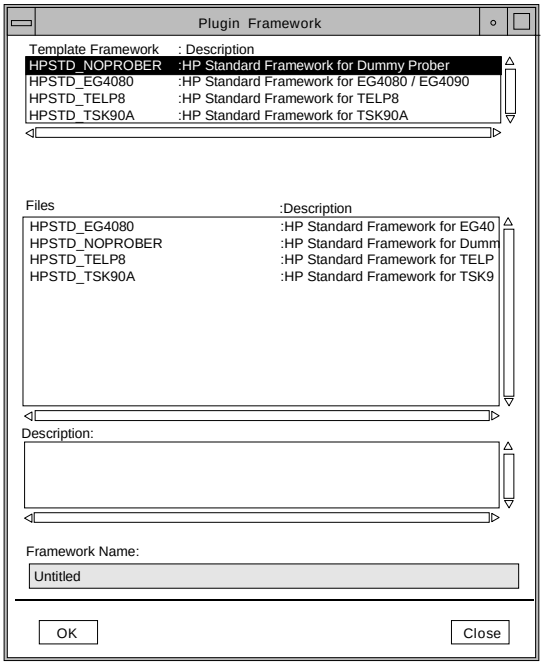


To Generate New Framework



Generate your framework by combining the plug framework to HPSTD framework as follow:

1. Open the plug framework. See “To Open Plug Framework” on page 3-8.
2. Select the “Framework:Plug In...” menu to open the “Plug In Framework” dialog.



3. Select a HPSTD framework from the “Template Framework” list in the dialog.
4. Enter a new framework file name in the “Framework Name:” field in the dialog.
5. Click on the “OK” button in the dialog.

To Add Algorithms to the Utility Algorithm Library

The Utility Algorithm Library contain algorithms which are utilities and for processing string, character, numeric data. HP SPECS provides the following Utility algorithm libraries:

- UTILITY (for HP4070 series tester)
- UTILITY4062 (for HP4062 series tester)
- UTILITY_NOTESTER (for offline environment)

These provided algorithms are used in HPSTD framework. If you modify the provided Utility algorithm library, the following problems may occur:

- Error occurs when you execute the HPSTD framework
- Your modification will be lost when you update the HP SPECS to the latest version.

So you should create other Utility algorithm library and link it to the provided Utility algorithm library when you add new algorithms to the Utility algorithm library. Do as shown below:

1. Open the Utility algorithm library for your tester using Algorithm Builder.
2. Define new algorithms in the algorithm library.
3. Save the algorithm library as a new file in “/opt/SPECS/usr/alg/utility” or “/opt/SPECS/usr/calg/utility” directory.
4. Execute the following command from command line to link the new algorithm library to the Utility algorithm library:

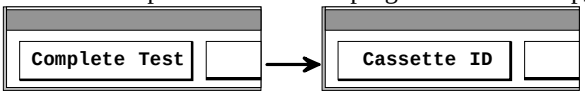
   $ alglink -o /opt/SPECS/usr/alg/utility/*AlgName* /opt/SPECS/usr/alg/utility
   (*AlgName*: new algorithm library name to be created)
5. Open the “/opt/SPECS/sys/config/sysconf” file using text editor. Then, set the new algorithm name created in the previous step to the UTILLIB parameter.

Modification Examples

This section shows five modification examples.

To Change Label in Operation Panel

In HPSTD framework, labels in operation panel are set using message files which are stored in "/opt/SPECS/usr/fwk/plug/*Framework*~/Op_Lang" directory.



You can change panel messages by editing message files as follows:

Step 1. Create a plug framework if not exists

Refer to "To Open Plug Framework" on page 3-8.

Step 2. Edit a message file

1. Change directory which stores message files as follow:

   $ cd /opt/SPECS/usr/fwk/plug/*Framework*~/*Op_Lang*

   Framework~: Framework name. Needs a tilde(~) after the framework name.
   OP_Lang: OP_LANG parameter value set in the sysconf file

2. Search the message file which store the message you will change using the following command:

   $ grep *string* * *string*: a message you will change

3. Open the message file found at the previous step and edit it.

Step 3. Generate new framework using plug-in function

Refer to "To Generate New Framework" on page 3-16.

NOTE

"$" + "*filename*" means using message file

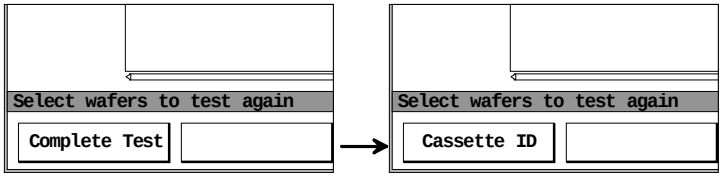
The list below is an example to the label parameter for SingleDisp Widget.

"**message**" Displays the string in Operation Panel directly.

$message Displays the string stored in the "/opt/SPECS/usr/fwk/*Framework*~/OP_Lang/message" file.

Example

This section shows an example to change a softkey label in Operation Panel



To change “Complete Test” label to “Cassette ID” in the “Test Result” panel for the HPSTD_NOPROBER framework.

1. Create a plug framework if not exists. See “To Open Plug Framework” on page 3-8.
2. Search the message file.

   In the terminal emulator window, execute the following commands:

   $ cd /opt/SPECS/usr/fwk/plug/*MyPlugFwk*~/C (*MyPlugFwk* : opened framework name)
   $ grep “Complete Test” *

   (The following result will be listed in the terminal)

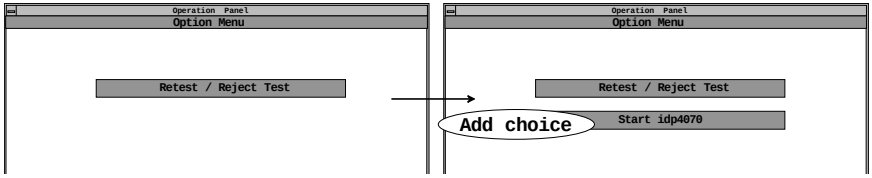
   ```
   END_TEST:Complete Test
   INS_TEST_END_RESULT:Click Complete Test to start the next test.
   ```

   In this case, “END_TEST” is the message file which stores the softkey label.
3. Edit the message file

   Open the message file, “END_TEST”, using text editor and change the label from “Complete Test” to “Cassette ID”.
4. Generate a new framework using plug-in function. Refer to “To Generate New Framework” on page 3-16.

NOTE Message files for plug framework are copied to the “/opt/SPECS/usr/fwk/*Framework~/OP_LANG”* directory by plug-in function.

To Add New Choices to Menu Panels



You can add new choices to the Utility menu and Option menu. Do as follows:

Step 1. Open a plug framework

Refer to “To Open Plug Framework” on page 3-8.

Step 2. Define new Action specifications

Define new Action specifications for new choices of Utility or Option menu.

Step 3. Edit the *.cf file

Add a new choice to the configuration file (*.cf) as follows:

1. Open “/opt/SPECS/usr/fwk/plug/*Framework*.cf file using text editor.
2. Add a title string and Action Spec name defined at “Step 2” for new button. The Table below lists parameters in *.cf file to modify the menu panels.

Configuration file parameters for “Utility” and “Option” menu

Parameter	Description
MNU_UTIL1_LABEL	“$MASTER” is already set for “Change Master File” menu. You can modify this setting if you don’t use the “master.dat” file.
MNU_UTIL2_LABEL	String for new button of Utility menu.
MNU_UTIL3_LABEL	String for new button of Utility menu.
MNU_UTIL1_ACT	“HP_MODIFY_MASTER “ is already set.
MNU_UTIL2_ACT	Action spec name to be assigned to “MNU_UTIL2_LABEL”.
MNU_UTIL3_ACT	Action spec name to be assigned to “MNU_UTIL3_LABEL”.
MNU_OPT1_LABEL	String for new button of Option menu.
MNU_OPT2_LABEL	String for new button of Option menu.
MNU_OPT3_LABEL	String for new button of Option menu.
MNU_OPT1_ACT	Action spec name to be assigned to “MNU_OPT1_LABEL”.
MNU_OPT2_ACT	Action spec name to be assigned to “MNU_OPT2_LABEL”.
MNU_OPT3_ACT	Action spec name to be assigned to “MNU_OPT3_LABEL”.

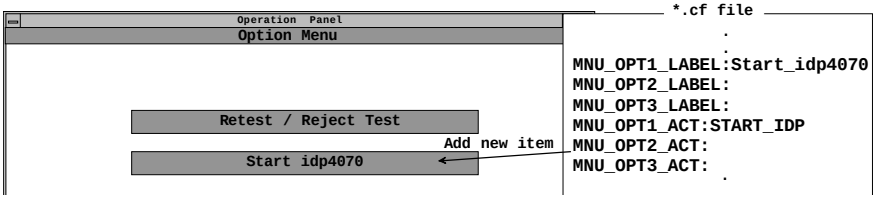
3. Save the configuration file.

Step 4. Generate new framework using plug-in function

Refer to “To Generate New Framework” on page 3-16.

Example

This section shows an example to add a new choice to the “Option” menu for starting idp4070 program.



1. Open the plug framework. Refer to “To Open Plug Framework” on page 3-8.
2. Define an Action spec.

   Add a new Action spec, “START_IDP”, to your plug framework. Then, enter as in the table below using SPEC EDITOR:

Table 3-5 Contents of “START_IDP” Action spec

Command	Parameter	Comment
PANEL	“PNL_START_IDP”,2	Open the message panel
EXECUTE	“/opt/hp4070/bin/idp4070”	Start idp4070
PANEL	“PNL_START_IDP”,0	Close the message panel

3. Define a Panel spec for the action.

   Add a new Panel spec,”PNL_START_IDP”, to the plug framework. In the SPEC EDITOR for the Panel spec, enter as follows:

Table 3-6 Contents of “PNL_START_IDP” Panel spec

Widget Name	Widget Type	Parameter
IDP_MES	Label	label=”idp4070 is running”

4. Open the “/opt/SPECS/usr/fwk/plug/*Framework*.cf” file using text editor. In the text editor, set “Start_idp4070” to “MNU_OPT1_LABEL” and “START_IDP” to “MNU_OPT1_ACT” as in the figure above.
5. Generate a new framework using plug-in function. Refer to “To Generate New Framework” on page 3-16.

To Run Desired Test Report Commands

You can set report commands to run test report commands at the end of test plan execution. Do as follows:

Step 1. Open a plug framework

Refer to “To Open Plug Framework” on page 3-8.

Step 2. Edit the *.cf file

Add a report command to the configuration file (*.cf) as follows:

1. Open “/opt/SPECS/usr/fwk/plug/*Framework*.cf file using text editor. MyPlugFwk is a framework name opened in “Step 1.”.
2. Set a report command and options to RPT_USERx_xxx in configuration file. The table below lists parameters in *.cf file to set report command.

Configuration file parameters for “Utility” and “Option” menu

Parameter	Description
RPT_USER1_CMD	Report command name
RPT_USER2_CMD	Report command name
RPT_USER3_CMD	Report command name
RPT_USER4_CMD	Report command name
RPT_USER5_CMD	Report command name
RPT_USER1_ARG	Options for RPT_USER1_CMD
RPT_USER2_ARG	Options for RPT_USER2_CMD
RPT_USER3_ARG	Options for RPT_USER3_CMD
RPT_USER4_ARG	Options for RPT_USER4_CMD
RPT_USER5_ARG	Options for RPT_USER5_CMD

3. Save the configuration file.

Step 3. Generate new framework using plug-in function

Refer to “To Generate New Framework” on page 3-16.

Example

This section shows an example to run “wafsum.file” command and creates a summary file “/var/opt/SPECS/data/data.txt” at the end of test plan execution.

1. Open the plug framework. Refer to “To Open Plug Framework” on page 3-8.
2. Open the “/opt/SPECS/usr/fwk/plug/*Framework*.cf” file using text editor. *Framework* is a plug framework name opened in step the previous step. In the text editor, enter report command and option as shown in Table 3-7.

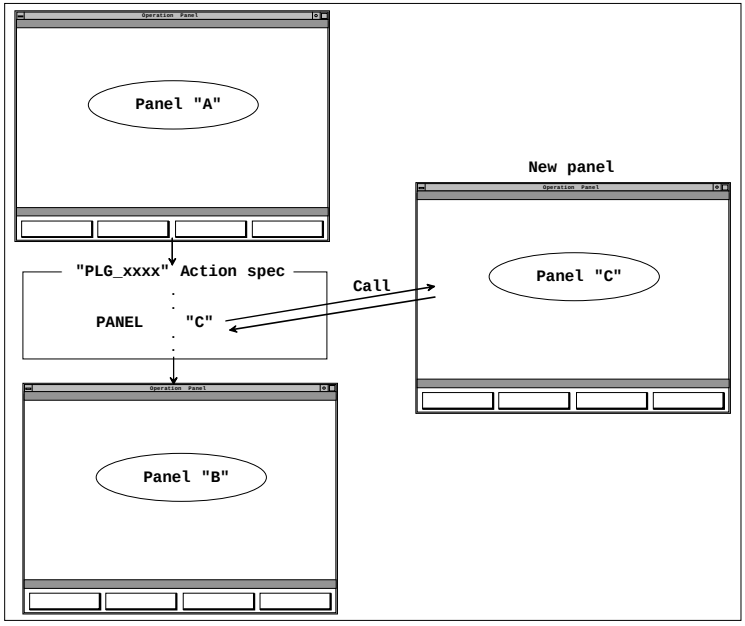
Table 3-7 Parameters in configuration file

Parameter	Value
RPT_USER1_CMD	wafsum.file
RPT_USER1_ARG	/var/opt/SPECS/data/data.txt

3. Generate a new framework using plug-in function. Refer to “To Generate New Framework” on page 3-16.

To Add a New Panel

You can define a new operation panel and call it from the PLG_xxxx Action spec.



Step 1. Open a plug framework

Refer to “To Open Plug Framework” on page 3-8.

Step 2. Define a new Panel spec

Define a new Panel spec using OUTLINER and SPEC EDITOR.

Step 3. Call the added panel from “PLG_xxxx” Action spec

Edit “PLG_xxxx” Action spec to call the added panel using “PANEL” command. “HPSTD Framework Execution Flow” on page 3-4 shows HPSTD framework execution flow and all “PLG_xxxx” Action specifications already defined in the HPSTD framework.

Step 4. Generate new framework using plug-in function

Refer to “To Generate New Framework” on page 3-16.

Example

This section shows an example to add a new panel for setting a data file name.

The procedure below is for adding a new panel to the HPSTD_NOPROBER framework.

1. Open the plug framework. Refer to “To Open Plug Framework” on page 3-8.
2. Add a Panel spec.

   Define a new Panel Spec,”PNL_DATAFILE_NAME”, as in the table below.

Table 3-8 Contents of “PNL_DATAFILE_NAME” Panel spec

Widget Name	Widget Type	Parameter	Variable
FILE_ENTRY	SingleInput	label=”Data File Name”	TST_DATA_NAME [a]
FILE_ACCEPT	AcceptKey	label=”Next”	

a. Already defined in Tag spec.

3. Edit the “PLG_NAME_DATAFILE” Action Spec to call the added panel as follow:

Table 3-9 Statement to call the added panel

Command	Parameter	Comment
ACTION [a]	“HP_NAME_DATAFILE”	
PANEL	“PNL_DATAFILE_NAME”	File name entry panel

a. Already defined in this Action spec.

4. Generate a new framework using plug-in function. Refer to “To Generate New Framework” on page 3-16.

To Replace an Existing Panel with New Panel

You can replace the panels listed in Table 3-10 with your new panel if the panel is called from “PLG_xxxx” Action Spec.

Table 3-10 Panels called from “PLG_xxxx” Action Spec

Panel Title	Panel Spec Name	Action Spec [a]
Cassette ID	HP_SET_KEY_CASS	HP_GET_RECIPE_SINGLE HP_GET_RECIPE_DOUBLE
Set Test Condition (1 lot 1 cassette)	HP_SET_RECIPE_1L1C	HP_SET_RECIPE_SINGLE
Set Test Condition (1 lot 2 cassette)	HP_SET_RECIPE_1L2C	HP_SET_RECIPE_DOUBLE

a. Action spec name which calls the panel.

Step 1. Open the plug framework

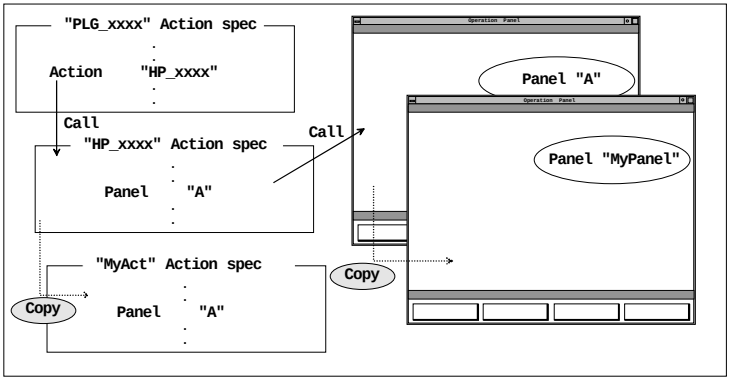
Refer to “To Open Plug Framework” on page 3-8.

Step 2. Create a new panel, “MyPanel”, using copy function

Copy a Panel spec to the other name using the “Spec:Copy....” menu in OUTLINER.

Step 3. Create a new action spec, “NewAct”, using copy function.

Copy the HP_xxx Action Spec which calls the panel to be replaced using the “Spec:Copy....” menu in OUTLINER.



Step 4. Modify the copied Panel Spec

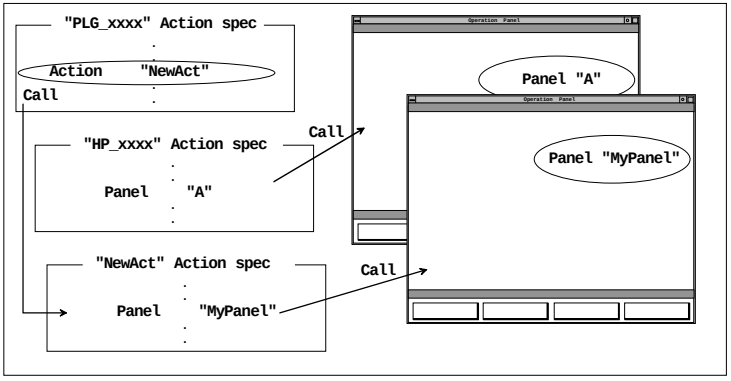
Modify the copied Panel Spec using SPEC EDITOR.

Step 5. Modify the copied Action Spec to call the “MyPanel”.

Open the Action Spec copied in the “Step 3”. Then, replace the panel name to the new name created in the “Step 2”

Step 6. Modify the “PLG_xxx” Action Spec to call the “NewAct”.

Open the “PLG_xxxx” Action Spec which calls the “HP_xxxx” Action Spec and replace the action name to the new name created in the “Step 3”.

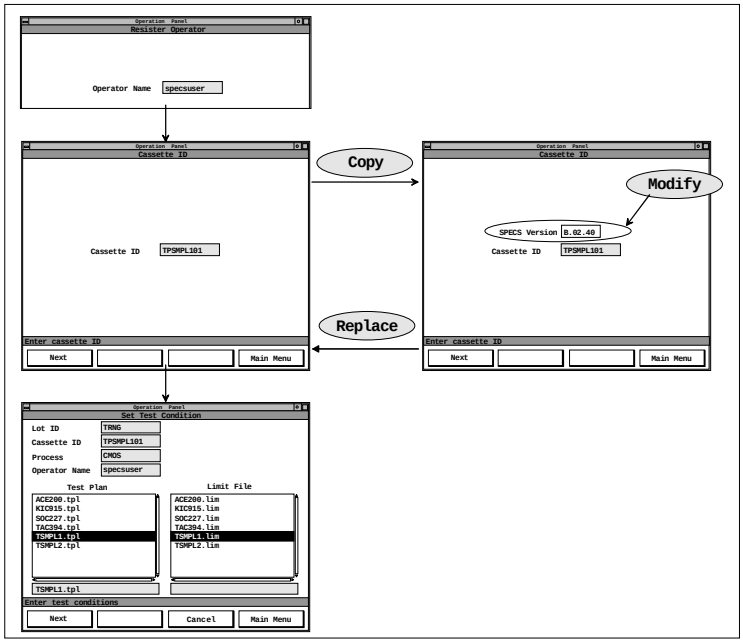


Step 7. Generate new framework using plug-in function

Refer to “To Generate New Framework” on page 3-16.

Example

This section shows an example to display a version number of HP SPECS in the “Cassette ID” panel.



The procedure below is for replacing the “Cassette ID” panel with new panel.

1. Open the plug framework. Refer to “To Open Plug Framework” on page 3-8.
2. Copy a Panel Spec.

   Copy the “HP_SET_KEY_CASS” Panel Spec to “NEW_SET_KEY_CASS”.
3. Copy a Action Spec.

   Copy the “HP_GET_RECIPE_SINGLE” Action Spec to “NEW_GET_RECIPE_SINGLE”.

4. Edit the copied Panel Spec.

   Add the following widget at the end of the “NEW_SET_KEY_CASS” Panel Spec.

Table 3-11 New widget of “NEW_SET_KEY_CASS” Panel spec

Widget Name	Widget Type	Parameter	Variable
VERSION	SingleDisp	label=”SPECS Version”, x=30,y=30, font=PNL_FONT_L	SYSTEM.VERSIONID [a]

a. System variable name defined and updated by HP SPECS automatically.

5. Edit the Action Spec to call the copied panel.

   Open the “NEW_GET_RECIPE_SINGLE” Action Spec. In the Action spec, replace the parameter of the line which calls the “HP_SET_KEY_CASS” panel with the “NEW_SET_KEY_CASS” as follow:

Table 3-12 New statement of “HP_GET_RECIPE_SINGLE” Action spec

Command	Parameter	Comment
PANEL	“NEW_SET_KEY_CASS”	

6. Edit the Action Spec to call the copied Action Spec.

   Open the “PLG_GET_RECIPE_SINGLE” Action Spec. In the Action spec, replace the parameter of the line which calls the “HP_GET_RECIPE_SINGLE” Action Spec with the “NEW_GET_RECIPE_SINGLE” as follow:

Table 3-13 New statement of “PLG_GET_RECIPE_SINGLE” Action spec

Command	Parameter	Comment
ACTION	“NEW_GET_RECIPE_SINGLE”	

7. Generate a new framework using plug-in function. Refer to “To Generate New Framework” on page 3-16.

4 Operation Panel Reference

Operation panels are visual user interfaces for executing the test plan.

The PANEL command is used to display a panel specification (operation panel) for a job, system, action, or test specification. A panel specification must be defined for each operation panel you want to display.

This chapter explains how to create an operation panel.

Creating an Operation Panel

This section describes how to create an operation panel.

1. In Outliner, open Spec Editor for Panel specification. Refer to Chapter 2, “Using Outliner.”.
2. Specify the desired widget name in Widget Name column. This is your user-defined name and cannot be duplicated in a panel specification.
3. The following widget types may be defined in a Panel specification. For more information, refer to the individual widget types described later in this chapter

   The following widget types may be defined in a Panel specification. For more information, refer to the individual widget types described later in this chapter

Widget Name	Description	Refer to
Inputs data		
SingleInput	Displays a data entry field	page 44
ListInput	Displays a multi column data entry field	page 34
ToggleInput	Displays data entry fields with toggle buttons	page 54
Selects items		
BtnSelect	Displays list areas for selecting items	page 14
RadioSelect	Displays radio buttons for selecting items	page 38
ToggleSelect	Displays toggle buttons for selecting items	page 56
WidgetSelect	Displays radio buttons for selecting widgets	page 62
WaferSelect	Displays wafermap and selects Die and Module during framework execution	page 60
Displays data		
SingleDisp	Displays a data display field	page 42
TextDisp	Displays a multi line data display field	page 52
ListDisp	Displays a multi column data display field	page 32
DataDisp	Displays a measurement data table	page 20
StatDisp	Displays a statistical or binning data table	page 46
WaferDisp	Displays a wafermap	page 58
Bitmap	Displays a bitmap image file	page 12

Widget Name	Description	Refer to
Lists data, status or file		
FileDisp	Lists a ASCII format file	page 24
FileView	Lists a tab-separated ASCII format file	page 28
DataView	Displays a measurement data table which can be configured	page 22
StatView	Displays a statistical or binning data table which can be configured	page 48
CassetteDisp	Displays a cassette map	page 18
Calls an user defined action spec		
ActionBtn	Displays a buttun which calls an user defined action spec.	page 8
ActionKey	Displays a softkey which calls an user defined action spec.	page 10
Creates a softkey to control the panel		
AcceptKey	Keeps the panel open until the softkey is clicked, then closes the panel. Any changes made in the panel become effective.	page 6
CancelKey	Closes the panel. Any changes made in the panel are ignored.	page 16
ResetKey	Resets all panel settings to the same status as when the panel was first opened. The panel does not close.	page 40
For a decoration		
Label	Displays a label	page 30
Separator	Displays a separator line	page 41

NOTE

For panels that use some of the widgets, the use of the AcceptKey widget in the same panel specification may be required, so that the panel stays open until the desired input or selection is performed. Refer to the specific widget descriptions, later in this chapter, for more information.

