
FTI STUDIO™ USER INTERFACE MANUAL

Copyright © 2007-10 Focused Test Inc

TABLE OF CONTENTS

1. INTRODUCTION.....4

2. QUICK MENUBAR OVERVIEW 10

..... 15

3. OPENING, SAVING AND CLOSING TEST PROGRAMS 16

4. BROWSING AND MODIFYING TEST PROGRAMS 20

4.1 ADDING AND REMOVING NODES 20

 4.1.1 *Configuration Nodes*.....21

 4.1.1.1 *Binning*.....21

 4.1.2 *Flow Node Types*23

 4.1.3 *Shared Flow Nodes*.....24

 4.1.4 *Sequences*25

 4.1.5 *Step Node Types*.....25

4.2 STEPS AND METHODS..... 27

 4.2.1 *Step Properties*.....29

 4.2.2 *Configuration Parameters*30

4.3 DUAL DIE TESTING 32

4.4 PARALLEL TESTING 33

4.5 INDEX PARALLEL 35

4.6 PROGRAM TOOLS..... 37

 4.6.1 *Accessing the Node Tools*.....37

 4.6.2 *Program Node Tools*..... 40

 4.6.2.1 *Profiler Tool*..... 40

 4.6.2.2 *Instrument Tool* 42

 4.6.2.3 *Variables Tool*..... 45

 4.6.3 *Step Node Tools* 47

 4.6.4 *Some Points on Scope and Shmoo* 57

 4.6.5 *Data Manager* 58

4.7 GENERAL TOOLS..... 64

 4.7.1 *Binner*..... 64

 4.7.2 *Commander* 65

 4.7.3 *Data Logger*..... 67

 4.7.4 *Equipment*..... 74

5. USER INTERFACE MODES (PRODUCTION/ENGINEERING)..... 75

6. PRODUCTION OPERATION 78

..... 78

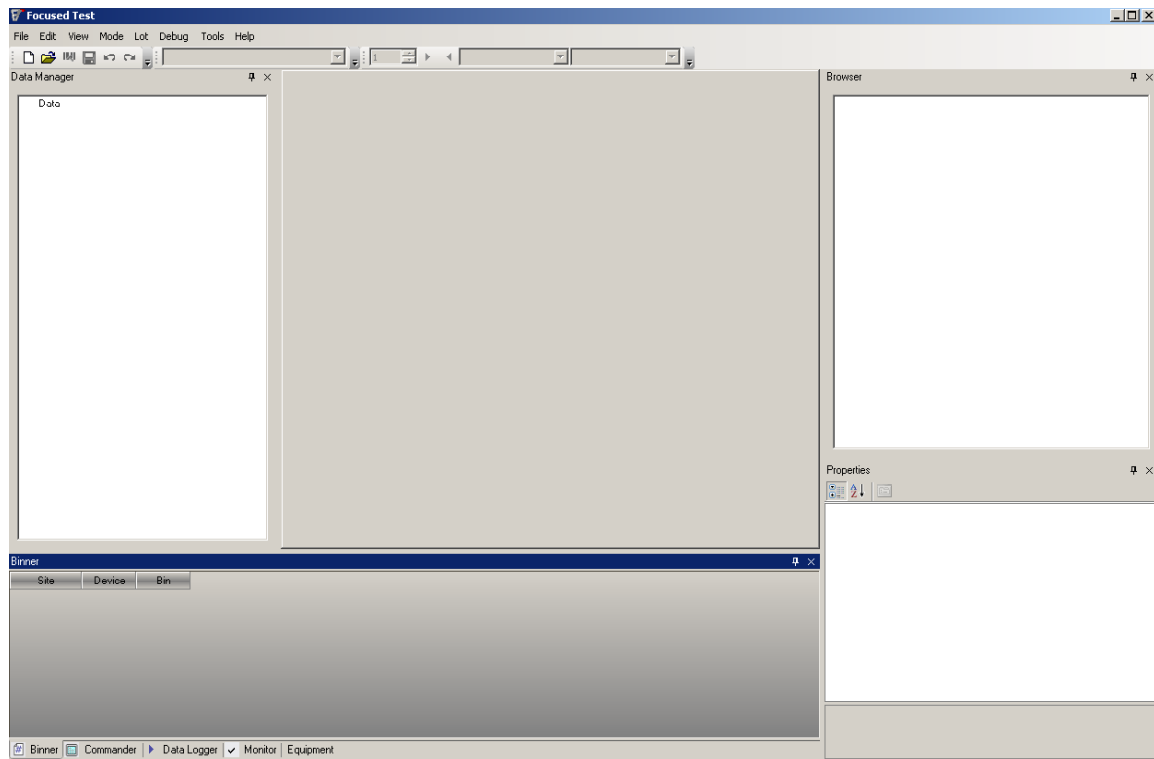
7. MISCELLANEOUS	81
7.1 VIEW MENU.....	81
7.2 DEBUG MENU.....	82
7.3 HELP MENU.....	83

1. INTRODUCTION

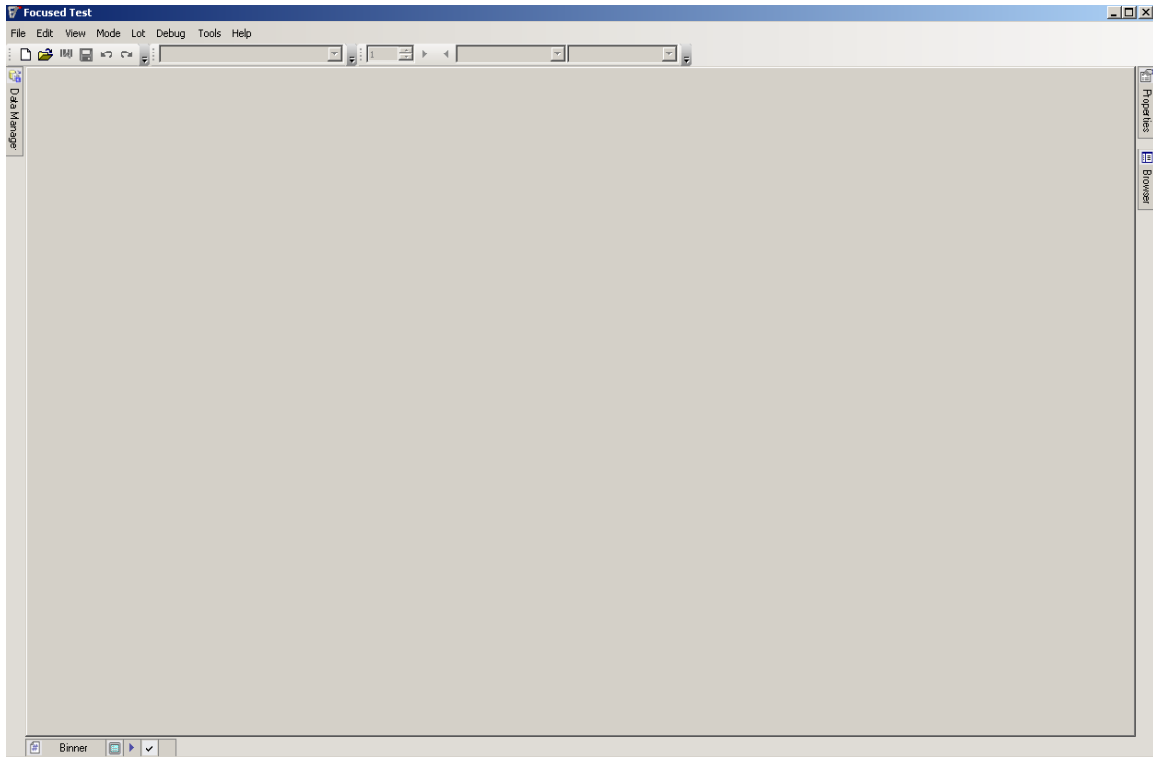
This manual is an introduction to the FTI Studio user interface. It contains an introductory section on its various aspects, as well as details about each function. The FTI Studio Production user interface is documented in the FTI Studio Operator Manual.

The user interface contains three main areas:

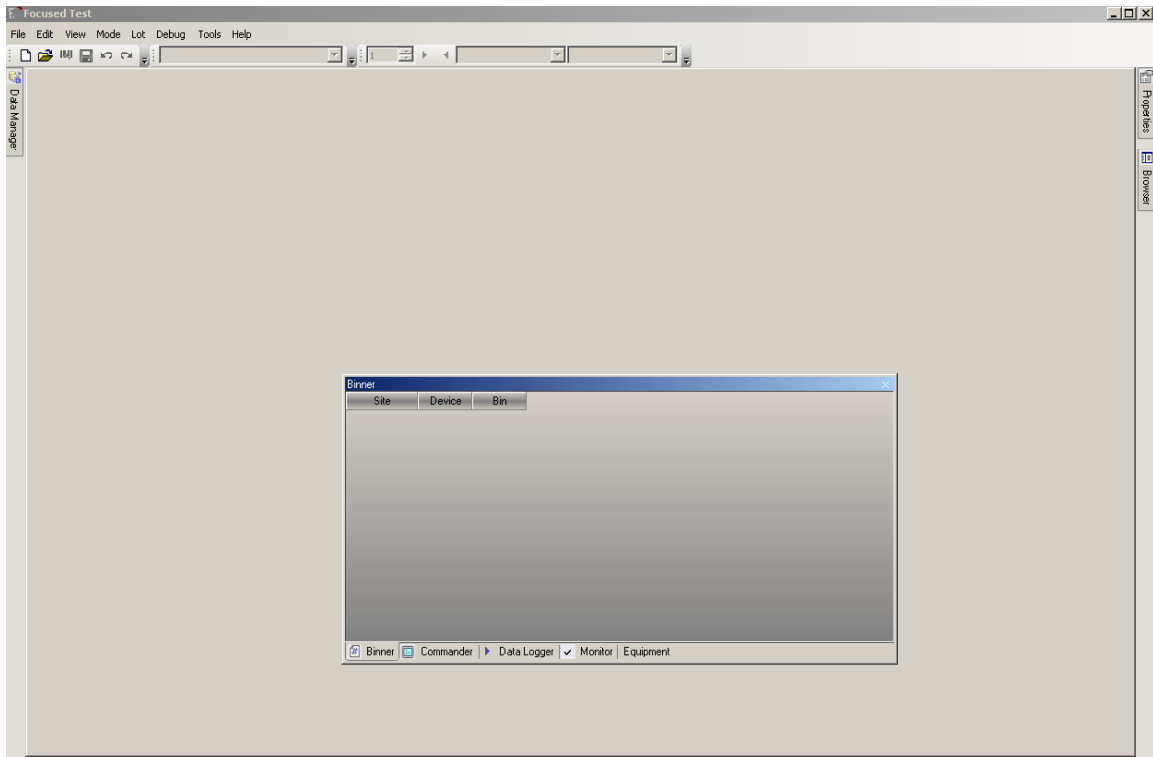
- Menubar
- Toolbar
- Workarea



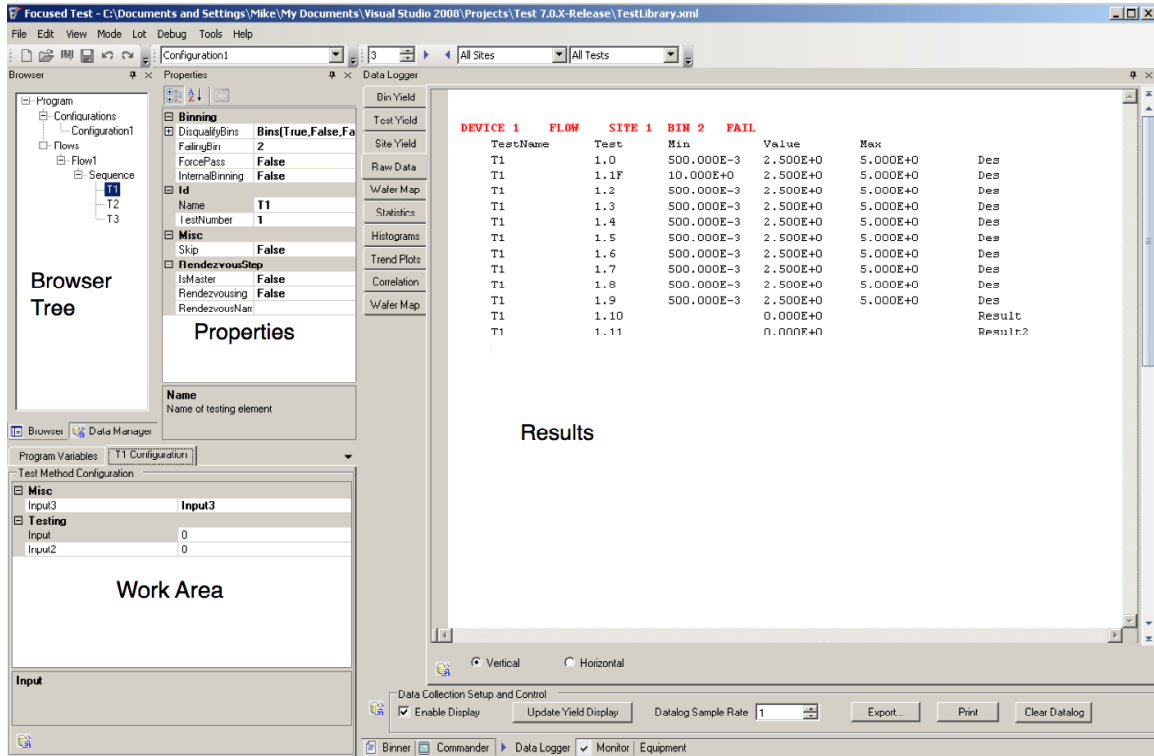
The workarea uses docking windows to organize information about a test program. Each docking window has several tabs that group information. Windows can be un-pinned so that the window closes to the edge:



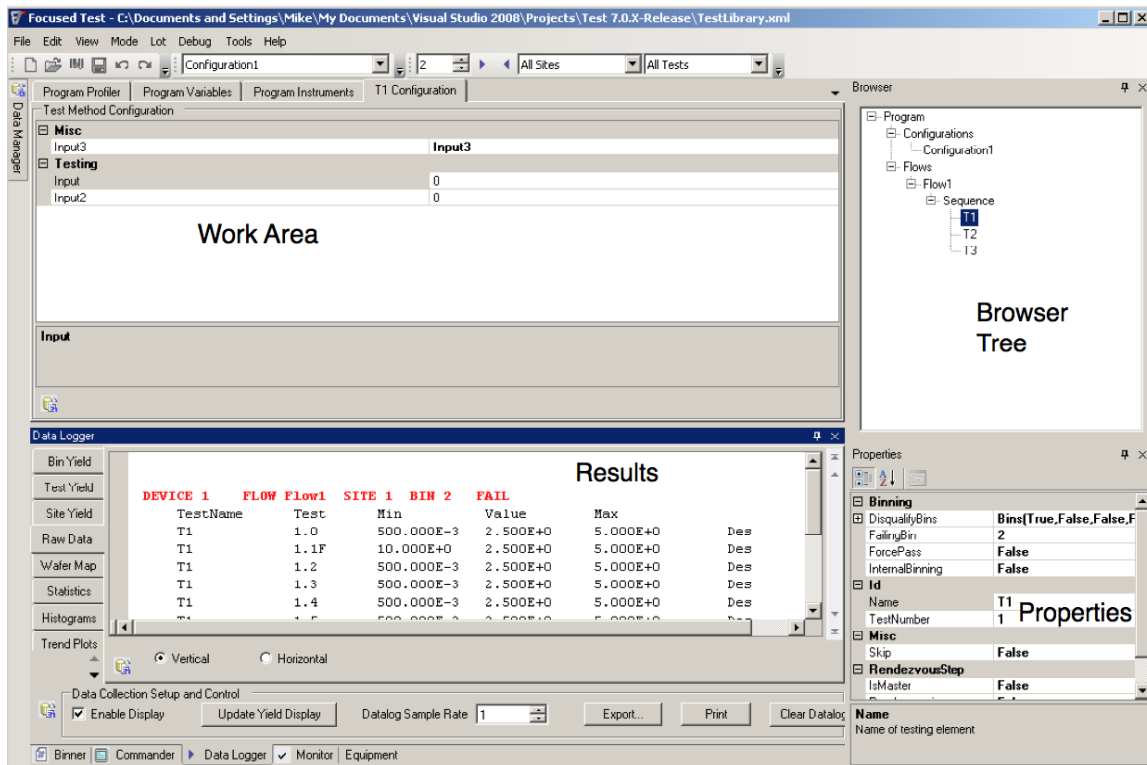
Un-pinning windows gives more working space on small screens. Windows can also be un-docked and free floating. Simply drag the title bar of a window to un-dock it:



It is also possible to drag a tab, pull off a tool, put it in its own window, or move the tab to another window. A window can also be re-docked to any side by dragging to the edge of the main window. Here is an example from one customer who wanted to organize their work differently. Their goal was to focus on result data while making minor changes to the test program, so the datalog window was docked on the right side and uses more than half the space.



Here is the traditional layout that you get by default.



The windows are organized logically into three areas:

- Browser and Properties
- Results
- Work Area

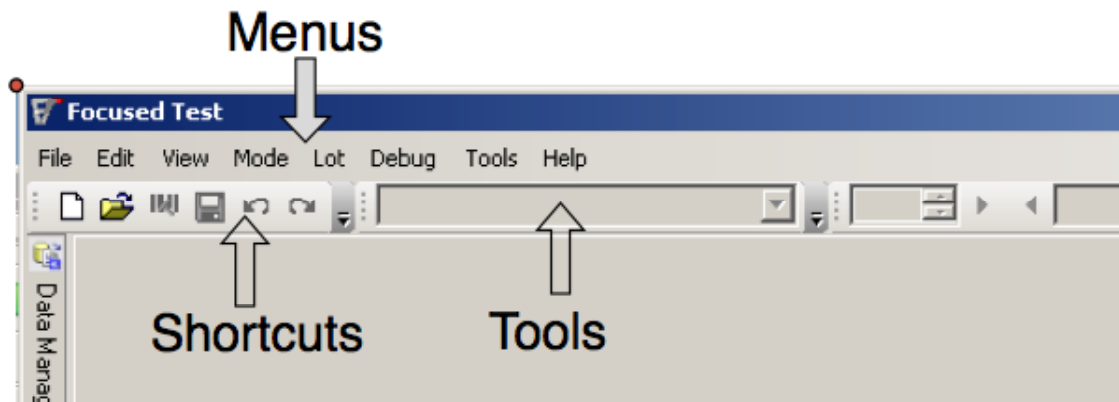
The Browser contains a tree representation of a test program. Right clicking on a node changes the Properties so you can configure the node.

The Results area contains the datalog, yield, prober/handler setup, and command line.

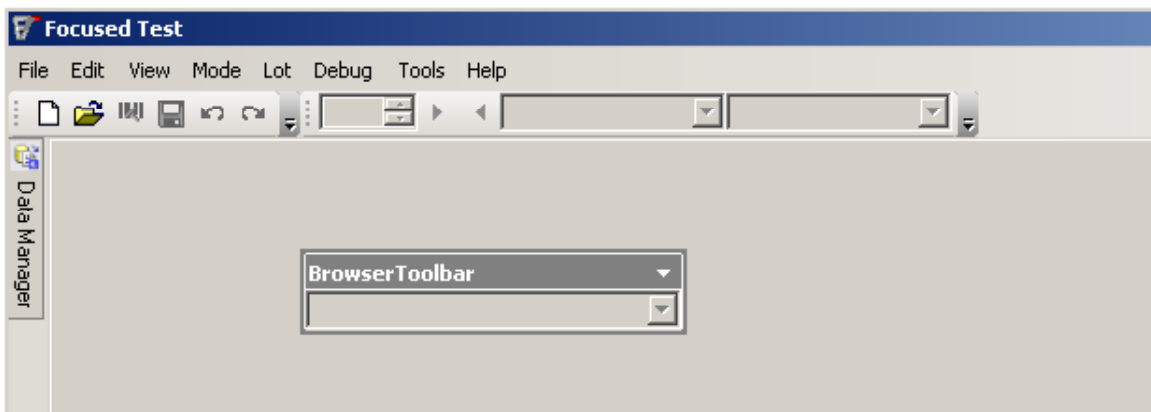
The Data Manager manages reusable tool setups.

The Work Area contains test configurations, program variables and limits, and tools. Right clicking a test program node and using the menu will create windows in the Work Area.

The toolbar contains simple short cuts to menu items, and simple tools that are not found in the menus:

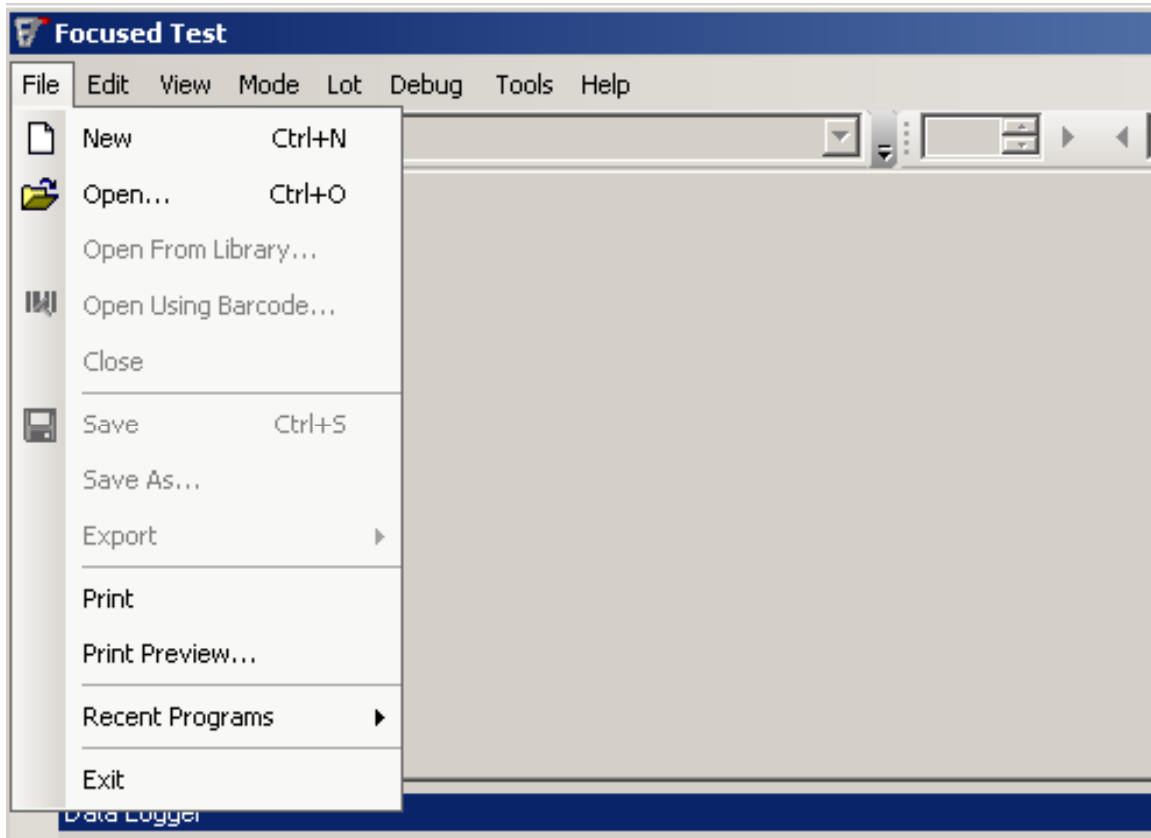


The toolbars can be un-docked by dragging their handle, found at the left of each toolbar:



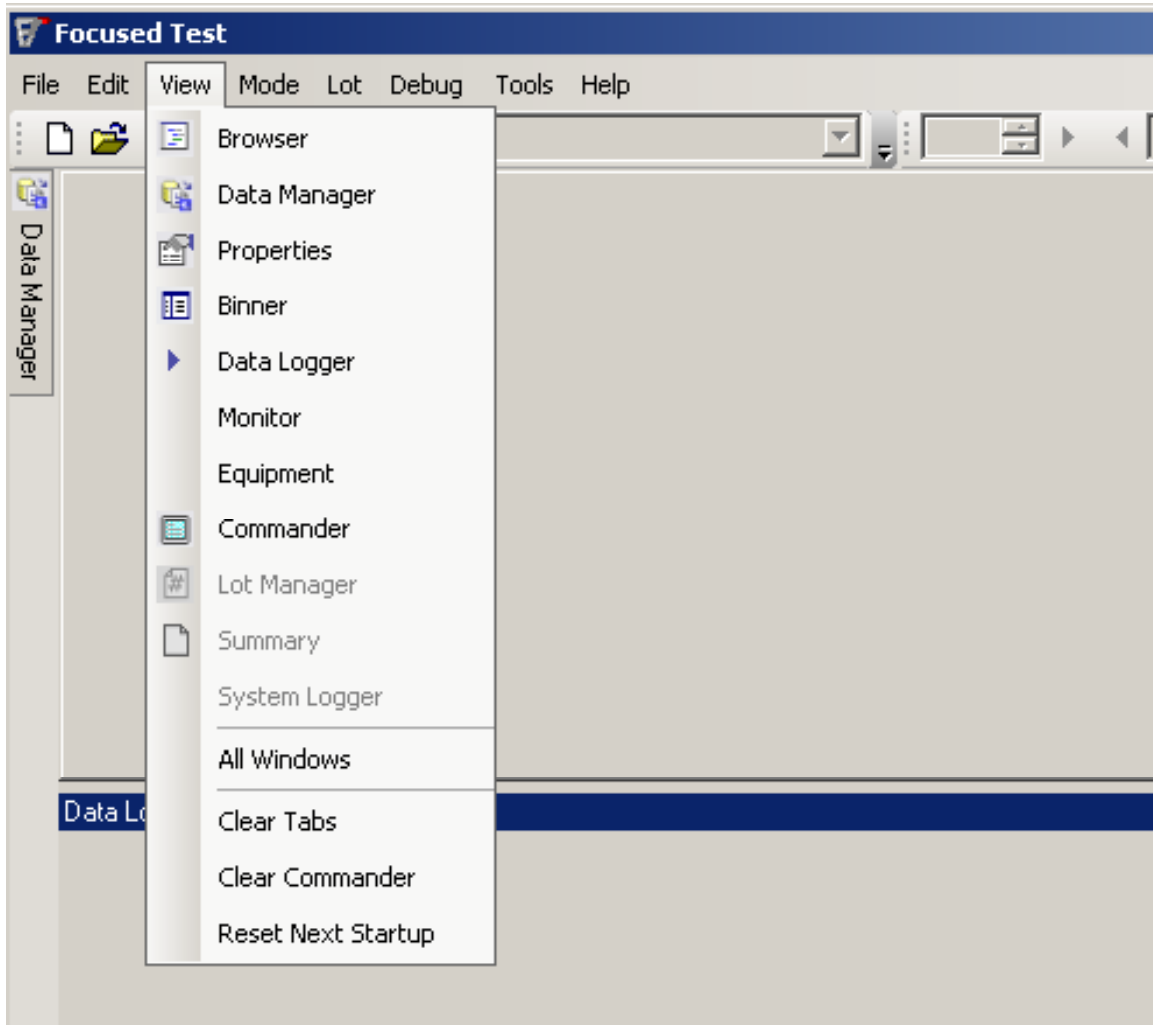
Toolbars and menus are reset to their default locations when with the application is restarted.