4. Enter the parameters for the widget in the Parameter column by using the Assist dialog. A description for each parameter appears in the Explanation field. For more information, refer to the widget descriptions in this chapter.

NOTE

The panel size is fixed: top left corner (0,0), bottom right corner (100,100). The location in which the widget appears in the panel is specified using the X-Y coordinates to set the bottom-left corner position of the widget.

5. Enter values in the Variable and Range columns if necessary.
6. Enter a message for the widget in the Message column. When the widget is selected, the entered string appears at the lower-left corner of the panel.
7. Repeat steps 2 through 6 for each widget.

Common parameters in Panel Spec

Table 4-1 Columns in Panel Spec

Widget Name	Widget type	Parameter	Variable	Range	Message
Name	*Type*	*parms*	*var_name*	*range*	*message*

Widget Name HP SPECS uses this parameter to manage widgets. Set unique name in this field. Must start with alphabet character and can have maximum 32 alphabetic, numeric and _(underline) characters.

Widget Type Widget type can be entered using Assist dialog. For example, SingleInput, DataDisp...

Parameter Parameters for widget can be entered using Assist dialog. Refer to after the next page.

Variable Tag variable for storing any input or sections in the operation panel, or setting strings as input parameters of widget.

Range Range to check the entered value in the operation panel. For example, if “0,1,2” is set in this column, error occurs when a string except 0, 1 or 2 is entered.

Message Message that appears in lower left of panel when the widget field is selected. Double quote is not necessary.

AcceptKey

Activates a softkey at bottom of panel, and panel stays open until this softkey is clicked. Then panel closes and any changes you made in panel become effective. If any input data you input to panel input field is out of range, an improper value error occurs.

You need to use this widget in most panel specifications. If not, the panel only stays open very briefly, then closes. So, you must use this widget for panels that require user input, such as panels that have a SingleInput, BtnSelect, or ListSelect widget.

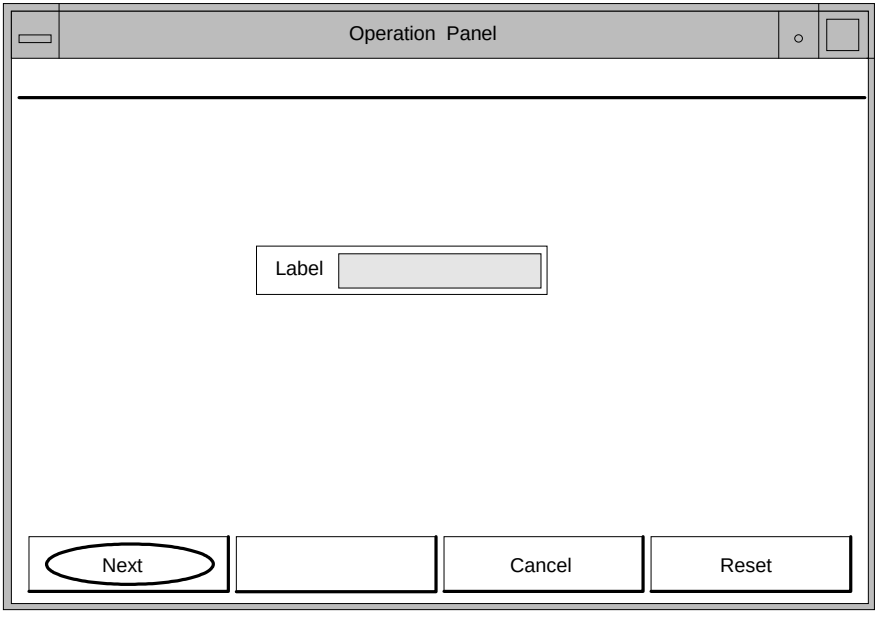
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**AcceptKey**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

Parameters



This figure is an example of AcceptKey widget.

Parameter	Description
label	Label for softkey
pos	Which softkey to use (1:left, 2:second, 3:third, 4:right)
fgcolor, bgcolor	Foreground (label) and background color for softkey. Set color using resource name of X window.
font	Font for widget.

Operation

Click the softkey on the panel. Any field inputs, list selections, or button selections you made becomes effective, and the panel closes.

ActionBtn

Displays a button on the panel. When button is clicked, the related action (specified in Action specification) is executed. The button is effective even while measurement is being performed.

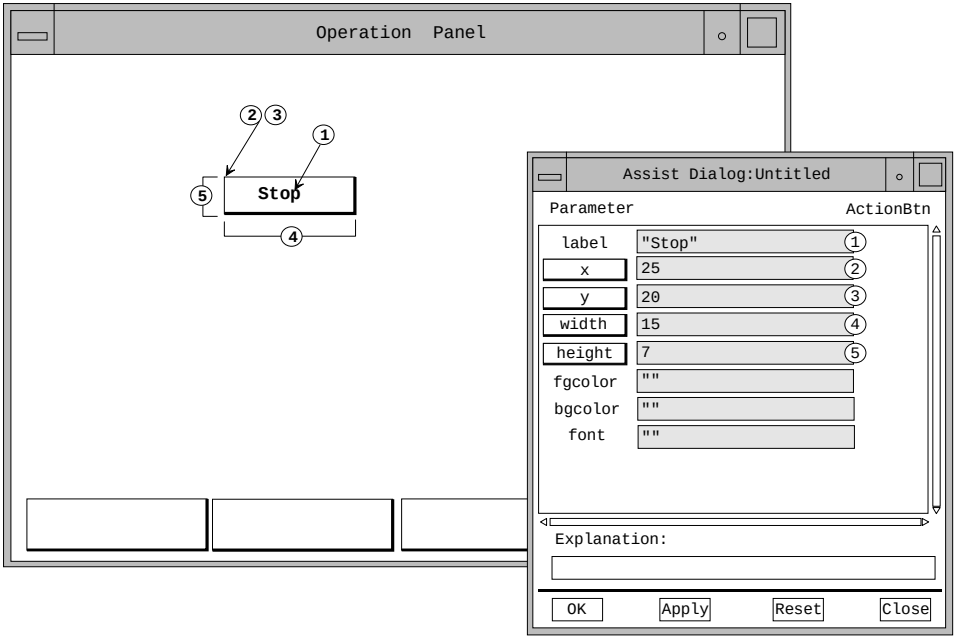
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**ActionBtn**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

Parameters



This figure is an example of ActionBtn widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
label	Label for button
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
width, height	Button width and height (relative size: 1 - 100 %)
fgcolor, bgcolor	Foreground (label) and background color for button. Set color using resource name of X window.
alignment	Position of text (1:left, 2:center, 3:right)
font	Font for widget.

Operation

Click the action button on panel. The action specified in Action specification starts. This widget is effective even while test plan is running.

ActionKey

Activates a softkey that is at bottom of panel.

When softkey is clicked, the related action (specified in Action specification) is executed. The softkey is effective even while measurement is being performed.

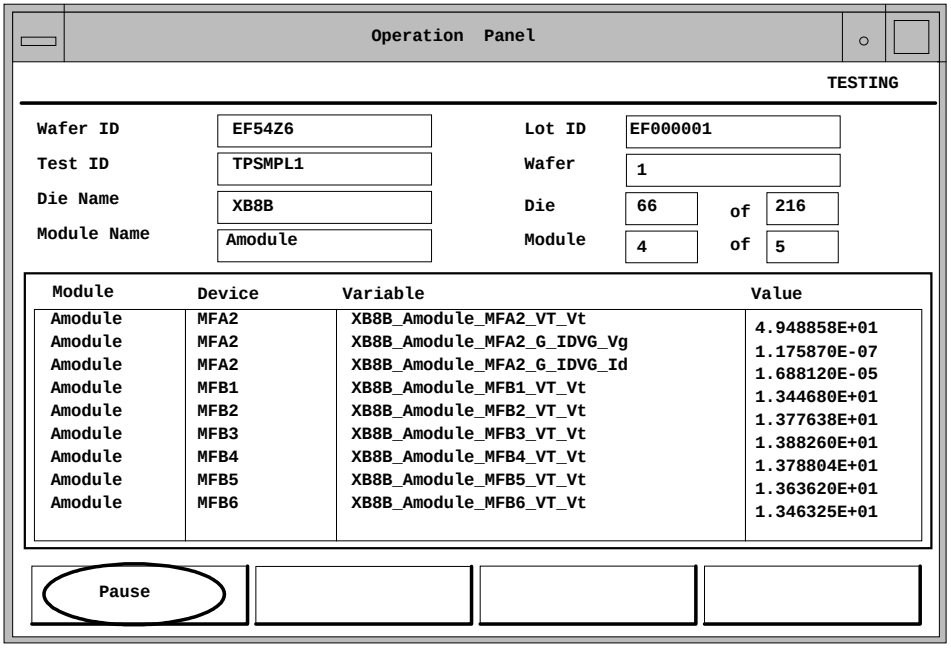
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**ActionKey**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

Parameters



This figure is an example of ActionKey widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
label	Label for softkey
pos	Which softkey to use (1:left, 2:second, 3:third, 4:right)
fgcolor, bgcolor	Foreground (label) and background color for softkey. Set color using resource name of X window.
font	Font for widget.

Operation

Click the action softkey on the panel. The action specified in Action specification starts. This widget is effective even while test plan is running.

Bitmap

Displays a bitmap image file. Only .xbm, .xpm formatted file can be displayed.

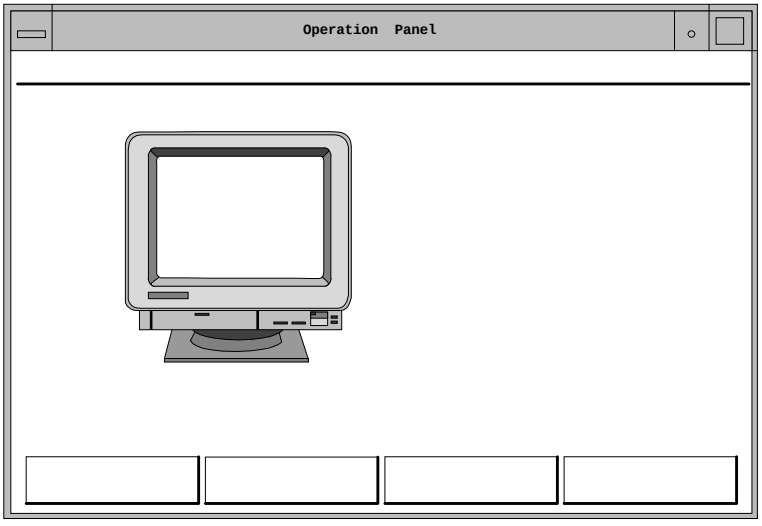
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**Bitmap**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

Parameters



Parameter	Description
filename	Bitmap file name (Only XBM or XPM format file is available)
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
width, height	Bitmap width and height (relative size: 1 - 100 %)
color, bgcolor	Bitmap color and background color. Set color using resource name of X window.
alignment	Widget alignment (1:Left, 2:Center, 3:Right)
margin	Margin around the image (0:no margin, 1:margin)

BtnSelect

Displays buttons that you can click to select desired items.

Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**BtnSelect**	*parms*	*register*	*labels*	*message*

name A label of the line. Name must be unique in the framework.

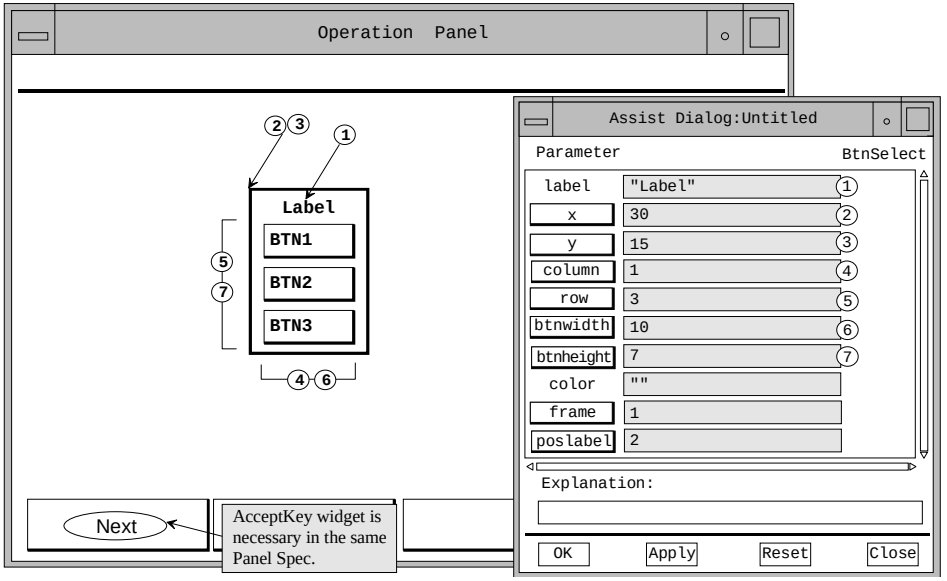
parms Values to be set using Assist dialog. Refer to “Parameters”.

register STRING[ ] type tag variable to store the selected button labels (for LANG=C) or message file name (for LANG=japanese). Array size determines the maximum number of selectable buttons. Array size determines the maximum number of selectable buttons. For example, if a STRING[3] type tag variable is used here, a maximum 3 buttons can be selected. If 4 or more selected, selection denied error occurs.

labels STRING[ ] type tag variable to set button labels.

message Message appears in lower left of panel when the widget is selected.

Parameters



This figure is an example of BtnSelect widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
label	Button area title.
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
column	Number of buttons (horizontal)
row	Number of buttons (vertical)
btnwidth	Button width (relative size: 1 - 100 %)
btnheight	Button height (relative size: 1 - 100 %)
color	Background color of widget. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
poslabel	Label position (1:left, 2:top, 3:right, 4:bottom)
beep	Beep on error (0: no beep, 1:beep)
errmsg	Message for input error.
font	Font for widget.

Operation

When you click on desired button, the button is selected and its color changes.

You can select multiple buttons, and maximum number of selectable buttons is limited as described later. If you select more than the maximum number of buttons, error message appears and warning beep sounds. So, if you want to select other buttons, you need to click and unselect some buttons first.

If button area is too small to display all the buttons, a vertical scroll bar appears.

If you use this widget type, you must also specify `AcceptKey` widget type in the same specification so that panel stays open until you input the data.

CancelKey

Activates a softkey at bottom of panel. If you click this softkey, any input (such as field input, button selection, list selection) you made on the panel is canceled and the panel closes.

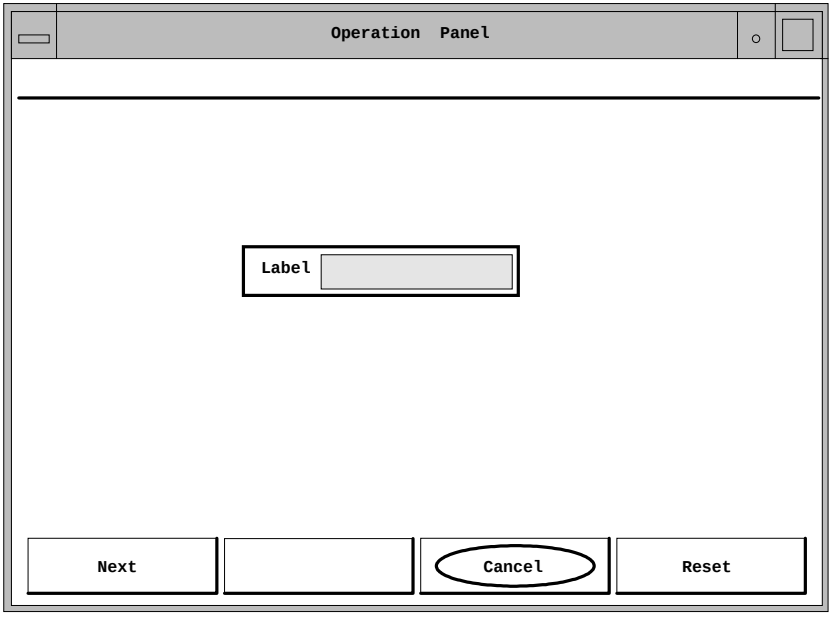
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**CancelKey**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

Parameters



This figure is an example of CancelKey widget.

Parameter	Description
label	Label for softkey
pos	Which softkey to use (1:left, 2:second, 3:third, 4:right)
fgcolor, bgcolor	Foreground (label) and background color for softkey. Set color using resource name of X window.
font	Font for widget.

Operation

Click the softkey on the panel. The panel closes and any changes you made in panel are ignored.

CassetteDisp

Displays a graph which is used to list the status of all wafers in the cassette.

This display's content is updated any time the UPDATE command is executed.

NOTE

CassetteDisp does not list the cassette status automatically.
To list the cassette status, you must obtain the cassette information using the Prober Algorithms and set the parameters in the framework.

Panel Spec Syntax

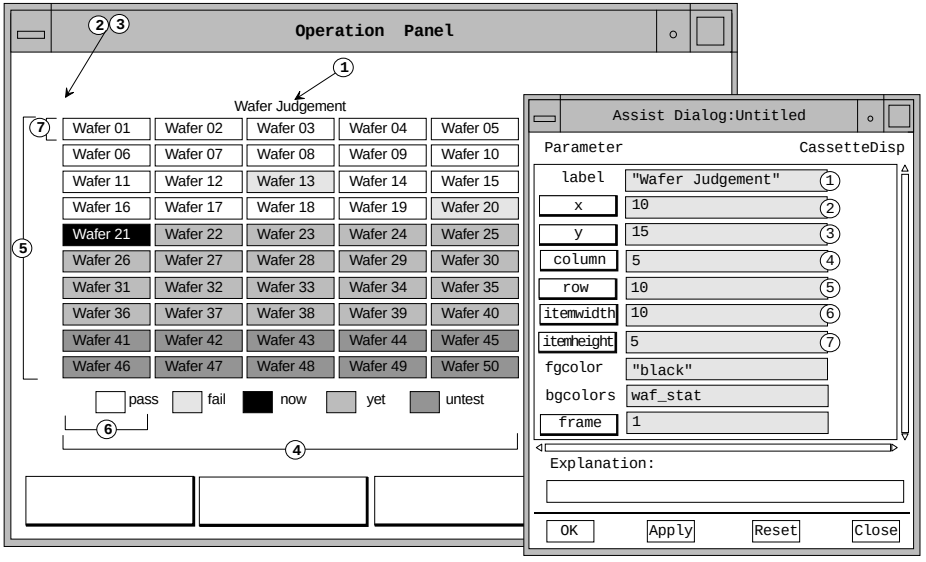
Widget Name	Widget type	Parameter	Variable	Range	Message
name	**CassetteDisp**	*status*	*legend*		

name A label of the line. Name must be unique in the framework.

status STRING[ ] type tag variable to assign each wafer status which is specified by *legend*.

legend STRING[ ] type tag variable to set the wafer legend (Example: "pass", "fail"). Each item appears as a legend string and corresponds to the items for diecolors in the Parameter column.

Parameters



This figure is an example of CassetteDisp widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
label	Title for cassette map
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
column	Number of items to display (horizontal)
row	Number of items to display (vertical)
itemwidth	Item width (relative size: 1-100 %)
itemheight	Item height (relative size: 1 - 100 %)
fgcolor	Item foreground color [a]
bgcolors	STRING[ ] type tag variable to set item background color.
poslabel	Label position (1:right, 2:left)
poslegend	Legend position (1:right, 2:bottom)
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
alignment	Item label alignment (1:left, 2:center, 3:right)
font	Font for list area
items	STRING[ ] type tag variable to set wafer names. For example, “Wafer01”, “Wafer02”, “Wafer03”, ... , “Wafer50” are set in the framework to display the panel as in the figure in the previous page.

a. Set using color resource name for X window.

DataDisp

Displays the algorithm results (module names, device names, data variable names and values) using fixed form. The results are updated after each module (MODULE_END) if sysconf file sets “OP_AUTOUPDATE= TRUE”
This Widget is not updated if “OP_AUTOUPDATE= FALSE”.

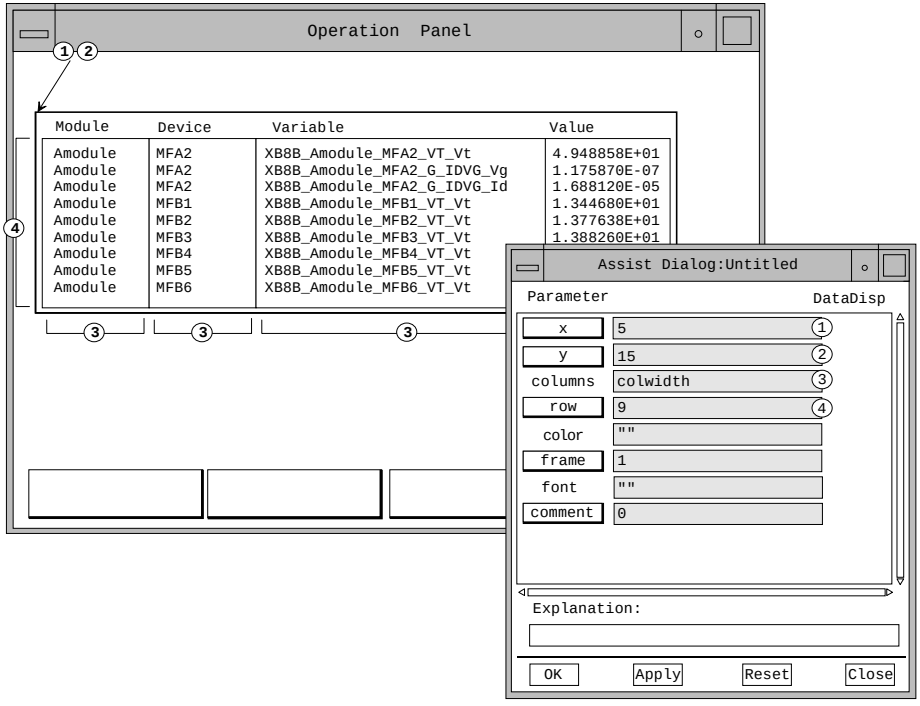
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**DataDisp**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

Parameters



This figure is an example of DataDisp widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
columns	INTEGER[ ] type tag variable to set column width (number of characters) for list
row	Number of lines of list area (number of characters)
color	Color for list. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
font	Font for widget.
comment	Width (number of characters) for comment column

Operation

If the list area is too small to display all the results, a vertical scroll bar will appear. When the test is paused, you can use the scroll bar to scroll the list. New results are appended at end of module test.

DataView

Displays the algorithm results. You can specify which items to display.

This display's content is updated any time the UPDATE command is executed

NOTE

Disp flag in Limit File

Output variable will not be listed in the table if the Disp flag of the output variable is "OFF" status.

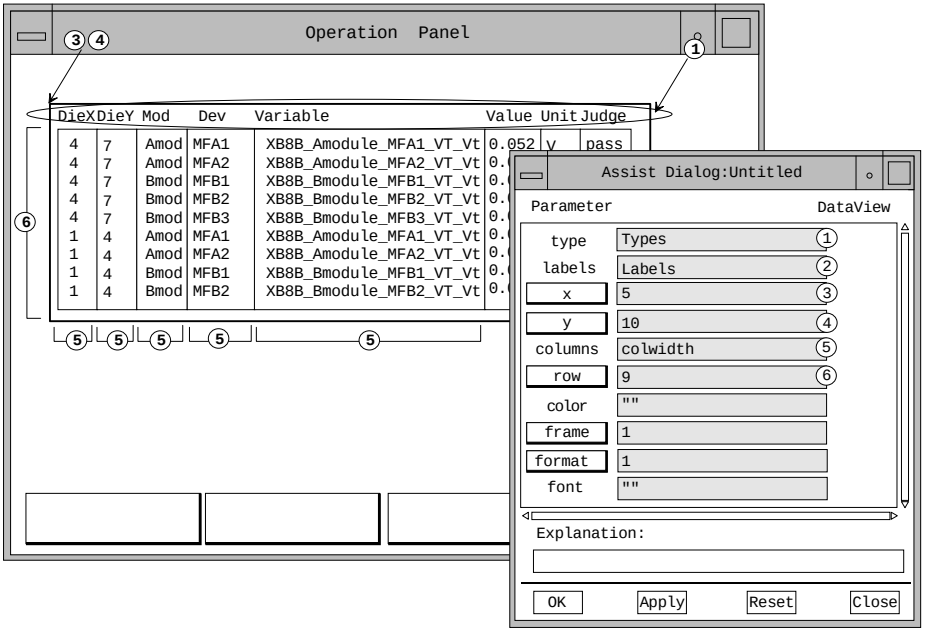
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**DataView**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to "Parameters".

Parameters



This figure is an example of DataView widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
type	STRING[ ] type tag variable to set format types to display. Available format types are listed in “Format type”. For example, types[ ] = “Diex”,”Diey”,”Mod”,”Dev”,”Name”,”Value”,”Unit”, “Judge” are set to display the panel as in the previous page.
labels	STRING[ ] type tag variable to set labels of columns.
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
columns	INTEGER[ ] type tag variable to set column width (number of characters) for list
row	Number of lines of list area (number of characters)
color	Color for list. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
format	Format to display decimal numbers. (1:fixed point, 2:floating point, 3:fixed point with unit)
font	Font for widget.

Format type

Type	Description	Type	Description
Wafer	Wafer ID	Unit	Unit Name
Dielbl	DieLabel	Judge[a]	Judge of output variable
Dienum	Die sequence number	Cmnt	Comment
Diex	X index of Die	RLim1 - 10[b]	Binning limit value
Diey	Y index of Die	ULow1 - 6	Low limit for user bin
Mod	Module Name	UHigh1 - 6	High limit for user bin
Dev	Device Name	JLow	Low limit for “pass” status
Alg	Algorithm Name	JHigh	High limit for “pass” status
Name	Variable Name	VLow	Low limit for “valid” status
Value[c]	Output value	VHigh	High limit for “valid” status

a. String to be displayed as Pass/Fail/Invalid/Untested status can be specified using “Judge:*Pass:Fail:Invalid:Untested*” format.
b. Number of digit can be specified using “RLimX:*num*” format.
c. Number of digit, string for array type output variable and not measured variable can be specified using “Value:*num*:1 *dimension:2 dimension:not measured*” format. Example: ”Value:8:ARRAY[ ]:ARRAY[ ][ ]:not meas”

FileDisp

Displays the contents of the user-created ASCII file on the panel.

This display's content is updated any time the UPDATE command is executed.

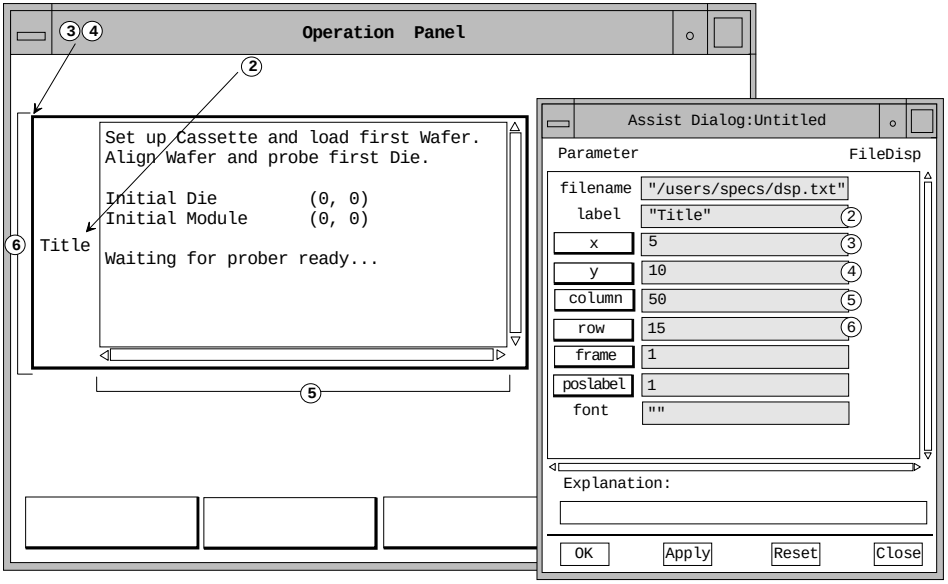
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**FileDisp**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to "Parameters".

Parameters



This figure is an example of FileDisp widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
filename	File name to display. Only ASCII format file can be displayed.
label	Title for list
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
column	List area width (number of characters)
row	Number of lines of list area (number of lines)
color	Color for list. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
poslabel	Label position (1:Left, 2:Upper, 3:Right, 4:Bottom)
font	Font for widget.

Operation

To scroll through the list, click the vertical or horizontal scroll bar. The list can be scrolled from the top to the bottom of the file, even when the test plan is running.

FileSelect

Selects a file or directory.

Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**FileSelect**	*parms*	*register*		*message*

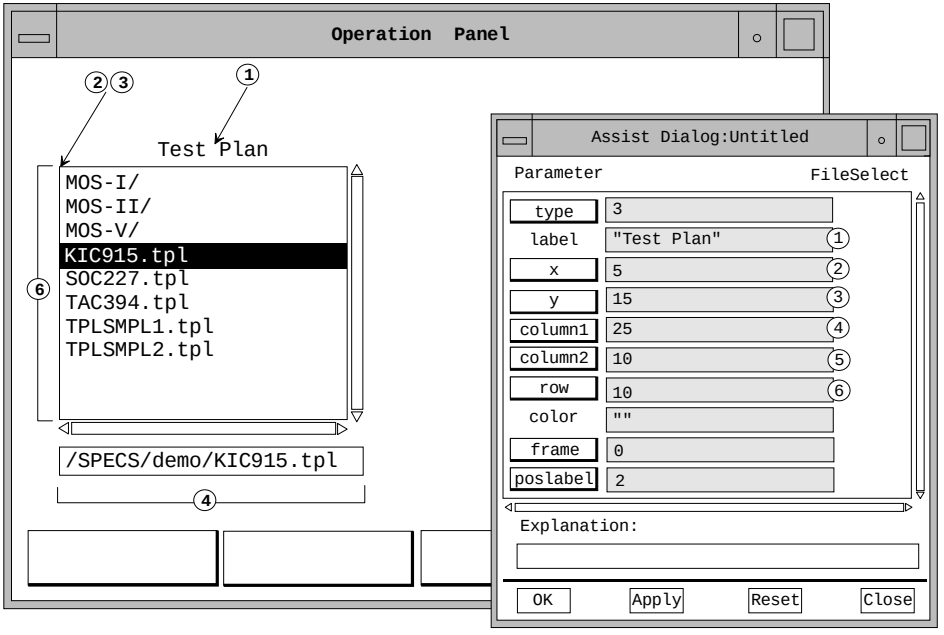
name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

register STRING type tag variable name to set default path and file name and store the selected file name.

message Message appears in lower left of panel when the widget is selected.

Parameters



This figure is an example of FileSelect widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
type	File list type (1: Dir only (one column), 2:File only (one column), 3:File and Dir (one column, 4:File and Dir (two column))
label	Widget title
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
colulmn1	Width for column1 (number of characters)
colulmn2	Width for column2 (number of characters) Effective only “type” parameter is “4”.
row	Number of lines of list
color	Color for list. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
poslabel	Label position (1:Left, 2:Upper, 3:Right, 4:Bottom)
posinput	Position of selected file name (1:Upper, 2:Bottom)
editinput	Editing selected file (0:uneditable, 1:editable)
sort	Order for listing files. (1:no sort, 2:sort in order of ASCII code, 3:sort in order of time stamp)
filter	Filter for listing files.
root	Root directory path
chdir	Changing directory (0:disable, 1:enable)
beep	Beep on error (0:no beep, 1:beep)
errmsg	Message for input error
font	Font for widget.

Operation

Select a file or directory in the list area.

If you use this widget type, you must also specify `AcceptKey` widget type in the same specification so that panel stays open until you input the data.

FileView

Displays the contents of the user-created ASCII file on the panel. The content of the tab separated file is displayed column by column.

This display's content is updated any time the UPDATE command is executed.

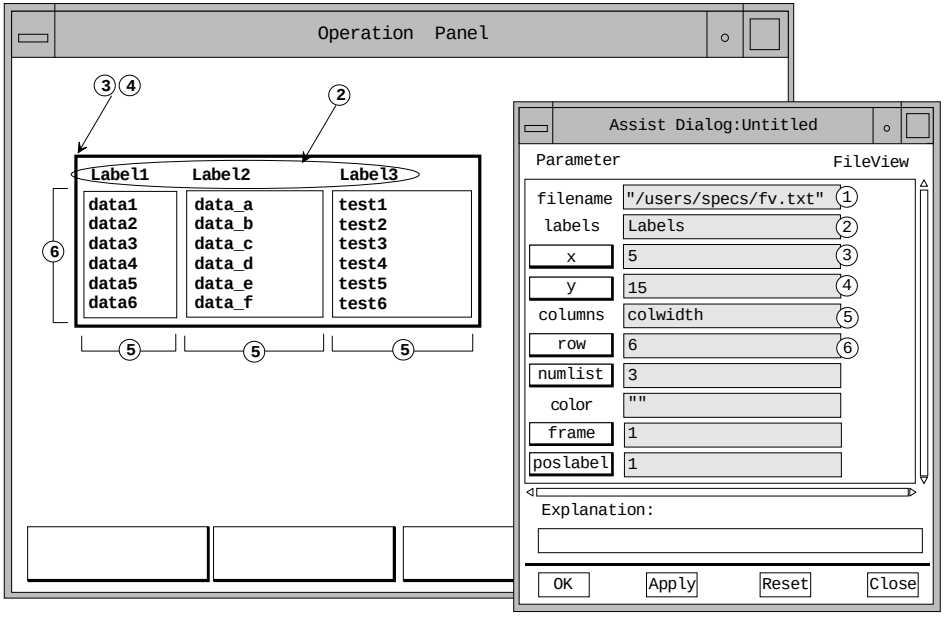
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**FileView**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to "Parameters".

Parameters



This figure is an example of FileView widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
filename	File name to display. Only ASCII format file can be displayed.
labels	STRING[ ] type tag variable to set labels for list
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
columns	INTEGER[ ] type tag variables to set column width (number of characters) for list.
row	Number of lines of list area (number of characters)
numlist	Number of lists
color	Color for list. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
font	Font for widget.

Label

Displays a string on the panel.

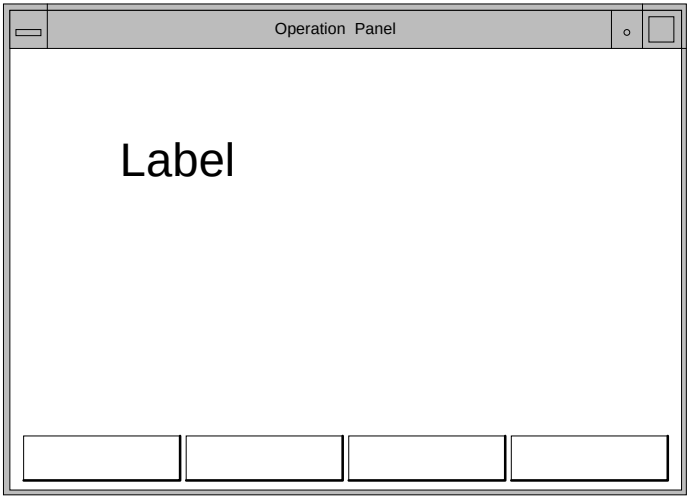
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**Label**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

Parameters



Parameter	Description
label	String to be displayed
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
width, height	Label width and height (relative size: 1 - 100 %)
Color	Text color for label. Set color using resource name of X window.
bgcolor	Background color. Set color using resource name of X window.
alignment	Widget alignment (1:Left, 2:Center, 3:Right)
margin	Margin around the widget (0: no margin, 1:margin)
font	Font for widget.

ListDisp

Displays data in a list. Lists are updated after each module (MODULE_END) if OP_AUTOUPDATE is set TRUE in /opt/SPECS/sys/config/sysconf file.
Also This contents is updated any time when UPDATE command is executed.

Panel Spec Syntax

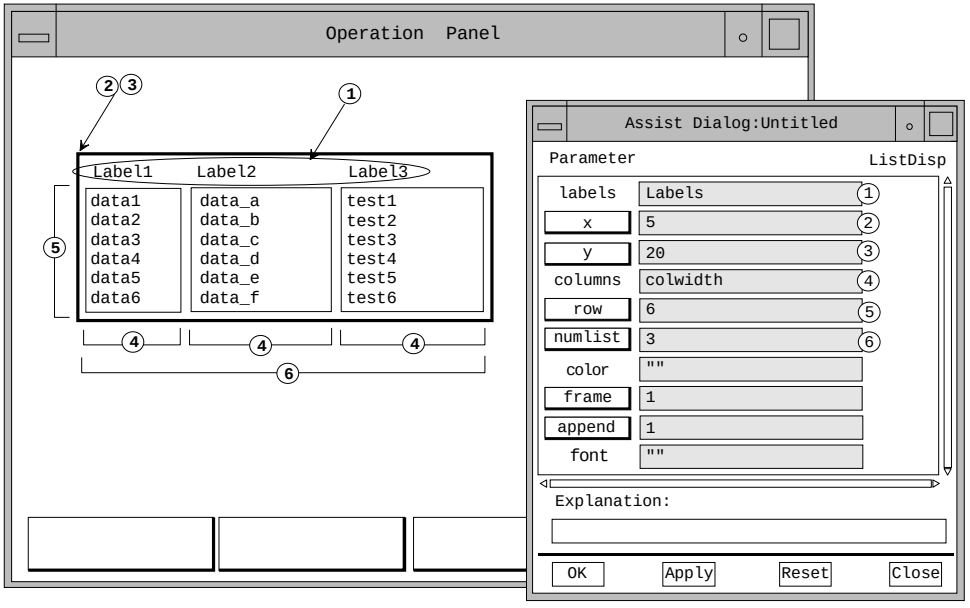
Widget Name	Widget type	Parameter	Variable	Range	Message
name	**ListDisp**	*parms*	*items*		

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

items STRING[ ][ ] type tag variable name to set items to be listed. Refer to “Item Definition” in the next page.

Parameters



This figure is an example of ListDisp widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
labels	STRING[ ] type tag variable to set column labels.
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100 %)
columns	INTEGER[ ] type tag variable to set width (number of characters) of each column.
row	Number of lines of list
numlist	Number of columns of list
color	Color for list. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
append	New data update (0:overwrite, 1:append)
font	Font for widget.
blank	Displays the blank line in the list (0:no blank, 1:blank)

Item Definition

Items to be listed are set using STRING[ ][ ] type tag variable in “Variable” column. For example, to define a list in the figure of previous page, the following strings should be set.

items[1][1] - items[1][6]=”data1”, “data2”, “data3”, “data4”, “data5”, “data6”
items[2][1] - items[2][6]=”data_a”,“data_b”,“data_c”,“data_d”,“data_e”,“data_f”
items[3][1] - items[3][6]=”test1”, “test2”, “test3”, “test4”, “test5”, “test6”

Refer to “Panel Spec Syntax” in previous page.

ListInput

Displays a field in which you can input multi column CHARACTER, STRING, INTEGER, or REAL type data from keyboard.

Panel Spec Syntax

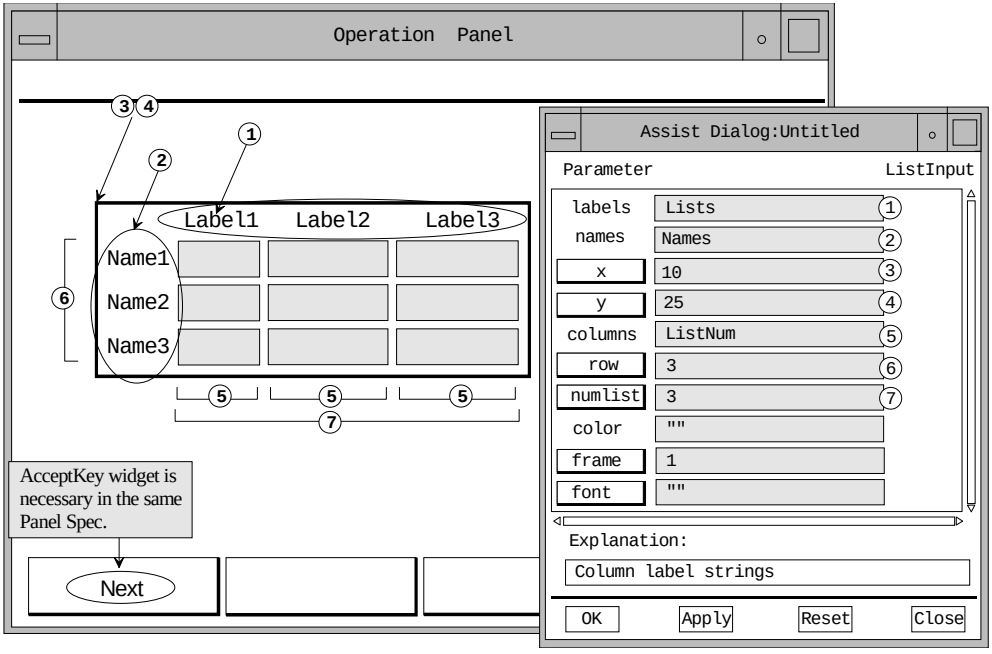
Widget Name	Widget type	Parameter	Variable	Range	Message
name	**ListInput**	*parms*	*register*		

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

register STRING[ ] [ ] type tag variable name to store the value entered in panel. Must be defined in Tag Spec.

Parameters



This figure is an example of ListInput widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
labels	STRING[ ] type tag variable to set column labels.
names	STRING[ ] type tag variable to set row labels.
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
columns	INTEGER[ ] type tag variable to set width (number of characters) of each column.
row	Number of lines of list
numlist	Number of columns of list
color	Color for input field. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
font	Font for widget.

Operation

Click the desired field and input a CHARACTER, STRING, INTEGER, or REAL type data from keyboard.

For inputting REAL type value, you can use an exponent or unit.

To verify the input data, press the Return key or click the AcceptKey softkey. If input data is out of range, an improper value error occurs.

If you use this widget type, you must also specify `AcceptKey` widget type in the same specification so that panel stays open until you input the data.

ListSelect

Displays hierarchical list areas in which you can select items.

Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**ListSelect**	*parms*	*register*	*items*	*message*

name A label of the line. Name must be unique in the framework.

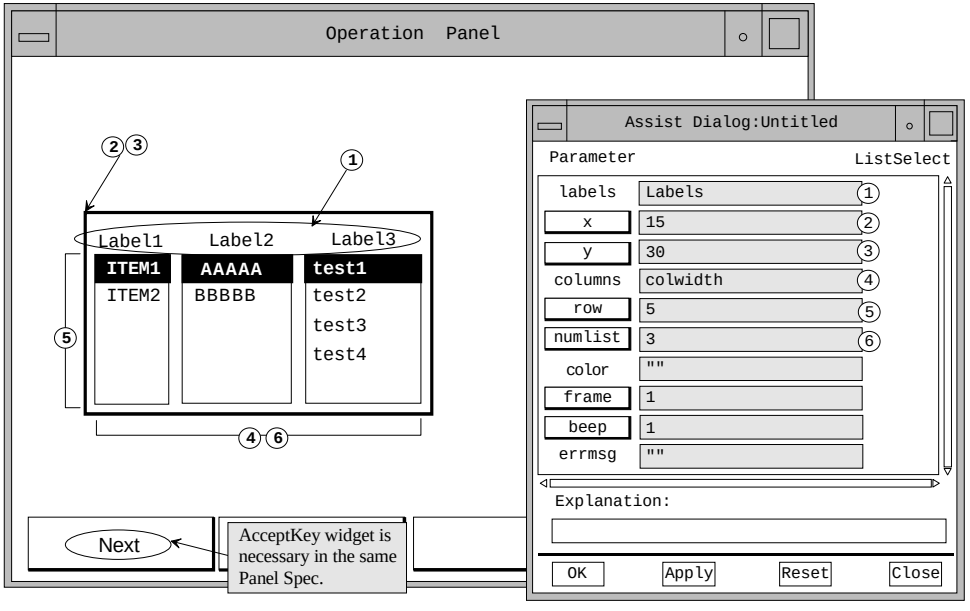
parms Values to be set using Assist dialog. Refer to “Parameters”.

register STRING[ ] type tag variable name to store the selected list items. Must be defined in Tag Spec. Array size determines the maximum number of selectable items. For example, if a STRING[3] type tag variable is used here, a maximum 3 items can be selected. If 4 or more selected, selection denied error occurs.

items STRING[ ] type tag variable to set items. Refer to “Item definition”.

message Message appears in lower left of panel when the widget is selected.

Parameters



This figure is an example of ListSelect widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
labels	STRING[ ] type tag variable to set column labels.
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
columns	INTEGER[ ] type tag variable to set width (number of characters) of each column.
row	Number of lines of list
numlist	Number of columns of list
color	Color for list. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
beep	Beep on error (0:no beep, 1:beep)
errmsg	Message for input error
font	Font for widget.

Item Definition

Items to be listed are set using STRING[ ] type tag variable in “Range” column. Assign strings to the tag variable using slash (/). For example, to define a hierarchy of lists in the figure of previous page, the following strings should be set.

items[1]=”aaa/bbb/test1”, items[2]=”aaa/bbb/test2”, items[3]=”aaa/ccc/test3”
items[4]=”aaa/ccc/test4”, items[5]=”aaa/ccc/test5”, items[6]=”aaa/ddd/test6”

Refer to “Panel Spec Syntax” in previous page.

Operation

When you click on an item in left list area, the item is highlighted, and contents of item are displayed in the list area to the right.

When you get to the right-most list area, you can click to select multiple items in the list area. The maximum number of selectable items is limited as described later. If you select more than the maximum number of items, error message appears and warning beep sounds. So, if you want to select other items, you need to click and unselect some items first.If list area is too small to display all the items, a vertical scroll bar appears.

If you use this widget type, you must also specify `AcceptKey` widget type in the same specification so that panel stays open until you input the data.

RadioSelect

Displays radio buttons that can be used to select desired items.

Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**RadioSelect**	*parms*	*register*	*labels*	*message*

name A label of the line. Name must be unique in the framework.

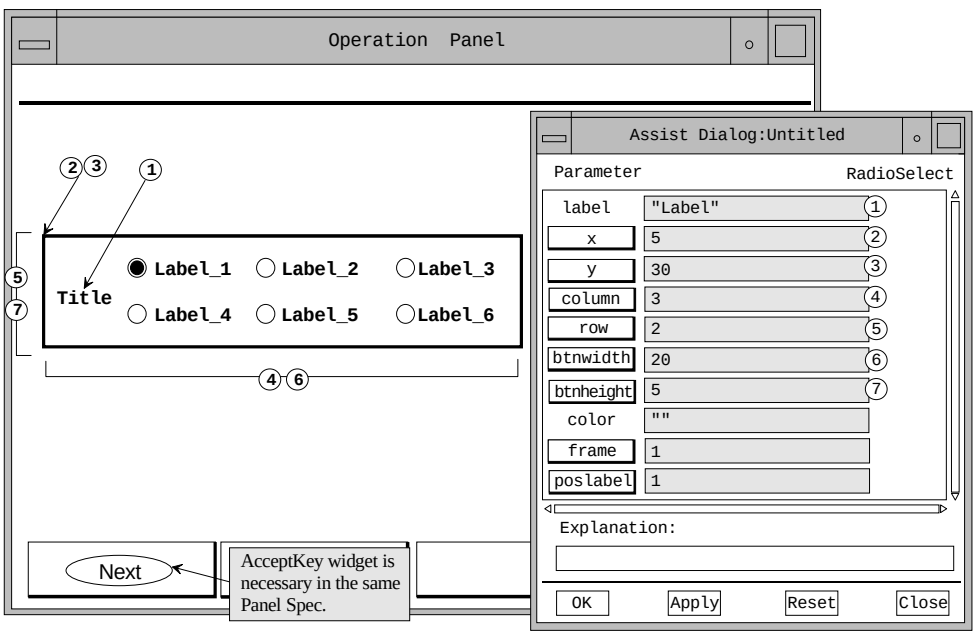
parms Values to be set using Assist dialog. Refer to “Parameters”.

register STRING[ ] type tag variable to store the selected button labels (for LANG=C) or message file name (for LANG=japanese). Array size determines the maximum number of selectable buttons.Only the first array index is valid.

labels STRING[ ] type tag variable to set button labels.

message Message appears in lower left of panel when the widget is selected.

Parameters



This figure is an example of RadioSelect widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
label	Button area title.
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
column	Number of buttons (horizontal)
row	Number of buttons (vertical)
btnwidth	Button width (relative size: 1 - 100 %)
btnheight	Button height (relative size: 1 - 100 %)
color	Background color of widget. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
poslabel	Label position (1:left, 2:top, 3:right, 4:bottom)
alignment	Position of text (1:left, 2:center, 3:right)
font	Font for widget.

Operation

When you click on desired button, the button is selected and other buttons are unselected..

If you use this widget type, you must also specify `AcceptKey` widget type in the same specification so that panel stays open until you input the data.

ResetKey

Activates a softkey at bottom of panel. If you click this softkey, all settings are reset to same as when the panel first opened.

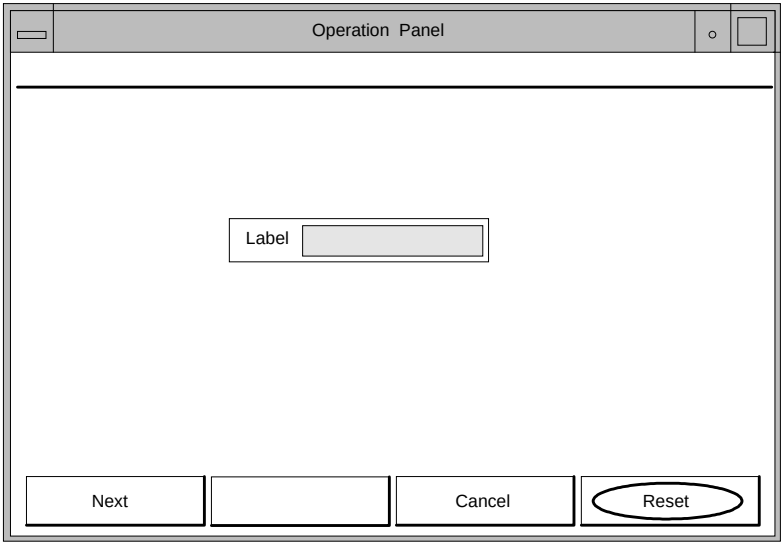
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**ResetKey**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

Parameters



Parameter	Description
label	Label for softkey
pos	Which softkey to use (1:left, 2:second, 3:third, 4:right)
fgcolor, bgcolor	Foreground (label) and background color for softkey. Set color using resource name of X window.
font	Font for widget.