2. QUICK MENUBAR OVERVIEW

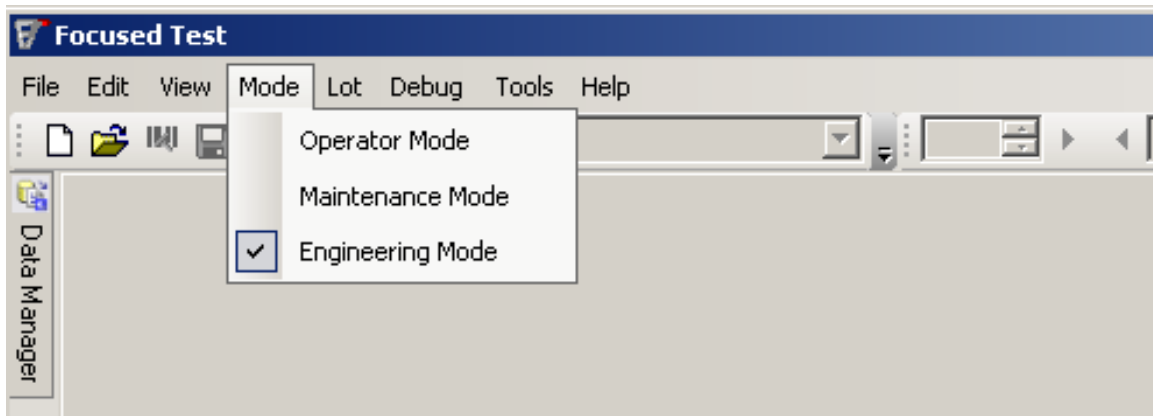


The File menu is for opening, closing, and saving test programs. In operator mode, Open From Library... opens a dialog that allows the operator to open programs from the test program library. Open Using Barcode... allows the operator to load programs from the test program library using a bar code reader. You can also open programs from the command line by starting FTI Studio™ like:

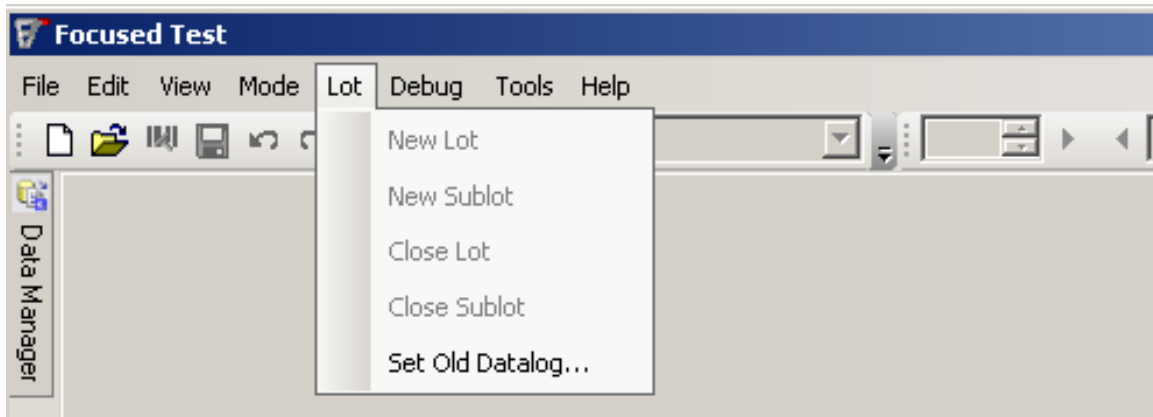
```
FTIStudio -program C:\Directory\TestProgram\program.xml
```



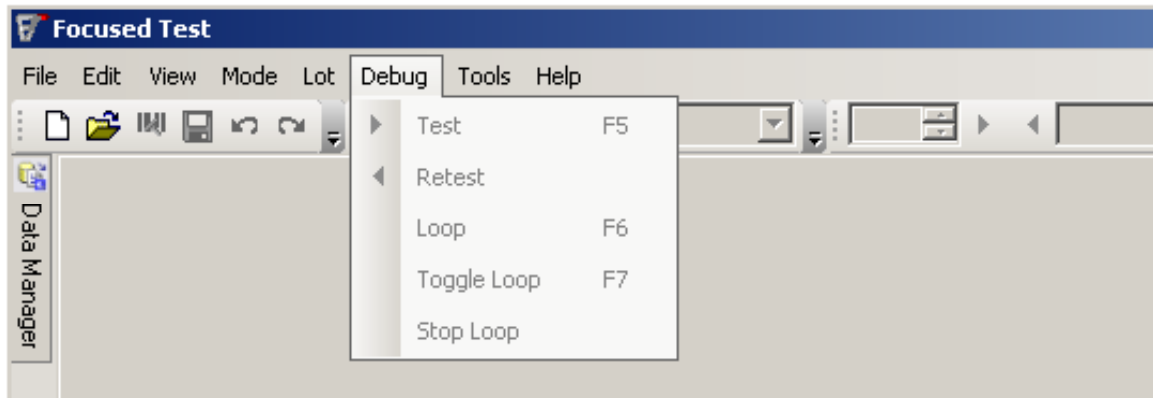
The View menu is for displaying windows that have been closed and are not visible.



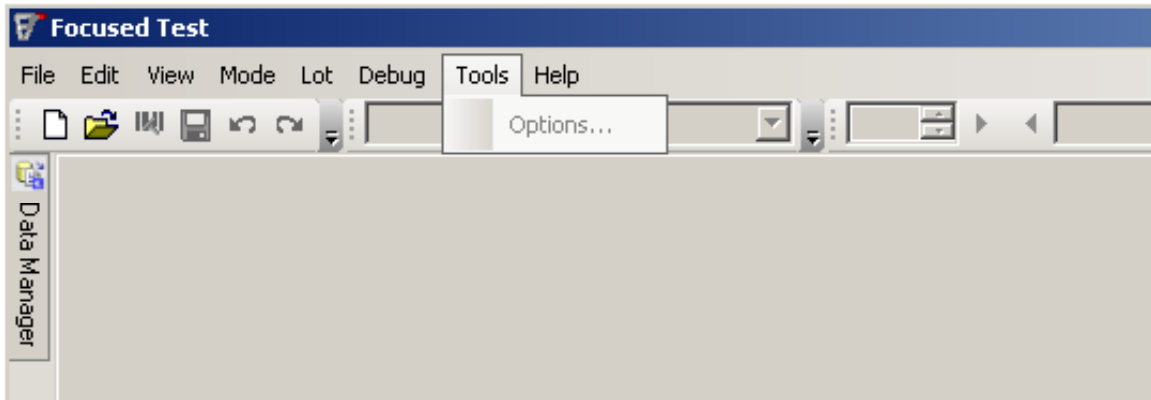
The Mode menu changes the mode of the user interface, which determines what tools are available. For example, an operator can't use the program browser.



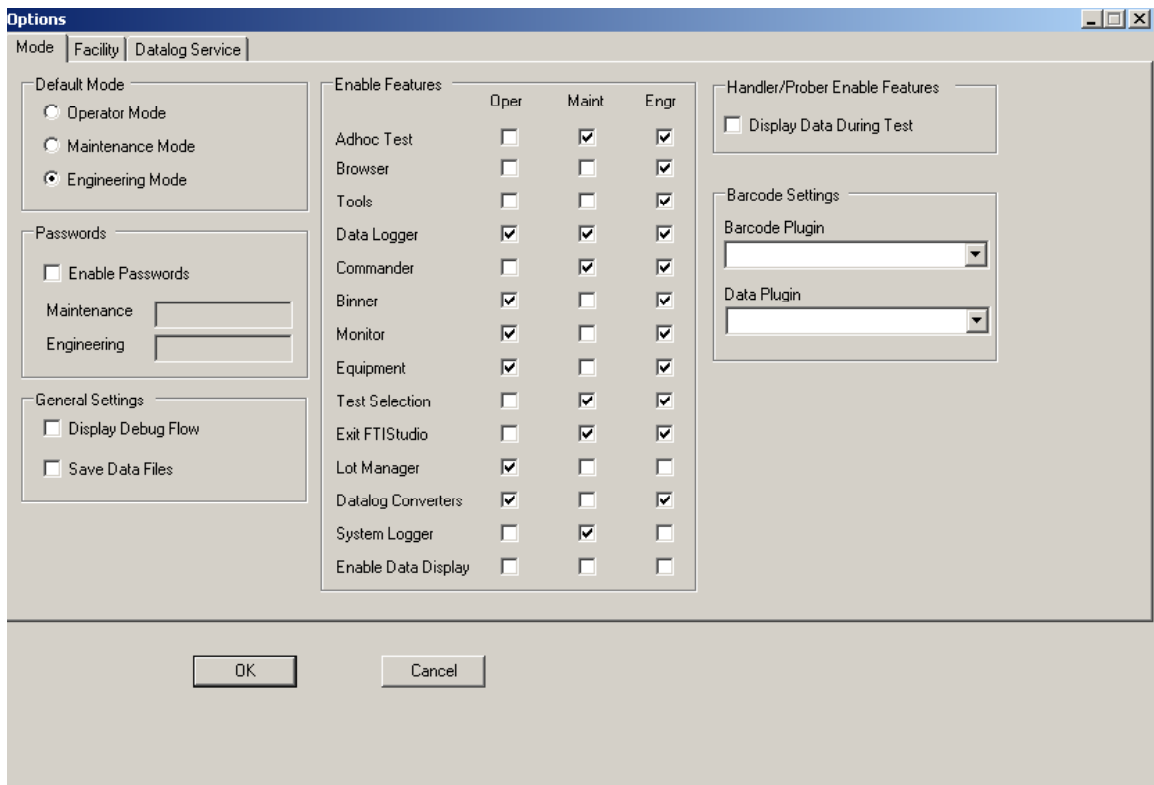
The Lot menu is for creating and closing lots during production operation.



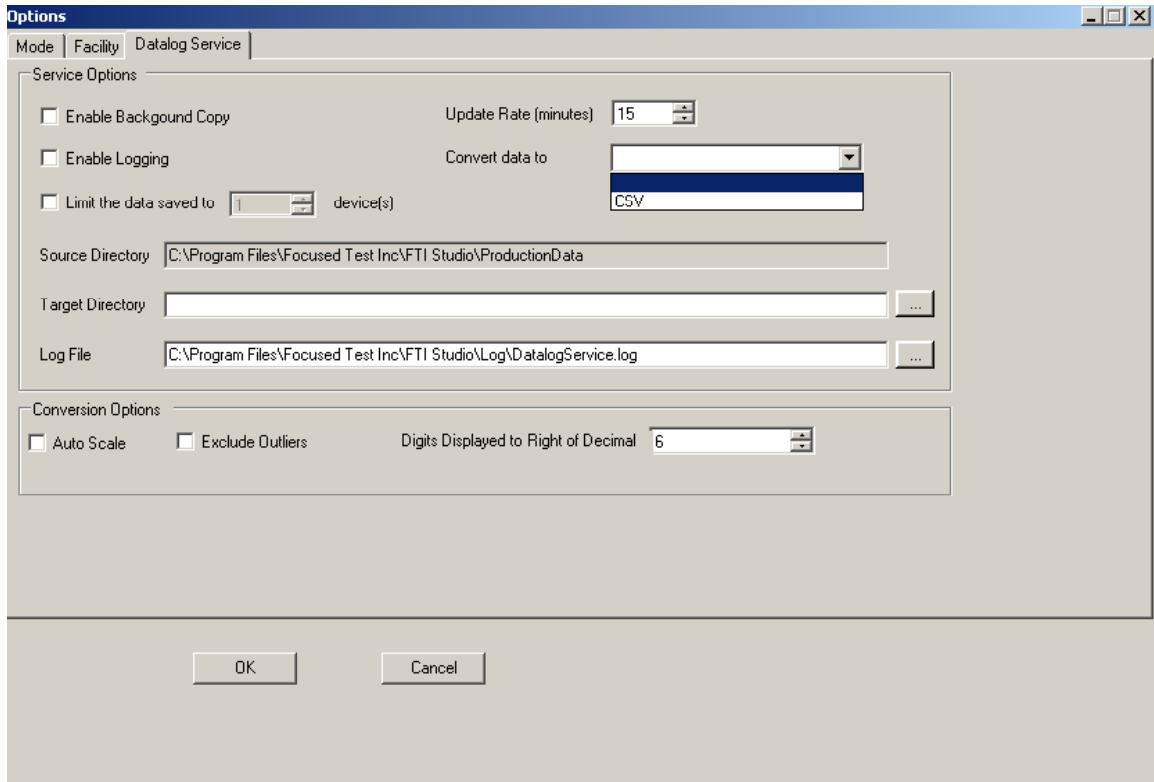
The Debug menu is for running and looping test programs during test program debug.



The Tools menu is for setting persistent user interface options. Options can only be changed in maintenance mode.



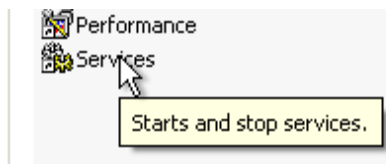
The Mode tab is used to set the default user interface mode and manage passwords. It also allows control over which features are available for each mode and sets the bar code reader plugins.

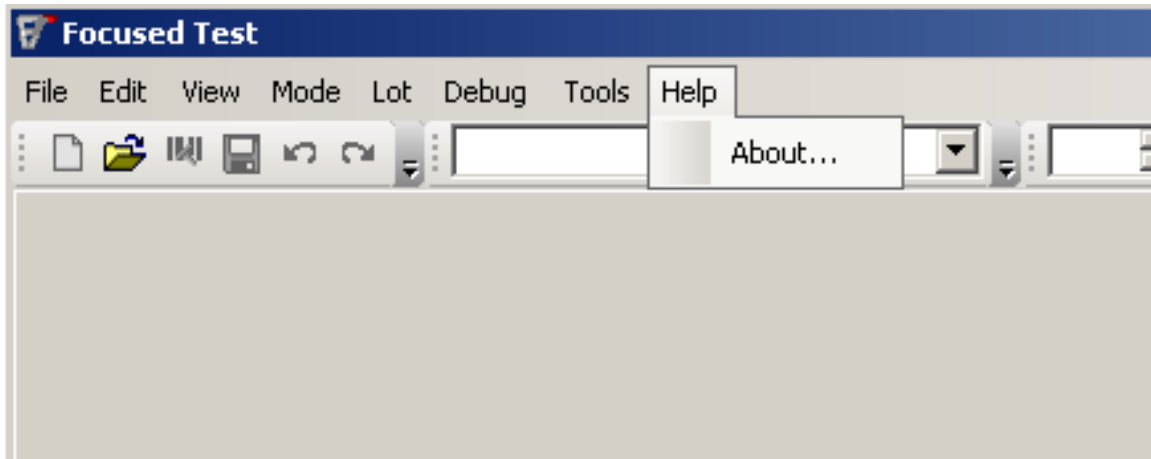


The Datalog Service tab controls automatic data conversion. By default all raw files generated by a Close Sublot are copied to (\$INSTALL_DIR)\ProductionData. When Enable Background Copy is checked, the datalog files are moved to the TargetDirectory. The Update Rate controls the interval of the copy operation. If a Convert data to is set, the data is also converted to the specified format. You can enable logging to debug the service and limit data saved. Changing the Log File from the default is not recommended.

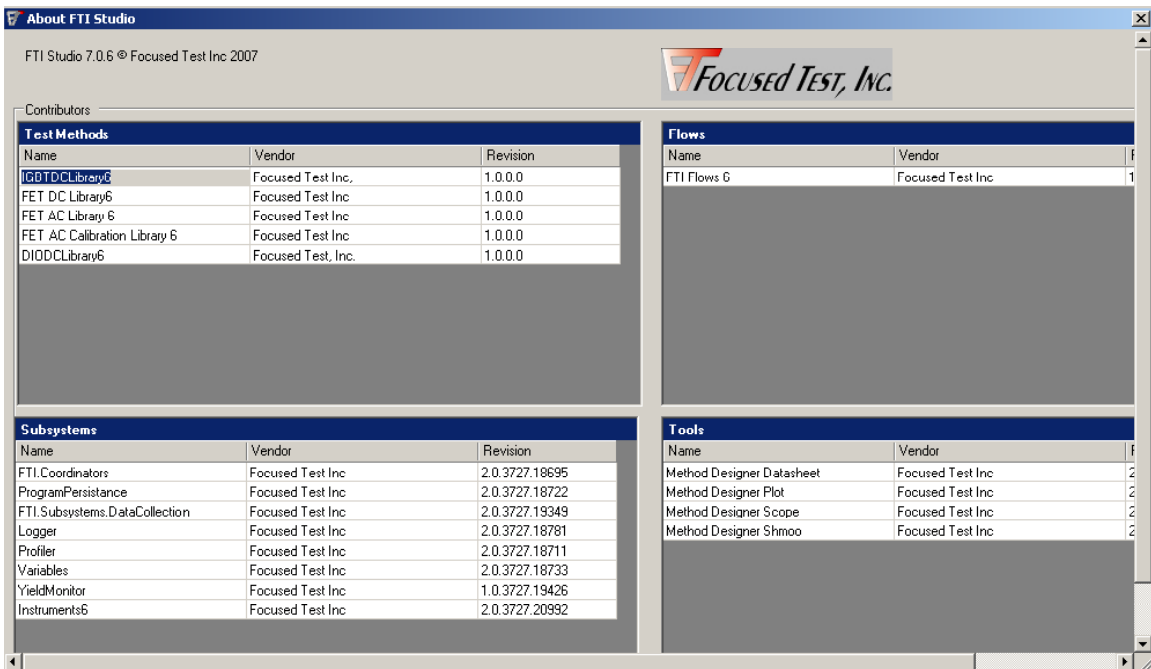
The conversion options will apply to any converters that choose to use them, mainly the CSV converter. Typically people set Auto Scale and use 4 decimals.

For the data to be copied and converted, the Datalog Service service must be running. Use the Services plugin in the Control Panel under Administrative Tools. This is a MS Windows tool, not a FTI Studio tool.



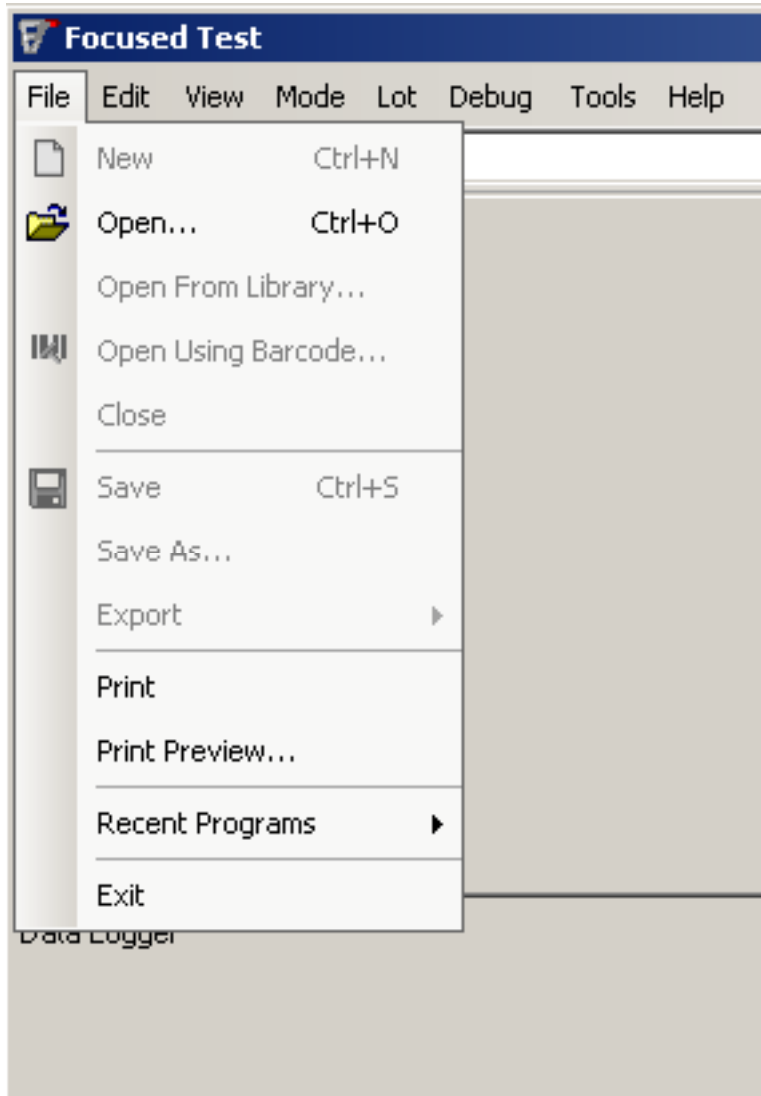


The Help menu contains the About box where you can find the FTI Studio revision and the revision of all tools and plugins.