Separator

Displays a separator line on the panel.

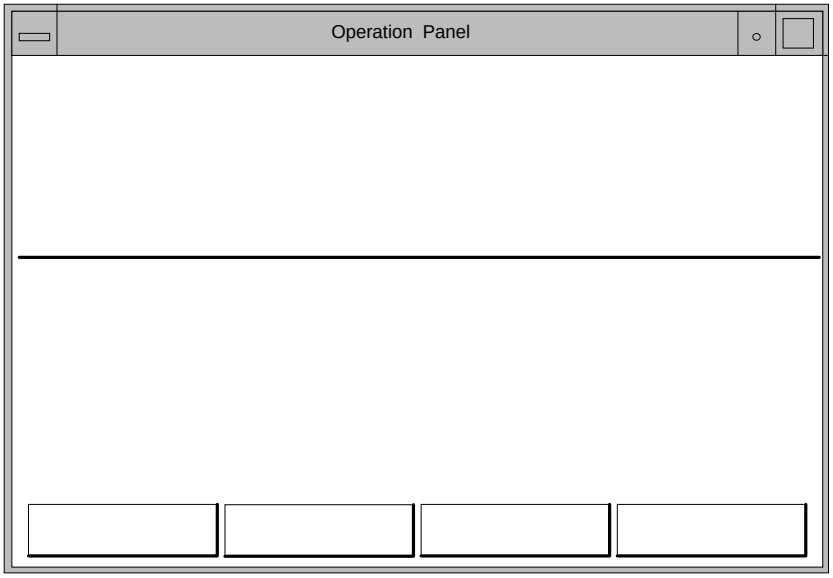
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**Separator**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

Parameters



Parameter	Description
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
length	Line length (relative size: 1 - 100 %)
orientation	Orientation for separator (1:horizontal, 2:vertical)
thick	Thickness of line. (2 - 32767)
margin	Margin around the line (0: no margin, 1:margin)

SingleDisp

Displays a CHARACTER, STRING, INTEGER, or REAL type data in a display field on the panel. Data is updated after each module (MODULE_END) if OP_AUTOUPDATE is set TRUE in /opt/SPECS/sys/config/sysconf file.
Also this contents is updated any time when UPDATE command is executed.

Panel Spec Syntax

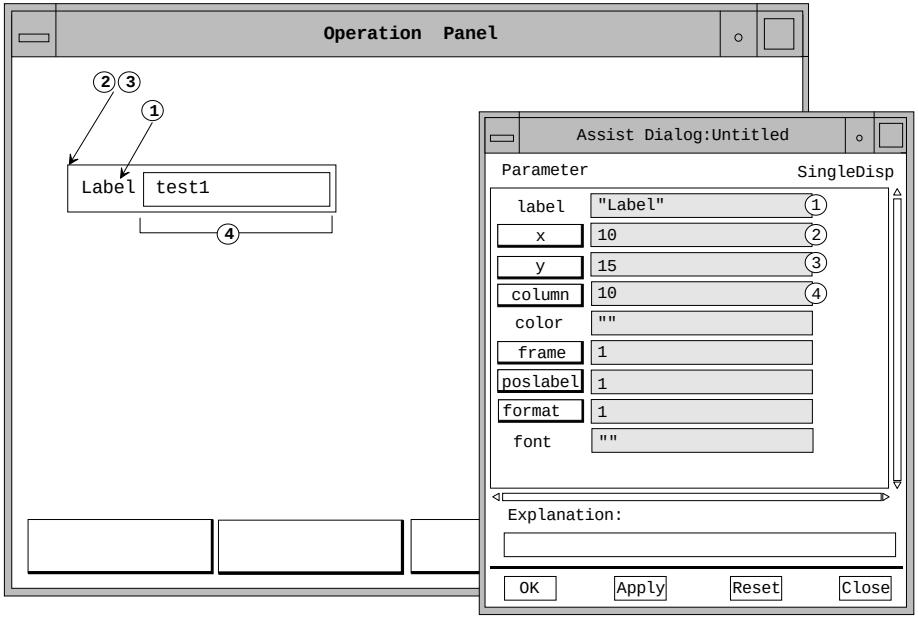
Widget Name	Widget type	Parameter	Variable	Range	Message
name	**SingleDisp**	*parms*	*register*		

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

register Tag variable name to set data to display.

Parameters



This figure is an example of SingleDisp widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
label	Label for field.
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
column	Width (number of characters) of field.
color	Color for input field. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
poslabel	Position of label (1:left, 2:top, 3:right, 4:bottom)
format	Display format for REAL type data (1:fixed point, 2:floating point, 3:fixed point with unit)
font	Font for widget.

SingleInput

Displays fields in which you can input a CHARACTER, STRING, INTEGER, or REAL type data from keyboard or bar-code reader.

Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**SingleInput**	*parms*	*register*	*range*	*message*

name A label of the line. Name must be unique in the framework.

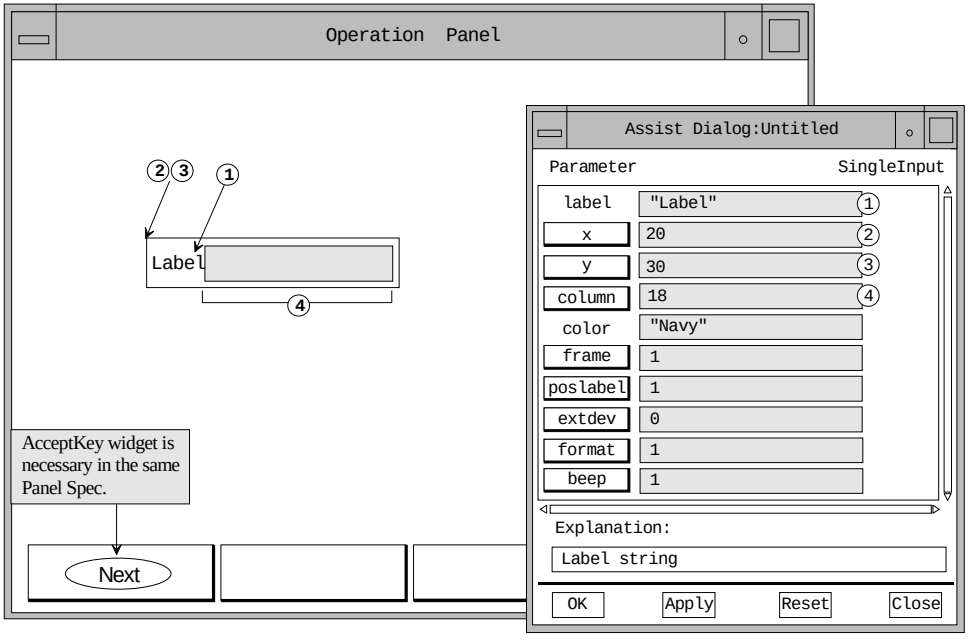
parms Values to be set using Assist dialog. Refer to “Parameters”.

register Tag variable name to store the value entered in panel. Must be defined in Tag Spec. When the Tag variable has an initial value, SingleInput displays the value in the input area.

range Range to check the entered value.

message Message appears in lower left of panel when the widget is selected.

Parameters



This figure is an example of SingleInput widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
label	Label for input field.
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
column	Width (number of characters) of input field.
color	Color for input field. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
poslabel	Position of label (1:left, 2:top, 3:right, 4:bottom)
extdev	Input device (0:keyboard, 1:Bar-code, 2:both)
format	Display format for REAL type data (1:fixed point, 2:floating point, 3:fixed point with unit)
beep	Beep on error (0:no beep, 1:beep)
errmsg	Message for input error
font	Font for widget.

Operation

Click the desired field and input a CHARACTER, STRING, INTEGER, or REAL type data from keyboard or bar-code reader.

For inputting REAL type value, you can use an exponent or unit.

To verify the input data, press the Return key or click the AcceptKey softkey. If input data is out of range, an improper value error occurs.

If you use this widget type, you must also specify `AcceptKey` widget type in the same specification so that panel stays open until you input the data.

NOTE

Initial Value in the Input Field

The initial value of Tag variable appears in the input field for following cases:

- Range column is specified and initial value is within the range
- Range column is *not* specified and initial value is specified

The minimum or first value of Range column appears in the input field in following case:

- Range column is specified and initial value is *not* specified or is out of range.

StatDisp

Displays the statistical and binning data on the panel.using fixed form This data is updated after each die (DIE_END) if OP_AUTOUPDATE is set to TRUE in /opt/SPECS/sys/config/sysconf file.
This Widget is not updated if OP_AUTOUPDATE is set to FALSE.

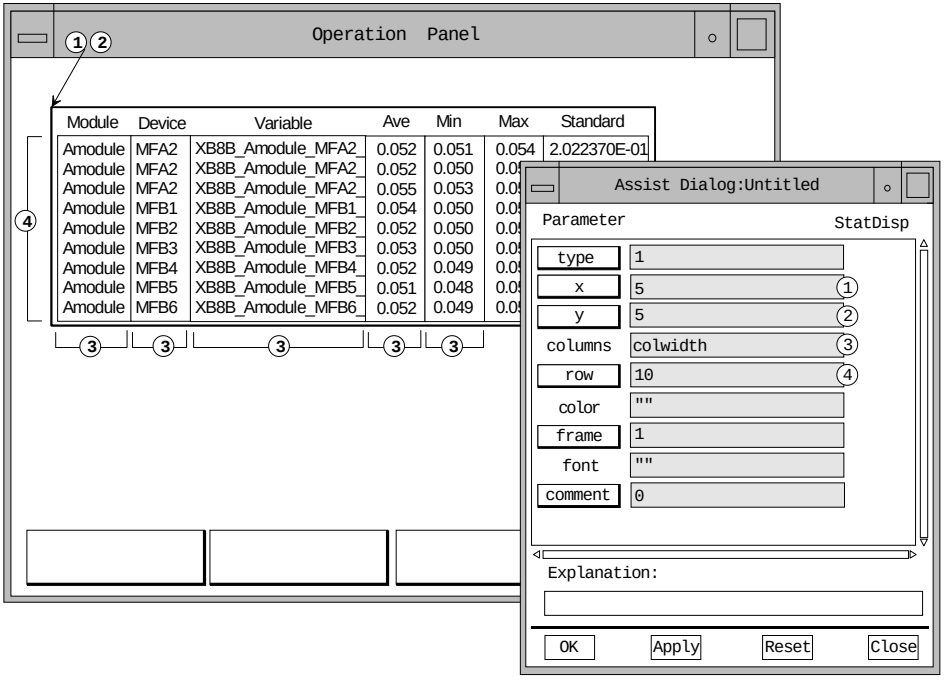
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**StatDisp**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

Parameters



This figure is an example of StatDisp widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
type	Display style (1:Statistical data, 2:Binning data)
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
columns	INTEGER[ ] type tag variable to set column width (number of characters) for list
row	Number of lines of list area (number of characters)
color	Color for list. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
font	Font for widget.
comment	Width (number of characters) for comment column

Figure 4-1 **Binning data style (type=2)**

Module	Device	Variable	1	2	3	4	5	6	7	8	9	10
Amodule	MFA2	XB8B_Amodule_MFA2_	0	9	0							
Amodule	MFA2	XB8B_Amodule_MFA2_	0	3	6							
Amodule	MFA2	XB8B_Amodule_MFA2_	0	9	0							
Amodule	MFB1	XB8B_Amodule_MFB1_	0	9	0							
Amodule	MFB2	XB8B_Amodule_MFB2_	0	9	0							
Amodule	MFB3	XB8B_Amodule_MFB3_	0	0	9							
Amodule	MFB4	XB8B_Amodule_MFB4_	0	9	8							
Amodule	MFB5	XB8B_Amodule_MFB5_	0	9	8							
Amodule	MFB6	XB8B_Amodule_MFB6_	0	6	6							

StatView

Displays the statistical and binning data on the panel. You can specify which items to display.

This display's content is updated any time the UPDATE command is executed.

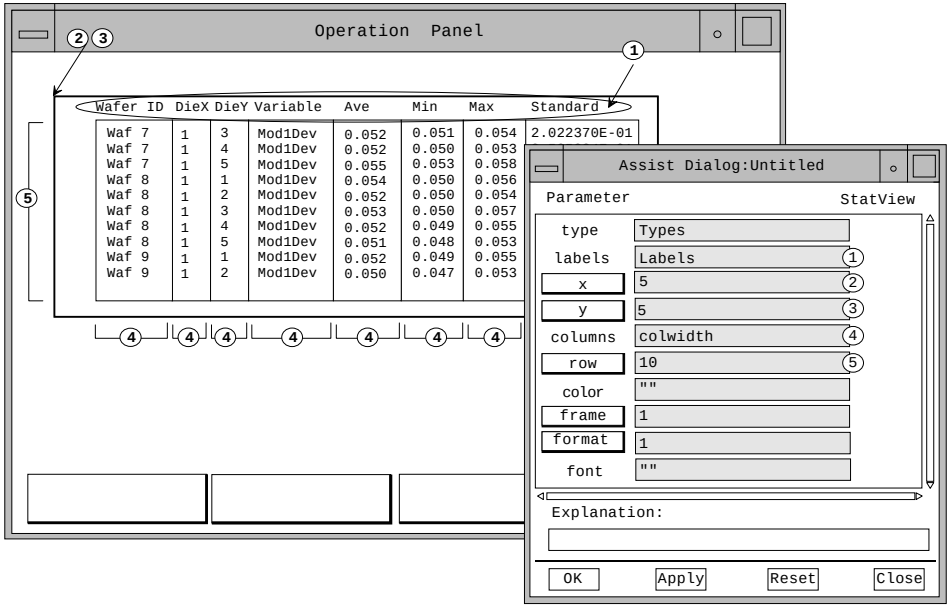
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**StatView**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to "Parameters".

Parameters



This figure is an example of StatView widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
type	STRING[ ] type tag variable to set format types to display. Available format types are listed in “Format type”.
labels	STRING[ ] type tag variable to set labels of columns.
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
columns	INTEGER[ ] type tag variable to set column width (number of characters) for list
row	Number of lines of list area (number of characters)
color	Color for list. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
format	Format to display decimal numbers. (1:fixed point, 2:floating point, 3:fixed point with unit)
font	Font for widget.

Format type

Type	Description	Type	Description
Wafer	Wafer ID	UMin1 - 6	Minimum limit of each User Binning Limit
Mod	Module Name	UMax1 - 6	Maximum limit of each User Binning Limit
Dev	Device Name	UStd1 - 6	Standard deviation of each User Bin
Alg	Algorithm Name	UBin1 - 6	Binning count of each User Bin
Name	Output variable name	JAve	Average value of “pass” status value
Yield	Yield	JMin	Minimum value of “pass” status value
Unit	Unit	JMax	Maximum value of “pass” status value
Judge	Pass/Fail judgement	JStd	Standard deviation of “pass” status value
Cmnt	Comment	JBin	Binning count of “pass” status value
RAve	Average	VAve	Average value of “valid” status value
RMin	Minimum value	VMin	Minimum value of “valid” status value
RMax	Maximum value	VMax	Maximum value of “valid” status value
RStd	Standard Deviation	VStd	Standard deviation of “valid” status value
RBin1 - 11	Binning count	VBin	Binning count of “valid” status value
UAve1 - 6	Average value of each User Bin		

TextDisp

Displays the text data on the panel. Text is updated after each module (MODULE_END) if OP_AUTOUPDATE is set TRUE in /opt/SPECS/sys/config/sysconf file.
Also this contents is updated any time when UPDATE command is executed.

Panel Spec Syntax

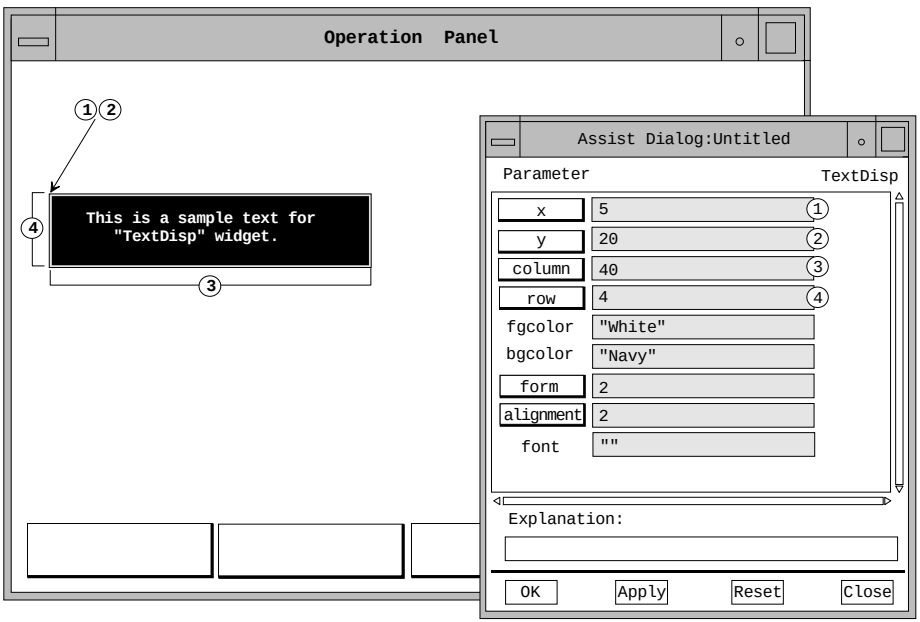
Widget Name	Widget type	Parameter	Variable	Range	Message
name	**TextDisp**	*parms*	*register*		

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

register Tag variable name to set text to display. For example, “This is a sample text for \n \”TextDisp\” widget” is assigned to the tag variable.

Parameters



This figure is an example of TextDisp widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
column	Width (number of characters) of field.
row	Height (number of lines) of field.
fgcolor	Foreground (text) color for input field. Set color using resource name of X window.
bgcolor	Background color for input field. Set color using resource name of X window.
form	-1:no frame, no margin, 0:no frame, margin, 1:shadow etched in 2:shadow etched out, 3:shadow in, 4:shadow out
alignment	Position of text (1:left, 2:center, 3:right)
font	Font for widget.

ToggleInput

Displays fields where you can input STRING type data from the keyboard. The toggle button for each field enables or disables the field.

Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**ToggleInput**	*parms*	*size, status*		

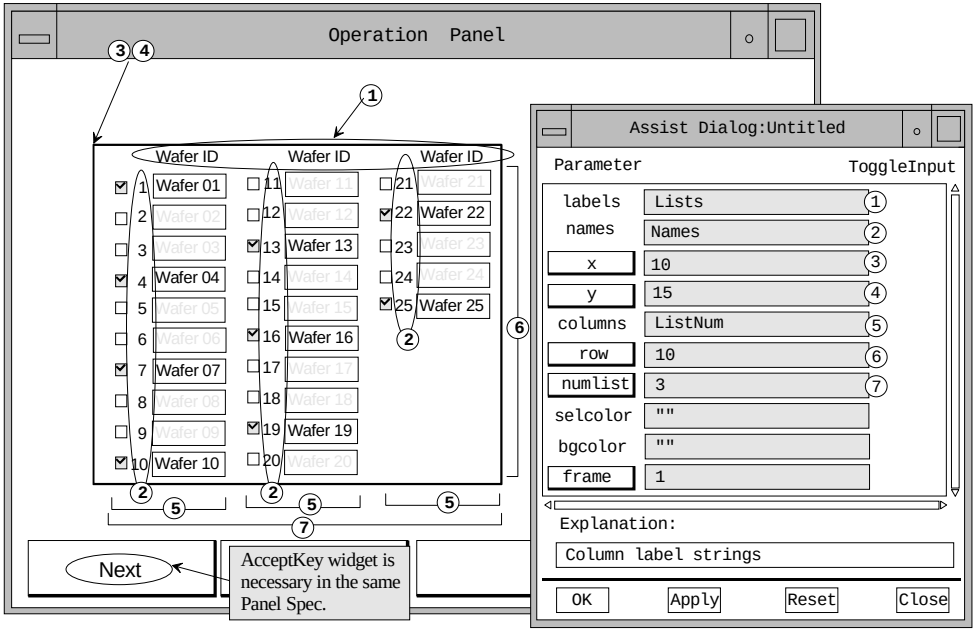
name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

size STRING[ ] [ ] type tag variable to set the button labels.

status INTEGER[ ] type tag variable to set and store the status of buttons. For details of status, refer to “Status”. Array size must be more than or equal to the number of rows.

Parameters



This figure is an example of ToggleInput widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
labels	STRING[ ] type tag variable to set column labels.
names	STRING[ ] type tag variable to set labels of each button.
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
columns	INTEGER[ ] type tag variable to set width (number of characters) of each column.
row	Number of fields by a column.
numlist	Number of columns
selcolor	Background color of selected items. Set color using resource name of X window.
bgcolor	Background color of widget. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
font	Font for widget.

Status

The table below lists valid status value to be used in “Variable” column. Refer to “Panel Spec Syntax” in previous page.

Status	Toggle button	Edit string	Value after clicking
-1	Unselected	OK	1
1	Selected	OK	-1
-2	Unselected	NG	2
2	Selected	NG	-2
-3	Unselected	NG	3
3	Selected	OK	-3
-4	Unselected	NG	Fixed
4	Selected	OK	Fixed

Operation

When you click on the desired toggle button, the button is selected or unselected and the toggle status changes. You can enter a string to the field. Refer to the Syntax and Row style blocks discussed on the next page.

If you use this widget type, you must also specify `AcceptKey` widget type in the same specification so that panel stays open until you input the data.

ToggleSelect

Displays toggle buttons that can be used to select desired items.

Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**ToggleSelect**	*parms*	*register*	*labels*	*message*

name A label of the line. Name must be unique in the framework.

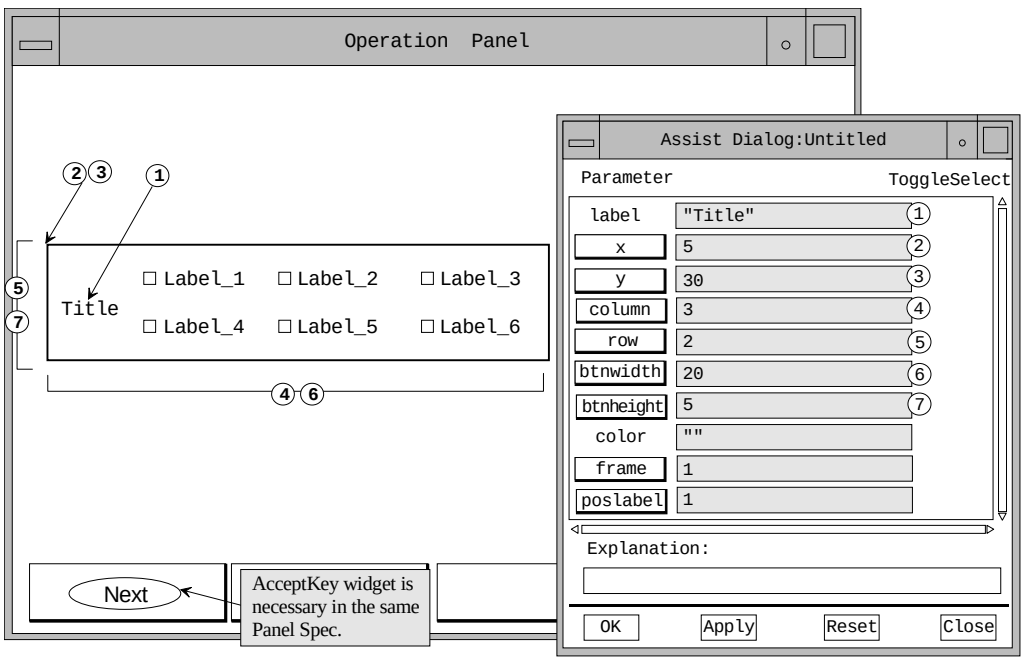
parms Values to be set using Assist dialog. Refer to “Parameters”.

register STRING[ ] type tag variable to store the selected button labels (for LANG=C) or message file name (for LANG=japanese). Array size determines the maximum number of selectable buttons.

labels STRING[ ] type tag variable to set button labels.

message Message appears in lower left of panel when the widget is selected.

Parameters



This figure is an example of ToggleSelect widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
label	Button area title.
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
column	Number of buttons (horizontal)
row	Number of buttons (vertical)
btnwidth	Button width (relative size: 1 - 100 %)
btnheight	Button height (relative size: 1 - 100 %)
color	Background color of widget. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
poslabel	Label position (1:left, 2:top, 3:right, 4:bottom)
alignment	Position of text (1:left, 2:center, 3:right)
beep	Beep on error (0: no beep, 1:beep)
errmsg	Message for input error.
font	Font for widget.

Operation

When you click on the desired toggle button, that button is selected or unselected and toggle status changes.

Multiple items can be selected. The maximum number of selectable buttons is limited, as described later in this chapter. If you select more than the maximum number of buttons, an error message appears and a warning beep sounds. In this case, you must deselect some items before selecting more.

If the button area is too small to display all of the buttons selected, a vertical scroll bar will appear.

If you use this widget type, you must also specify `AcceptKey` widget type in the same specification so that panel stays open until you input the data.

WaferDisp

Displays the wafermap which shows the die status on the wafer.

This display's content is updated any time the UPDATE command is executed

NOTE You can use this widget only in the Action Spec from LOOP_BEGIN to LOOP_END.

Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**WaferDisp**	*parms*	*status*	*legend*	

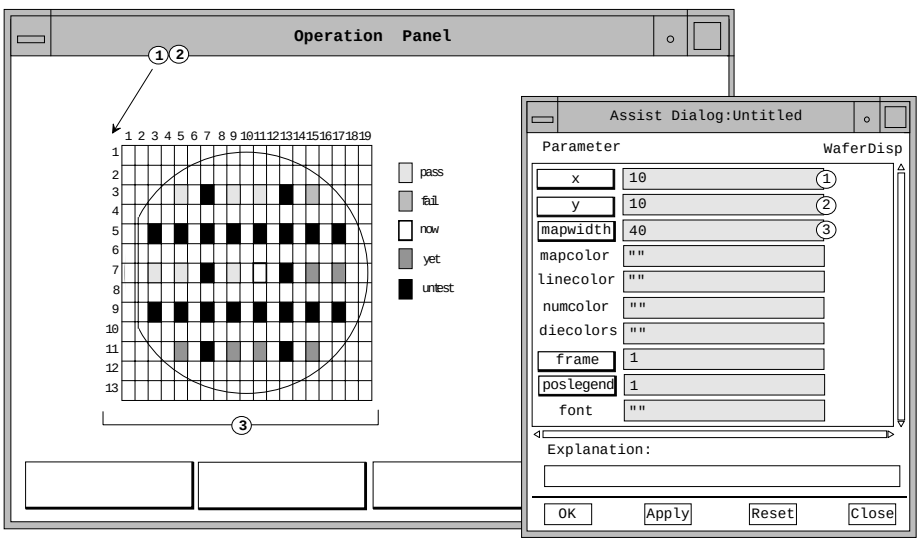
name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to "Parameters".

status STRING[ ] type tag variable to assign each die status which is specified by *legend*.

legend STRING[ ] type tag variable to set the die legend (Example: "pass", "fail"). Each item appears as a legend string and corresponds to the items for diecolors in the Parameter column.

Parameters



This figure is an example of WaferDisp widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
mapwidth	Wafermap width (relative size: 1 - 100 %)
mapcolor	Color for wafermap [a]
linecolor	Color for frame and axis line of wafermap [a]
numcolor	Color of number string for die [a]
diecolors	STRING[ ] type tag variable to set colors for each die status [a]
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
poslegend	Legend position (1:Right, 2:Bottom)
font	Font for list area
mapfont	Font for numbers on wafermap

a. Set color resource name for X window.

WaferSelect

Displays the wafermap and selects the die type and module name.

NOTE You can use this widget only in the Action Spec from LOOP_BEGIN to LOOP_END.

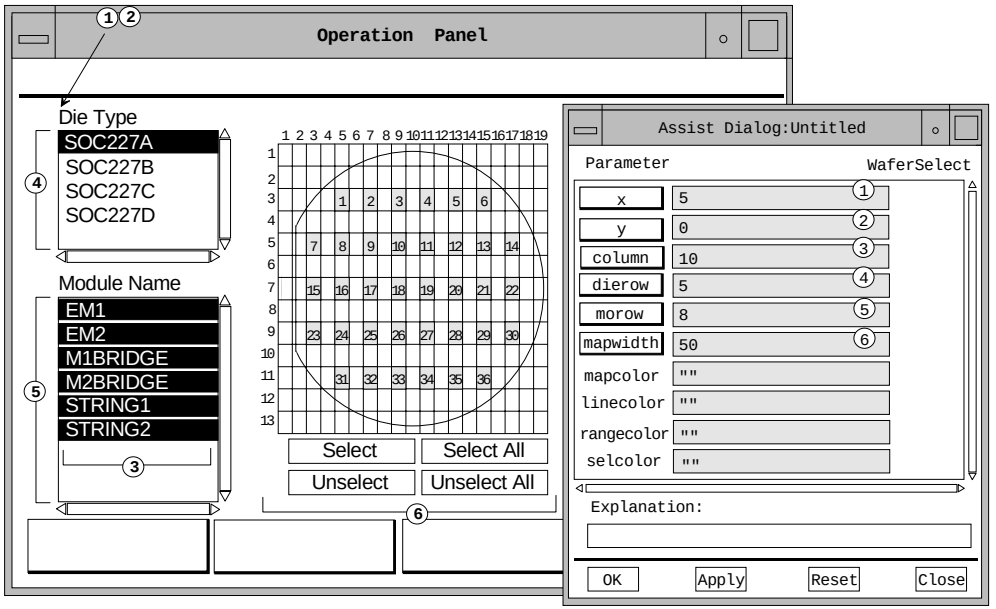
Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
name	**WaferSelect**	*parms*			

name A label of the line. Name must be unique in the framework.

parms Values to be set using Assist dialog. Refer to “Parameters”.

Parameters



This figure is an example of WaferSelect widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
column	Die, Module list area width. (Number of characters)
dierow	Number of lines of Die list area.
morow	Number of lines of Module list area.
mapwidth	Wafermap width (relative size: 1 - 100 %)
mapcolor	Background color of widget. [a]
linecolor	Color for frame and axis line of wafermap [a]
rangecolor	Color for dice in region [a]
selcolor	Color for selected dice [a]
unselcolor	Color for unselected dice [a]
donecolor	Color for dice which have already tested [a]
yetcolor	Color for dice which have not tested yet [a]
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
font	Font for list area
mapfont	Font for numbers on wafermap

a. Set color using resource name of X window.

Operation

Click on the desired die in the “Die Type” list area. The “Module Name” list area lists the modules in the selected dice and the wafermap displays the selected dice.
Click on the desired module in the “Module Name” list area to select or unselect modules.
Click on a desired die in the wafermap to select or unselect a die.
To select or unselect multiple dice, drag the mouse to highlight the dice, then click the “Select” or “Unselect” button.
To select or unselect all dice in the wafer, click “Select All” or “Unselect All”.

If you use this widget type, you must also specify `AcceptKey` widget type in the same specification so that panel stays open until you input the data.

WidgetSelect

Displays radio buttons that correspond to the widget name specified in the same panel specification. Only one item may be selected.

The selected widget becomes visible on the panel, while others become hidden. However, the widget types ActionKey, AcceptKey, CancelKey, ResetKey and WidgetSelect are ignored by WidgetSelect and will always be visible.

Panel Spec Syntax

Widget Name	Widget type	Parameter	Variable	Range	Message
*name*1	**WidgetSelect**	*parms*	*default*	*labels*	*message*
name2	*type2*	*parm2*			
name3	*type3*	*parm3*			

name1,2,3 A label of the line. Name must be unique in the framework. *name2,3* must be assigned to “candidate” parameter.

parms Values to be set using Assist dialog. Refer to “Parameters”.

default STRING[ ] type tag variable to set the selected widget initially. Only the first array index is valid.

labels STRING[ ] type tag variable to set button labels.

message Message appears in lower left of panel when the widget is selected.

Operation

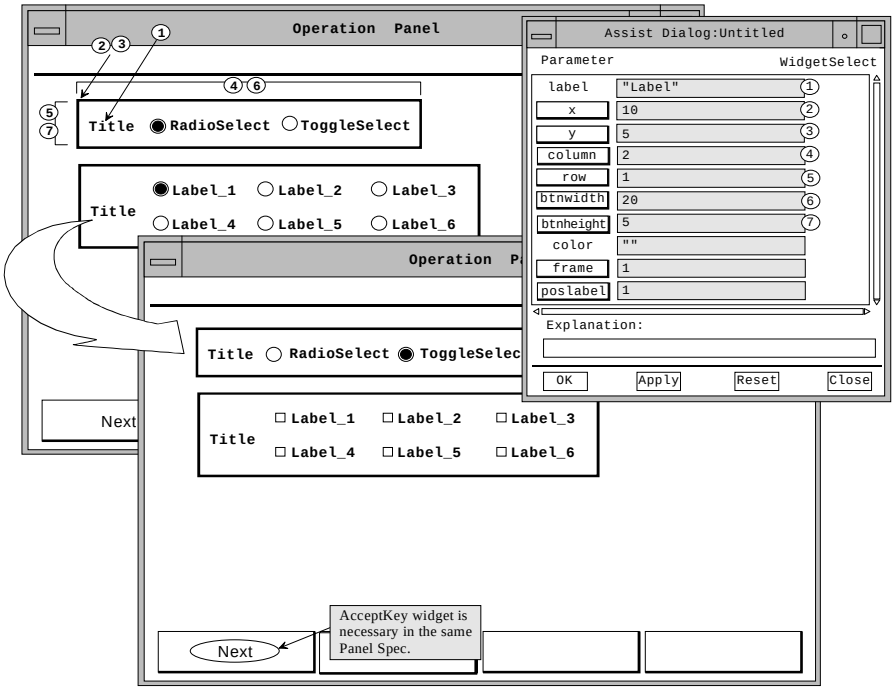
When you click on the desired radio button, the button is selected and the toggle status changes.

You can select only one item. When you select one button, the others revert to an unselected status.

If the button area is too small to display all the buttons, a vertical scroll bar will appears.

If you use this widget type, you must also specify `AcceptKey` widget type in the same specification so that panel stays open until you input the data.

Parameters



This figure is an example of Widget widget. This example is defined using parameters and values in the Assist dialog.

Parameter	Description
label	Button area title.
x,y	X,Y location of upper left of the widget. (Relative value: 0 - 100)
column	Number of buttons (horizontal)
row	Number of buttons (vertical)
btnwidth	Button width (relative size: 1 - 100 %)
btnheight	Button height (relative size: 1 - 100 %)
color	Background color of widget. Set color using resource name of X window.
frame	-1:no frame, no margin, 0:no frame, margin, 1:frame
poslabel	Label position (1:left, 2:top, 3:right, 4:bottom)
alignment	Position of text (1:left, 2:center, 3:right)
font	Font for widget.
candidate	STRING[ ] type tag variable to set callable widget names.

Example This procedure is an example to define a panel as in the figure in previous page.

Step 1. Define a new Panel Spec, “Widget_Select”, as below.

Widget Name	Widget type	Parameter	Variable	Range
ws1	WidgetSelect	*Refer to the table below*	default	labels1
ws2	RadioSelect	*Refer to the table below*	register1	labels2
ws3	ToggleSelect	*Refer to the table below*	register2	labels3
ws4	AcceptKey	Label=”Next”		

Parameter	WidgetSelect (ws1)	RadioSelect (ws2)	ToggleSelect (ws3)
label	“Title”	“Title”	“Title”
x,y	(10, 5)	(10, 20)	(10, 20)
column	2	3	3
row	1	2	2
btnwidth	10	10	10
btnheight	5	5	5
candidate	items		

Step 2. Define tag variables in “SYSTEM” class Tag Spec as below.

Tag Name	Type	Unit	Value	Range	Comment
labels1	STRING[2]				
labels2	STRING[6]				
default	STRING[1]				
register1	STRING[1]				
register2	STRING[6]				
items	STRING[1]				

Step 3. Set labels for widget, selected widget initially and selectable widget names to the tag variables using LET command. And call the “Widget_Select” panel. Add the following lines to the System Spec.

Branch Label	Command	Parameter	Comment
	LET	labels1[1]=”RadioSelect”	
	LET	labels1[2]=”ToggleSelect”	
	LET	labels2[1]=”Label_1”	
	LET	labels2[2]=”Label_2”	
	LET	labels2[3]=”Label_3”	
	LET	labels2[4]=”Label_4”	
	LET	labels2[5]=”Label_5”	
	LET	labels2[6]=”Label_6”	
	LET	default[1]=”ws2”	
	LET	items[1]=”ws2”	
	LET	items[2]=”ws3”	
	PANEL	“Widget_Select”	

5 TPL Command Reference

As described in Chapter 2, “Using Outliner,” on page 1, you define test plan and framework specifications by using the Spec Editor.

This chapter describes the data types, variables, expressions, and test plan commands that you can use to define the specifications.

- “Data Type” Describes the data types that you can use to declare variables in specifications.
- “Variable Range” Describes the syntax for setting the range or binning limits for a variable of a specification.
- “Variables” Describes how to use various Tag variables, output variables of algorithm, and System variables in a specification.
- “Expressions” Describes the operators you can use in expressions for some columns of a specification, such as for assigning parameter or variable values or for determining branch conditions of a test plan command.
- “Test Plan Commands” Describes the test plan commands you can use in some specifications.

Data Type

The following describes the data types that you can use to declare variables in specifications, such as in Type column of Tag specification or Var Type column of Algorithm specification.

Var Type	Description
REAL	For numeric floating point data: -3.402823E38 to +3.402823E38
INTEGER	For an integer: -2147483648 to +2142483647
CHARACTER	For 1 byte character
STRING	For characters. Maximum 255 bytes.
VAR1	REAL type one-dimensional array (maximum size: 1024). Available only for the “Output” variable of algorithm specification. VAR1 type variable is for storing primary sweep output values, and for variable that is X-axis data of X-Y Graph. Only one VAR1 type variable can be used in an algorithm specification.
VAR2	REAL type one dimensional array (maximum size: 128). Available only for the “Output” variable of algorithm specification. VAR2 type variable is for storing secondary sweep output values, and can only be used for algorithms that use VAR1 type variable. Only one VAR2 type variable can be used in an algorithm specification.

- For REAL type variables, you can use the following unit specifiers instead of an exponent expression:

```
P: 10E15        m: 10E-3
T: 10E12        u: 10E-6
G: 10E9         n: 10E-9
M: 10E6         p: 10E-12
k: 10E3         f: 10E-15
```

  Any characters after these units are ignored. For example: 10pA = 10p = 10E-12
  50mV = 10m = 50E-3

- CHARACTER or STRING type data must be enclosed by double quotes. To use a double quote as a character, put \ before the double quote.

- Following escape commands can be used for STRING type data:

```
New line character:    \n
Tab character:         \t
```

- INTEGER, REAL, STRING and CHARACTER type arrays can be used.
  - Data size is fixed when data is declared

```
Example:
Num = INTEGER[3]      - Num[1], Num[2], Num[3]
Message = STRING[2]   - Message[1], Message[2]
R = REAL[2][3]        - R[1][1], R[1][2], R[1][3]
                          R[2][1], R[2][2], R[2][3]
```

  - Maximum size is 1024 (for a one dimensional array), and 128 x 1024 (for a two dimensional array).
  - For copying arrays or other array operations, you can point to a whole array or levels of an array as in following examples:

```
A = REAL[2][2]

A[1][1]  > A[1][1]
A[1]     > A[1][1], A[1][2]
A        > A[1][1], A[1][2], A[2][1], A[2][2]
```

  - Contents of array can be copied to another array. If their sizes are different, overflow data is ignored.

```
Example:
A = INTEGER[2]   A[1] = A[2] = 0
B = INTEGER[3]   B[1] = B[2] = B[3] = 1

LET A = B  -> A[1] = A[2] = 1
              B[1] = B[2] = B[3] = 1

LET B = A  -> A[1] = A[2] = 0
              B[1] = B[2] = 0, B[3] = 1
```

Variable Range

This section describes the syntax of setting the range or binning limits for a variable of a specification, such as in Range column of the Algorithm specification, Tag specification, or Panel specification, or Output column of Test specification:

- **For setting a continuous range:**

  To define a valid range for REAL type or INTEGER type variables, you can use the ~ (tilde) as in the following example:

  ```
  50~100
  ```

  The above example means allowed *variable* values are: $50 \leq variable \leq 100$

- **For setting specific values for a range:**

  For the range of REAL, INTEGER, CHARACTER, or STRING type variables, you can also set specific values as in following example:

  ```
  "A","B","C"
  ```

  This example means the allowed variable values are `A`, `B`, and `C`.

- **For setting the binning limits for output variable:**

  You specify these binning values for the output variable definition in a test specification or algorithm specification.

  You can specify a maximum 10 binning values for checking or binning the algorithm results. Two values define one bin, so including two out-of-range bins, you can have maximum 11 bins.

  Test report commands (`lot_sum` and `waf_sum`) use the specified binning limits for statistical calculation. All measurement result data are viewable, but the out-of-range data are not used in the statistical calculations.

  For an output variable, you define the binning limits for a REAL or INTEGER type variable by using the ~ (tilde) as in following example:

  ```
  0~1~2~3~4~5
  ```

  This example defines the following binning limits:

  - Out of range ≤0
  - 0< `Bin 1` ≤1
  - 1< `Bin 2` ≤2

- 2< `Bin 3` ≤3
- 3< `Bin 4` ≤4
- 4< `Bin 5` ≤5
- Out of range >5

In this example, data ≤ `0`, and data > `5` are *not* used for statistical calculation.

If you want this data to be used in statistical calculations, define as follows:

```
~0~1~2~3~4~5~
```

Variables

The following describes how to use various Tag variables, output variables of algorithm, and System variables in a specification:

Tag Variable:

- What's a Tag variable?

  Tag variables are global variables that can be referred to from all specifications, and are very useful for transferring the parameters for commands.

  There are two types of Tag variables: pre-defined by HP SPECS (see below) and user-defined, which you define in the tag specification.

  User-defined Tag variables are saved to the lot test data file. However, if you enclose a variable with `{ }`, variable is not saved to file.

  Tag variables have the following four classes:

SYSTEM	Information of test plan and global variables.
LOT	Information of lot. When a lot ends, LOT classed tag variables are saved to lot test data file, then set to initial value that is assigned in Tag specification.
WAFER	Information of wafer. When a wafer ends, WAFER classed tag variables are saved to lot test data file, then set to initial value that is assigned in Tag specification.
DIE	Information of die. When a die ends, DIE classed tag variables are saved to lot test data file, then set to initial value that is assigned in Tag specification.

- Syntax to refer to Tag variable

  TagVariableName: `for SYSTEM class Tag variable`

  TagClass`.`*TagVariableName*: `for getting information about present lot, wafer, and die.`

  For example, `WAFER.SYS_DIECOUNT` means number of dice already tested on present wafer.

- Pre-defined Tag variables

  The following Tag variable names are pre-defined in HP SPECS, and are saved to the lot test data file:

Table 5-1

Class	Name	Type	Description
SYSTEM	SYS_USERID	STRING	User ID
	SYS_CASSID	STRING	Cassette ID
	SYS_PRODID	STRING	Product ID
	SYS_LOTID	STRING	Lot ID
	SYS_TESTID	STRING	Test Plan ID
WAFER	SYS_WAFERID	STRING	Wafer ID
	SYS_TESTSTART	STRING	Test start time
	SYS_TESTEND	STRING	Test end time
	SYS_DIECOUNT	INTEGER	Number of dice tested until now
	SYS_WAFERPASS	INTEGER	Pass(1)/Fail(0) result of wafer
	SYS_SLOTID	STRING	Slot ID
DIE	SYS_DIEPASS	INTEGER	Pass(1)/Fail(0) result of die

Output Variable:

- What's an output variable?

  Output variables are for storing the results of algorithm defined in Algorithm specification. The output variable name determines the default name of test data variable that is stored in the lot test data file:

  DieName_ModuleName_DeviceName_AlgorithmName_OutputVariableName

  If you do not want to use this default test data variable name, you can specify desired name in the Test specification.

  If you enclose the output variable name by { } in Algorithm specification, output variable data will not be saved to lot test data file.

  Also, you can enclose the algorithm name by { } in the Test specification, so the algorithm output is not saved to the lot test data file.

- Syntax for referring to an output variable

  You can refer to an output variable by using following syntax:

  AlgorithmName`.`*OutputVariableName*
  `(refers to latest output variable value for the algorithm)`

  `DEVICE.`*AlgorithmName*`.`*OutputVariableName*
  `(refers to the latest output variable for present device)`

  `MODULE.`*[DeviceName.]AlgorithmName*`.`*OutputVariableName*
  `(refers the latest output variable for present module)`

`DIE.`*`[ModuleName.][DeviceName.]AlgorithmName.OutputVariableName`
`(refers the latest output variable for present die)`

For example,

- `Leak.current` means latest `current` variable value of `Leak` algorithm.
- `DEVICE.Leak.current` means `current` variable value of `Leak` algorithm for present device.
- `DIE.Mod1.FET1.Leak.current` means `current` variable value of `Leak` algorithm for `FET1` device on `MOD1` module of present die.

System Variable:

- What's a System variable?

  A System variable stores the information of the executing test plan and can be referred to from the specifications of test plan. You *cannot* assign new System variable values while the test plan is executing. Some system variable values change while the test plan is executing.

- The System variables pre-defined by HP SPECS are as follows:

Name	Type	Description
SYSTEM.TESTPLAN	STRING	Present test plan name (no path/suffix)
SYSTEM.FRAMEWORK	STRING	Present framework name (no path/suffix)
SYSTEM.ALGORITHM	STRING	Present algorithm library name (no path/suffix)
SYSTEM.RETEST	STRING	Most recently executed RETEST parameter(Ex. WAFER) *1
SYSTEM.REJECT	STRING	Most recently executed REJECT parameter(Ex. LOT) *1
SYSTEM.SESSIONID	STRING	Present SPECS "Session ID" ranging from 1 to 4 which identifies Editor/Navigator
SYSTEM.PROCESSID	STRING	HP-UX "Process ID" of Test Plan Interpreter
SYSTEM.USERID	STRING	HP-UX "User ID"
SYSTEM.VERSIONID	STRING	SPECS version number

Name	Type	Description
SYSTEM.SYSTEMID	STRING	HP-UX hostname
SYSTEM.ERRORNUMBER	INTEGER	Most recent TPL command error number
SYSTEM.ERRORMESSAGE	STRING	Most recent TPL command error message
WAFER.NAME	STRING	Name of effective wafer specification *1
WAFER.NUMBER	INTEGER	Number of wafers tested (Note: RETEST WAFER does not increment count) *1
WAFER.MINX	INTEGER	Minimum die X index *1
WAFER.MAXX	INTEGER	Maximum die X index *1
WEAFR.MINY	INTEGER	Minimum die Y index *1
WAFER.MAXY	INTEGER	Maximum die Y index *1
WAFER.LOADLIST	STRING[]	List of wafer specifications defined in test plan
WAFER.LOADCOUNT	INTEGER	Number of wafer specifications defined in test plan
DIE.NAME	STRING	Present die name *1
DIE.INDEXX	INTEGER	Present die X index *1
DIE.INDEXY	INTEGER	Present die Y index *1
DIE.POSX	REAL	Present die X position in microns *1
DIE.POSY	REAL	Present die Y position in microns *1
DIE.NUMBER	INTEGER	Present die count *1
DIE.TOTAL	INTEGER	Total number of dice in present die type (contiguous dice of same type) *1
DIE.TYPENUMBER	INTEGER	Present die type count *1
DIE.TYPETOTAL	INTEGER	Total number of die types in present wafer *1
DIE.LOADLIST	STRING[]	List of die specifications defined in test plan
DIE.DEFINELIST	STRING[]	List of die names defined in present wafer *1

Name	Type	Description
DIE.SELECTLIST	STRING[]	List of die names selected in present wafer *1
DIE.LOADCOUNT	INTEGER	Number of die specifications defined in test plan
DIE.DEFINECOUNT	INTEGER	Number of die names defined in present wafer *1
DIE.SELECTCOUNT	INTEGER	Number of die names selected in present wafer *1
MODULE.NAME	STRING	Present module name *1
MODULE.POSX	REAL	X distance (microns) of module from die origin *1
MODULE.POSY	REAL	Y distance (microns) of module from die origin *1
MODULE.NUMBER	INTEGER	Present module count in all test specifications which are assigned to present die *1
MODULE.TOTAL	INTEGER	Total number of modules in all test specifications which are assigned to present die *1
MODULE.TYPENUMBER	INTEGER	Present module count in present die *1
MODULE.TYPETOTAL	INTEGER	Total number of modules defined in present die *1
MODULE.DEFINELIST	STRING[]	List of defined module names in present test specification *1
MODULE.SELECTLIST	STRING[]	List of selected module names in present test specification *1
MODULE.DEFINECOUNT	INTEGER	Number of defined module names in present test specification *1
MODULE.SELECTCOUNT	INTEGER	Number of selected module names selected in present test specification *1
DEVICE.NAME	STRING	Present device name *1
DEVICE.NUMBER	INTEGER	Present device count in present module *1
DEVICE.TOTAL	INTEGER	Total number of defined devices under the present module. *1

Name	Type	Description
PROBE.NAME	STRING	Name of effective probe specification *1
PROBE.LOADLIST	STRING[]	List of probe specifications defined in test plan
PROBE.LOADCOUNT	INTEGER	Number of probe specifications defined in test plan
TEST.NAME	STRING	Present test specification name *1
TEST.NUMBER	INTEGER	Present test count for present die *1
TEST.TOTAL	INTEGER	Total number of tests assigned to present die *1
TEST.LOADLIST	STRING[]	List of test specifications defined in test plan *1
TEST.DEFINELIST	STRING[]	List of test specifications assigned to dice in the test plan. (Before die measurement starts) List of test specifications assigned to present die. (After die measurement starts)
TEST.SELECTLIST	STRING[]	List of test specifications selected and assigned to dice in the test plan. (Before die measurement starts) List of test specifications selected and assigned to present die.(After die measurement starts) *1
TEST.LOADCOUNT	INTEGER	Number of test specifications defined in test plan *1
TEST.DEFINECOUNT	INTEGER	Number of test specifications assigned to dice in the test plan. (Before die measurement starts) Number of test specifications assigned to present die. (After die measurement starts) *1

Name	Type	Description
TEST.SELECTCOUNT	INTEGER	Number of test specifications selected and assigned to dice in the test plan. (Before die measurement starts) Number of test specifications selected and assigned to present die.(After die measurement starts) *1
PANEL.RETURN	STRING	Widget name (specified by AcceptKey or CancelKey widget) is returned to this variable when operation panel closes.
REPORT.DIR	STRING	Directory path where lot test data file is stored
REPORT.FILE	STRING	File name of present lot test data file

*1 Has meaning only while LOOP command is running.