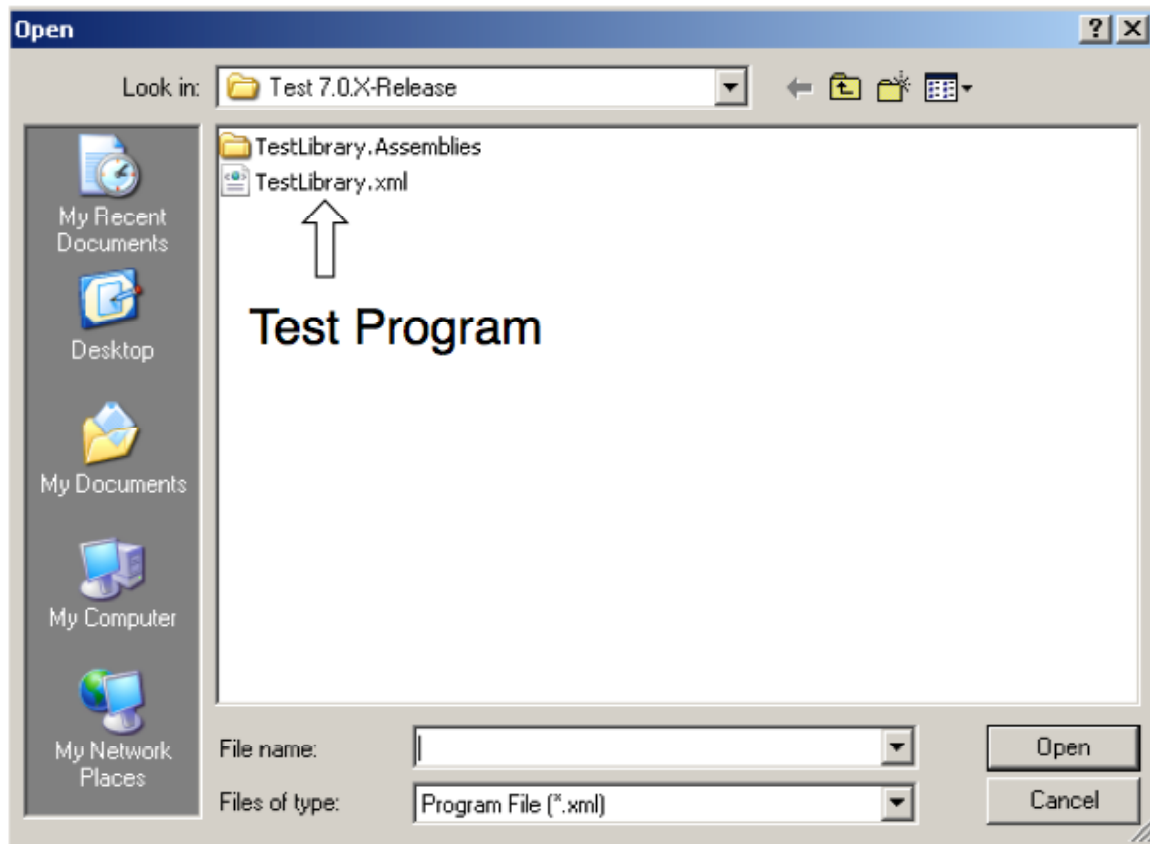


3. OPENING, SAVING AND CLOSING TEST PROGRAMS

Programs can be opened, saved, and closed via the menubar or toolbar:



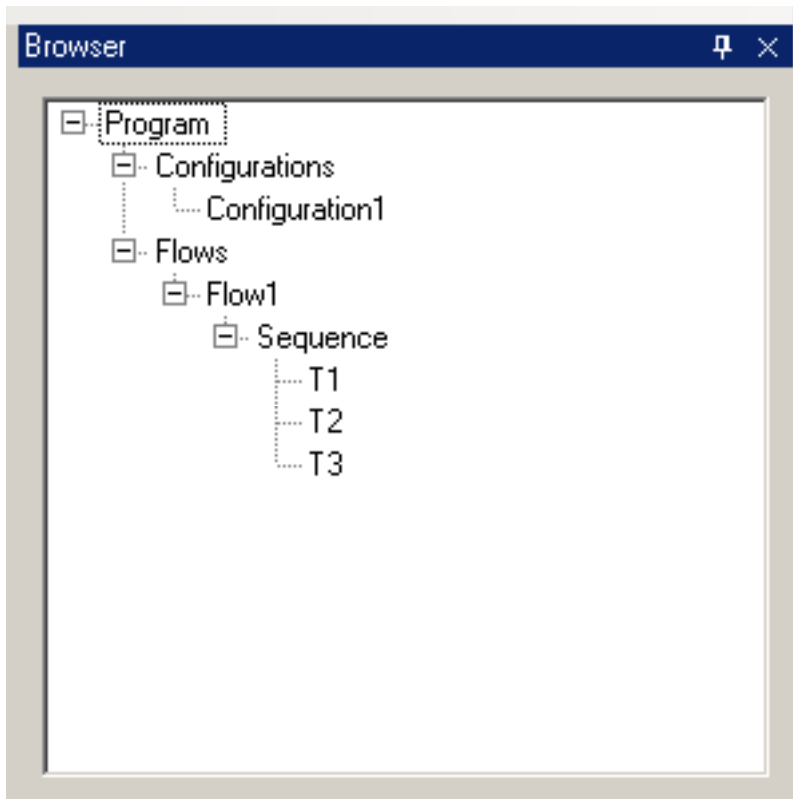
When Open... is pressed, a file dialog pops up. This open file dialog is used to browse for a test program. Test programs are saved as xml files:



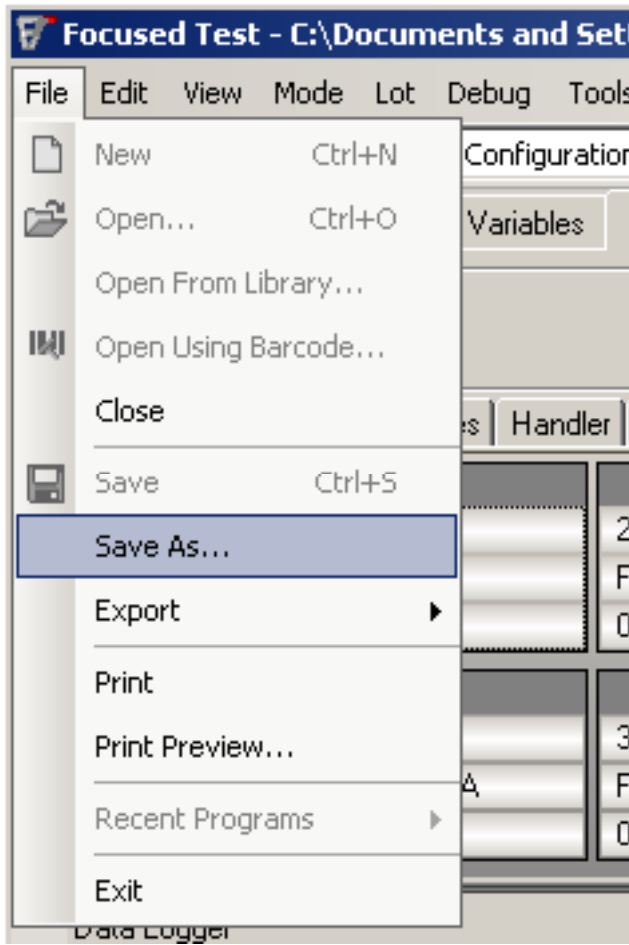
The above dialog shows a test program that resides in a directory called Test 7.0.X-Release. The entry point to the program is `TestLibrary.xml`. Test programs can use any name, but must end in `.xml`.

The `TestLibrary.Assemblies` directory contains local test methods (not a shared library). If you move the program file, move this directory with it if you are using tests in it. Most people are not using local test methods and are using FTI supplied libraries that are checked into the library.

The program is opened by clicking on the entry file (`TestLibrary.xml` in this case), and then pressing the Open button. Once the program is opened, the program tree is displayed in the browser:



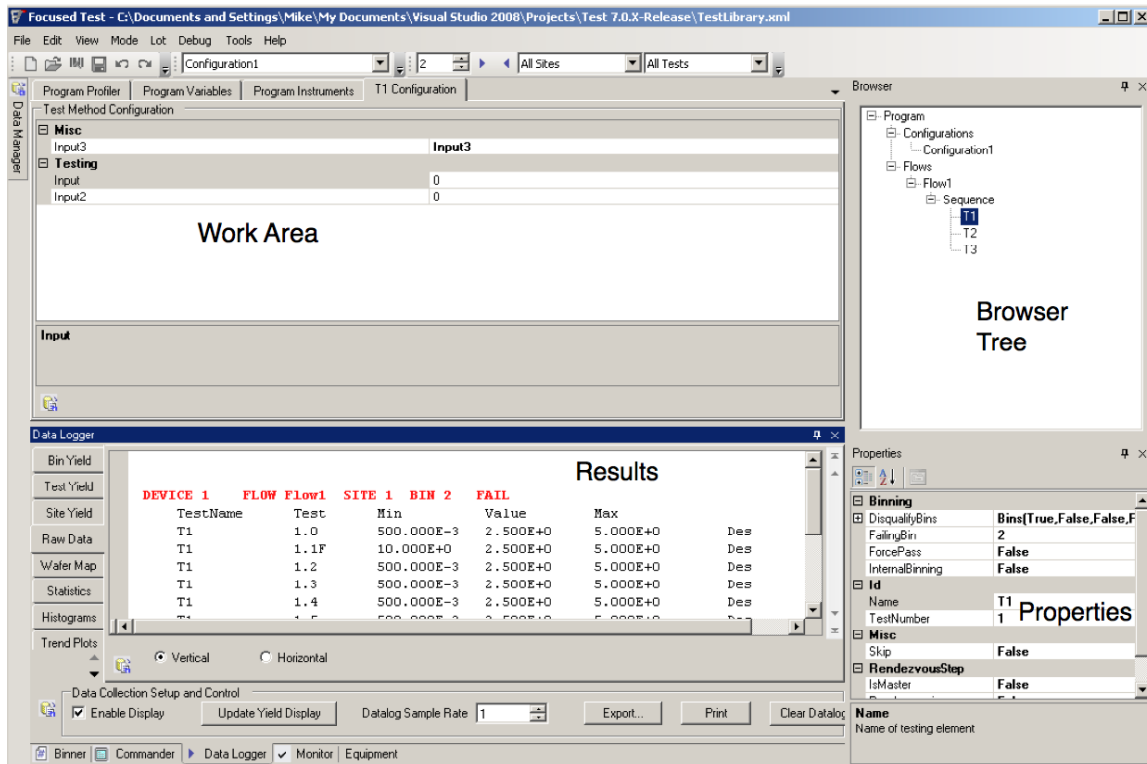
Saving a program is simply a matter of using the Save or Save As... option in the menubar or toolbar. However, menubar and toolbar options are only enabled when they have meaning. For example, the Save As... is disabled until a program is opened, and Save is disabled until a program is modified. Open... will be disabled until a program is closed:



Therefore, to open a different program, the close menu must be actuated.

4. BROWSING AND MODIFYING TEST PROGRAMS

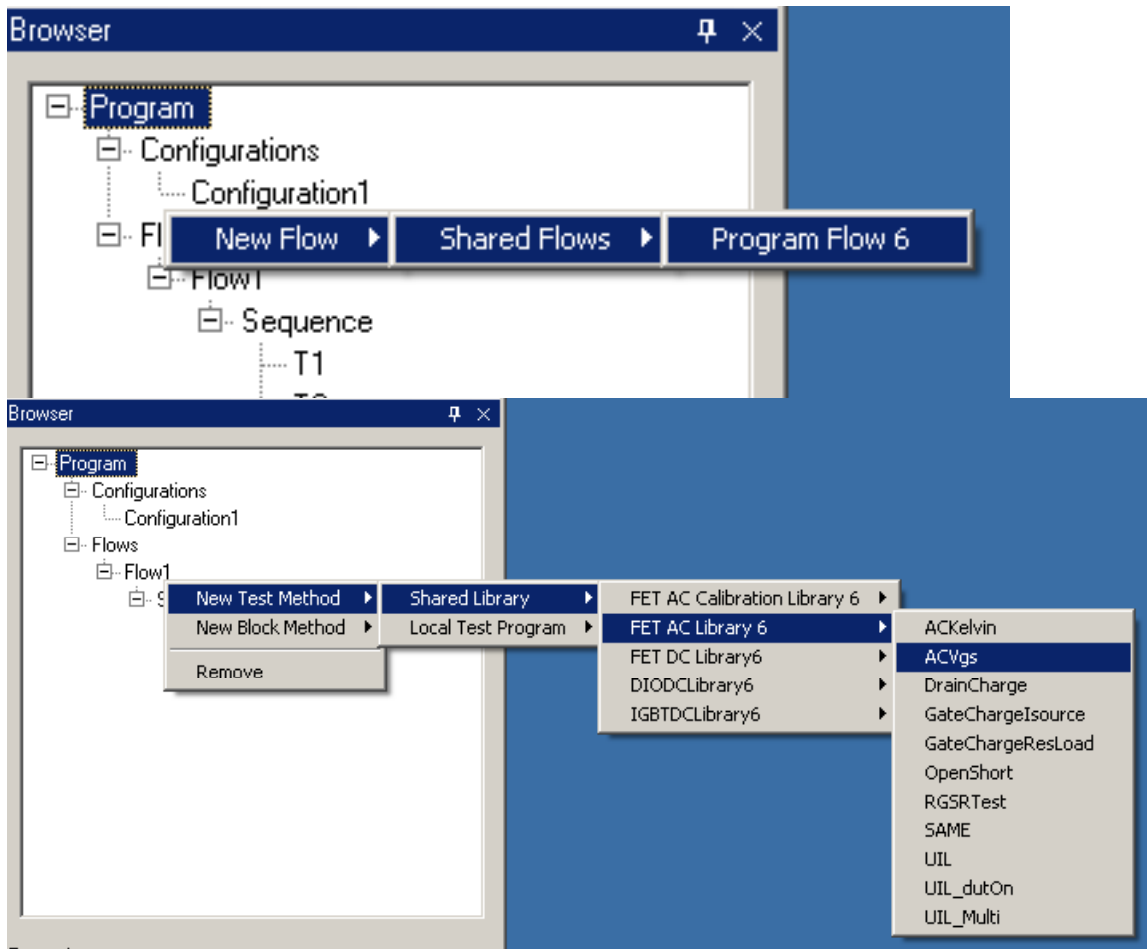
Working with programs involves three areas of the display:



The program is represented as a program tree in the browser. When a node of the tree is selected, the node properties will be updated in the Properties window. Any tools selected from the node are placed in tabs to the left of the browser. The above diagram shows the Instruments tool, the Variables tool, the profiler tool, and the configuration of test T1.

4.1 ADDING AND REMOVING NODES

To add a node, right click on a node that contains flows or steps. This will cause a popup menu to be displayed. For example, right clicking on the flows or steps node results in:



The menus can be navigated to the item you want to insert.

4.1.1 CONFIGURATION NODES

Configuration nodes represent a complete setup that can be run. It contains a number of flows, limit selection, acceptable bins, and other properties required to run. The toolbar contains a dropdown menu to select which configuration will run when the user presses F5 or uses a handler/prober to test devices.

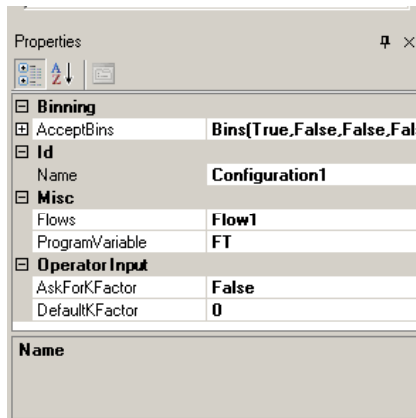
4.1.1.1 BINNING

Binning works of a principle of removing goodness. Internal to the system is the idea of current bins. There are 32 current bins in the system (1 to 32), and when testing begins all 32 current bins are good (true). When testing completes, the device bin number is the smallest current bin that is good (true).

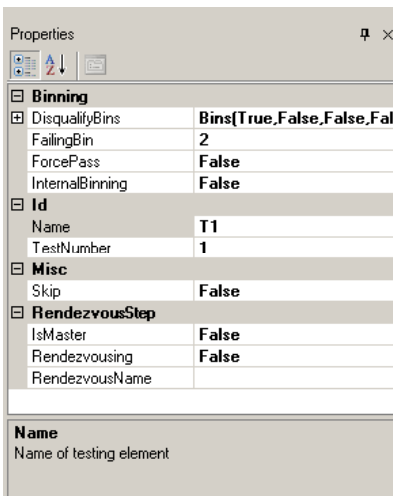
Accept bins are the bins that are good parts. For example, suppose we have a two grade part. Let's say grade A is bin 1, and grade B is bin 2. The accept bins would be set to bins 1-2. Any other bin would be a device failure (3-32).

During test, bins are disqualified. For example, if the current bin 1 is disqualified (false), then the smallest current bin remaining (true) is bin 2. If testing completes and current bin 2 is still good (true), the device bin is 2 and since this is an accept bin, the device passes. If both accept bin 1 and accept bin 2 are disqualified (false), then the device bin is 3, which is a device failure because it is not in the accept bins.

Accept bins is set in the configuration properties. Disqualify bins are set in the step properties.



Configuration Properties



Step Properties

A typical two grade test program is easy to setup. Suppose we have the following scheme:

AcceptBins = 1-2

T1 Continuity, DisqualifyBins = 1-2

T2 Grade A Test, DisqualifyBins = 1

T3 Grade B Test, DisqualifyBins = 1-3

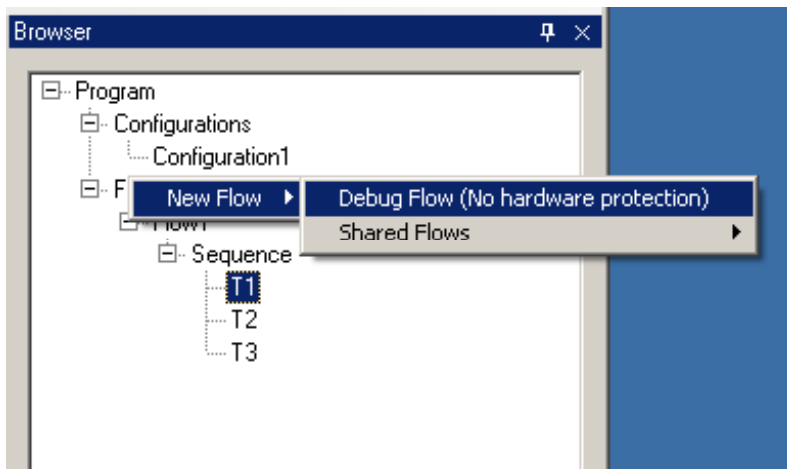
T4 Test, DisqualifyBins = 1-4

T5 Test, DisqualifyBins = 1-4

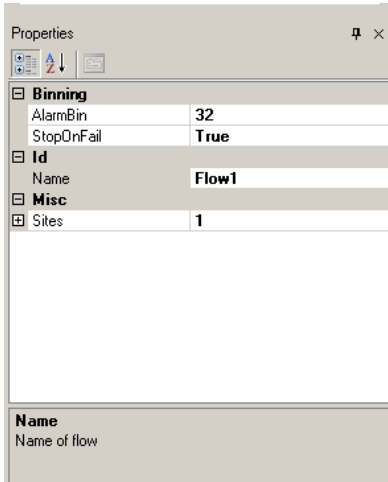
A bin 1 is an A grade, bin 2 is a B grade, bin 3 is a continuity failure, bin 4 is a grade B failure, and bin 5 is a failure.

Note: The step also has a property called FailingBins. This is a short cut for DisqualifyBins. When you set this property, it sets all the bins less than the value. For example, setting FailingBins to 5 sets the DisqualifyBins to 1-4. For simple binning schemes, this is easier than setting individual disqualify bins.

4.1.2 FLOW NODE TYPES



There are two types of flow nodes. A default flow is a standard node provided by the system. It has the following properties:



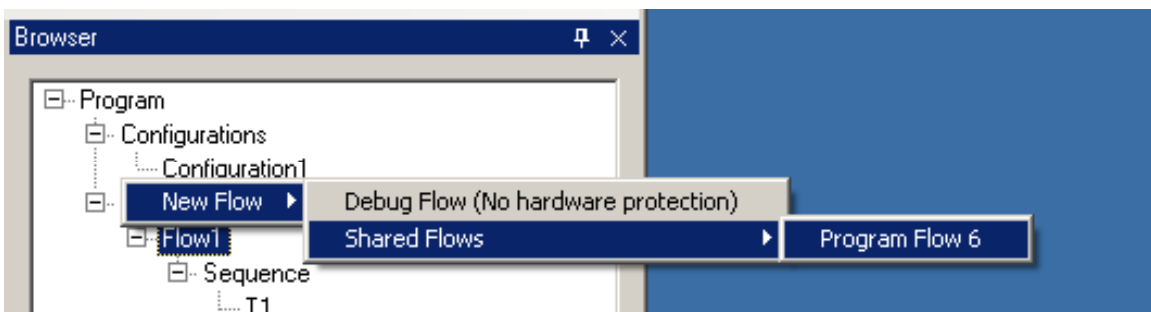
The alarm bin is the bin used when there is an alarm. Normally this is never set to an accept bin. These settings are global to the flow.

The stop on fail property determines whether testing continues after a failure occurs. All of these properties are persistent, therefore, if stop on fail is set to false, and the program is saved, the next time the program is run, it will still be false. The name property is the name that is displayed in the tree and is used to uniquely identify the flow.

The sites property sets the sites that this flow supports.

4.1.3 SHARED FLOW NODES

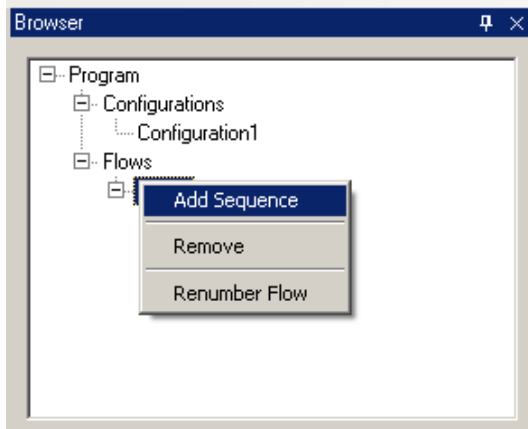
Shared flow nodes are nodes that are created by users and placed in the flow library. To create a shared flow, one creates a Visual Studio C# project, and then creates one or more flow classes that inherit from the system supplied Flow class. Once the assembly is compiled, it is checked into the flow library within the Library Manager. The next time FTI Studio™ is run, the new flows will show up in the menus. In this example you see the standard flow shipped with a FTI 1000 that will reset instruments at the end of test.



Note: The FTI Studio installer comes with a custom flow for FET testing which should always be used with the FTI 1000.

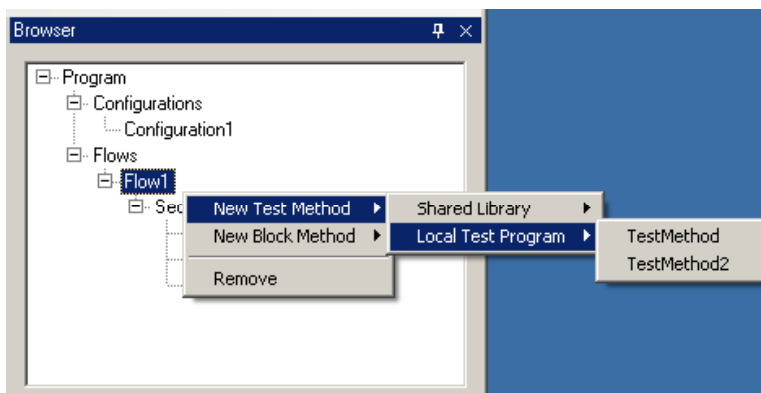
4.1.4 SEQUENCES

Sequences are added to flows with the pop up menu like this:



You can only add steps to a sequence, not directly to a flow.

4.1.5 STEP NODE TYPES



Step nodes represent instances of a test method class. The concept is that a test method can be instantiated multiple times. So when adding a step, you choose a test method, and a step will be created that uses that method.

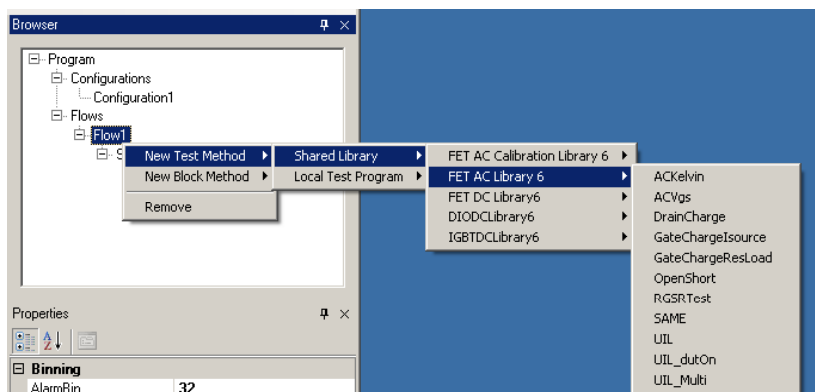
There are two categories of test methods. A shared test method is one that is found in the Library Manager. Test methods can be added to libraries by creating C# assemblies with test method classes in them, and then checking them into the Test Assemblies tab in the Library Manager. The library can either be local to the computer running FTI Studio™ or a shared directory on the network.

The other category contains test methods that are local to the test program. These methods are in C# assemblies that reside in the test program file tree and are not used by any other test program.

The main differentiator in the two categories is that shared methods are methods that are designed to be reused. This means they can be configured in the test program such that they can test more than one device type. Local methods are usually highly specific to a device.

A typical usage is to write local tests, evolve them, and when they become stable and reusable, move them to the central library for everyone to use.