Expressions

This section describes the operators you can use in expressions for some columns of a specification.

For example, to define the SIZE attribute of an 8 inch wafer, you can use `8*25.4` instead of `203.2` in Value field of wafer specification.

For another example, if you use REPEAT — UNTIL command in a test specification, you can use an expression, such as `Leakchk.Igsd >= -14`, in the Input column as a branch condition.

Table 5-2 **Arithmetic operators**

Operator	Function
+	addition
-	subtraction
*	multiplication
/	division
%	remainder (ex. 5 % 2 = 1)

Table 5-3 **Assignment operator**

Operator	Function
=	assignment equal

Table 5-4 **Relational operators**

Operator	Function
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
=, ==	equal to
<>, !=	not equal to

Table 5-5 **String operators**

Operator	Function
+	combines 2 strings into 1

Table 5-6 Logical operators

Operator	Function
NOT	logical opposite
AND	logical AND
OR	logical OR

Examples:

```
If=50mA
T1=21
Leakchk.Igsd >= -14
i=i+1
B="Pass"
PANEL.RETURN=="ONLINE"
(Leak.current<5n) OR (i>10)
8*25.4
OUTPUTFILE="/tmp/"+SYS_WAFERNAME+WRITE_INTEGER.String
"/CMOS/"+SYS_WAFERNAME+"/"+SYS_LOTID+"/"+SYSTEM.TESTPLAN
```

Test Plan Commands

Test plan commands can be used in Job, System, Action, and Test specifications. See note below.

NOTE

Test plan commands can be used in Job, System, Action, and Test specifications.

In the reference for each Test plan command, the column headers of **Syntax** table indicate which specifications can use the command.

For example, the following Syntax table indicates that the IF command can be used in all four specification types: Job, System, Action, and Test specifications.

Syntax

Job/System Action Test	Branch Label Event Label	Command Command Algorithm	Parameter Parameter Input	Comment Comment Comment
	[label]	IF statement1 [ELSE] [statement2] END IF	expression	[comment] [comment] [comment] [comment] [comment]

But following example indicates the LOOP command can be used in only the Job specification.

Syntax

Job	Branch Label	Command	Parameter	Comment
	[label]	LOOP	[count]	[comment]

ACTION

This command calls an Action Spec like a sub-routine.

Syntax

Job/System Action Test	Branch Label Event Label	Command Command Algorithm	Parameter Parameter Input	Comment Comment Comment
	[label]	ACTION	act_spec	[comment]

label line label (optional)

act_spec Action Spec name

BREAK

This command exits the present loop of REPEAT or WHILE, and goes to next line after end of loop.

Syntax

Job/System Action Test	Branch Label Event Label	Command Command Algorithm	Parameter Parameter Input	Comment Comment Comment
	[label]	BREAK		[comment]

label line label (optional)

Example (Test Spec)

Module	Device	Algorithm	Input
Mod1	Dev1		
		LET	i=0
		REPEAT	
		LET	i=i+1
		Leak	
		IF	Leak.current < 2n
		BREAK	
		END IF	
		UNTIL	i > 10

DATA

This command specifies the full directory path in which to save the lot test data file. Optionally you can specify the file name.

This command can be executed in job specification or system specification or with WAFER_BEGIN in action specification.

Syntax

Job/System	Branch Label	Command	Parameter	Comment
	[label]	DATA	“path”[,”file”]	[comment]

label line label (optional)

path Path name (STRING expression)

file Data file name (optional)
Default is hostname.mmddyy
(mm: month, dd: data, yy: year)
Suffix “.adt” is added automatically and you cannot change this.

comment maximum 64 characters (optional)

Example (System Spec)

Branch Label	Command	Parameter
	DATA	“/CMOS/test1”

You can use + to specify variables as in following example:

```
"/SPECS/usr/spool/data/"+SYS_WAFERNAME+"/"+SYS_LOTID+"/"+SYSTEM.T
ESTPLAN
```

EXECUTE

This command executes the specified HP-UX command.

Syntax

Job/System Action Test	Branch Label Event Label	Command Command Algorithm	Parameter Parameter Input	Comment Comment Comment
	[label]	EXECUTE	“command”	[comment]

label line label (optional)

command HP-UX command and its parameters (STRING expression). Cannot use special operators for HP-UX shell, such as * and >.

comment maximum 64 characters (optional)

Example (Job Spec)

Branch Label	Command	Parameter
	EXECUTE	“pcsstart”

GOTO

This command jumps to the specified destination label.

Syntax

Job/System Action Test	Branch Label Event Label	Command Command Algorithm	Parameter Parameter Input	Comment Comment Comment
	[label]	GOTO	destination	[comment]

label line label (optional)

destination Destination line label.
The specified destination label must exist in the present specification.
In a Test Spec, you can use a Module or Device name as the destination label.
(For Device name, must be name in present module)

comment maximum 64 characters (optional)

Example (Test Spec)

Module	Device	Algorithm	Input
Mod1	Dev1		
		LET	i=0
		REPEAT	
		LET	i=i+1
		Leak	
		IF	Leak.current < 2n
		GOTO	Dev2
		END IF	
		UNTIL	i > 10
Mod1	Dev1		

IF - END IF

This command is a conditional branch.

Syntax

Job/System Action Test	Branch Label Event Label	Command Command Algorithm	Parameter Parameter Input	Comment Comment Comment
	[label]	IF statement1 [ELSE] [statement2] END IF	expression	[comment] [comment] [comment] [comment] [comment]

label line label (optional)

IF INTEGER expression. Refer to “Expressions” on page 14

statement1 Performed if expression is TRUE.

statement2 Performed if expression is FALSE.(optional)

END IF End of IF statement.

comment maximum 64 characters (optional)

Example (Test Spec)

Module	Device	Algorithm	Input
Mod1	Dev1		
		LET	i=0
		IF	Leak.current > 2n
		Alg1	
		ELSE	
		LET	Flg=1
		END IF	

Restrictions

- IF/.REPEAT/WHILE statements can be nested up to 256 levels.
- IF and END IF cannot be used across multiple devices.

LET

This command assigns values to variables.

Syntax

Job/System Action Test	Branch Label Event Label	Command Command Algorithm	Parameter Parameter Input	Comment Comment Comment
	[label]	LET	var=expr [,var=expr]	[comment]

label line label (optional)

var variable name

expr expression. Refer to “Expressions” on page 14.
Data type must be same type as variable.
For data type, refer to “Data Type” on page 3.
Enclose STRING or CHARACTER type with double quotes.

comment maximum 64 characters (optional)

Example (Test Spec)

Module	Device	Algorithm	Input
		LET	A=1, B=”Pass”
		LET	C=1.25

LOOP

This command starts executing the tests assigned for each selected die on the wafer, then repeats the specified times. For assigning tests, see “TEST”.

This command can be used only in a job specification.

One Wafer specification and one Probe specification must be selected to be effective before executing the LOOP command. For selecting, see “SELECT”.

Syntax

Job	Branch Label	Command	Parameter	Comment
	[label]	LOOP	[count]	[comment]

label line label (optional)

count Number of times to repeat the testing(optional).
If count is not defined, testing is repeated until REJECT LOT is executed.

comment maximum 64 characters (optional)

Example (Job Spec)

Branch Label	Command	Parameter
SELECT	WAFER,”Wafers”	
SELECT	PROBE,”P1”	
TEST	“Die1”,”T1”	
LOOP		

PANEL

This command displays the specified operation panel.

Syntax

Job/System Action Test	Branch Label Event Label	Command Command Algorithm	Parameter Parameter Input	Comment Comment Comment
	[label]	PANEL	“name”[,parm]	[comment]

label line label (optional)

name Name of Panel Spec (STRING type)

parm Parameter of Panel Spec. Must be integer type data:
0 , 1 or 2

- **0** Pop. Closes present Panel, then opens the saved Panel(see 2 below).
  Panel Spec name is ignored.
- **1** (Default) Swap. Closes the present Panel, then opens specified Panel.
- **2** Push. Saves and closes the present Panel, then opens specified Panel. You can display saved Panel again by using parameter “0” above.

comment maximum 64 characters (optional)

Example (System Spec)

Branch Label	Command	Parameter
Initialize	PANEL	“INITIALIZE”
	IF	PANEL.RETURN==”ONLINE”
	GOTO	TesterInit
	END IF	
	PANEL	“MAIN”

PAUSE

This command pauses the executing test plan, that is, pauses execution started by "LOOP" command in job specification.

Paused test plan can be continued from the Run Panel.

Syntax

Job/System Action Test	Branch Label Event Label	Command Command Algorithm	Parameter Parameter Input	Comment Comment Comment
	[label]	PAUSE		[comment]

label line label (optional)

comment maximum 64 characters (optional)

RETURN

This command exits the present event which is specified in “Action Specification” on page 12. This command can be used in “Action Specification”.

Syntax

Action	Branch Label	Command	Parameter	Comment
	[label]	RETURN		[comment]

label line label (optional)

comment maximum 64 characters (optional)

Example (Action Spec)

Branch Label	Command	Parameter	Comment
chk	IF	Skipflag	
	RETURN		
	END IF		
	LET	N=N+1	
	PANEL	“CONT”	

REJECT

This command immediately ends testing of present lot, wafer, die, or test. This command can be used in a test specification or an action specification.

Syntax

Action Test	Event Label	Command Algorithm	Parameter Input	Comment Comment
	[label]	REJECT	LOT	[comment]
	[label]	REJECT	WAFER	[comment]
	[label]	REJECT	DIE	[comment]
	[label]	REJECT	TEST	[comment]

label line label (optional)

REJECT LOT Ends testing of present lot, and terminates the “RUN” command.

REJECT WAFER Ends testing of present wafer.

REJECT DIE Ends testing of present die.

REJECT TEST Ends the present test.

comment maximum 64 characters (optional)

NOTE

Data recording on REJECT

Depends on CLEARONREJECT parameter of “/opt/SPECS/sys/config/sysconf” file. It is configured as follow:
TRUE: Clears current test data of rejected unit on REJECT.
FALSE: Don’t clear the current test data on REJECT.

REPEAT - UNTIL

Defines a loop that is repeated until expression becomes TRUE. Whether to repeat the loop is determined at *end* of loop (UNTIL statement).

Syntax

Job/System Action Test	Branch Label Event Label	Command Command Algorithm	Parameter Parameter Input	Comment Comment Comment
	[label]	REPEAT		[comment]
		statement1		[comment]
		UNTIL	expression	[comment]

label line label (optional)

statement1 Performed if expression is TRUE.

expression INTEGER expression. Refer to “Expressions” on page 14. If expression is TURE, end loop(continue to next line). If expression is FALSE, go back to REPEAT statement.

comment maximum 64 characters (optional)

Example (Test Spec)

Module	Device	Algorithm	Input
Mod1	Dev1		
		LET	i=0
		REPEAT	
		LET	i=i+1
		Leak	
		UNTIL	(Leak.current > 5n) OR (i > 10)
		END IF	

Restrictions

- IF/.REPEAT/WHILE statements can be nested up to 256 levels.
- As shown in example in Test Spec, REPEAT and UNTIL cannot be used across multiple devices.

REPORT

This command reports the measurement results in text format. This command can be used in the system specification or job specification. This command cannot be used while “LOOP” command is executing.

Syntax

Job/System	Branch Label	Command	Parameter	Comment
	[label]	REPORT	“type” [,”parm” [,”file”]]	[comment]

label line label (optional)

type Report type(STRING type). Specify one of following:

lotsum	Lot summary report format.
wafsum	Wafer summary report format.
wafraw	Wafer raw data report format.
grecipe	Make a graph report.

Following options for grecipe type must be specified:

-w	Wafer Select (ex. -w1,3,5~21)
-x	Die-X Select (ex. -x1,3,5~21)
-y	Die-Y Select (ex. -y1,3,5~21)
-g	Graph Type (1:XY, 2:Hist, 3:Wafmap)
-u	Output Block Unit Select (1:Lot, 2:Wafer, 3:Die)
-v	Variable Name Select
-t	Variable Type (1:SysTag, 2:Vari, 3:VariStat, 4:VariLotStat, 5:WaferTag, 6:DieTag, 7:LotTag)
-f	Data converter Select. Specify executable converter name. Default is “FlatComb”.

Following option s for grecipe type are optional:

-p	Dump Mode (1:PCL, 2:PostScript)
-e	No Arg. Exit after dump operation.

At the end of option string, you have to specify raw data file name. You can display the test report on the operation panel, print out the report, or save the test report to a data file by using appropriate suffix:

xxxx.term	Displays test report on operation panel.
xxxx.file	Saves test report to a data file.
xxxx.lp	Prints test report to default printer or your controller.

comment maximum 64 characters (optional)

Example (Job Spec)

Branch Label	Command	Parameter
	LOOP	
	REPORT	“lotsum”
	REPORT	“wafsum.lp”

RETEST

This command executes testing again from beginning of present wafer, die, test, module or device. This command can be used in a test specification or an action specification. In test specification, the retest starts immediately. In action specification, the retest starts after all statements in the event are performed.

Syntax

Action Test	Event Label	Command Algorithm	Parameter Input	Comment Comment
	[label]	RETEST	WAFER	[comment]
	[label]	RETEST	DIE	[comment]
	[label]	RETEST	TEST	[comment]
	[label]	RETEST	MODULE	[comment]
	[label]	RETEST	DEVICE	[comment]

label line label (optional)

RETEST WAFER Executes testing again from first test on first die of present wafer.

RETEST DIE Executes testing again from first test of present die.

RETEST TEST Executes testing again from beginning of present test.

RETEST MODULE Executes testing again for present module.

RETEST DEVICE Executes testing again for present device.

comment maximum 64 characters (optional)

NOTE Data recording on RETEST

Depends on CLEARONRETEST parameter of “/opt/SPECS/sys/config/sysconf” file. It is configured as follow:
TRUE: Clears current test data of retested unit on RETEST.
FALSE: Don’t clear the current test data on RETEST.

RUN

This command opens the specified test plan, then moves control to the job specification of the test plan.

The RUN command can only be executed in the system specification.

Syntax

System	Branch Label	Command	Parameter	Comment
	[label]	RUN	[“tpl_file”[, “lim_file”]]	[comment]

label line label (optional)

tpl_file Name of test plan file (full path without suffix) to load and execute. STRING type (optional). If omitted, default is the presently loaded test plan.

lim_file Name of limit file (full path without suffix) to use.(optional)

comment maximum 64 characters (optional)

Example (System Spec)

Branch Label	Command	Parameter
	RUN	“/users/anyone/tpl/plan1”

SELECT

A test plan can contain multiple specifications of each type. This command selects which wafer specification and which probe specification are effective for test plan execution.
If wafer is selected, all dice on wafer, and all modules on dice are automatically selected. So, usually you do not need to select die specifications and modules.
If die is selected, all modules on die are automatically selected.
The “TEST” command automatically selects the test specification. So, usually you do not need to select test specifications.

Syntax

Job Action Test	Branch Label Event Label	Command Command Algorithm	Parameter Parameter Input	Comment Comment Comment
	[label]	SELECT	type,”name”	[comment]

label line label (optional)

type Specification type: WAFER|PROBE|DIE|MODULE|TEST

name Specification name(STRING or STRING array type)
For MODULE, use module name specified in die specification.
For WAFER or PROBE, must specify a name. Optional for DIE, MODULE, or TEST, default is all specifications that belong to the specified type.

comment maximum 64 characters (optional)

Example (Job Spec)

Branch Label	Command	Parameter
	SELECT	WAFER,”Waf1”
	SELECT	PROBE,”Prb1”

Restrictions

The “LOOP” command is executed in the job specification.

- Before the “LOOP” command is executed, one Wafer Specification and one Probe Specification must be selected.
- While the “LOOP” command is running, SELECT/UNSELECT commands can be executed, but *not* with WAFER or PROBE type.

STOP

This command stops the executing test plan, that is, stops execution started by "LOOP" command in job specification.

Stopped test plan cannot be continued, that is, the test plan can only be started from beginning.

Syntax

Job/System Action Test	Branch Label Event Label	Command Command Algorithm	Parameter Parameter Input	Comment Comment Comment
	[label]	STOP		[comment]

label line label (optional)

comment maximum 64 characters (optional)

TEST

This command assigns a test to a die.

The test assigned by the TEST command is automatically selected. To make sure the die is selected, see “SELECT” on page 34 command.

Multiple tests can be assigned to one die.

The TEST command cannot be executed while “LOOP” command is executing.

Syntax

Job	Branch Label	Command	Parameter	Comment
	[label]	TEST	“die”,”test”	[comment]

label line label (optional)

die Die specification name (STRING type)

test Test specification name (STRING or STRING array type)

Example (Job Spec)

Branch Label	Command	Parameter
	SELECT	WAFER,”Waf1”
	SELECT	PROBE,”Prb1”
	TEST	“Die1”,”LeakChk”
	TEST	“Die1”,”Rmeas”
	TEST	“Die2”,”LeakChk”

UNSELECT

This command disables (for execution) a specification of test plan. If wafer is unselected, all dies on wafer, and modules on dies are automatically unselected. If die is unselected, all modules on die are automatically unselected.

Syntax

Job Action Test	Branch Label	Command	Parameter	Comment
	[label]	UNSELECT	type,"name"	[comment]

label line label (optional)

type Specification type: WAFER|PROBE|DIE|MODULE|TEST
MODULE is not a specification type.

name Specification name(STRING or STRING array type)
For MODULE, use module name specified in die specification. For WAFER or PROBE, must specify a name. Optional for DIE, MODULE, or TEST, default is all specifications that belong to the specified type.

comment maximum 64 characters (optional)

Example (Job Spec)

Branch Label	Command	Parameter
	SELECT	WAFER,"Waf1"
	UNSELECT	DIE,"Die1"
	UNSELECT	DIE,"Die5"
	UNSELECT	MODULE,"Mod1"
	UNSELECT	MODULE,"Pfet"

Restrictions

The "LOOP" command is executed in the job specification.

- Before the "LOOP" command is executed, one Wafer Specification and one Probe Specification must be selected.
- While the "LOOP" command is running, SELECT/UNSELECT commands can be executed, but *not* with WAFER or PROBE type.

UNTEST

This command cancels the die and test assignment, but does not change the select/unselect condition.

The UNTEST command cannot be executed while “LOOP”command is executing.

Syntax

Job	Branch Label	Command	Parameter	Comment
	[label]	UNTEST	“die”,”test”	[comment]

label line label (optional)

die Die specification name (STRING type)

test Test specification name (STRING or STRING array type)

Example (Job Spec)

Branch Label	Command	Parameter
	SELECT	WAFER,”Waf1”
	SELECT	PROBE,”Prb1”
	TEST	“Die1”,”LeakChk”
	TEST	“Die1”,”Rmeas”
	TEST	“Die2”,”LeakChk”
	UNTEST	“Die1”,”LeakChk”
	UNTEST	“Die2”

UPDATE

This command updates selected widgets of specified panel.

Syntax

Job/System Action	Branch Label Event Label	Command Command	Parameter Parameter	Comment Comment
	[label]	UPDATE	“name, parm”	[comment]

label line label (optional)

name Panel spec name (STRING type)

parm Widget name (STRING type)
When specified, UPDATE command updates only specified widgets. This command can update the following widgets:

SingleDisp, TextDisp, ListDisp, FileDisp, FileView, CassetteDisp, WaferDisp, DataView, StatView

WHILE - END WHILE

This command defines a loop that is repeated while expression is TRUE. Whether to repeat the loop is determined at *beginning* of loop.

Syntax

Job/System Action Test	Branch Label Event Label	Command Command Algorithm	Parameter Parameter Input	Comment Comment Comment
	[label]	WHILE statement1 END WHILE	expression	[comment] [comment] [comment]

label line label (optional)

statement1 Performed statement. (optional)
If algorithm is used here, only measurement result of last loop is save to the lot test data file.

expression INTEGER expression. Refer to “Expressions” on page 14. If expression is TRUE, go to next line. If expression is FALSE, end loop(go to line after END WHILE)

comment maximum 64 characters (optional)

Example (Test Spec)

Module	Device	Algorithm	Input
Mod1	Dev1		
		LET	i=0
		WHILE	(Leak.current > 5n) OR (i < 10)
		LET	i=i+1
		Leak	
		END WHILE	

Restrictions

- IF/REPEAT/WHILE statements can be nested up to 256 levels.
- As shown in example in Test Spec, WHILE and END WHILE cannot be used across multiple devices.

6 Algorithm Library Reference

This chapter is a reference for the algorithms used by the frameworks that are provided with the HP SPECS software. The frameworks use following algorithms:

- “Tester Algorithms” Algorithms in tester libraries. (HP4071A.lib, HP4062UX.lib and HP4062F.lib).
- “Utility Algorithms” Algorithms in `UTILITY`*`xxxx`*`.lib(bas)`.
- “Prober Driver Algorithms” Algorithms in prober libraries.(TEL20S.lib, TSK60A.lib, .etc)

For making test plan, you need to make a measurement algorithm. You can refer measurement algorithm `ICMS` which is described in *HP SPECS User's Guide*.

`Demo` algorithm is for demonstrator, `TRNG` algorithm is for trainer. You will not use them.(They are not supported)

Tester Algorithms

This section is a reference for the "Tester Algorithms". "Tester Algorithms" contain algorithms which are for controlling HP4071A, HP4062UX and HP 4062F. "Tester Algorithms" are named HP4071A.lib, HP4062UX.lib and HP4062F.lib and stored in `/opt/SPECS/usr/alg` directory.

HP SPECS provides following "Tester Algorithms".

- Common Algorithms
- HP4071A Algorithms
- HP4062UX Algorithms
- HP4062F Algorithms

Common Algorithms

Algorithms in below table can be used for controlling all testers(HP 4071A,4062UX/F). They are stored in each tester algorithm.(HP4071A, HP4062UX, HP4062F)

Algorithm Name	Description
TESTER_START	Executes HP 4071A,4062UX/F START program.
TESTER_STOP	Exits from the test session.
TESTER_INIT	Initializes HP 4071A,4062UX/F test system.
TESTER_RESET	Disables and disconnects HP 4071A,4062UX/F output.
TESTER_IDENT	Check which TIS is linked for HP 4071A, HP 4062UX or HP 4062F.

HP 4071A Algorithms

Algorithms in below table can be used for controlling HP 4071A. They are stored in `HP4071A.lib(bas)`.

Algorithm Name	Description
HP4071A_CAL	Calibrates HP 4071A test system.
HP4071A_ERROR	Gets HP 4071A error number and error message.

HP 4062UX Algorithms

Algorithms in below table can be used for controlling HP 4062UX. They are stored in `HP4062UX.lib(bas)`.

Algorithm Name	Description
HP4062UX_CAL	Calibrates HP 4062UX test system.
HP4062UX_ERROR	Gets HP 4062UX error number and error message.
HP4062UX_LOCK	Locks HP 4062UX test system.
HP4062UX_UNLOCK	Release Hp 4062UX from lock condition.

HP 4062F Algorithms

Algorithms in below table can be used for controlling HP 4062F. They are stored in `HP4062F.lib(bas)`

Algorithm Name	Description
HP4062F_CAL	Calibrates HP 4062F test system.
HP4062F_ERROR	Gets HP 4062F error number and error message.
HP4062F_LOCK	Locks HP 4062F test system.
HP4062F_UNLOCK	Release Hp 4062F from lock condition.

TESTER_START

Calls and executes the HP 4071A,4062UX/F START program.

Variable Type	Variable Name	Data Type	Description
Input	`Mode`	INTEGER	Execution mode. `0`: offline, `1`: online, `2`: online debug. Default: 1.
Output	`Stat`	INTEGER	Execution result. `1: already login status` `0`: successful, `–1`: error, -2: already login status but *Mode* input variable is different

NOTE

For starting the HP 4071A, iconified “hp4070” window opens automatically and returns error status when user attempts to initialize HP 4071A with inconsistent mode.

TESTER_INIT

Calls the HP 4071A/4062UX/F TIS command `Init_system`, which initializes the HP 4071A,4062UX/F test system.

Variable Type	Variable Name	Data Type	Description
Input			none
Output	`Stat`	INTEGER	Initialization result. `0`: successful, `–1`: error.

TESTER_RESET

Calls HP 4071A,4062UX/F TIS commands `Disable_port` and `Connect`, which disable and disconnect the outputs of HP 4071A,4062UX/F test system.

Variable Type	Variable Name	Data Type	Description
Input			none
Output	`Stat`	INTEGER	Reset result.`0`: successful, `–1`: error

HP4071A_CAL

Calls HP 4071A TIS command `Long_cal`, which calibrates the HP 4071A test system.

Variable Type	Variable Name	Data Type	Description
Input			none
Output	`Stat`	INTEGER	Calibration result. `0`: successful, `–1`: error.

HP4071A_ERROR

Calls HP 4071A TIS command `Error_info`, which gets the error number and error message of the HP 4071A test system.

Variable Type	Variable Name	Data Type	Description
Input			none
Output	`Errornum`	INTEGER	Error number
	`Errormsg`	STRING	Error message

HP4062UX_CAL

Calls HP 4062UX TIS command `Long_cal`, which calibrates the HP 4062UX test system.

Variable Type	Variable Name	Data Type	Description
Input			none
Output	`Stat`	INTEGER	Calibration result. `0`: successful, `–1`: error.

HP4062_LOCK

Calls HP 4062UX TIS command `Lock_system`, which locks the HP 4062UX test system.

Variable Type	Variable Name	Data Type	Description
Input			none
Output	`Stat`	INTEGER	Lock result. `0`: successful, `–1`: fail.

HP4062_UNLOCK

Calls HP 4062UX TIS command `Unlock_system`, which releases the LOCK condition of the HP 4062UX test system.

Variable Type	Variable Name	Data Type	Description
Input			none
Output	`Stat`	INTEGER	Unlock result. `0`: successful, `–1`: fail.

HP4062_ERROR

Calls HP 4062UX TIS command `Error_info`, which gets the error number and error message of the HP 4062UX test system.

Variable Type	Variable Name	Data Type	Description
Input			none
Output	`Errornum`	INTEGER	Error number
	`Errormsg`	STRING	Error message

HP4062F_CAL

Calls HP 4062F TIS command `Long_cal`, which calibrates the HP 4062F test system.

Variable Type	Variable Name	Data Type	Description
Input			none
Output	`Stat`	INTEGER	Calibration result. `0`: successful, `–1`: error.

HP4062F_ERROR

Calls HP 4062F TIS command `Error_info`, which gets the error number and error message of the HP 4062F test system.

Variable Type	Variable Name	Data Type	Description
Input			none
Output	`Errornum`	INTEGER	Error number
	`Errormsg`	STRING	Error message

HP4062F_LOCK

Calls HP 4062F TIS command `Lock_system`, which locks the HP 4062F test system.

Variable Type	Variable Name	Data Type	Description
Input			none
Output	`Stat`	INTEGER	Lock result. `0`: successful, `–1`: fail.

HP4062F_UNLOCK

Calls HP 4062F TIS command `Unlock_system`, which releases the LOCK condition of the HP 4062F test system.

Variable Type	Variable Name	Data Type	Description
Input			none
Output	`Stat`	INTEGER	Unlock result. `0`: successful, `–1`: fail.

Utility Algorithms

This section is a reference for the "Utility Algorithms" which are stored in `/opt/opt/SPECS/usr/alg/UTILITY.lib(bas)`. "Utility Algorithms" contain algorithms which are utilities and for processing string, character, numeric data. Also sweep measurement algorithms are contained.

- Utility algorithms

Algorithm Name	Description
MASTER_LOOKUP	Returns specified field of specified line in an ASCII data file.
FILE_LOOKUP	Searches for and returns all the file names (in all directories under a specified directory) that have a specified suffix.
STRING_LOOKUP	Returns the position of specified character(s) in a STRING variable.
STRING_SPLIT	Returns data (without separators) from a STRING variable, which contains data separated by data separators.
TIMEDATE	Returns the present date and time.
ARRAY_LOOKUP1	Returns the array index for a specified data element of a STRING array.
ARRAY_LOOKUP2	Returns the array indices for the specified data elements of a STRING array.
ARRAY_LOOKUP3	Returns the data elements for the specified indices of a STRING array.
READ_REAL	Reads a value from a STRING variable, and returns it to a REAL variable.
WRITE_REAL	Writes a REAL value to a STRING variable.
READ_INTEGER	Reads a value from a STRING variable, and returns it to an INTEGER variable.

Algorithm Name	Description
WRITE_INTEGER	Writes an INTEGER value to a STRING variable.
READ_CHAR	Reads a character from a STRING variable, and returns it to a CHARACTER variable.
WRITE_CHAR	Writes a CHARACTER value to a STRING variable.
READ_STRING	Reads string data from a STRING variable, and returns it to another STRING variable.
WRITE_STRING	Writes a STRING value to a STRING variable.
BEEP	Specifies frequency and duration for the beep, and sounds the beep.
CLOCK	Returns the present date and time.
WAIT	Waits for the specified duration.
PRINT	Prints the specified STRING variable to the present PRINTER IS device.
PRINT_REAL	Writes a REAL value to a STRING variable, then prints it to the present PRINTER IS device.
PRINT_INTEGER	Writes an INTEGER value to a STRING variable, then prints it to the present PRINTER IS device.
PRINT_CHAR	Writes a CHARACTER value to a STRING variable, then prints it to the present PRINTER IS device.
PRINT_STRING	Writes a STRING value to a STRING variable, then prints it to the present PRINTER IS device.
PRINTER_IS	Changes the PRINTER IS device, which is the destination device of the PRINT algorithms described earlier.
PLOTTER_IS	Changes the PLOTTER IS device.

Algorithm Name	Description
CREATE_WINDOW	Opens a new HP BASIC/UX window.
DESTROY_WINDOW	Closes an existing HP BASIC/UX window.
XWUD	Executes the `xwud(1)` (HP-UX undump) command to display a window image, which is stored in the specified dump file.

- Data recording algorithms

Algorithm Name	Description
DATALOG_DIEEND	Make a data log file.
STATLOG_DIEEND	Make a statistical data log file.
YIELDLOG_WAFEND	Make a yieldlog file.

- Other algorithms

Algorithm Name	Description
SETUSRBIN_LPBGN	Sets "User Bin" for all real- or integer-scalar variables, using limits defined in Test Spec and "TPLSETUSRBIN" algorithm utility function.
JDG_DIE	Judge pass/fail of the current die by parameter yield.
JDG_DIE_AT_POS	Judge pass/fail of the specified die by parameter yield.
JDG_WAF_BY_DIE	Judge pass/fail of the current wafer by die yield.
JDG_WAF_BY_PARA	Judge pass/fail of the current wafer by parameter yield.

- Sweep measurement algorithms

NOTE If “NOTESTER” is selected at installation, “UTILITY_NOTESTER” is used as Utility algorithm library. Sweep measurement algorithms in “UTILITY_NOTESTER” cannot be used for actual measurement.

Algorithm Name	Description
G_IDVD	Measures the MOSFET Drain Current-Drain Voltage characteristics (Id-Vd characteristics), and displays the measurement results on a graph.
G_IDVG	Measures the MOSFET Drain Current-Gate Voltage characteristics (Id-Vg characteristics), and displays the measurement results on a graph.
G_ICVC	Measures the bipolar transistor Collector Current-Collector Voltage characteristics (Ic-Vc characteristics), and displays the measurement results on a graph.
G_ICVB	Measures the bipolar transistor Collector Current-Base Voltage characteristics (Ic-Vb characteristics), and displays the measurement results on a graph.
G_HFE	Measures the bipolar transistor Collector Current-hFE characteristics (Ic-hFE characteristics), and displays the measurement results on a graph.
G_CV	Measures the Capacitance-Voltage characteristics (C-V characteristics), and displays the measurement results on a graph.

MASTER_LOOKUP

Returns the contents of a specified field in a specified line of a specified ASCII data file.

A maximum 199 fields can be defined on one line. The fields must be separated by colon (:) as in following example:

```
# Example of the data file for MASTER_LOOKUP
CASSETTE_1:LOT_1:PROD_1:PROCESS_1:TEST_1:5
CASSETTE_2:LOT_2:PROD_2:PROCESS_2:TEST_1:5
```

The first field of each line contains a keyword that identifies the line. In above example, `CASSETTE_`*n* (*n*: Integer) are keywords that the MASTER_LOOKUP algorithm uses to find the specified line.

The MASTER_LOOKUP algorithm is required by the SHELL1 framework to get required data from the `master.dat`. For the `master.dat` file, refer to *HP SPECS Operator's Guide*.

Variable Type	Variable Name	Data Type	Description
Input	`File`	STRING	ASCII file name.
	`Key`	STRING	Keyword that specifies a line in the ASCII file. The keyword must be the first field of a line in the ASCII file.
	`Fnum`	INTEGER	Number that specifies a field in the line. Range: 1 to 199, where 1 indicates the first (leftmost) field of the line.
Output	`Field`	STRING	Contents of specified field.
	`Stat`	INTEGER	Error status. `0`: No error. `–1`: Invalid keyword. `–2`: Invalid field number. `–3`: Invalid file name.

The following is an example for a file named `master.dat` that contains a line as follows:

```
TAC39401:HP98574YQ4484100:EF54Z6:CMOS:TAC394:1
```

- Input variables:
  - `File`: `"master.dat"`
  - `Key`: `"TAC39401"`
  - `Fnum`: `4`
- Output variables:
  - `Field`: `CMOS`
  - `Stat`: `0`

FILE_LOOKUP

Searches for and returns the names of all files (in all directories under a specified directory) that have the specified suffix. The suffix means a file extension, such as `tpl` in the file name `testplan.tpl`.

Variable Type	Variable Name	Data Type	Description
Input	`Dir`	STRING	Directory to search. All directory levels below this directory will be searched. Default: your home directory.
	`Suffix`	STRING	File suffix to search for. Default: `tpl`. If you enter `""` (a pair of double quotes), all file names will be returned.
Output	`Files`	STRING[1024]	The found file names (including directory names relative to the specified directory) are returned to this variable. Maximum 1024 file names can be returned. Maximum file name (*directory/file*) length is 255 characters.
	`Count`	INTEGER	Number of files found.
	`Depth`	INTEGER	Level of deepest directory at which a file was found. This is the number of directory levels below the specified directory plus 1, which is added for the file name. For example, if `level1/level2/level3/testplan.tpl` is deepest file returned to `Files`, then `4` is returned to `Depth`.

Example:

- Input Variables:
  - `Dir`: `"/SPECS"`
  - `Suffix`: `"dat"`
- Output Variables:
  - `Files`: `usr/tpl/master.dat`
  - `Count`: `1`
  - `Depth`: `3`

STRING_LOOKUP

Returns the position of specified character(s) in a STRING variable.

Variable Type	Variable Name	Data Type	Description
Input	`Src`	STRING	String data.
	`Pat`	STRING	Character(s) to search for.
Output	`Pos`	INTEGER	Position of the characters specified by `Pat`. 0: Not found. *N*: Position.

Example:

- Input Variables:
  - `Src`: `"abcdefghij"`
  - `Pat`: `"ef"`
- Output Variable:
  - `Pos`: `5`

STRING_SPLIT

Returns the data (without separators) from a STRING variable, which contains data and data separators.

This algorithm can return a maximum 128 elements.

Variable Type	Variable Name	Data Type	Description
Input	`Src`	STRING	String data.
	`Del`	STRING	Data separator. You can specify multiple characters.
Output	`Elem`	STRING[128]	Returned data. Maximum 128 data can be returned.
	`Count`	INTEGER	Number of returned data.

Example:

- Input Variables:
  - `Src`: `"zabyabx"`
  - `Del`: `"ab"`
- Output Variables:
  - `Elem`:

    ```
    Elem[1]="z"
    Elem[2]="y"
    Elem[3]="x"
    ```

  - `Count`: `3`

TIMEDATE

Returns the present date and time.

Variable Type	Variable Name	Data Type	Description
Input			none
Output	`Date`	STRING	Date (month/day/year). mm/dd/yy
	`Time`	STRING	Time (hour/minute/second). hh:mm:ss

ARRAY_LOOKUP1

Returns the array index for the specified data element of a STRING array.

Variable Type	Variable Name	Data Type	Description
Input	`Src`	STRING[256]	String array to search.
	`Pat`	STRING	Data to search for in array. Cannot use abbreviation.
Output	`Pos`	INTEGER	Array index returned. 0: Not found. *N* (integer): Array index of found data.

Example:

- Input Variables:
  - `Src`:

    ```
    Src[1]="abc"
    Src[2]="bcd"
    Src[3]="cde"
    Src[4]="def"
    ```

  - `Pat`: `"def"`
- Output Variable:
  - `Pos`: `4`

ARRAY_LOOKUP2

Returns the array indices for the specified data elements of a STRING array.

Variable Type	Variable Name	Data Type	Description
Input	`Src`	STRING[256]	String array to search.
	`Pat`	STRING[256]	Data to search for in array. Cannot use abbreviation.
Output	`Pos`	INTEGER[256]	Array index for each specified data: 0: Not found. *N* (integer): Array index.

Example:

- Input Variables:
  - `Src`:

```
Src[1]="abc"
Src[2]="bcd"
Src[3]="cde"
Src[4]="def"
```

  - `Pat`:

```
Pat[1]="def"
Pat[2]="cde"
Pat[3]="bcd"
Pat[4]="abc"
```

- Output Variable:
  - `Pos`:

```
Pos[1]=4
Pos[2]=3
Pos[3]=2
Pos[4]=1
```

ARRAY_LOOKUP3

Returns the data elements for the specified indices of a STRING array.

Variable Type	Variable Name	Data Type	Description
Input	`Src`	STRING[256]	String array to search.
	`Pos`	INTEGER[256]	Array indices to search for.
Output	`Sub`	STRING[256]	Data element for each index.

Example:

- Input Variables:
  - `Src`:

    ```
    Src[1]="abc"
    Src[2]="bcd"
    Src[3]="cde"
    Src[4]="def"
    ```

  - `Pos`:

    ```
    Pos[1]=4
    Pos[2]=3
    Pos[3]=2
    Pos[4]=1
    ```

- Output variable:
  - `Sub`:

    ```
    Sub[1]="def"
    Sub[2]="cde"
    Sub[3]="bcd"
    Sub[4]="abc"
    ```

READ_REAL

Reads a value from a STRING variable, and returns it to a REAL variable.

Variable Type	Variable Name	Data Type	Description
Input	`Format`	STRING	Format for reading string data. Refer to `scanf(3S)` in HP-UX manual for how to specify the format.
	`String`	STRING	String data to read.
Output	`Value`	REAL	Returned value. REAL.
	`Stat`	INTEGER	Read status. `0`: No data. `1`: successful. `–1`: EOF.

Example:

- Input Variables:
  - `Format`: `"%5*c%f"`
  - `String`: `"abcde1"`
- Output Variables:
  - `Value`: `1.0`
  - `Stat`: `1`

WRITE_REAL

Writes a REAL value to a STRING variable.

Variable Type	Variable Name	Data Type	Description
Input	`Format`	STRING	Format in which to write the value. Refer to `printf(3S)` in HP-UX manual for how to specify the format.
	`Value`	REAL	REAL value to write to the STRING variable.
Output	`String`	STRING	Written value. STRING
	`Length`	INTEGER	Number of characters written to new string.

Example:

- Input Variables:
  - `Format`: `"%.6f"`
  - Value: `1.23`
- Output Variables:
  - String: `1.230000`
  - Length: `8`

READ_INTEGER

Reads a value from a STRING variable, and returns it to an INTEGER variable.

Variable Type	Variable Name	Data Type	Description
Input	`Format`	STRING	Format for reading the string data. Refer to `scanf(3S)` in HP-UX manual for how to specify the format.
	`String`	STRING	String data to read.
Output	`Value`	INTEGER	Returned value. INTEGER.
	`Stat`	INTEGER	Read status. `0`: No data. `1`: successful. `–1`: EOF.

Example:

- Input Variables:
  - `Format`: `"%5*c%d"`
  - `String`: `"abcde1"`
- Output Variables:
  - `Value`: `1`
  - `Stat`: `1`

WRITE_INTEGER

Writes an INTEGER value to a STRING variable.

Variable Type	Variable Name	Data Type	Description
Input	`Format`	STRING	Format in which to write the value. Refer to `printf(3S)` in HP-UX manual for how to specify the format.
	`Value`	INTEGER	INTEGER value to write to the STRING variable.
Output	`String`	STRING	Written value. STRING.
	`Length`	INTEGER	Number of characters written to STRING variable.

Example:

- Input Variables:
  - `Format`: `"%.6d"`
  - `Value`: `123`
- Output Variables:
  - `String`: `000123`
  - `Length`: `6`

READ_CHAR

Reads a character from a STRING variable, and returns it to a CHARACTER variable.

Variable Type	Variable Name	Data Type	Description
Input	`Format`	STRING	Format for reading string data. Refer to `scanf(3S)` in HP-UX manual for how to specify the format.
	`String`	STRING	String data to read.
Output	`Value`	CHARACTER	Returned value. CHARACTER.
	`Stat`	INTEGER	Read status. `0`: No data. `1`: successful. `–1`: EOF.

Example:

- Input Variables:
  - Format: `"%5*c%c"`
  - String: `"abcdef"`
- Output Variables:
  - Value: `f`
  - Stat: `1`

WRITE_CHAR

Writes a CHARACTER value to a STRING variable.

Variable Type	Variable Name	Data Type	Description
Input	`Format`	STRING	Format in which to write the value. Refer to `printf(3S)` in HP-UX manual for how to specify the format.
	`Value`	CHARACTER	CHARACTER value to write to the STRING variable.
Output	`String`	STRING	Written value. STRING.
	`Length`	INTEGER	Number of characters written to STRING variable.

Example:

- Input Variables:
  - `Format`: `"%c"`
  - `Value`: `"A"`
- Output Variables:
  - `String`: `A`
  - `Length`: `1`

READ_STRING

Reads string data from a STRING variable, and returns it to another STRING variable.

Variable Type	Variable Name	Data Type	Description
Input	`Format`	STRING	Format for reading the string data. Refer to `scanf(3S)` in HP-UX manual for how to specify the format.
	`String`	STRING	String data to read.
Output	`Value`	STRING	Returned value. STRING.
	`Stat`	INTEGER	Read status. `0`: No data. `1`: successful. `–1`: EOF.

Example:

- Input Variables:
  - `Format`: `"%5*c%s"`
  - `String`: `"abcdefghij"`
- Output Variables:
  - `Value`: `fghij`
  - `Stat`: `1`

WRITE_STRING

Writes a STRING value to a STRING variable.

Variable Type	Variable Name	Data Type	Description
Input	`Format`	STRING	Format in which to write the string. Refer to `printf(3S)` in HP-UX manual for how to specify the format.
	`Value`	STRING	Source string.
Output	`String`	STRING	Destination STRING variable
	`Length`	INTEGER	Number of characters written to STRING variable.

Example:

- Input Variables:
  - `Format`: `"%.4s\n"`
  - `Value`: `"data_1"`
- Output Variables:
  - `String`: `data`
  - `Length`: `4`

BEEP

Specifies the frequency and duration of the beep, and sounds the beep.

Variable Type	Variable Name	Data Type	Description
Input	`Freq`	REAL	Frequency.
	`Dur`	REAL	Duration.
Output			none.

CLOCK

Returns the present date and time.

Variable Type	Variable Name	Data Type	Description
Input			none.
Output	`Year`	INTEGER	Year.
	`Month`	INTEGER	Month.
	`Day`	INTEGER	Day.
	`Hour`	INTEGER	Hour.
	`Minute`	INTEGER	Minute.
	`Second`	INTEGER	Second.
	`Value`	REAL	Value returned by HP BASIC/UX TIMEDATE command.

WAIT

Waits for the specified duration.

Variable Type	Variable Name	Data Type	Description
Input	`Wait`	REAL	Duration.
Output			none.

PRINT

Prints the specified STRING variable to the present PRINTER IS device.

Variable Type	Variable Name	Data Type	Description
Input	`S`	STRING	String data to print.
Output			none.

PRINT_REAL

Writes a REAL value to a STRING variable, then prints it to the present PRINTER IS device.

Variable Type	Variable Name	Data Type	Description
Input	`Format`	STRING	Format in which to write the value. Refer to the WRITE_REAL algorithm.
	`Value`	REAL	REAL value to write to the STRING variable.
Output			none.

PRINT_INTEGER

Writes an INTEGER value to a STRING variable, then prints it to the present PRINTER IS device.

Variable Type	Variable Name	Data Type	Description
Input	`Format`	STRING	Format in which to write the value. Refer to the WRITE_INTEGER algorithm.
	`Value`	INTEGER	INTEGER value to write to the STRING variable.
Output			none.

PRINT_CHAR

Writes a CHARACTER value to a STRING variable, then prints it to the present PRINTER IS device.

Variable Type	Variable Name	Data Type	Description
Input	`Format`	STRING	Format in which to write the character. Refer to the WRITE_CHAR algorithm.
	`Value`	CHARACTER	CHARACTER value to write to the STRING variable.
Output			none.

PRINT_STRING

Writes a STRING value to a STRING variable, then prints it to the present PRINTER IS device.

Variable Type	Variable Name	Data Type	Description
Input	`Format`	STRING	Format in which to write the string. Refer to the WRITE_STRING algorithm.
	`Value`	STRING	Source string.
Output			none.

PRINTER_IS

Changes the PRINTER IS device, which is the destination device of the PRINT algorithms described earlier.

Variable Type	Variable Name	Data Type	Description
Input	`Sel`	INTEGER	Device selector. Use the parameters available for PRINTER IS, which is an HP BASIC/UX command.
Output			none.

PLOTTER_IS

Changes the PLOTTER IS device.

Variable Type	Variable Name	Data Type	Description
Input	`Sel`	INTEGER	Device selector. Use the parameters available for PLOTTER IS, which is an HP BASIC/UX command.
Output			none.

CREATE_WINDOW

Opens a new HP BASIC/UX window.

Variable Type	Variable Name	Data Type	Description
Input	`Window`	INTEGER	Window ID. Used when you want to close the window. Refer to DESTROY_WINDOW.
	`X`	INTEGER	X position for the window.
	`Y`	INTEGER	Y position for the window.
	`W`	INTEGER	X width of the window.
	`H`	INTEGER	Y height of the window.
	`Label`	STRING	Title of the window.
Output			none.

DESTROY_WINDOW

Closes an existing HP BASIC/UX window.

Variable Type	Variable Name	Data Type	Description
Input	`Window`	INTEGER	Window ID used in the CREATE_WINDOW algorithm that opened the window.
Output			none.

XWUD

Executes the `xwud(1)` (HP-UX undump) command to display a window image, which is stored in the specified dump file. This is a file that was created by the `xwd(1)` dump command.

Variable Type	Variable Name	Data Type	Description
Input	`File`	STRING	`xwd(1)` file to display.
	`Xpos`	INTEGER	X position for display.
	`Ypos`	INTEGER	Y position for display.
	`Noclick`	INTEGER	To set `-noclick` option for xwud command, enter `1`. If this option is not specified, clicking any button in the window terminates the command.
Output			none.

G_IDVD

Measures the MOSFET Drain Current-Drain Voltage characteristics (Id-Vd characteristics), and displays the measurement results on a graph.

Graph Setup:

- Graph scale: Linear
- Title: `Vd - Id`
- X-axis label: `Vd [V]`
- Y-axis label: `Id [A]`

Variable Type	Variable Name	Data Type	Unit	Default Value	Range	Description
Input	`Vdstart`	REAL	V	0.0	−100~100	Drain start voltage.
	`Vdstop`	REAL	V	20.0	−100~100	Drain stop voltage.
	`Vdstep`	REAL	V	0.25	−200~200	Drain step voltage.
	`Vgstart`	REAL	V	3.0	−100~100	Gate start voltage.
	`Vgstop`	REAL	V	6.0	−100~100	Gate stop voltage.
	`Vgstep`	REAL	V	1.0	−200~200	Gate step voltage.
Terminal	`S`					Source terminal.
Terminal	`G`					Gate terminal.
	`D`					Drain terminal.
	`Sub`					Substrate terminal.

G_IDVG

Measures the MOSFET Drain Current-Gate Voltage characteristics (Id-Vg characteristics), and displays the measurement results on a graph.

Graph Setup:

- Graph scale: Linear
- Title: `Vg - Id`
- X-axis label: `Vg [V]`
- Y-axis label: `Id [A]`

Variable Type	Variable Name	Data Type	Unit	Default Value	Range	Description
Input	`Vgstart`	REAL	V	0.0	–100~100	Gate start voltage.
	`Vgstop`	REAL	V	5.0	–100~100	Gate stop voltage
	`Vgstep`	REAL	V	0.05	–200~200	Gate step voltage.
	`Vsubstart`	REAL	V	0.0	–100~100	Substrate start voltage.
	`Vsubstop`	REAL	V	–1.2	–100~100	Substrate stop voltage.
	`Vsubstep`	REAL	V	–0.4	–200~200	Substrate step voltage.
	`Vd`	REAL	V	5.0	–100~100	Drain Voltage.
Terminal	`S`					Source terminal.
Terminal	`G`					Gate terminal.
	`D`					Drain terminal.
	`Sub`					Substrate terminal.

G_ICVC

Measures the bipolar transistor Collector Current-Collector Voltage characteristics (Ic-Vc characteristics), and displays the measurement results on a graph.