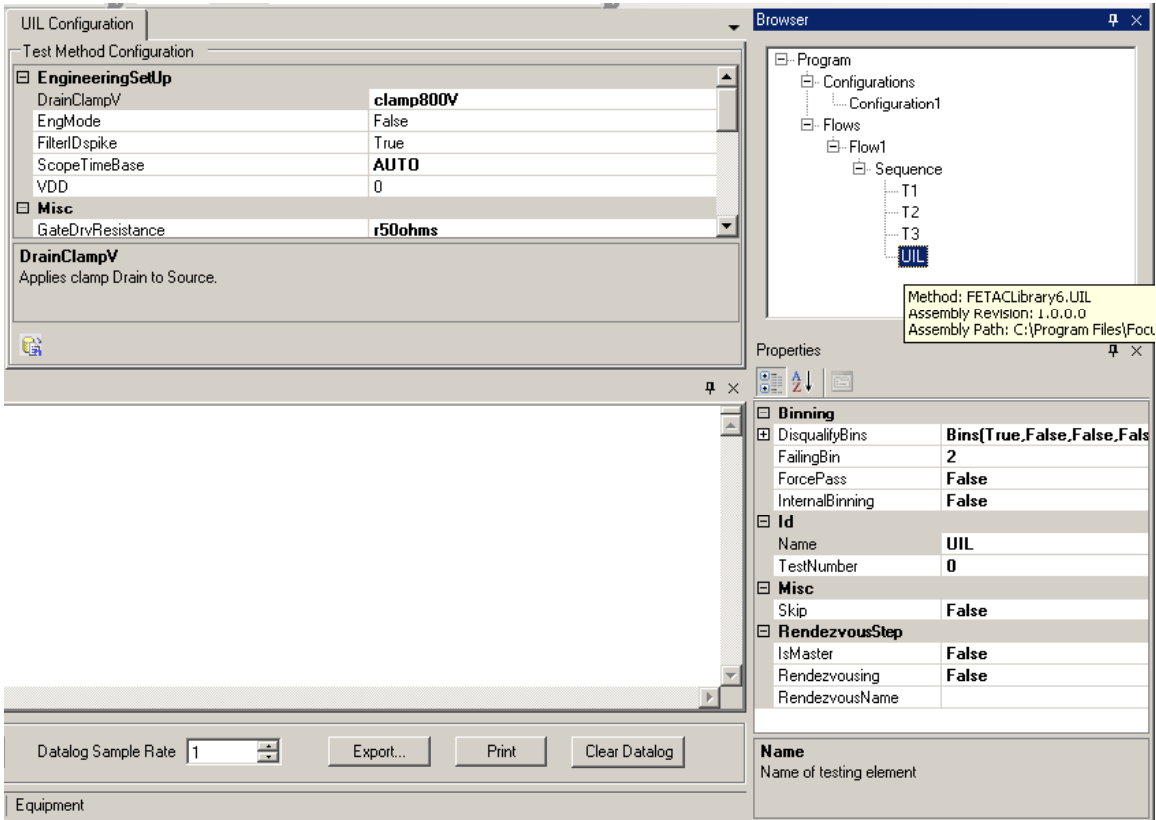
Each shared library has its own menu in the menu tree.



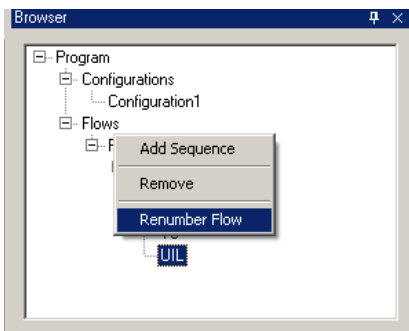
When the menus are navigated all the way to the right, the actual tests are displayed. Each test shown above is a test method class in the assembly.

Let's review this one last time. Test classes are put into a Visual Studio project and produce an assembly. Assemblies are .dll files. The assemblies are checked into the Library Manager, or placed in the Local Test Program directory. FTI Studio™ lists the assemblies under the Shared Library menu or the Local Test Program menu, and lists the classes under the assembly. There is a correspondence between the assembly/class and the menu/submenu.

Notice that there is a new node called “UIL”. Once the step is added, it is nice to change its name to something meaningful. Click on the new step and view the properties:



You should set the test number to some unique number. Two steps may not use the same test number. You can renumber all test by right clicking on a flow node:



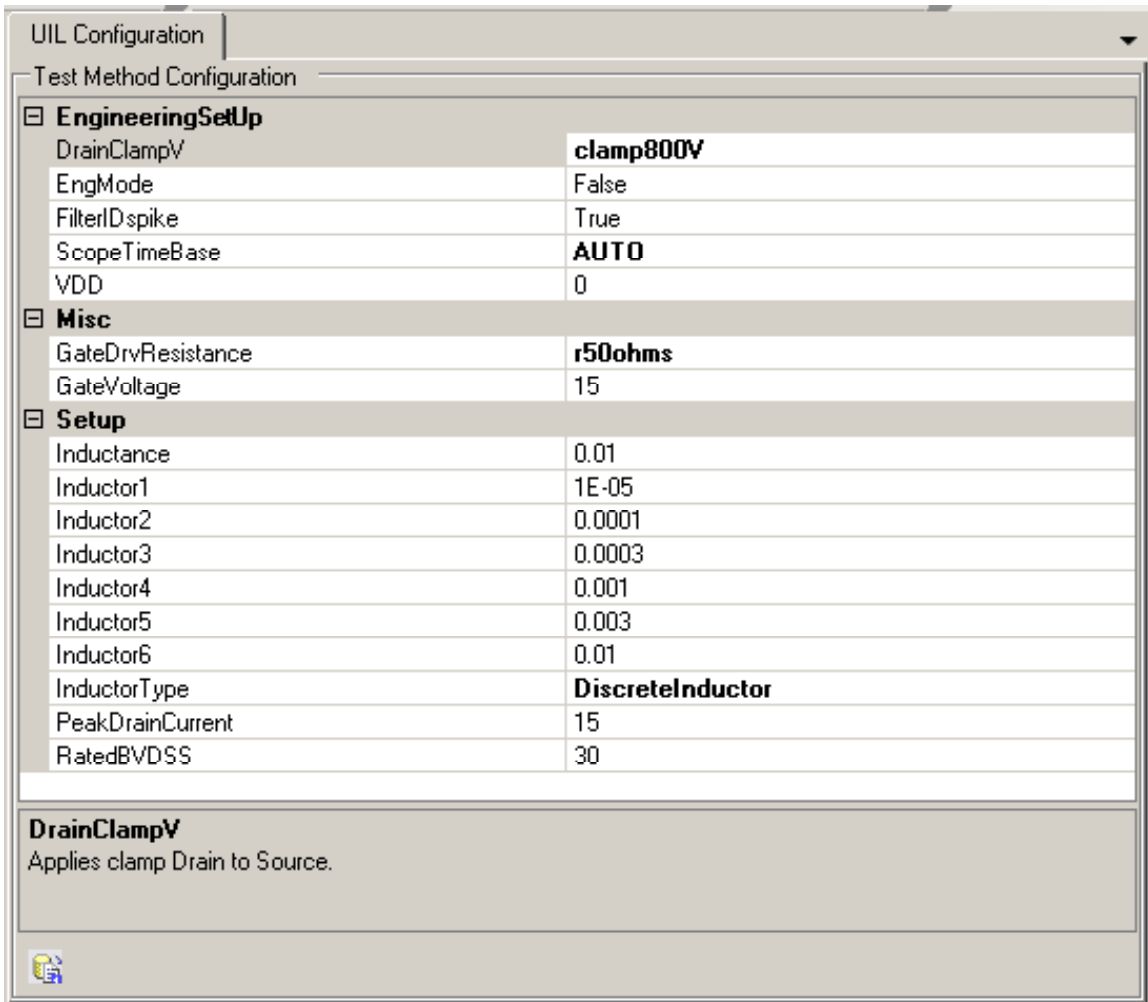
4.2 STEPS AND METHODS

As mentioned above, there is a distinction between steps and methods.

Steps are the visual nodes in the tree. Each step has a test number and logs data.

Methods are C# classes that perform the testing. Each step operates with an instance of a test method class.

The distinction becomes clearer when considering how a test is configured. Each test method class has a set of configuration parameters. These are displayed to the left of the browser when a step is selected:



The behavior of the test method is dependent on the settings in these parameters. Each test method specifies its own unique set of parameters. This allows one to use a test method more than once in a flow with different parameters.

For example, suppose one has a test method that measures a clock out frequency, and this test method has the following configuration parameters:

- DUT Supply Voltage

- Logic input and output levels

The test method must be run at three supply voltages: normal, high, low.

To implement this, three steps are added, where all three steps use the same test method. This will mean there are three steps, each with their own name, and each step will have its own instance of the test method. Each step will maintain its own set of configuration parameters.

A good way to conceptualize this is to think of the step as a step in the process of testing the device, where each step has a configuration. Think of a test method as the C# class that has the instrument code that makes the measurement, and specifies what the configuration parameters are and how they are applied to the instrument code.

4.2.1 STEP PROPERTIES

Each step has several properties:

Binning	
DisqualifyBins	Bins(True,False,False,Fals
FailingBin	2
ForcePass	False
InternalBinning	False
Id	
Name	UIL
TestNumber	0
Misc	
Skip	False
RendezvousStep	
IsMaster	False
Rendezvousing	False
RendezvousName	
Name	
Name of testing element	

The disqualify bins determine which bins are disqualified if this test fails. The idea is that there are 32 bins, and they are all initially qualified. As you disqualify bins during failure, the smallest bin number that is qualified is the current bin. As long as the current bin is in the accept bins, the device is good. When testing is complete, the current bin is the bin assigned to the device. These are soft bins. Hard bins are controlled by the handler/prober plug-in. The FailingBin property is a short cut for simple binning situations. It will automatically change the DisqualifyBins for you.

The Force Pass property will cause a test to pass no matter what the limits say. This is a way of ignoring failures during debug. Skip causes the step to not run.

The name is the name that is displayed in the tree.

The test number is used by the datalogger and must be unique. No two steps may have the same test number.

The Rendezvous properties are for dual channel devices. To use it you create two sequences, and put different steps in each one. Code a test that switches a mux so that AC and DC boards swap channels. Make that step a Rendezvous step and make one a master and the other slaves. When execution is stopped on all sequences, the master has its code run and you make the mux switch, then all the sequences are allowed to continue. You can have as many Rendezvous steps as you want and use them for any kind of synchronization you desire. Just be sure to use different group names and make sure each sequence keeps the order of the Rendezvous steps the same.

These properties are saved when the test program is saved.

4.2.2 CONFIGURATION PARAMETERS

The screenshot shows a software window titled "UIL Configuration" with a sub-tab "Test Method Configuration". The window contains a tree view with three main sections: "EngineeringSetup", "Misc", and "Setup". Each section contains a list of parameters and their values. Below the tree view, there is a detailed description for the "DrainClampV" parameter.

Section	Parameter	Value
EngineeringSetup	DrainClampV	clamp800V
	EngMode	False
	FilterIDspike	True
	ScopeTimeBase	AUTO
	VDD	0
Misc	GateDrvResistance	r50ohms
	GateVoltage	15
Setup	Inductance	0.01
	Inductor1	1E-05
	Inductor2	0.0001
	Inductor3	0.0003
	Inductor4	0.001
	Inductor5	0.003
	Inductor6	0.01
	InductorType	DiscreteInductor
	PeakDrainCurrent	15
	RatedBVDSS	30

DrainClampV
Applies clamp Drain to Source.

Configuration parameters are created in the test method C# code. See the programmer's manual for details on how to create them.

Configuration parameters can be modified by placing the mouse in any box on the right side and entering information, or using the dropdown combo box arrow to the right of an entry.

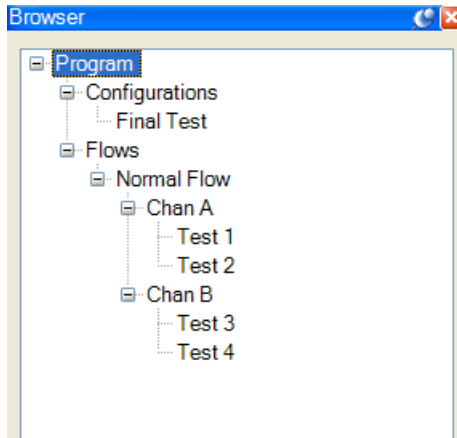
Often combo boxes allow you to choose variables from the limits table in place of hard coded numbers. In this example the limits were chosen from the limits table.

These parameters are saved with the test program.

Note: It is possible to create test methods for FTI Studio™ that have custom panels for configuration parameters rather than the simple table shown here.

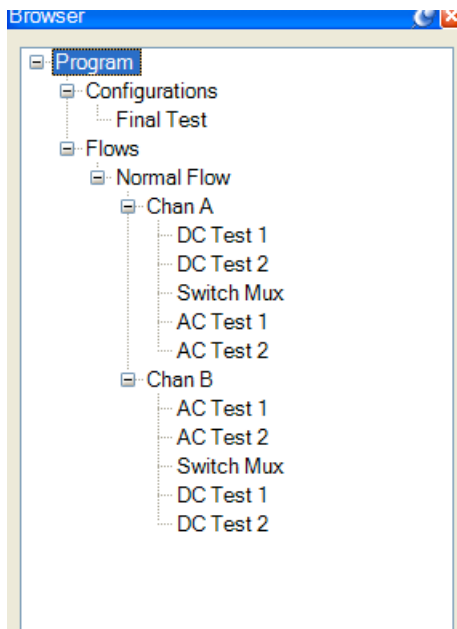
4.3 DUAL DIE TESTING

Dual die testing is accomplished by running tests on two channels or devices on a single die in parallel. To do this, setup two sequences like this:

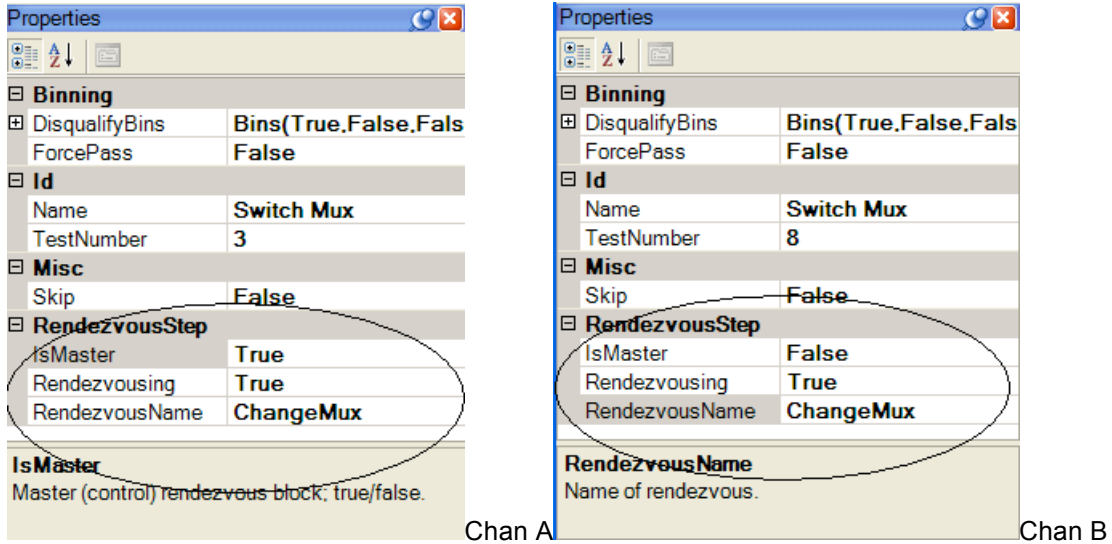


This example has a sequence called Chan A and one called Chan B. FTI Studio will run each sequence in its own thread. A special case occurs when the two channels have to mux hardware. For example, suppose your system has a FET AC Board and a FET DC Board, and you must use both boards on each channel. So you do DC on Chan A while doing AC on Chan B, then reverse them with a mux and do AC on Chan A, and DC on Chan B.

To do this you need to synchronize things so your program would look like:



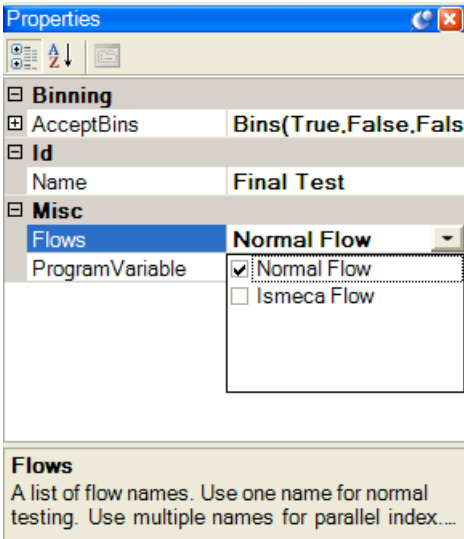
A key element is the Switch Mux step. We need execution on each sequence to stop at switch mux, and then after they are stopped, switch the mux, then continue. We do this by making Switch Mux a Rendezvous step. Here are the properties on the Chan A Switch Mux and Chan B Switch Mux:



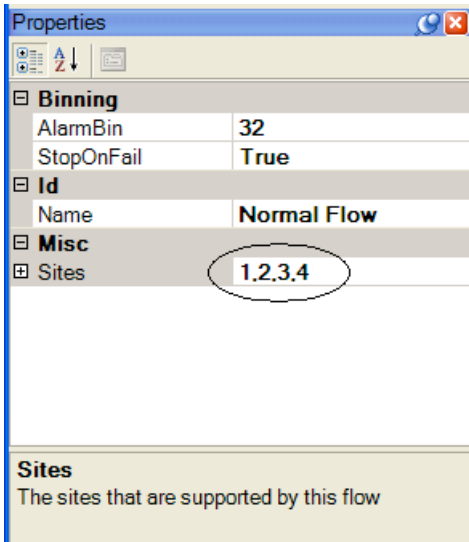
The master step is the one that will run the code to change the mux. The non-master steps will just wait until the master is done. It is important that all steps involved in a rendezvous have the same name. You can have multiple rendezvous in a test program as long as they have unique names.

4.4 PARALLEL TESTING

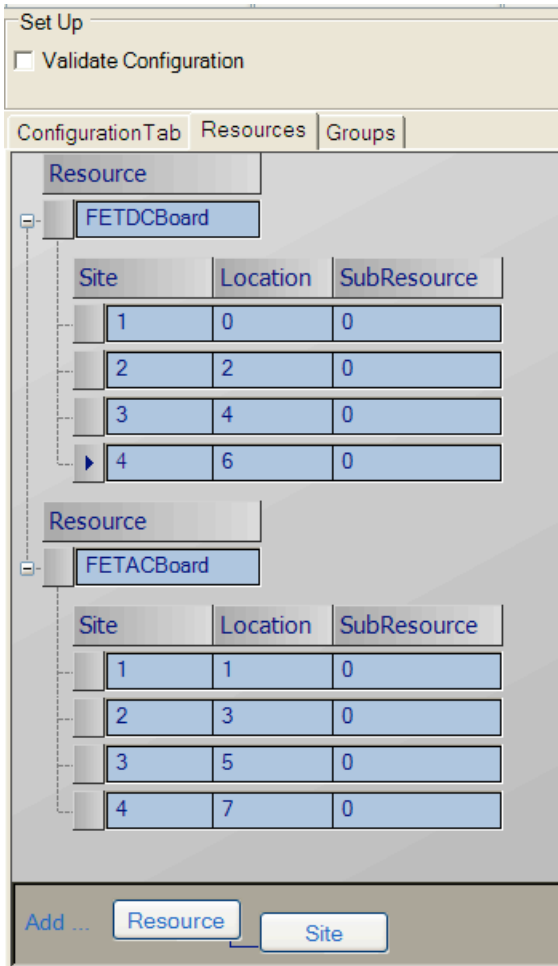
Parallel testing or traditional multisite is used for traditional multisite handlers that are not index-parallel type. To parallel test, one flow is used to test on all sites. This means that your configuration selects only one flow:



Never choose more than one flow for this type of handler. Then in the flow being used, list all the sites being tested:



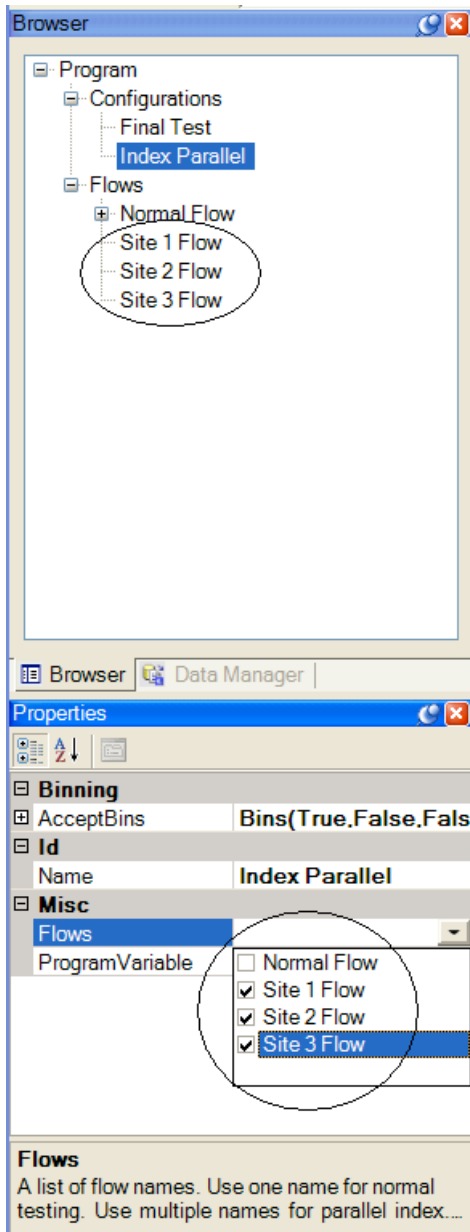
Now, you also need a resource map to support these sites:



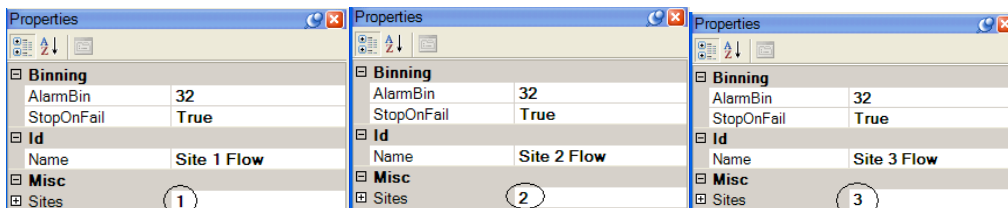
This tells the tests how to communicate with a particular site.

4.5 INDEX PARALLEL

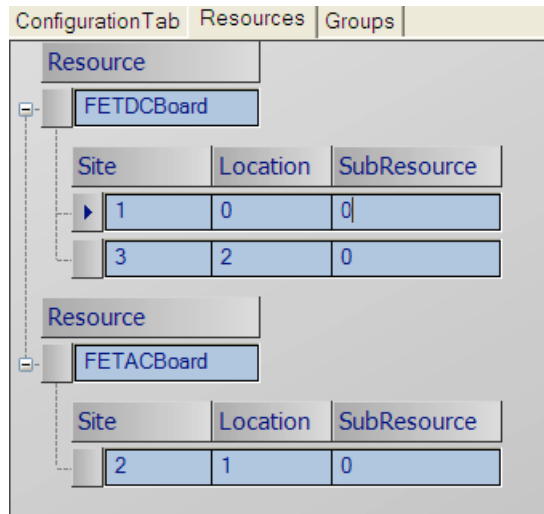
Index parallel testing is done differently than traditional. For this you must assign one flow to each site, and each flow must only support one site. Let's walk through a 3 site example. We will have three flows called Site 1 Flow, Site 2 Flow, and Site 3 Flow:



The configuration has all three flows selected. Each flow is assigned to a site:



You must also have a resource map. For example:



In this case we have a DC board on site one and three, and an AC board on site 2. This means we would add DC tests to the flows for site one and three, and add AC tests to the site 2 flow.

When the handler is run, data from sites 1, 2, and 3 will be merged and assigned to one device number in the datalog. All the statistics and counts will reflect the real number of devices tested.

Note: If you use an index parallel handler in parallel mode such that it behaves like a traditional handler, then refer to section 4.4 for creating your test program.

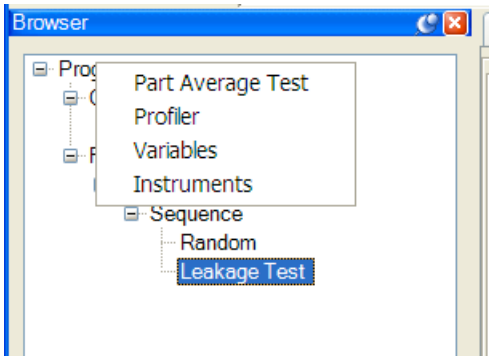
4.6 PROGRAM TOOLS

Program tools are associated with program nodes in the browser. There are tools for the program node and for step nodes. However, these tools do not appear in the Node Tools area without selecting them from the Program and Step nodes.

4.6.1 ACCESSING THE NODE TOOLS

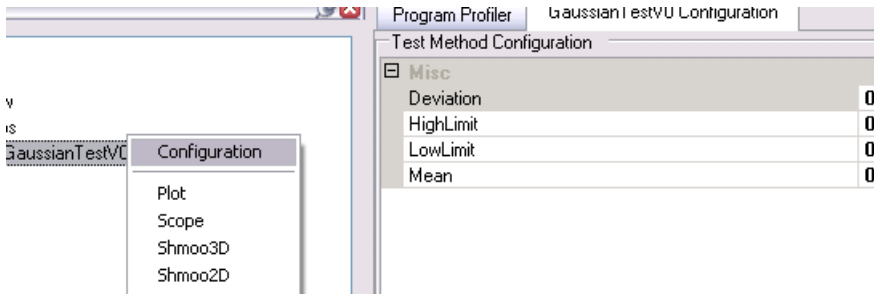
To acquire these tools, the user must right click on a node on the left and pick a tool available to that node.

In this example we will pick the profiler tool from the Program node:



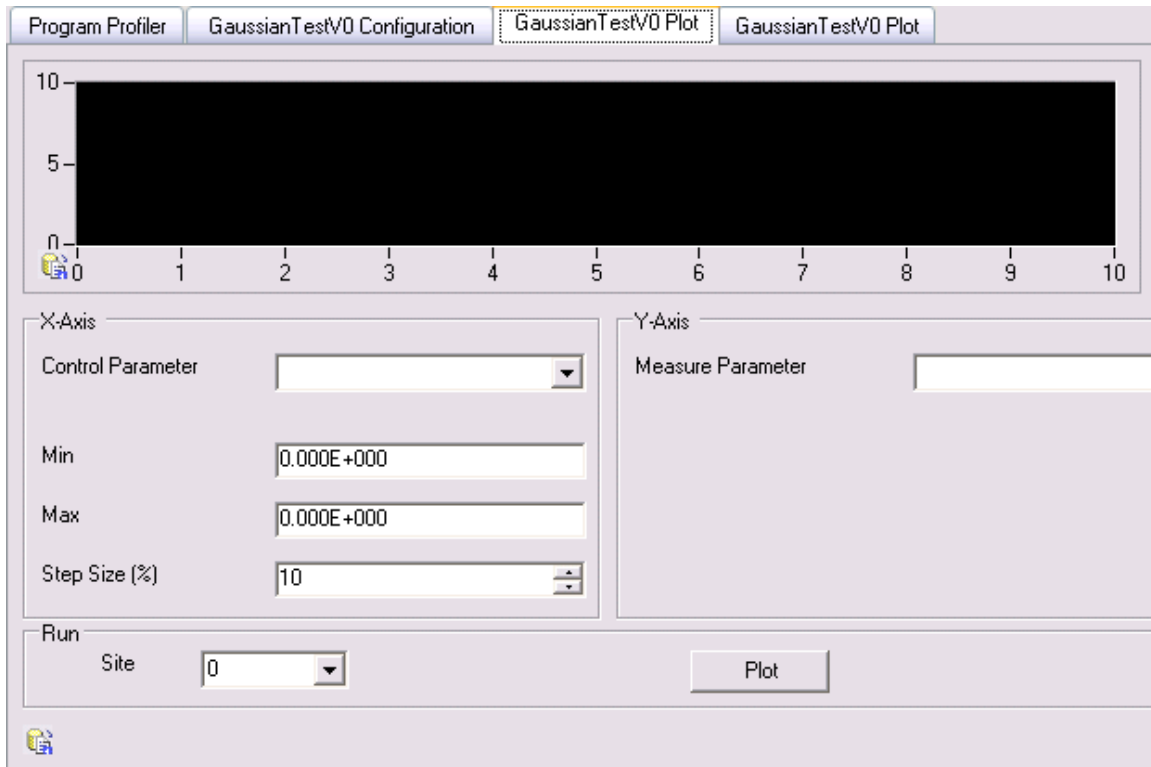
After right clicking the node and selecting the Profiler Tool you can see it appear in a tab on the right.

By right clicking on the example test node, Gaussian TestV0, the user can also select a Configuration Node Tool, or you can left click and get the configuration by default:



This process of accessing tools allows the user to have multiple tabs from multiple nodes showing at the same time. Also you can have multiple copies of the same tool showing simultaneously.

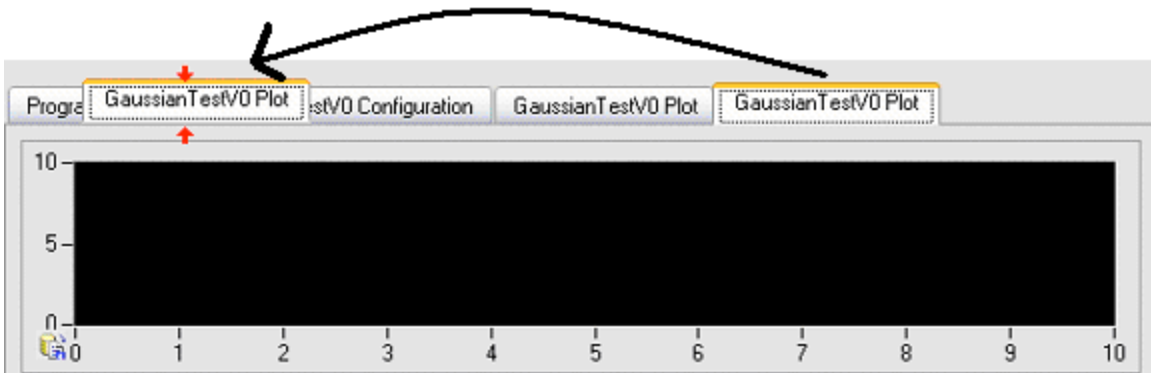
For example, it is possible to use two Plot Tools at the same time:



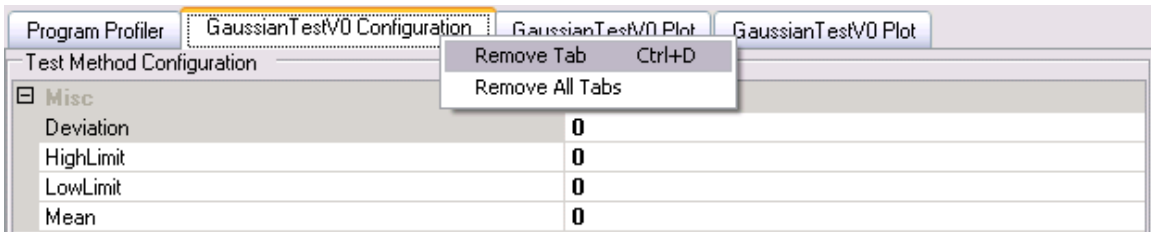
The users can input two different settings allowing for side by side comparisons.

When a particular test node is selected in the Program Tree, the Configuration Tool for that test node automatically pops up. This allows for quick navigation through your tabs.

You can move tabs by dragging and dropping:

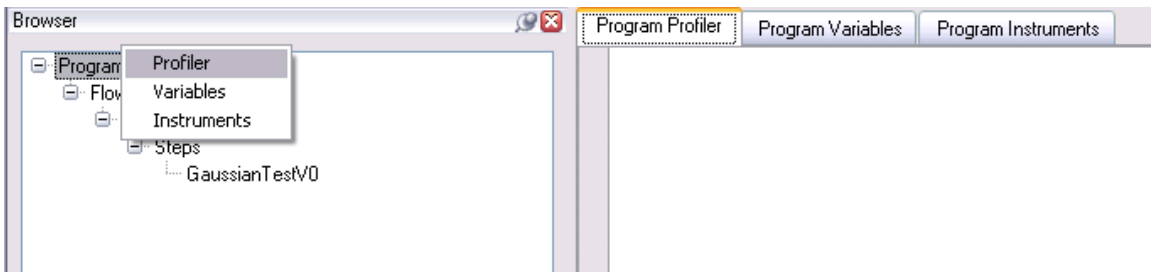


To close a Node Tool tab just right click the tab and select Remove Tab, or you can press Ctrl+D:



The user can also select Remove All Tabs to delete all of the tabs in the Node Tools browser.

4.6.2 PROGRAM NODE TOOLS



There are three tools which can be displayed when the program node is selected: Profiler, Instruments, and Variables.

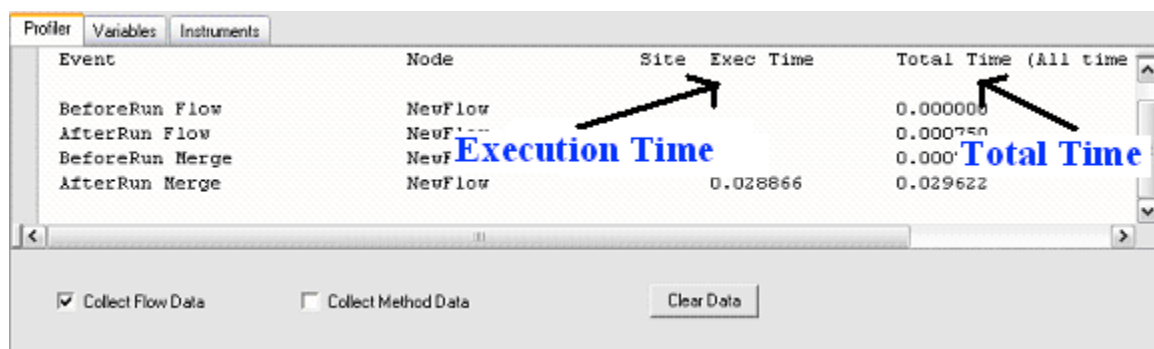
4.6.2.1 PROFILER TOOL

The Profiler Tool is used to maximize the efficiency of the tests. It measures the execution times of Flow Data, Method Data, or both of them together.

You can choose which you would like to measure by checking the related box:



When the Collect Flow Data box is checked, and you run the test, the Profiler gives you the Before-Run and After-Run flow execution time. It also provides you with the total time of the tests:

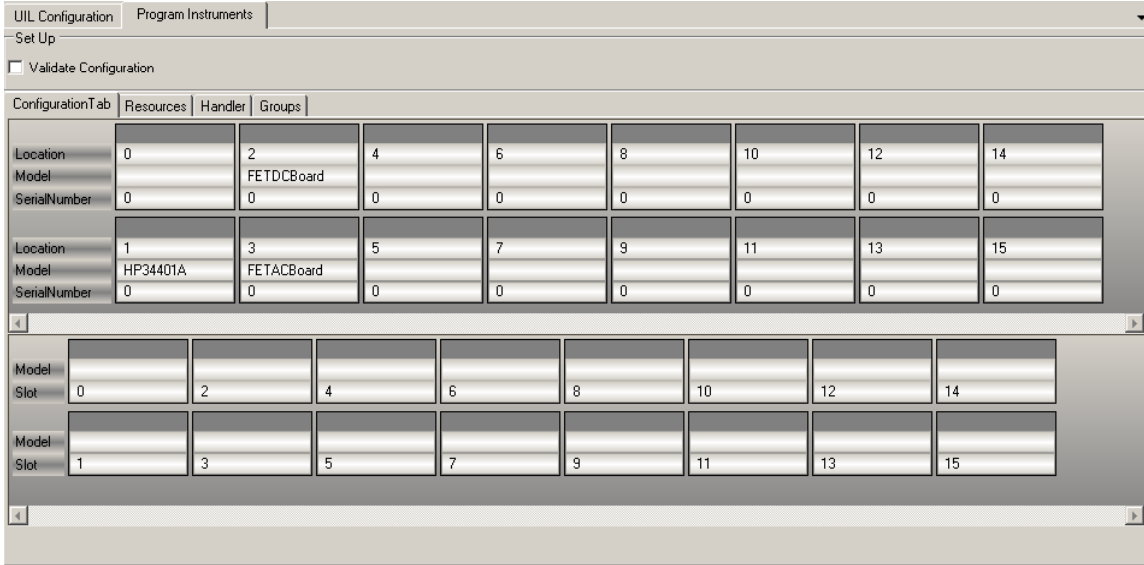


When the Collect Method Data box is checked, the Profiler gives you the times of the Pre-Run, Run, and also the Post-Run.

Remember you may check both boxes at once.

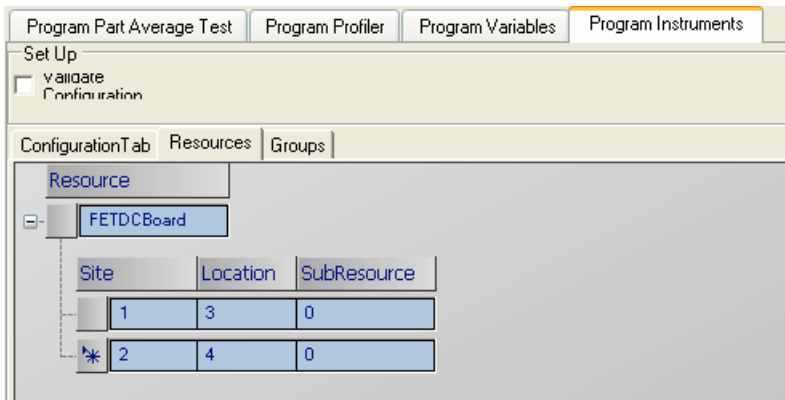
This information may all be used to maximize efficiency in the programmer's testing.

4.6.2.2 INSTRUMENT TOOL

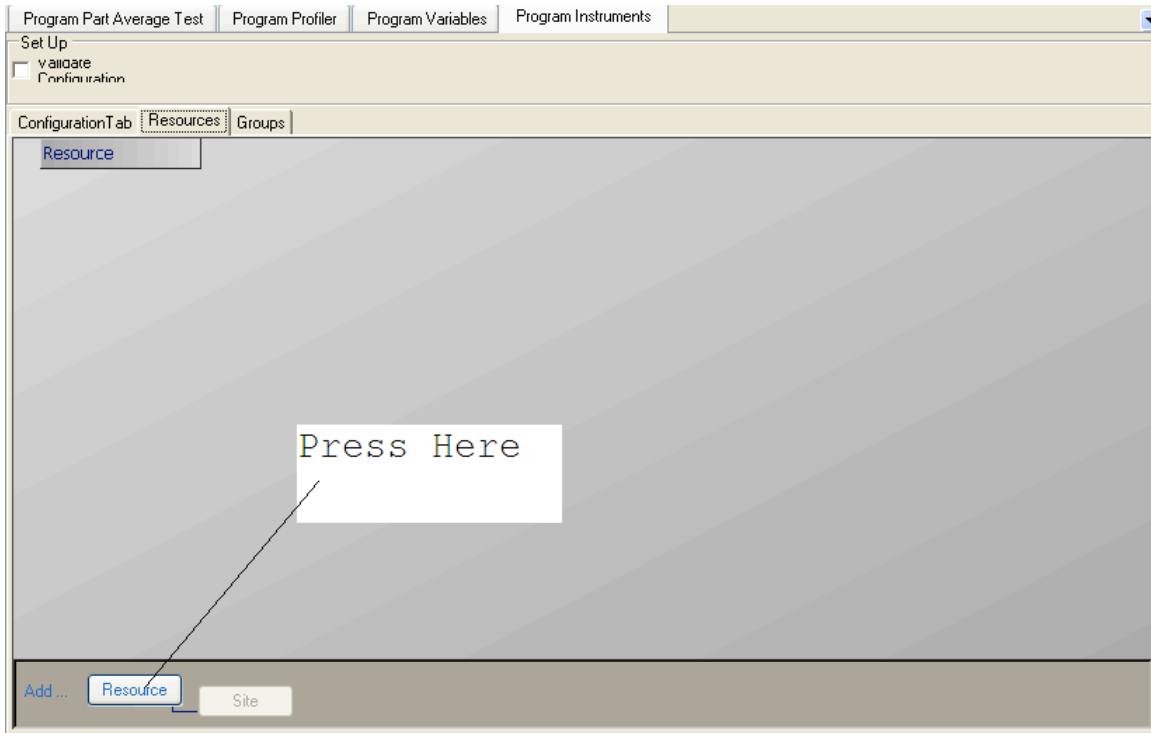


You can view the instruments installed in the tester under Current Configuration. You can set the instruments that must be installed for your application under Required Configuration. Check the Validate Configuration if you want to prevent the test program from running when the configurations don't match.

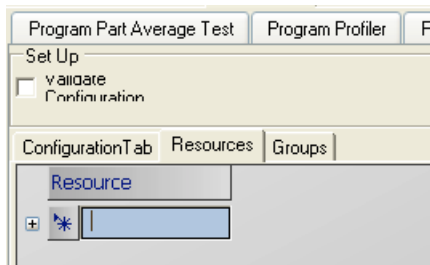
You also need to define a resource map:



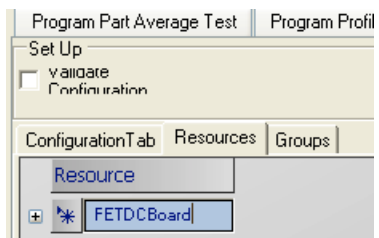
The first step to creating a map is to press the Resource button at the bottom of the map:



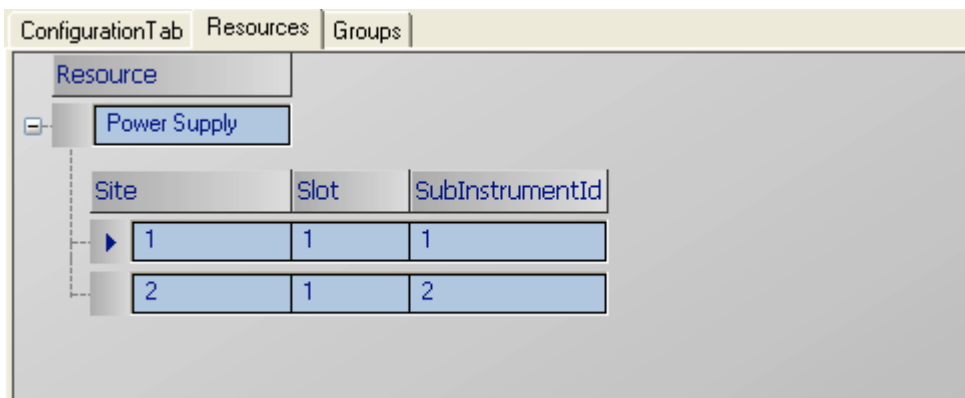
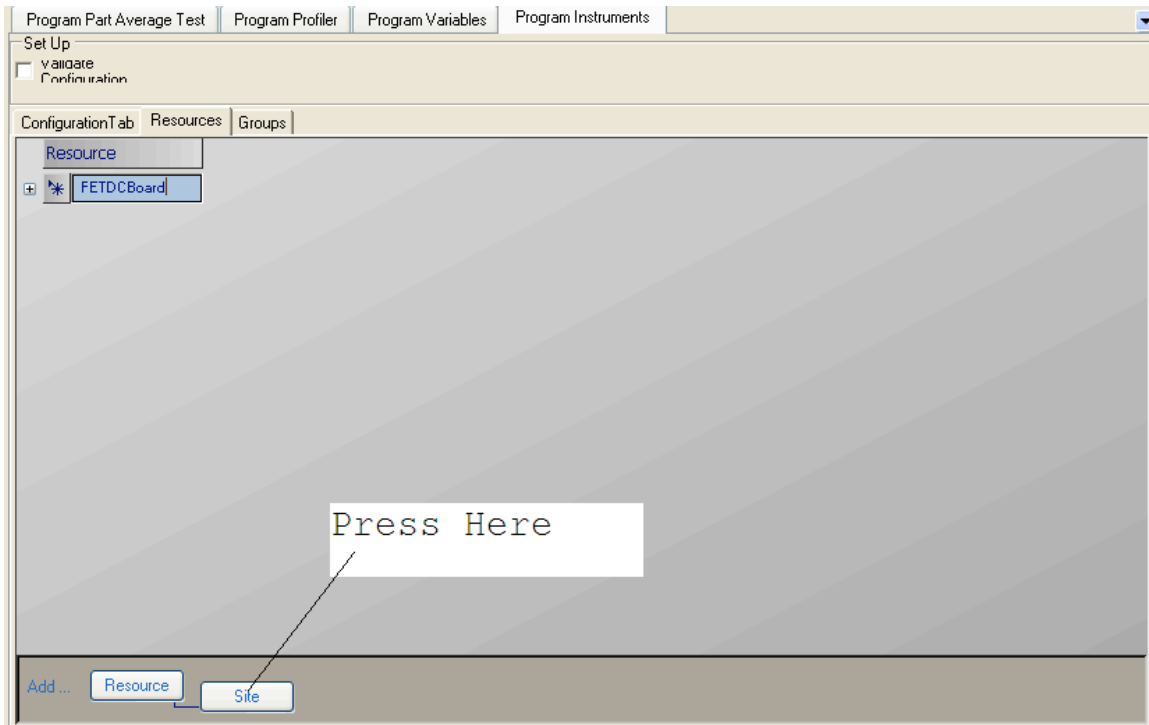
You will then have a table with one resource:



Enter a resource name:



Then press the Site button to add sites:



Enter the site, slot, and id in the table. This example assumes that Slot 1 contains an Octal VI, and each channel of the VI is numbered one through 8. Therefore, the Power Supply resource for Site 1 uses Slot 1-Channel 1, and for Site 2 uses Slot 1-Channel 2.

The test method code will then program in terms of the Power Supply resource rather than slot and channel numbers. Sub Instrument Id is used in place of a channel, because instruments may have their own concept for dividing up the instrument. Sub Instrument Id is just a general concept that can mean channel, slice, or other things. A VI might call each one a Sub-VI, whereas a digital card may call one a channel.

Note: The resource map is saved with the test program in an XML file. It is possible to create this file externally to the test program and place it in the test program directory as long as the file name and syntax are preserved.

4.6.2.3 VARIABLES TOOL

Profiler Variables Instruments						
Name	Type	FT	QA	Wafer	Custom	
Mean	DoubleVariable	0.5	0.5	0	0	
StandardDeviation	DoubleVariable	0.1	0.1	0	0	
MinLimit	DoubleVariable	0.43	0.1	0	0	
MaxLimit	DoubleVariable	0.7	0.9	0	0	
Global	DoubleVariable	10	10	0	0	
Bias Current Min	DoubleVariable	-1E-07	-1E-07	-1E-07	-1E-07	
Bias Current Max	DoubleVariable	1E-07	1E-07	1E-07	1E-07	
Offset Current Min	DoubleVariable	-1E-08	-1E-08	-1E-08	-1E-08	
Offset Current Max	DoubleVariable	1E-08	1E-08	1E-08	1E-08	

The variables tool is used to create/modify test program variables. These variables are used by the C# test method code.

It would be awkward to create variables in this table and then write C# code to use them, so generally, a test either creates the variable programmatically when a step is added to a flow, or a configuration parameter can refer to a variable. For example, in this configuration parameter, one can choose a variable from a pick list:

The screenshot shows the 'Variables' tab in the configuration tool. Under the 'Misc' section, the 'StandardDeviation' parameter is selected. A dropdown menu is open, showing a list of available variables: Mean, StandardDeviation, MinLimit, MaxLimit, and Global. The 'StandardDeviation' option is currently selected in the dropdown.

Notice that the drop down displays the variables show in the tool. One can also type a new variable name into the configuration window and it will automatically add the variable to the tool.

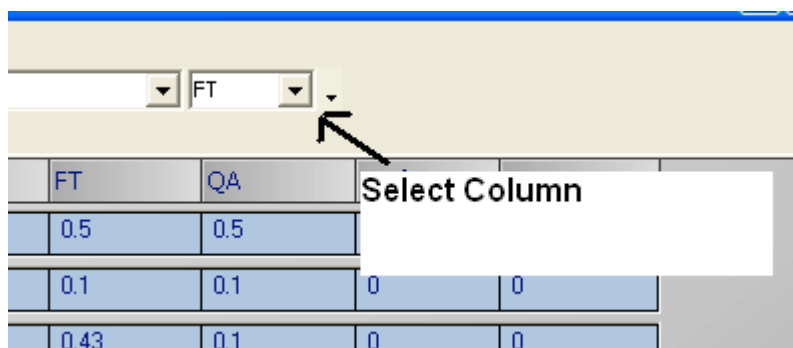
Variables are categorized by using columns:

Name	Type	FT	QA	Wafer	Custom
Mean	DoubleVariable	0.5	0.5	0	0

The categories are:

- Final Test
- Quality Test
- Wafer Test
- Custom Test

A test method is coded without knowledge of which column is active. The active column is selected on the toolbar:



Essentially, this allows the variables to be used as test program limits. The column selector on the toolbar is available in production mode, so the operator can choose what limits too use.

Note: Refer to the Programmer's Manual for more information on how to create configuration panels that use variables.