Graph Setup:

- Graph scale: Linear
- Title: `Vc - Ic`
- X-axis label: `Vc [V]`
- Y-axis label: `Ic [A]`

Variable Type	Variable Name	Data Type	Unit	Default Value	Range	Description
Input	`Vcestart`	REAL	V	0.0	–100~100	Collector start voltage.
	`Vcestop`	REAL	V	5.0	–100~100	Collector stop voltage.
	`Vcestep`	REAL	V	0.1	0.001~200	Collector step voltage.
	`Ibstart`	REAL	A	10u	–100m~100m	Base start current.
Input	`Ibstop`	REAL	A	50u	–100m~100m	Base stop current.
	`Ibstep`	REAL	A	10u	–200m~200m	Base step current.
Terminal	`E`					Emitter terminal.
	`B`					Base terminal.
	`C`					Collector terminal.

G_ICVB

Measures the bipolar transistor Collector Current-Base Voltage characteristics (Ic-Vb characteristics), and displays the measurement results on a graph.

Graph Setup:

- Graph scale: Log
- Title: `Vb - Ic/Ib`
- X-axis label: `Vb [V]`
- Y-axis label: `Ic/Ib [A]`

Variable Type	Variable Name	Data Type	Unit	Default Value	Range	Description
Input	`Vbestart`	REAL	V	0.0	–100~100	Base start voltage.
	`Vbestop`	REAL	V	1.0	–100~100	Base stop Voltage.
	`Vbestep`	REAL	V	0.01	–200~200	Base step Voltage.
Terminal	`E`					Emitter terminal.
	`B`					Base terminal.
	`C`					Collector terminal.

G_HFE

Measures the bipolar transistor Collector Current-hFE characteristics (Ic-hFE characteristics), and displays the measurement results on a graph.

Graph Setup:

- Graph scale: Log
- Title: `hFE`
- X-axis label: `Ic [A]`
- Y-axis label: `hFE`

Variable Type	Variable Name	Data Type	Unit	Default Value	Range	Description
Input	`Ibestart`	REAL	A	1p	−100m~100m	Base start current.
	`Ibestop`	REAL	A	500u	−100m~100m	Base stop current.
	`Ibestep`	INTEGER		25	2~1001	Number of base steps.
	`Vce`	REAL	V	5.0	−100~100	Collector voltage.
Terminal	`E`					Emitter terminal.
	`B`					Base terminal.
	`C`					Collector terminal.

G_CV

Measures the Capacitance-Voltage characteristics (C-V characteristics), and displays the measurement results on a graph.

Graph Setup:

- Graph scale: Linear
- Title: `C-V`
- X-axis label: `Bias [V]`
- Y-axis label: `Cap [F]`

Variable Type	Variable Name	Data Type	Unit	Default Value	Range	Description
Input	`Vstart`	REAL	V	-5	-40~40	Inversion voltage.
	`Vstop`	REAL	V	5	-40~40	Accumulation voltage.
	`Vstep`	REAL	V	0.1	-80~80	Step voltage.
	`Delay`	REAL	sec	0.1	0~650	Step delay time.
	`Hold`	REAL	sec	30.0	0~650	Hold time.
Terminal	`H`					Capacitance high terminal.
	`L`					Capacitance low terminal.

DATALOG_DIEEND

Logs parameters listed below on the specified file. You can use this file for FileView operation panel widget. This MUST be called in DIE_END.

- Wafer ID
- Die X position
- Die Y position
- Module label
- Device label
- Variable name
- Measured value
- Lower limit
- Upper limit
- Judgement

Variable Type	Variable Name	Data Type	Unit	Default Value	Range	Description
Input	Filename	STRING				Log file name.
	Mode	INTEGER		0	0.,1	0: Logs all, 1: failed only

NOTE If you may use several tpledit/tplnavi at a time, you must distinguish the file name using session ID, etc.

STATLOG_DIEEND

Logs parameters listed below on the specified file. You can use this file for FileView operation panel widget. This MUST be called in DIE_END.

- Wafer ID
- Die X position
- Die Y position
- Module label
- Device label
- Variable name
- Measured value
- Mininum value
- Maximum value
- Mean
- Standard deviation

Variable Type	Variable Name	Data Type	Unit	Default Value	Range	Description
Input	Filename	STRING				Log file name.
	Mode	INTEGER		0	0.,1	0: Logs all, 1: failed only

NOTE If you may use several tpledit/tplnavi at a time, you must distinguish the file name using session ID, etc.

YIELDLOG_WAFEND

Logs parameters listed below on the specified file. You can use this file for FileView operation panel widget. This MUST be called in DIE_END.

- Wafer ID
- Item Yield
- Die Yield
- Judgement

Variable Type	Variable Name	Data Type	Unit	Default Value	Range	Description
Input	Filename	STRING				Log file name.
	Dieyield	INTEGER	%			Die yield target

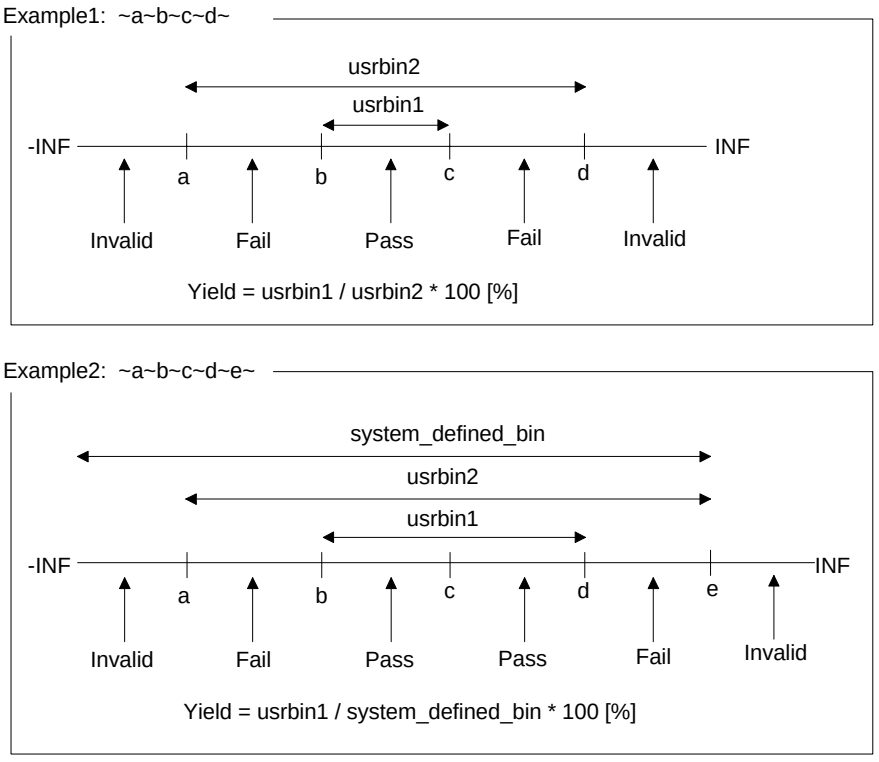
NOTE If you may use several tpledit/tplnavi at a time, you must distinguish the file name using session ID, etc.

SETUSRBIN_LPBGN

Sets "User Bin" for all real- or integer-scalar variables, using limits defined in Test Spec and "TPLSETUSRBIN" algorithm utility function. See examples below. "SPEC" range is always usrbin1 (in C, usrbin0), and "VALID" range is always the largest one. This MUST be called in LOOP_BEGIN.

Variable Type	Variable Name	Data Type	Unit	Default Value	Range	Description
Input	Waftgt	INTEGER	%	0	0~100	Wafer-level yield target
	Lottgt	INTEGER	%	0	0~100	Lot-level yield target

Figure 6-1 **Example of range**



JDG_DIE

Judges the pass/fail of the current die by the yield of test parameter.

Variable Type	Variable Name	Data Type	Description
Input	Tgt_yld	INTEGER	Target parameter yield.
Output	Jdg	INTEGER	0: fail, 1:pass, 2: unknown
	Yld	REAL	Parameter yield
	Pass_cnt	INTEGER	Number of passed parameters
	Fail_cnt	INTEGER	Number of failed parameters

JDG_DIE_AT_POS

Judges the pass/fail of the specified die by the yield of test parameter.

Variable Type	Variable Name	Data Type	Description
Input	Die_pos	INTEGER	Index number of the die
	Tgt_yld	INTEGER	Target parameter yield.
Output	Jdg	INTEGER	0: fail, 1:pass, 2: unknown
	Yld	REAL	Parameter yield
	Pass_cnt	INTEGER	Number of passed parameters
	Fail_cnt	INTEGER	Number of failed parameters
	Total_cnt	INTEGER	Number of total parameters

JDG_WAF_BY_DIE

Judges the pass/fail of the current wafer by the yield of the dice.

Variable Type	Variable Name	Data Type	Description
Input	Tgt_wafyld	INTEGER	Target wafer yield.
	Tgt_dieyld	INTEGER	Target die yield.
Output	Jdg	INTEGER	0: fail, 1:pass, 2: unknown
	Yld	REAL	Die yield
	Pass_cnt	INTEGER	Number of passed dice
	Fail_cnt	INTEGER	Number of failed dice
	Total_cnt	INTEGER	Number of total dice

JDG_WAF_BY_PARA

Judges the pass/fail of the current wafer by the yield of the parameters.

Variable Type	Variable Name	Data Type	Description
Input	Tgt_dieyld	INTEGER	Target parameter yield.
Output	Jdg	INTEGER	0: fail, 1:pass, 2: unknown
	Yld	REAL	Parameter yield
	Pass_cnt	INTEGER	Number of passed parameters
	Fail_cnt	INTEGER	Number of failed parameters
	Total_cnt	INTEGER	Number of total parameters

Prober Driver Algorithms

This section is a reference for the "Prober Algorithms". "Prober Algorithms" contain algorithms for controlling your prober. "Prober Algorithms" are named the same as prober model name as in the table below.

In installation of HP SPECS, prober model is specified by using `/opt/SPECS/install/bin/setup_prober` command.

Table 6-1 Prober Algorithm Libraries

Library name	Prober model name
TELP8	TELP8, TELP8LC
TSK90A	TSK90A, UF200
EG4080	EG4080, EG4090
EG2001	EG2001
TEL20S, 78S, 80S, 80W[a]	TEL20S, 78S, 80S, 80W
TSK60A, 80A[a]	TSK60A, 80A

a. These are unsupported drivers and stored in “/opt/SPECS/unsupported/alg” directory.

NOTE Special Prober Algorithm Library XXXX_HPSTD

TELP8_HPSTD and TSK90A_HPSTD are special libraries for HPSTD_TELP8 and HPSTD_TSK90A frameworks. You must change the algorithm library to XXXX_HPSTD when you use the HPSTD_XXXX framework.

Each "Prober Algorithms" organized same algorithms. HP SPECS can control all supported probers by using same algorithm. (ex. All supported probers can be initialized by using `PROBER_INITIAL` algorithm.)

Table 6-2 Algorithms stored of all Prober Algorithm Libraries

Algorithm Name	Description
PROBER_CONFIG	Defines the prober model and HP-IB address.
PROBER_SRQ	Waits until SRQ is asserted.
PROBER_INITIAL	Defines the initial die/module location.
PROBER_LOAD	Unloads the present wafer and loads the next wafer from cassette.
PROBER_HOME	Moves the wafer chuck (stage) to the initial position.
PROBER_MOVE	Moves wafer chuck (stage) to specified position.
PROBER_UP	Raises the wafer chuck (stage) to contact the probe card.
PROBER_DOWN	Lowers the wafer chuck (stage) to disconnect from probe card.
PROBER_INKER	Drives inker. This is a sample algorithm and not supported as product.
PROBER_CASSINFO	Returns cassette information.
PROBER_READY	Waits for prober ready SRQ.

Table 6-3 Special Algorithms for TELP8, TSK90A Prober Algorithm Libraries

Algorithm Name	Description	TEL	TSK
PROBER_ABORT	Puts the prober to the ready state	OK[a]	OK
PROBER_ALIGN	Executes wafer alignment	OK[a]	OK
PROBER_CASSMAP	Returns cassette map	OK	OK
PROBER_CLEAN	Executes probe tip cleaning	OK	OK
PROBER_ENTER	Returns HPIB ENTER result	OK	OK
PROBER_LOG	Defines HPIB I/O logmode and logfile name	OK	OK
PROBER_LOTID	Returns lot ID	OK	OK
PROBER_OUTPUT	OUTPUTs HPIB command	OK	OK
PROBER_PAUSE	Pauses the prober and turns alarm ON	OK	OK
PROBER_PRODID	Returns product ID	OK	OK
PROBER_REJECT	Unloads the current wafer to the pre-defined device	OK	NG
PROBER_SETUP	Specifies product file name	OK	OK
PROBER_SLOAD	Loads the specified wafer without pipelining	OK	OK
PROBER_TEMPMON	Returns hot chuck temperature	OK	OK
PROBER_TEMPSET	Sets hot chuck temperature	OK	OK
PROBER_UNLOAD	Unloads all the wafers to the original slot	OK[a]	OK
PROBER_WAFID	Returns wafer ID	OK	OK

a. Can be used only for non-pipelining mode

Error Information

PROBER_xxx algorithm returns the status and error code into “Status” and “Error” output variable. You can refer to the error code using “*AlgorithmName*.Error” syntax.
Example in Table 6-4 opens the “ERROR_PANEL” panel and lists the error code in it when the status of PROBER_ALGIN algorithm is “ERROR”

Table 6-4 Example of using Error Information

Branch Label	Command	Parameter
	IF	PROBER_ALIGN.Status == “ERROR”
	LET	PNL_MSG = PROBER_ALGN.Error
	PANEL	“ERROR_PANEL”

Table 6-5 lists all error codes which are returned by Prober Algorithm Library.

Table 6-5 Error codes to be returned into “Error” output variable

Error Information	Description
#ADDRESS	HP-IB address is invalid
#ALIGN_ASSIST	Alignment fail (within manual operation)
#ALIGN_REJECT	Alignment fail (auto reject)
#CASSNO	Cassette number is out of range
#ERROR *yy*	Any error occured in prober, and the generic error code is *yy*
#INVALID *zzz*	Invalid strings *zzz* was returned
#NOSUPPORT	Your prober doesn’t support this function.
#NOT_READY	Cassette mapping is not yet done
#NOWAFER	No wafer on the chuck
#NOFILE	Specified product file is not found
#PROBEAREA	Chuck moves out of pre-defined area
#SLOTNO	Slot number out of range

Error Information	Description
#SRQ *xx*	Unexpected SRQ *xx* is returned
#STEPX	StepX value is different from the prober's X index value.
#STEPY	StepY value is different from the prober's Y index value.
#TIMEOUT_IO	HPIB timeout occured on OUPUT/ENTER
#TIMEOUT_SPOLL	HPIB timeout occured on SPOLL
#TIMEOUT_SRQ	SRQ was not returned within timeout
#XDTRAVEL	X index movement out of range
#XMTRAVEL	X micro movement out of range
#YDTRAVEL	Y index movement out of range
#YMTRAVEL	Y micro movement out of range
#Z_LIMIT	Z up level is beyond limitation

PROBER_CONFIG

Sets the prober model name and its HP-IB address.
This algorithm must be called before any other prober independent algorithms.

Input Var Name	Data Type	Description
`Model`	STRING	Prober model name: Available names are TELP8_HPSTD, TSK90A_HPSTD, TELP8, TSK90A, EG4080, EG4085, EG2001, NOPROBER_HPSTD or NOPROBER.[a]
`Address`	INTEGER	HP-IB address of prober. (Port Address)*100 + (Prober Address) For off-line mode, assign 0 to Address.

a. For C language environment, EG2001 is not available.

PROBER_SRQ

Waits for SRQ from the prober, then returns status byte. If prober does not return SRQ within user-specified timeout, this algorithm returns `–1` to indicate that timeout occurred. This algorithm is mainly called by other algorithms to check the status byte. This algorithm can be called anytime after PROBER_CONFIG.

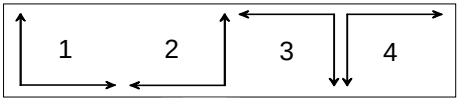
Variable Type	Variable Name	Data Type	Description
Input	`Timeout`	INTEGER	Timeout value in sec.
Output	`Statusbyte`	INTEGER	Status byte. `0~255`[a] or `–1`: SRQ timeout occurred. `-2`:SPOLL timeout occured.

a. EGxxxx returns 64(40H) usually, then the tester must read HPIB again to get the message MC(Motion Complete) or MF(Motion Fail) to determine if the specified action has been done correctly or not.

PROBER_INITIAL

Defines the initial die/module locations, which should be the actual location of a module on a die, and should match the location specified in the wafer specification.

To synchronize the actual wafer coordinates with prober driver coordinates, this algorithm should be called right after initial alignment of probe card to initial die/module. This algorithm must be called before PROBER_MOVE, PROBER_HOME, or PROBER_LOAD is called.

Input Var Name	Data Type	Description
`DieX`	REAL	Initial Die X position. Distance from wafer origin in µm. To match Wafer specification, use `WAFER.STEPX*WAFER.ALIGNDIEX`.
`DieY`	REAL	Initial Die Y position. Distance from wafer origin in µm. To match Wafer specification, use `WAFER.STEPY*WAFER.ALIGNDIEY`.
`ModX`	REAL	Initial Module X position. Distance from die origin in µm. To match Wafer specification, use `WAFER.ALIGNMODX`.
`ModY`	REAL	Initial Module Y position. Distance from die origin in µm. To match Wafer specification, use `WAFER.ALIGNMODY`.
StepX	REAL	X index size in [um][a]
StepY	REAL	Y index size in [um][a]
Coord	INTEGER	Defines the direction of the wafer coordinate. This parameter is for getting COORDINATE parameter defined in Wafer Spec. PROBER_INITIAL does not send this parameter to your prober. 

a. StepX,Y is used only in TELP8_HPSTD (Default value is 0).
If any value other than 0 is specified, it will be stored as COM then compared with the one defined in the prober.
If those value are different, it returns Status=”ERROR”,
Error=”# STEPX(Y)”.
On default, the value sent from the prober is stored as COM variable.

PROBER_LOAD

Unloads the current wafer on the chuck if any to the original slot of the cassette, and loads the next wafer onto the chuck. (After the loading, wafer profiling and auto-alignment is done if the prober has such capability, and those functions are enabled in the prober console.)

PROBER_HOME

Moves the chuck to the initial position, (the position right after load/alignment is done) previously defined at the prober console, which is the same location that SEPCS recognizes as the initial position, with the parameters defined by PROBER_INITIAL.

PROBER_MOVE

Moves chuck to the specified XY position, defined in SPECS coordinate system. (Independent of prober settings.)

When this function is called, the wafer chuck fall down to the down position, moves to the specified location,then raises up to the up position.

Input Var Name	Data Type	Description
`DieX`	REAL	Die X position. Distance from wafer origin in µm.
`DieY`	REAL	Die Y position. Distance from wafer origin in µm.
`ModX`	REAL	Module X position. Distance from die origin in µm.
`ModY`	REAL	Module Y position. Distance from die origin in µm.

PROBER_UP

Raises the chuck to the up position, previously defined at the prober console.

If the action is completed without any errors, it returns Status="OK" and Error="". If not, Status="ERROR" with error messages.

PROBER_DOWN

Moves the chuck to the down position, previously defined at the prober console.

PROBER_INKER

Drives inker, which is a prober function to mark defective dice on the actual wafer.

NOTE

PROBER_INKER is not supported as product.

This is a sample algorithm.

PROBER_CASSINFO

Returns the slot number of the current wafer on the chuck, the number of the wafers and the cassette remaining to be tested. This algorithm returns these parameters to the variable listed in the table below.

If those information are got correctly, it also returns Status="OK" and Error="", if not Status="ERROR" with some error messages.

Output Var Name	Data Type	Description
`Slotno`	INTEGER	Slot number of present wafer.
`Cassno`[a]	INTEGER	Cassette number of current wafer origin
`Wafcount`	INTEGER	Number of remaining wafers in cassette.
`Casscount`	INTEGER	Number of remaining cassettes in lot.

a. This parameter is only for TELP8_HPSTD and TSK90A_HPSTD.

PROBER_READY

Waits for the SRQ line become busy, and make sure if the prober status is ready to test the first die/module. If it detects the SRQ and ready status byte or message, it returns Status="OK" and Error="".

If the SRQ line doesn't turn to true in Timeout seconds, it will return Status="ERROR" and Error="#TIMEOUT_SRQ".

xxxx_HPSTD prober algorithm library uses a timeout input variable which specifies wait time for SRQ.

NOTE When using this algorithm with the XXXX_HPSTD frameworks

Timeout input variable can be set. This parameter specifies the time to wait the SRQ. Default is 300 seconds.

PROBER_ABORT

Aborts probing and quit the current lot.

NOTE For TELP8 the command to abort probing for the pipelining mode is not yet available.
Or, just available for non-pipeline mode only. So, only when you set TELP8 to non-pipelining mode, the abort functions is implemented with TELP8_SLOAD(9,99). Refer to "PROBER_SLOAD" on page 66.

PROBER_ALIGN

Executes auto wafer alignment of the current wafer.

NOTE If there's no wafer on the chuck, it returns Stats="ERROR".
For TELP8, the command to auto-align the wafer is available only when non-pipeline mode is selected.

PROBER_CASSMAP

Returns the cassette map in an “Cassmap” INTEGER[50] type output variable.

The cassette map contains which slot is occupied with a wafer(1), or empty(0).

PROBER_CLEAN

Cleans the probe needles, if the target probers have this automatic needle clean capability and it’s enabled.

CAUTION

The cleaning action must be properly programmed in the prober console. If not, it may damage the probe tip.

PROBER_ENTER

Reads data, if any from the HPIB address defined as /Hp_prbdvr/Prb_address. If Prb_address is set properly, this algorithm returns a result of HPIB read to “Ent_val” output variable.

If PROBER_LOG returns Logmode=1, it records the Ent_val to the log file “/opt/SPECS/sys/tmp/probera_log” with time log. The Logmode can be set using the “PROBER_LOG_MODE” global variable only when this algorithm is used with C language environment.

PROBER_LOG

Returns a log-mode to Logmode(0 or 1) output variable. If the Logmode=0, HPIB I/O logging will not be done.
Else, the log will be recorded in /opt/SPECS/sys/tmp/prober_log.

For logging the HPIB I/O(OUTPUT/ENTER/SPOLL), the internal parameter, Logmode must be edited to Logmode=1. (Default: Logmode=1)
The Logmode can be set using the “PROBER_LOG_MODE” global variable only when this algorithm is used with C language environment.

PROBER_LOTID

Returns the lot ID as “Lot id” defined at the prober console. processed to Lotid output variable. Lot ID is defined at the prober console.

PROBER_OUTPUT

Sends HPIB command defined as “Command” input variable to the address defined as /Hp_prbdvr/ Prb_address.

For logging the HPIB I/O(OUTPUT/ENTER/SPOLL), the internal parameter, Logmode must be edited to Logmode=1. (Default: Logmode=1)
The Logmode can be set using the “PROBER_LOG_MODE” global variable only when this algorithm is used with C language environment.

NOTE This algorithm only sends command without serial polling after that. If SPOLL is necessary after sending HPIB command, use the PROBER_SRQ algorithm after the PROBER_OUTPUT.

PROBER_PAUSE

Pauses the current lot test and turns the alarm light ON. The PAUSE state of prober will be continued/aborted by the operator.

NOTE This algorithm doesn’t clear the SRQ after calling the HPIB command. To clear the SRQ, call the PROBER_READY algorithm.

PROBER_PRODID

Returns the product file name currently processed as “Prodid” output variable.

PROBER_REJECT

Unloads the current wafer to the pre-defined device. If the pipeline mode is enabled in the prober, the PROBER_REJECT algorithm loads the next wafer.

PROBER_SETUP

Specifies the product file name by “Filename” input variable and the disk drive by “Drive” input variable where the file is stored.

If the specified file is not found in the specified drive, it returns “ERROR” to “Status” output variable and “#NOFILE” to the “Error” output variable and changes no parameters. In this case the operator must specify the file name manually at the prober console, and may need to change the file name either in the program, the database, or the prober’s disk.

PROBER_SLOAD

Unloads the current wafer on the chuck to the original slot, and loads the wafer specified by “Cassno” and “Slotno” to the chuck, without pipelining.

To end the lot and unload all of the wafer, put Cassno=9 and Slotno=99. If you want to unload the current wafer and stand by, put Cassno=0 and Slotno=0.

Input Var Name	Data Type	Description
`Cassno`	INTEGER	Cassette number of the wafer to be loaded. 1(EG4080), 1 or 2(TELP8, TSK90A)
`Slotno`	INTEGER	Slot number of the wafer to be loade3d. 1 to 25: load the wafer 0: unload the current wafer 99: end lot

PROBER_TEMPMON

Returns the temperature of the hot chuck to the “Montemp” output variable, if it was set properly.

PROBER_TEMPSET

Set the temperature by “Settemp” input variable to the hot chuck.

PROBER_UNLOAD

Unloads the all wafers from the prober to the original slot of the cassette.

NOTE

The PROBER_UNLOAD algorithm only unloads the all wafers from the prober. The PROBER_ABORT function rejects the lot.

PROBER_WAFID

Returns the ID of the wafer currently on the chuck to “Wafid” output variable.