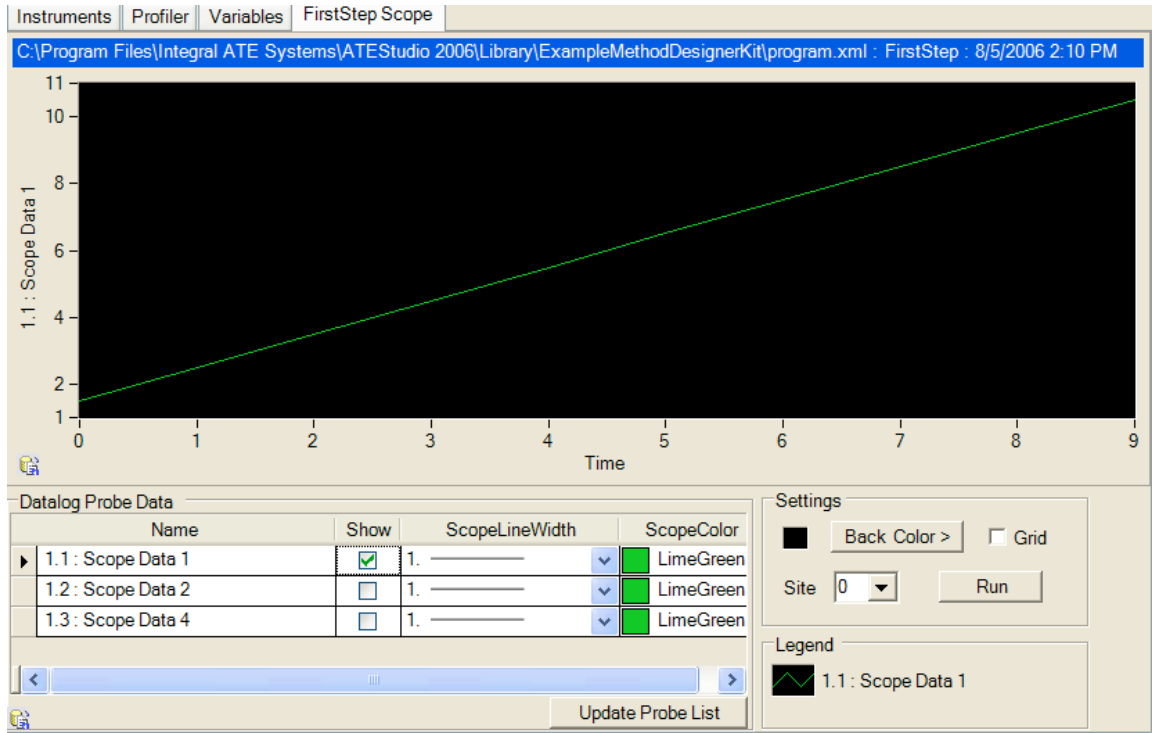
4.6.3 STEP NODE TOOLS

The step nodes contain four tools:

- Scope
- Plot
- Shmoo
- Characterize

One can add new tools by creating new tool plug-ins and putting the resulting C# assembly in a well know directory. These four tools are the default tools.

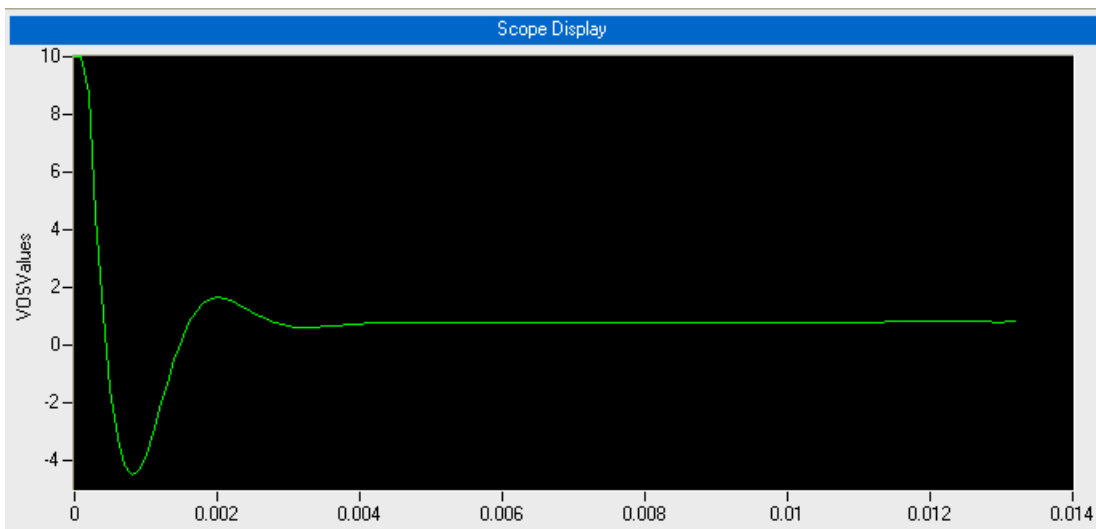
Step Node tools interact with a test method instance. For them to work, a test method must expose certain parameters using the test designer in Visual Studio. If parameters are exposed, then the tool will display choices. For example:



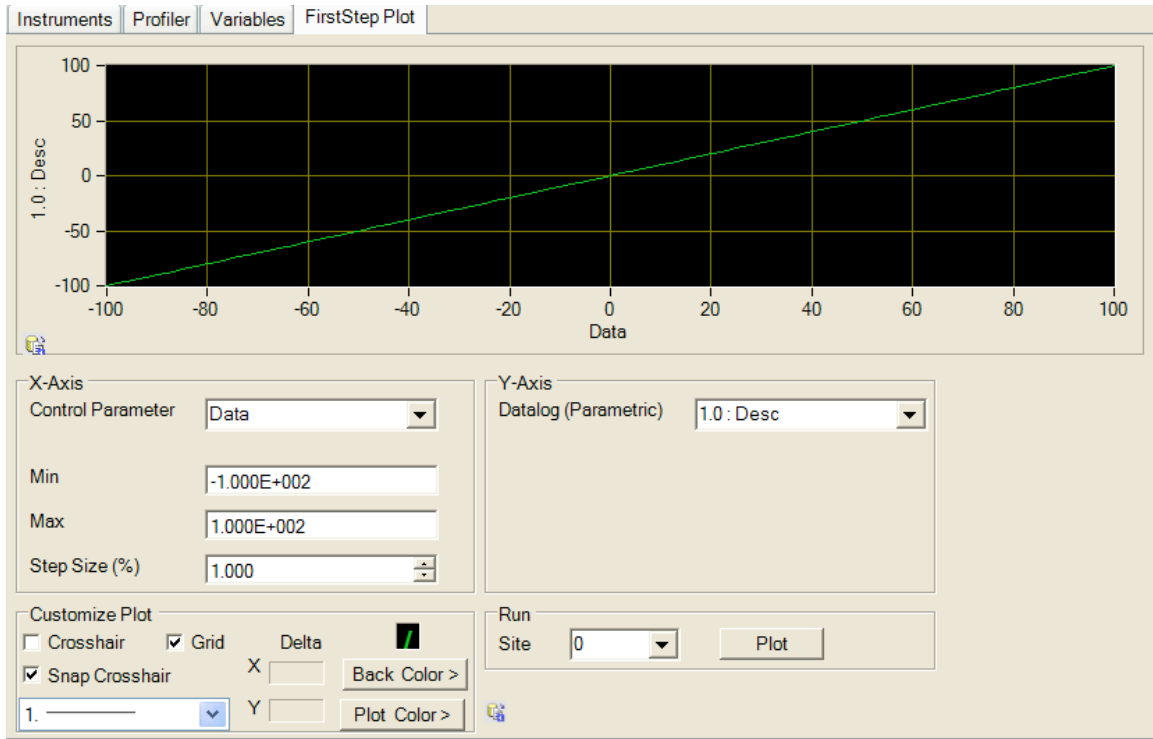
Notice the lower left window called Probes. Probes are created by logging an array of doubles in the test code. These probes are then listed in the table and can be selected in the Show column. It is possible to select any number of waveforms.

The Update Probe List button will fetch any new probes found in the latest datalog run. Pressing Run will cause the test to be run and displayed. This means you can change the configuration of a test and press Run to acquire new data.

This example plot was derived from a test that logged an array representing a straight line. Below is a VOS settling time plot from an Op Amp program:

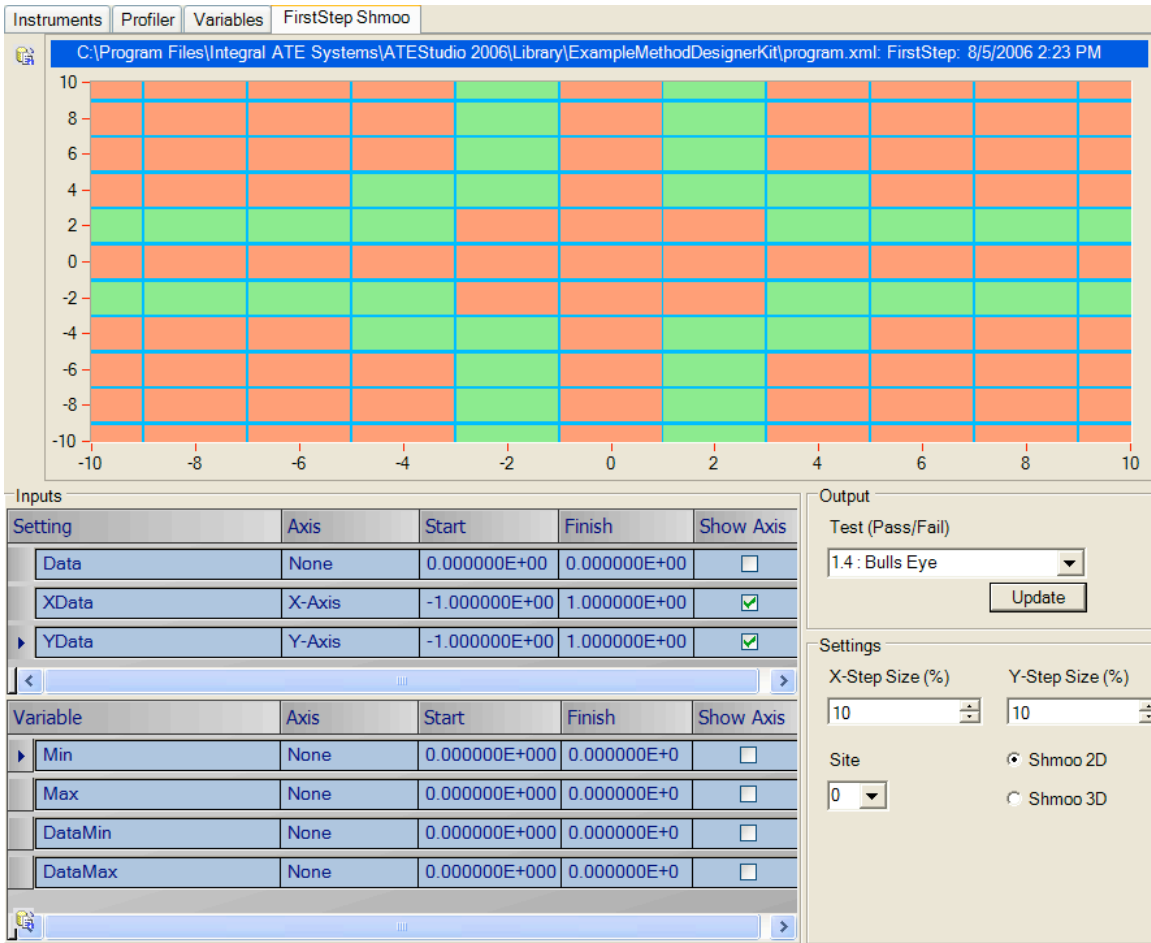


The Plot tool is used to plot a datalog result vs. a change in a test parameter. For example:



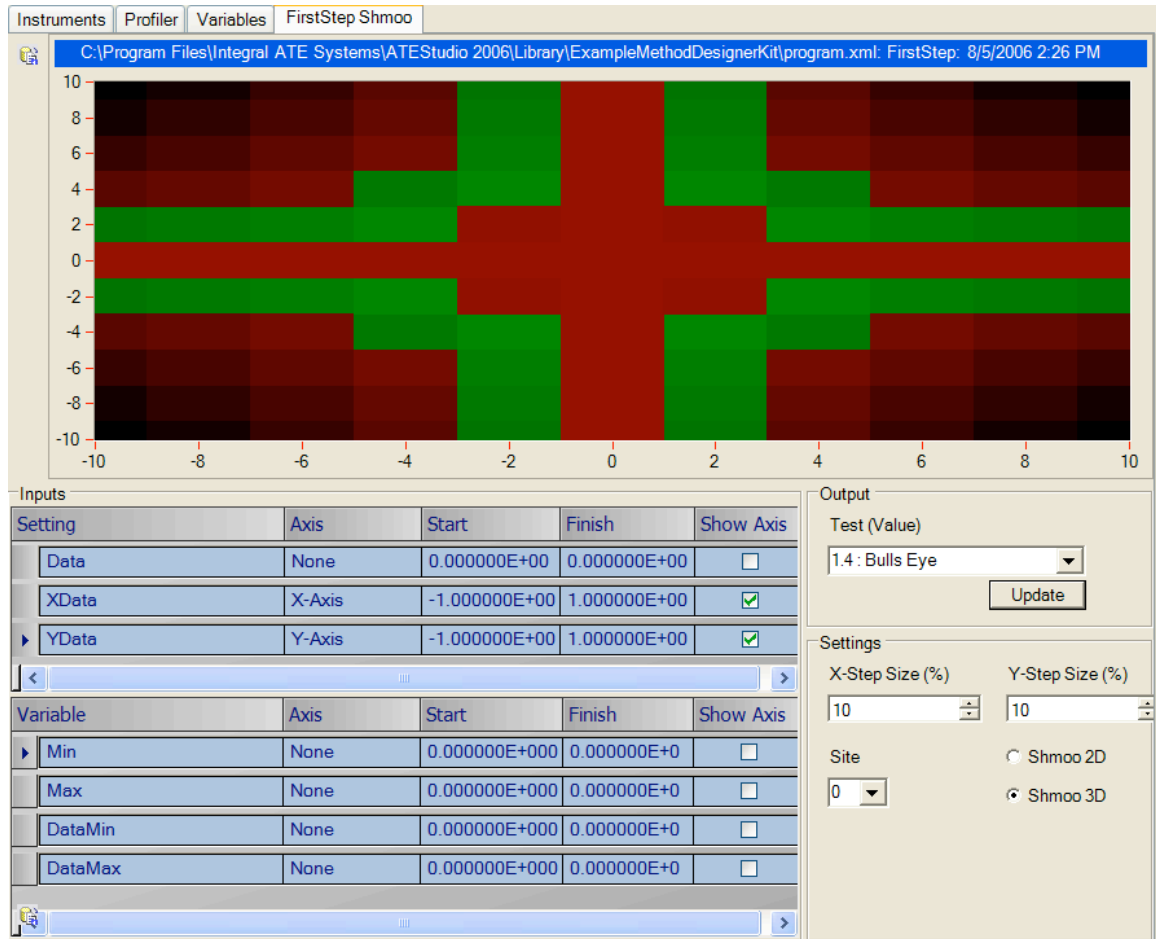
The X-Axis control parameter comes from the configuration panel of the test. The Y-Axis measured value comes from the datalog of a double value. In this case we are varying the Data value of the test from -100 to 100 in 1% steps and plotting against test 1.0. When you press the Plot button the test is run 100 X 100 times and the collected data is plotted.

The Shmoo tool is used to make two-dimensional plots. For example:



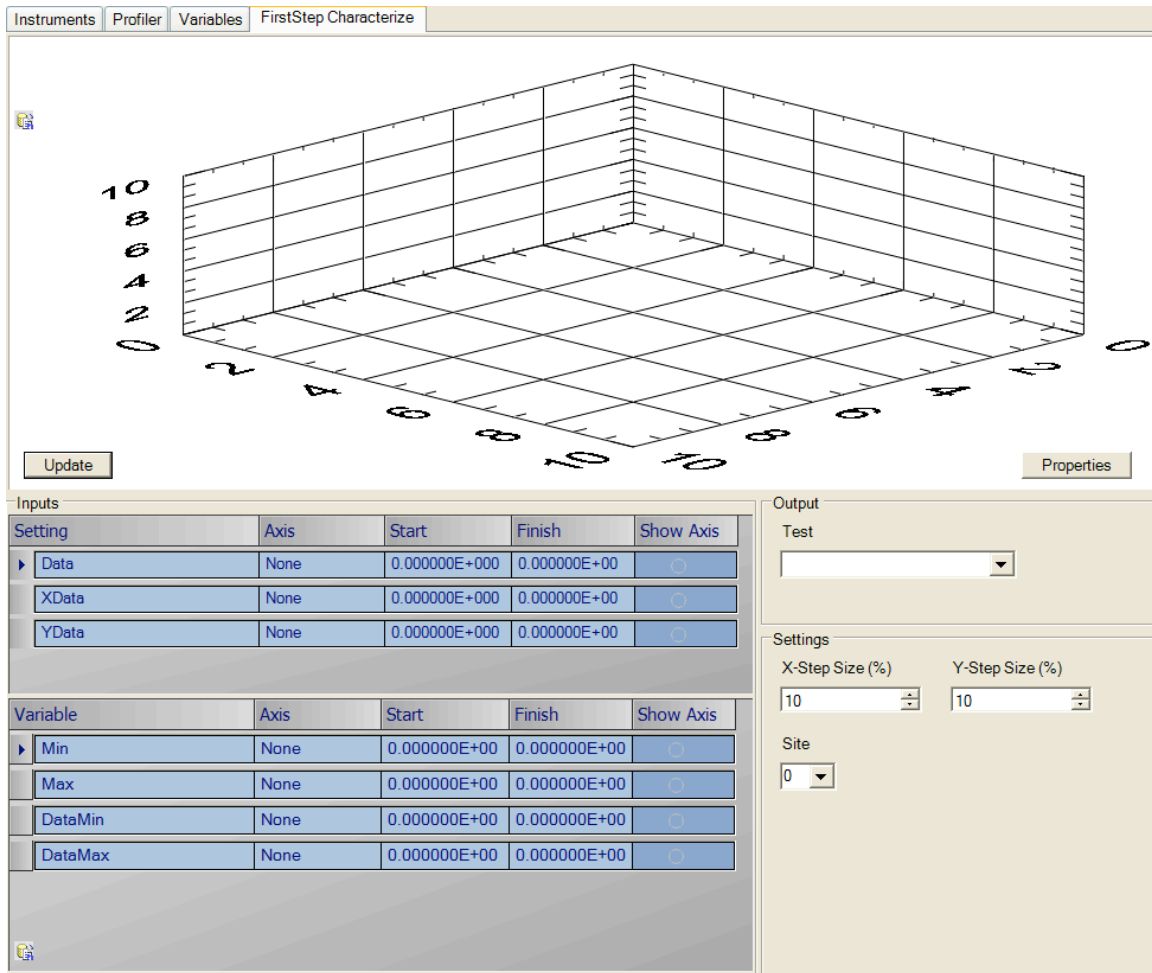
Both the X/Y axis are the independent axis and you can vary settings from the configuration panel or variables. You can assign more than one parameter to each independent axis. The dependent data is plotted as pass/fail using green/red. You can select the test in the Output area to select the test.

You can also make select Shmoo 3D to use color intensity to represent the parametric value of the test:



Green and red still represent pass/fail, but the intensity represents the relative measured value.

The Characterize tool is used to make three-dimensional characterization plots.



There are three lower windows for configuring a plot: Inputs, Output, and Settings.

Inputs are settings in the configuration data and variables. Each setting is stepped from a start to a finish value, and assigned to an axis like the Shmoo tool.

The Outputs are parametric values from the datalog. The result is plotted on the z-axis. You can think of this as a Shmoo 3D as a surface plot.

The Settings control the resolution of the x and y-axis, and the site that is used to measure.

When the measurements are being made, each setting is stepped via the Step Size. If there is more than one setting per axis, each setting is scaled by the percentage starting with the Start value.

Let's look at an example using a random number test (Note, this is a picture from FTI Studio 1.0, hence the different look and feel):

Inputs

Settings			
Setting	Axis	Start	Finish
SeedSetting	X-Axis	0	10
▶ ValueMultiplier	Y-Axis	5	15

The Seed Setting is the seed of the random number generator and starts at 0 and finishes at 10. The Value Multiplier is a scale value that scales the random number.

Output

Measurement

Value1

The output is Value1, and this value is just the result of the random number generator.

Settings

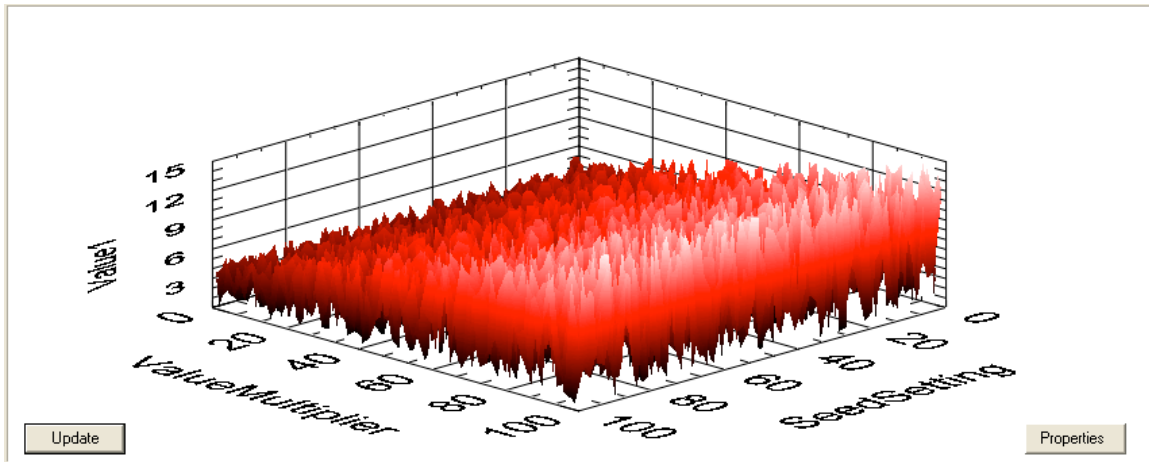
X-Step Size (%) Y-Step Size (%)

1.00 1.00

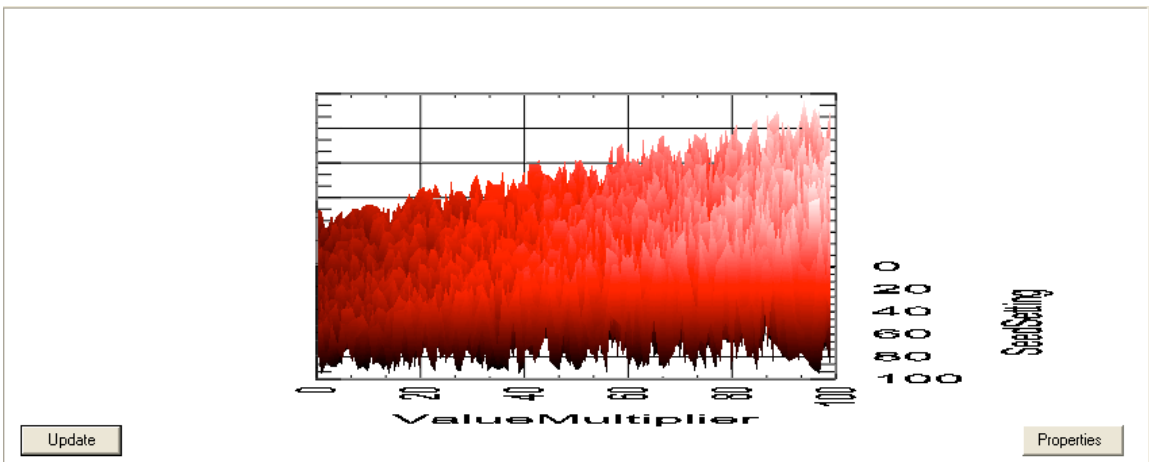
Site

0

The seed is assigned to the x-axis and is stepped in 1% increments. The multiplier is assigned to the y-axis and is also stepped in 1% increments.

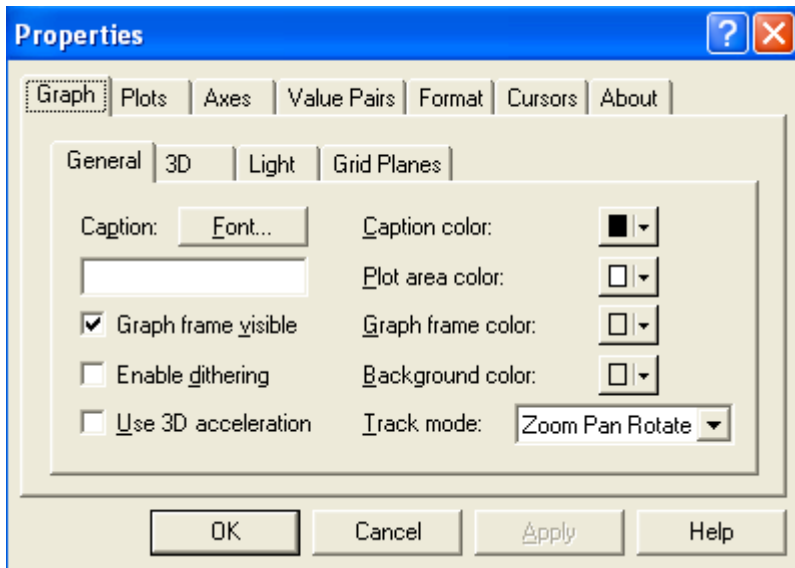


The plot is run by pressing the Update button. By rotating the plot on end you can see below that the multiplier axis (x-axis) ramps up from the start to finish:

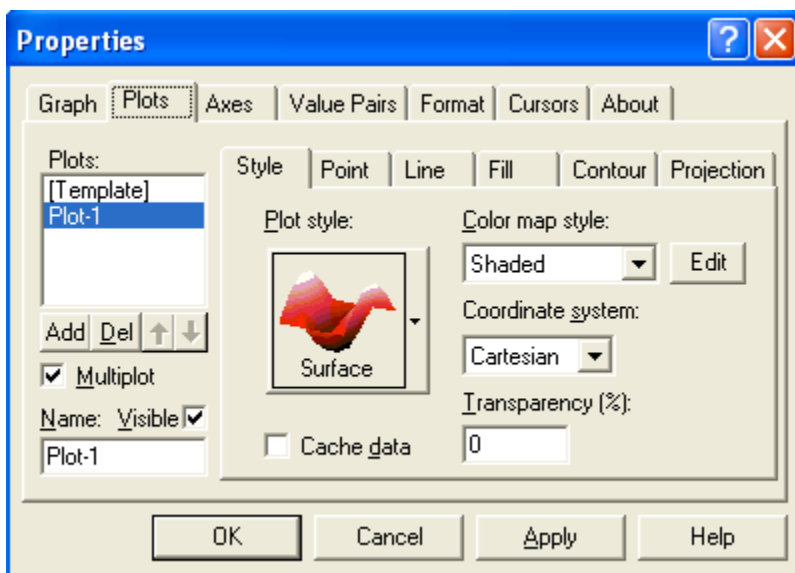


You can rotate the plot with the mouse by pressing the left button and dragging.

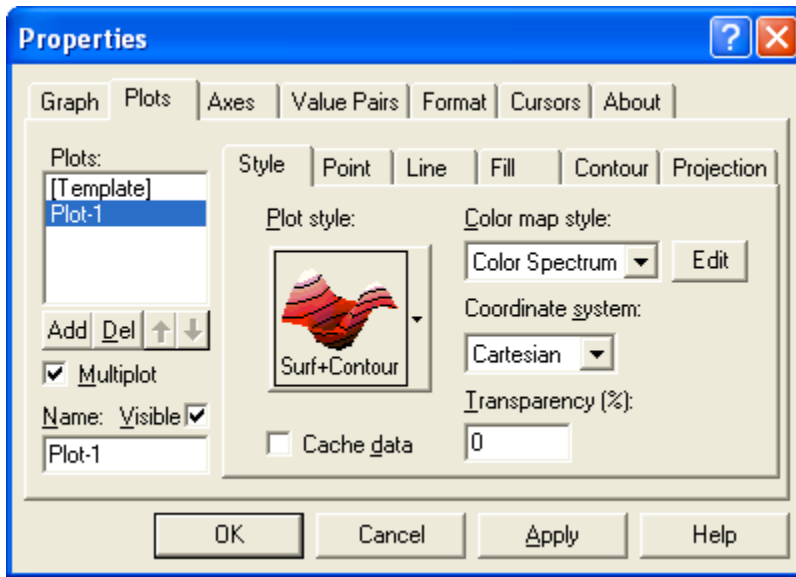
You can also change the properties of the graph. Pressing the Properties button brings up the following dialog:



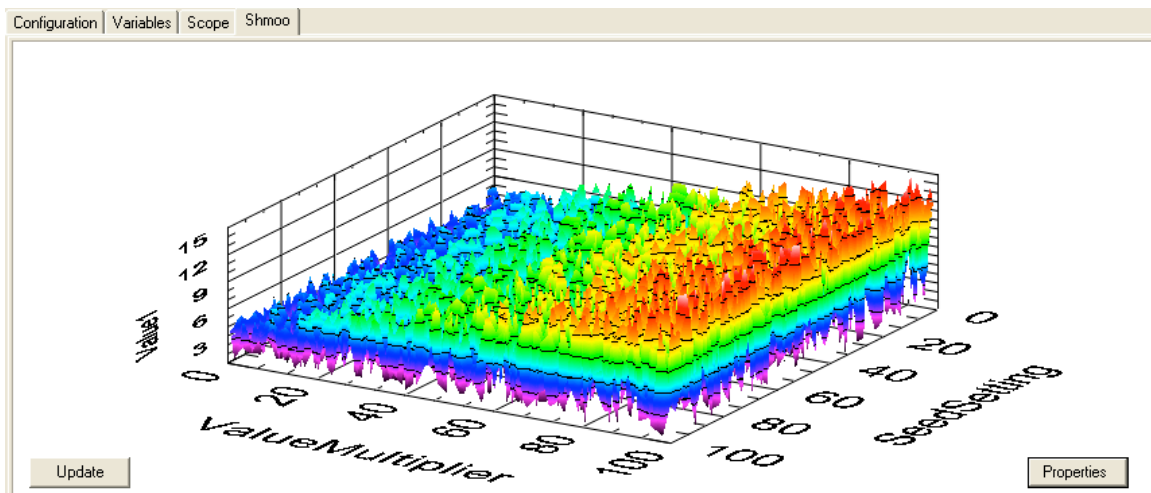
For an example, we will select the Plots tab:



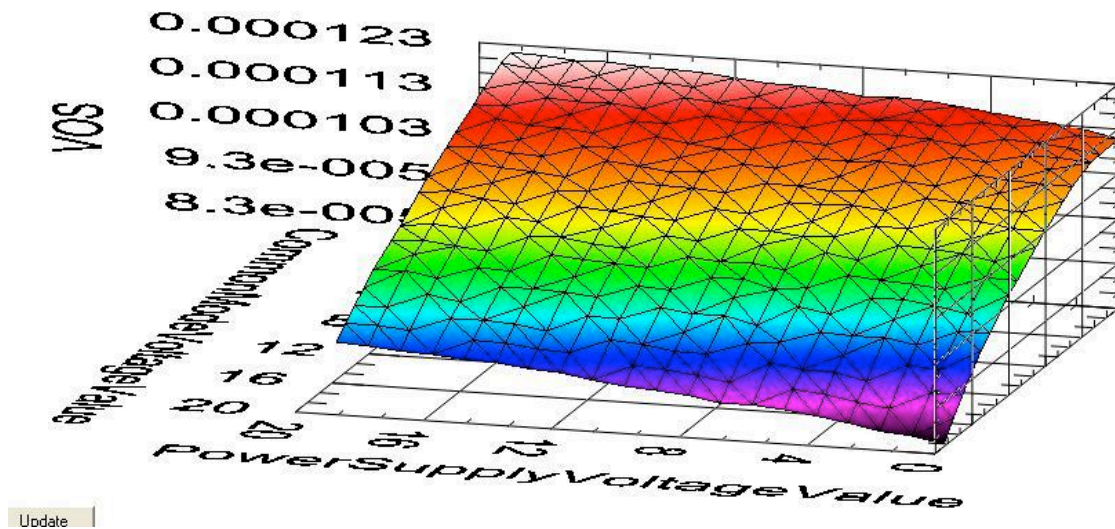
Then change the plot style and color map:



And the resulting plot is:



Here is a more realistic plot from an Op Amp test:



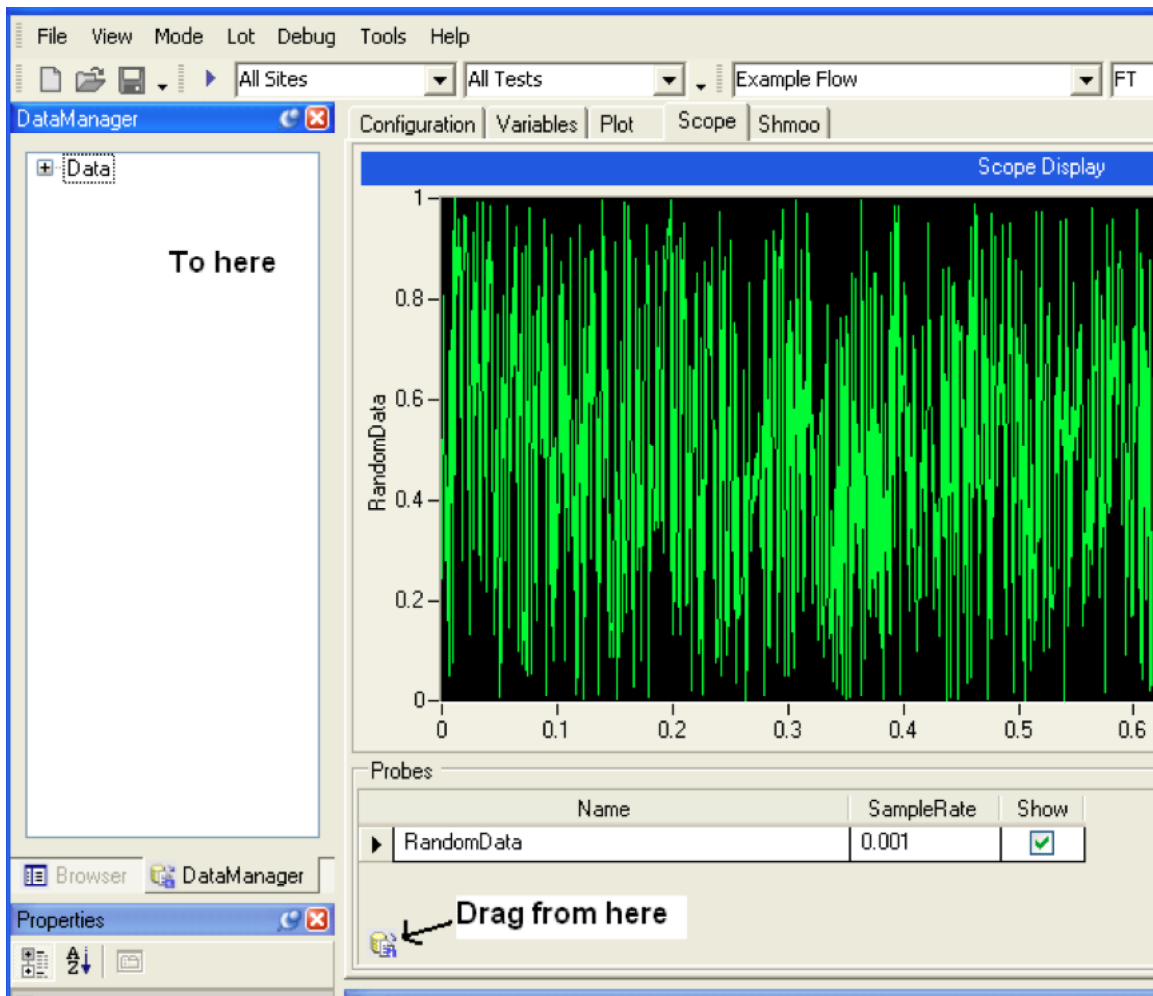
This plot shows Common Mode Voltage vs. Power Supply Voltage, while measuring the voltage offset. As the power supply voltage decreases, so does the offset, but as the common mode voltage increases, the offset decreases.

4.6.4 SOME POINTS ON SCOPE AND SHMOO

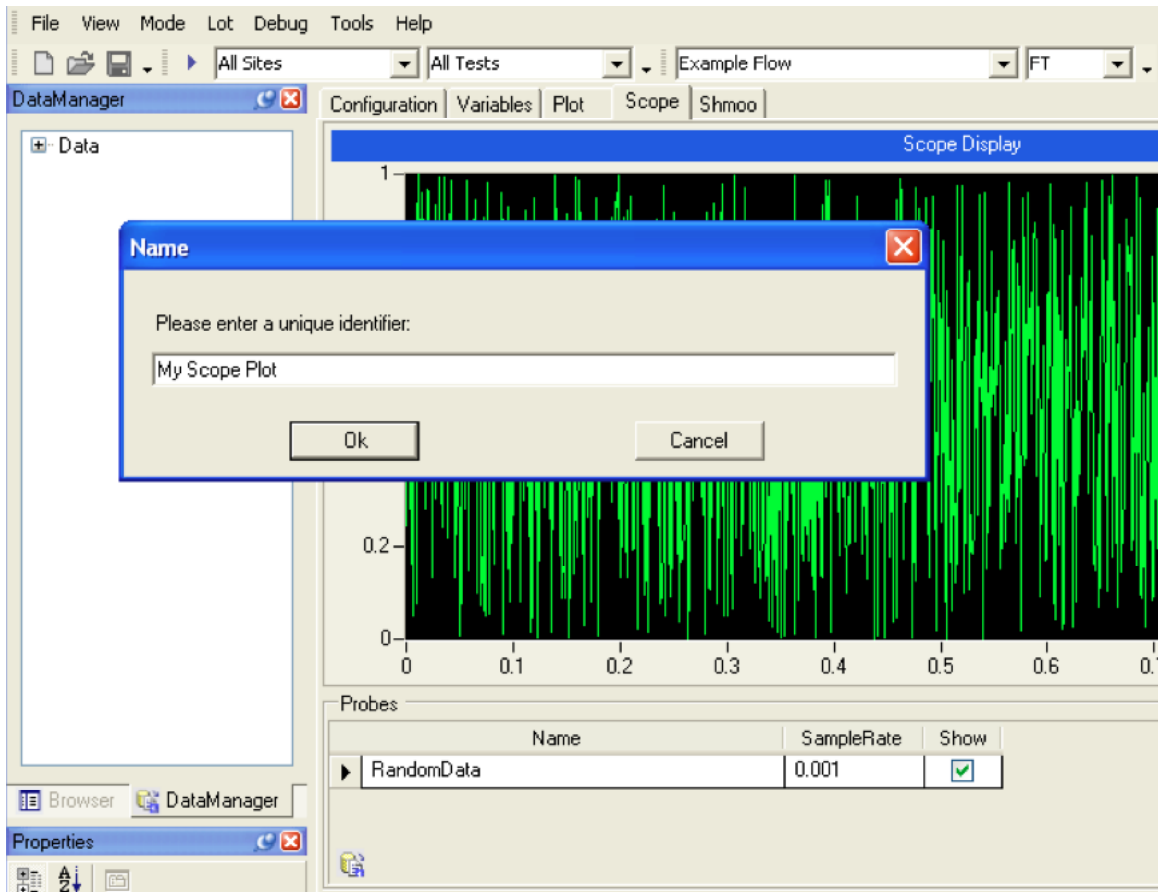
The Scope and Shmoo tool both operate off of configuration values and/or variables, and not by operating directly on hardware. This means that it is possible to make tests that can scope and shmoo for any hardware. In fact, you can take some simple PXI or PCI instrumentation and write tests that can use scope and shmoo. This decoupling from the hardware was done intentionally to make it easy to use standard tools with any hardware.

4.6.5 DATA MANAGER

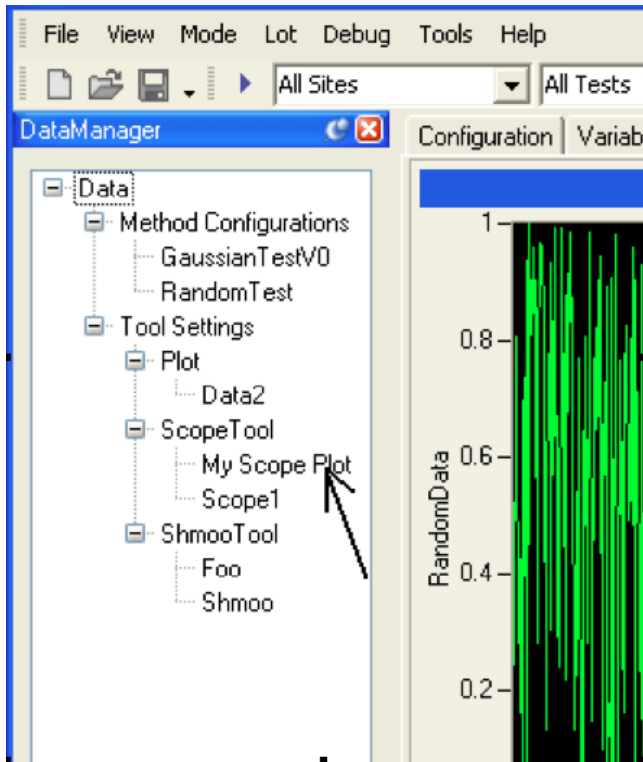
The data manager is used to store and retrieve tool setups, tool data, and data analysis results. To store a setup, drag from the tool to the manager:



Dropping on the data manager will raise a name dialog:



Enter a name and press ok:

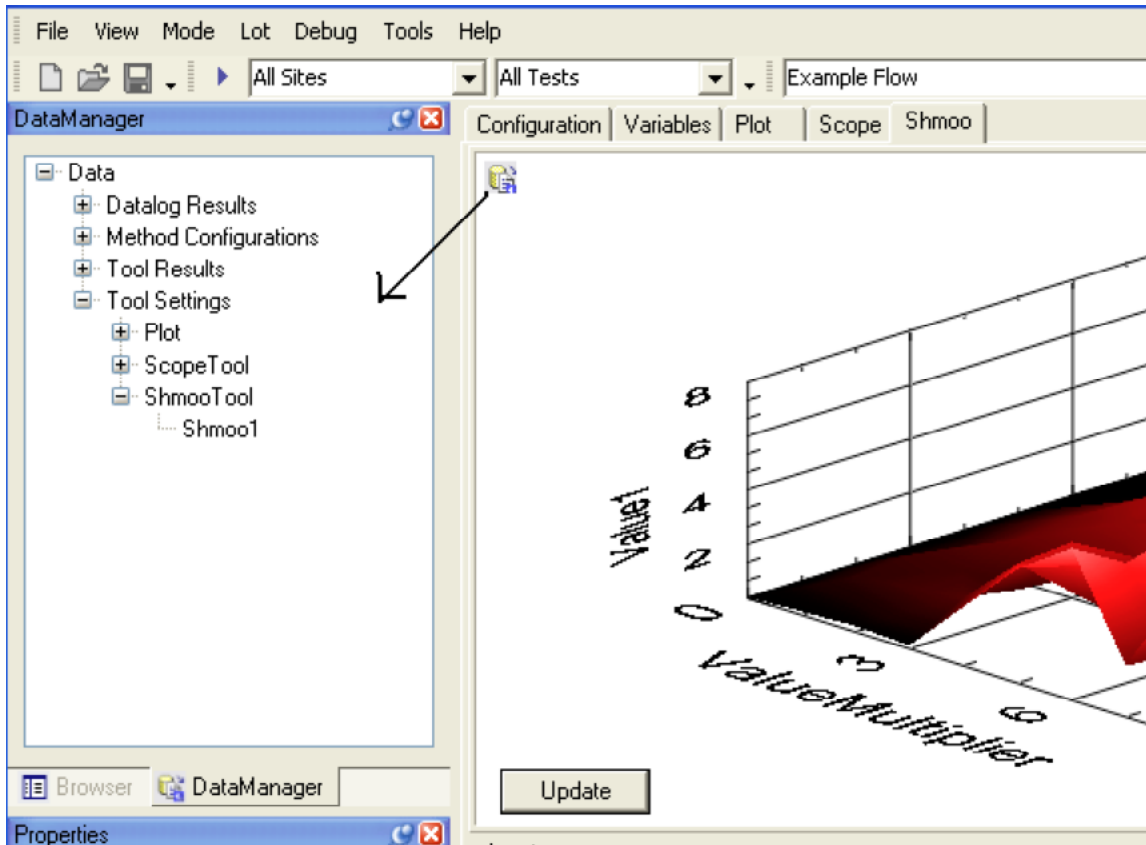


The data manager will now display the setup categorized by the tool name. To restore these settings back to the tool, drag the setup (My Scope Plot) from the data manager and drop it on the settings of the tool.

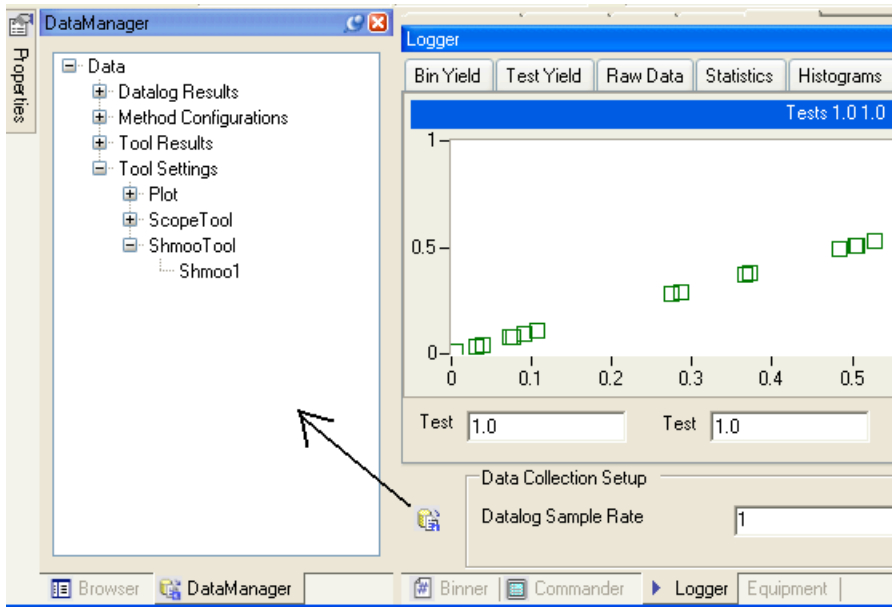
Remember, drag and drop from the icon part of a tool, and not the settings. You may also display or graph part of a tool, but drop on the settings or graph and not the icon.

You may also store test configurations in the data manager.

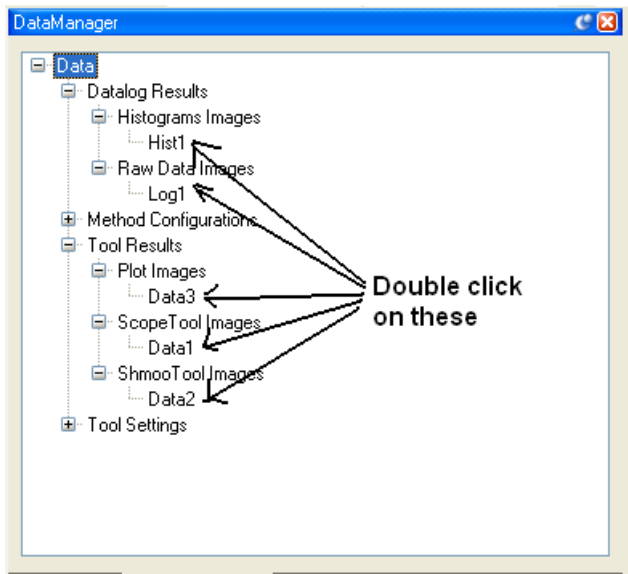
To store tool results, you drag from the icon on the graph or plot, rather than the icon next to the settings:



To save results from data analysis, use the icon that is next to the settings:



To view results, open the Datalog Results or Tool Results and double click on a node:



Double clicking will open the image.

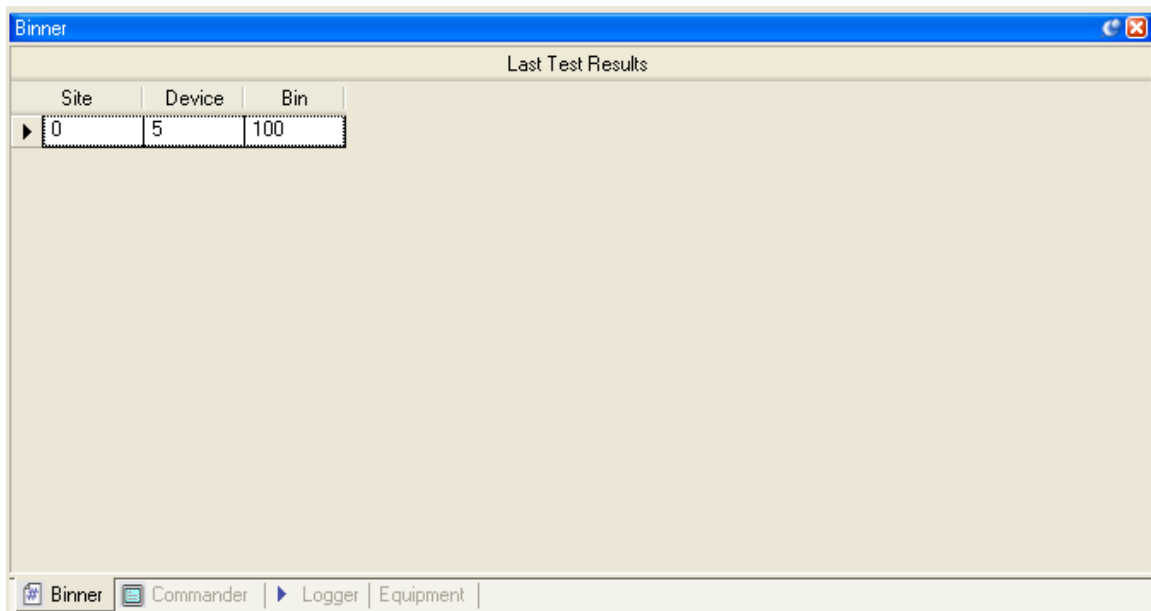
Note: Once the image viewer is open, you can use Control-C to copy it to MS Word.

4.7 GENERAL TOOLS

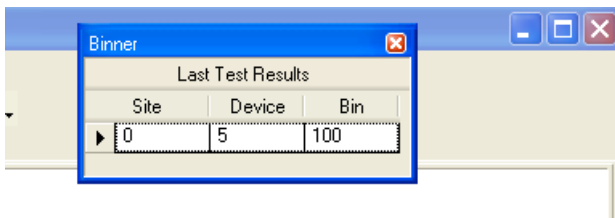
The lower right window houses four general tools:

- Binner
- Commander
- Logger
- Equipment

4.7.1 BINNER

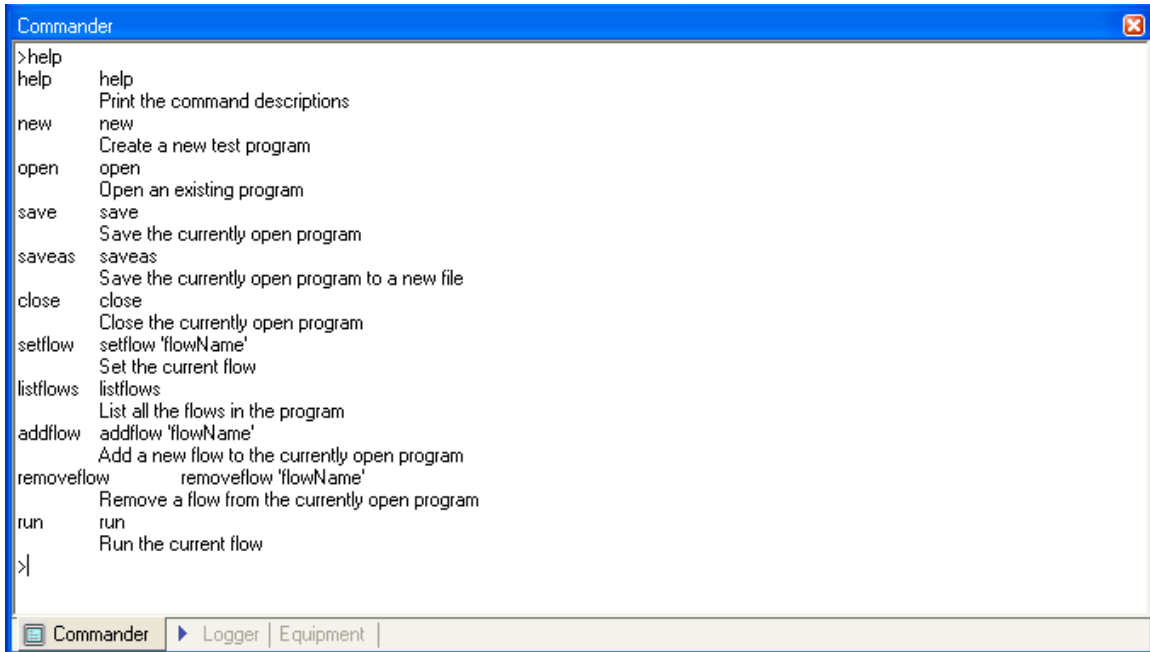


The Binner shows the results of the last test. A convenient way to use the Binner is to drag the tab off the window and move it somewhere convenient:



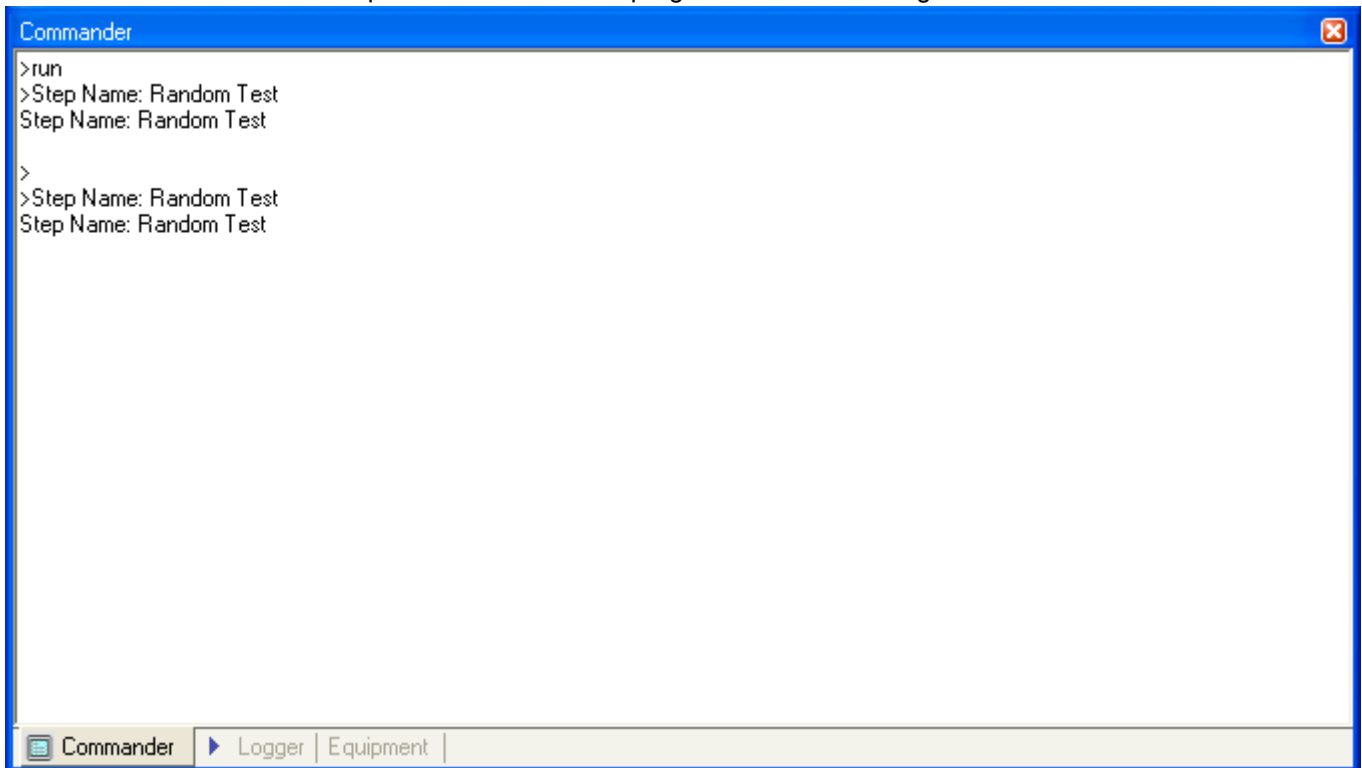
4.7.2 COMMANDER

The Commander is a command line window.



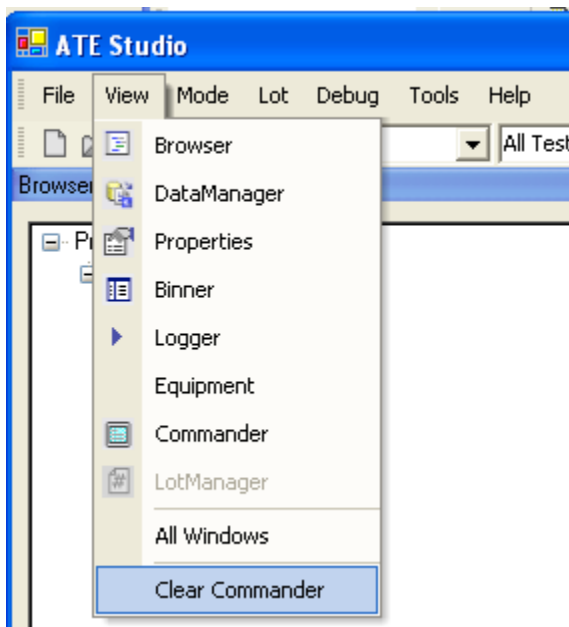
Typing “help” will list all the commands available.

The Commander is also a place where a test program can write diagnostic information.

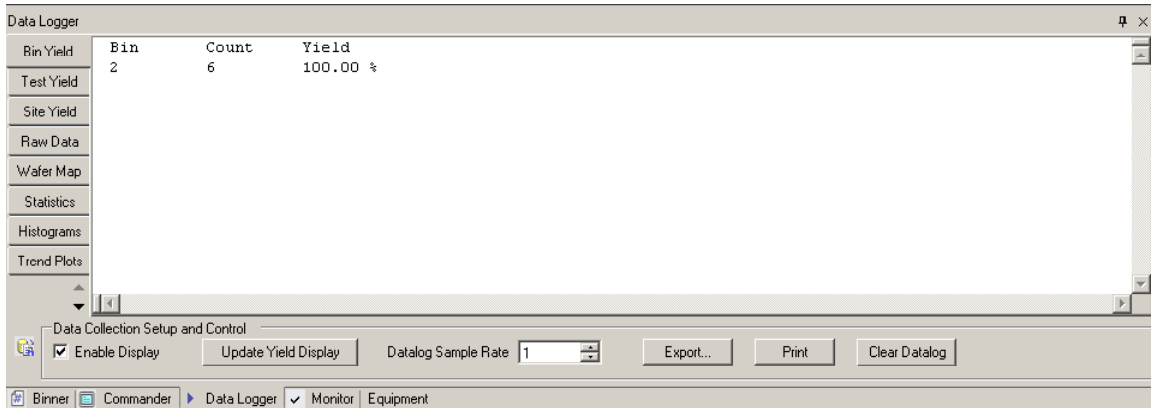


In the above example, the first run was via the command line “run” command. The second was generated by the Toolbar | Run icon. In both cases, the test method code wrote the message.

To clear the Commander window, use the View | Clear Commander menu:



4.7.3 DATA LOGGER



The Data Logger has six functions, one per tab:

- Bin Yield
- Test Yield
- Raw Data
- Statistics
- Histograms
- Scatter Plots

The Bin Yield tab shown above displays the yield in terms of bins:

Bin	Count	Yield
1	28	93.33
2	2	6.67

The Test Yield tab displays the yield in terms of tests:

Test	Count	Yield
1.0	28	93.33
1.1	30	100.00
1.2	30	100.00
1.3	30	100.00
1.4	30	100.00
1.5	30	100.00
1.6	30	100.00

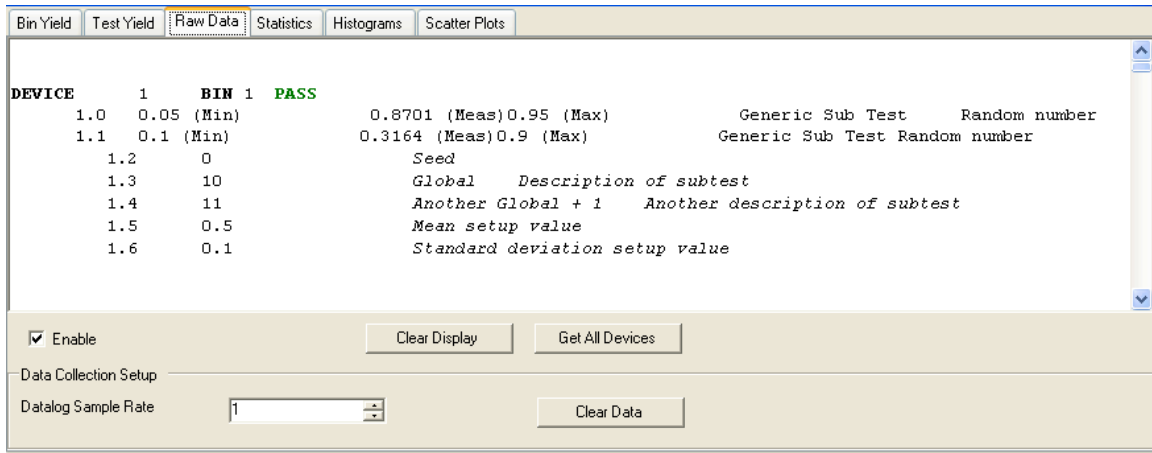
Both yield tabs update dynamically each time a device is tested.

The Raw Data tab displays the parametric data:

DEVICES	1	BIN 1	PASS			
1.0	0.05 (Min)			0.8701 (Meas)	0.95 (Max)	Generic Sub Test Random number
1.1	0.1 (Min)			0.3164 (Meas)	0.9 (Max)	Generic Sub Test Random number
1.2	0			Seed		
1.3	10			Global	Description of subtest	
1.4	11			Another Global + 1	Another description of subtest	
1.5	0.5			Mean setup value		
1.6	0.1			Standard deviation setup value		

The formatting of the data is defined by C# test code. There are formatter objects which the test code uses to describe how the values are printed, and control the tabbing.

The Raw Data tab has some controls that affect what data is displayed:

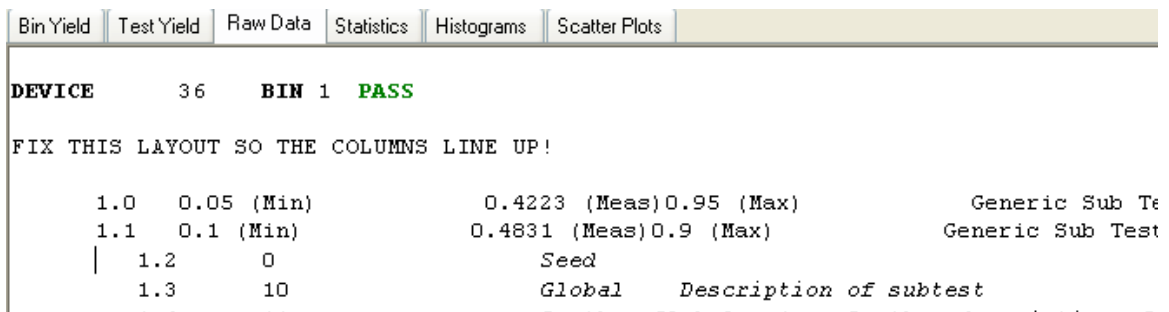


By default the data is displayed, but it can be disabled with the Enable check box. If disabled, the data is still collected, but it is not displayed. The display can be cleared with the Clear Display button, or one can fetch all data with the Get All Devices button.

It is also possible to control the data collection. The Datalog Sample Rate Numeric controls the sampling. The default value of 1 indicates that all device parametric data is recorded. A value of 2 means every other device is recorded. The Data Collection Setup portion of the window is displayed for every tab, and functions on all tabs equally. Thus the sample rate is independent of the tab chosen.

Pressing the Clear Data button erases the collected data from memory and from the UI. Once this is done, Get All Devices can not have the data redisplayed, because it has been completely deleted.

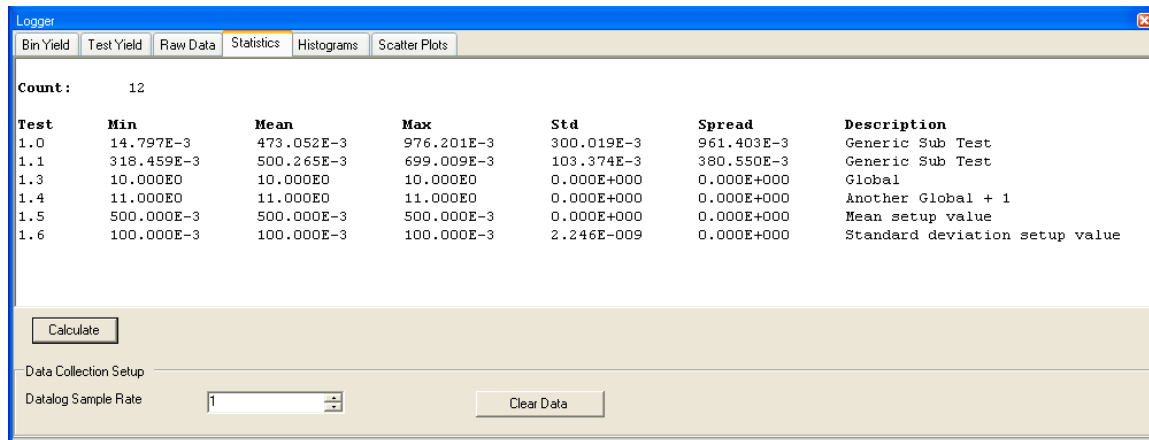
The raw data window is editable, so one can write notes in it:



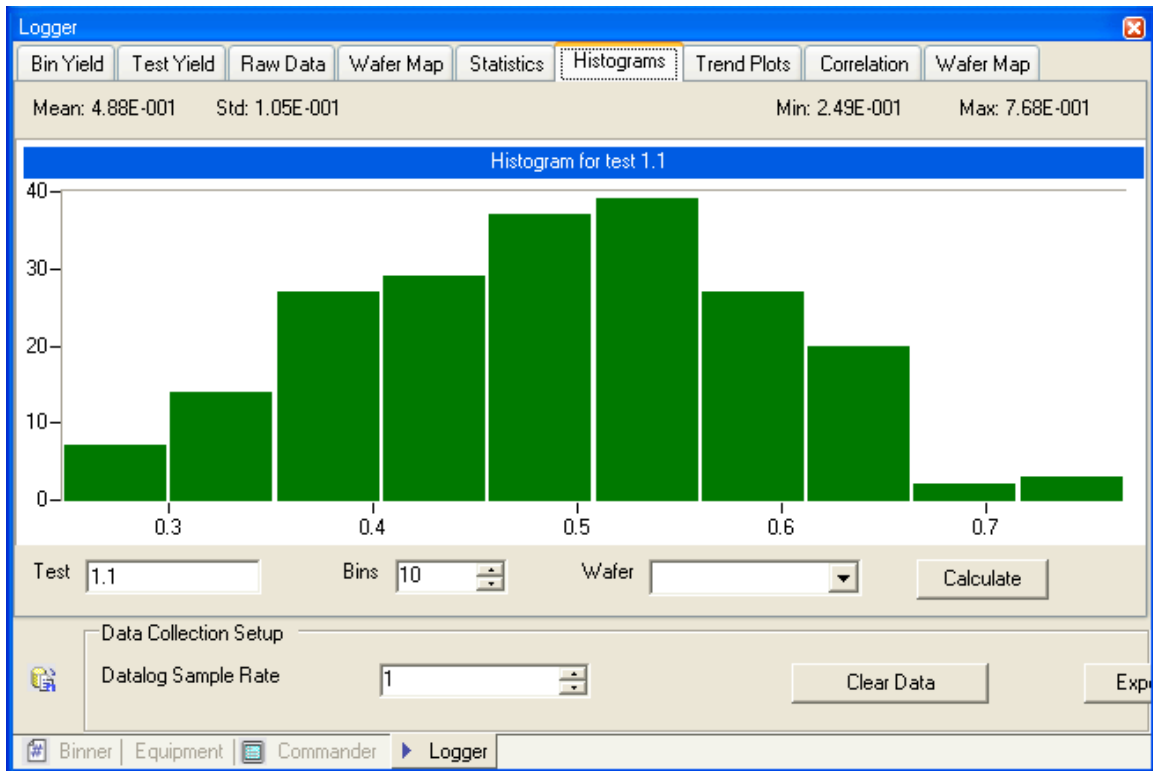
Edits are not saved by the system. However, you can cut and paste text to a word processor.

Raw data is also updated each time a device is tested, unless it is disabled. If the window is closed, the UI is still being updated, and when the window is opened, the raw data will display the devices that were tested while it was closed.

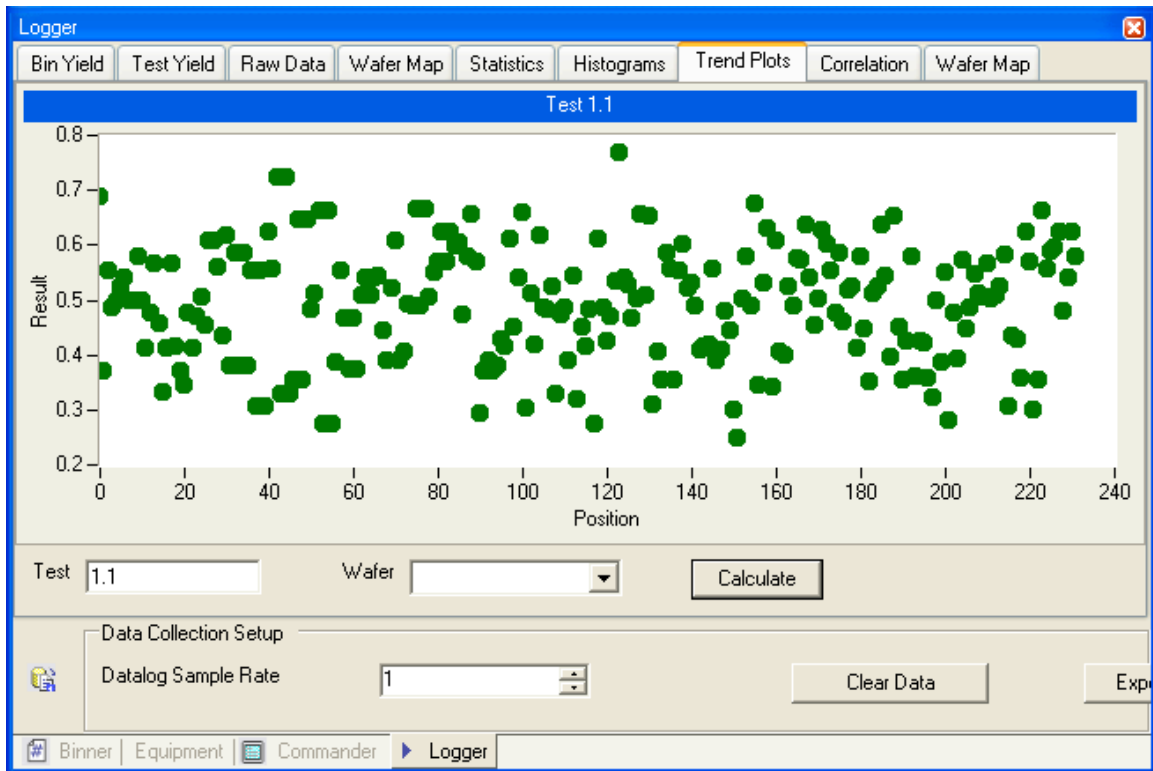
The Statistics tab displays statistical information about all devices tested:



Statistics are not updated each time a device is tested. Press the Calculate button to update the statistics. You can edit in the window and copy and paste to a word processor.

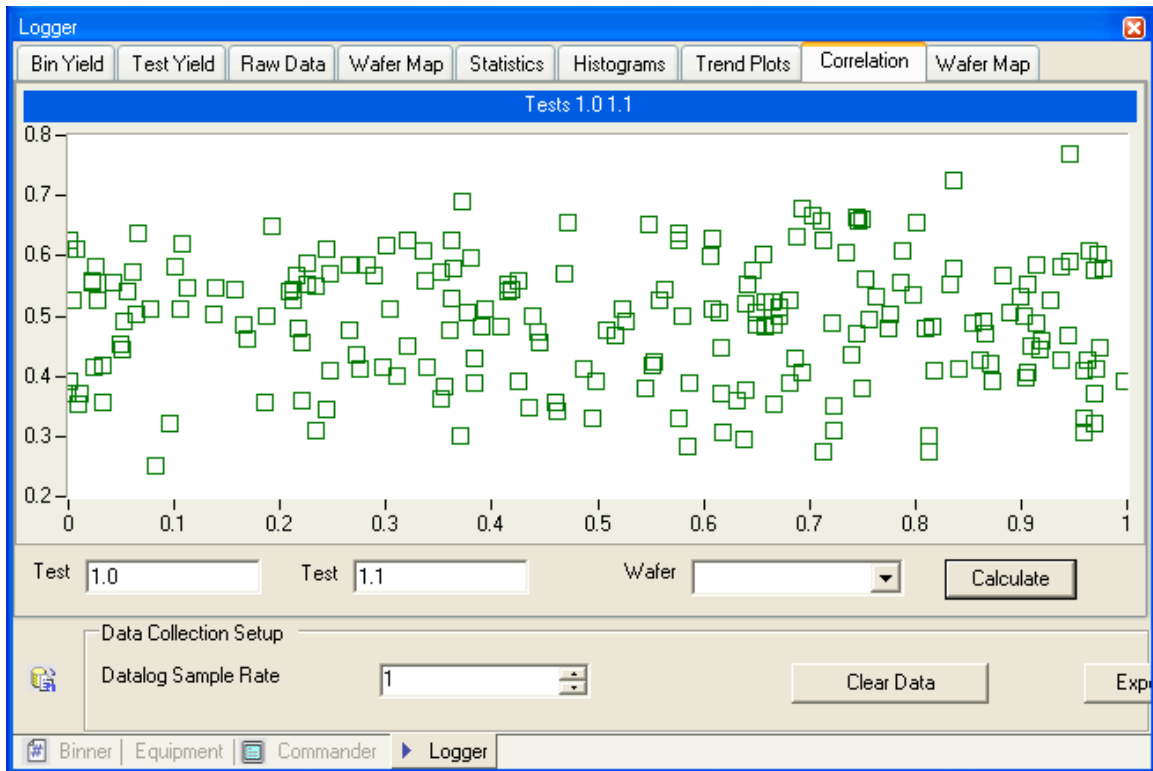


The histogram tab displays histograms. It does not update dynamically. To histogram a test, enter a test number, set the number of bins, and press Calculate.

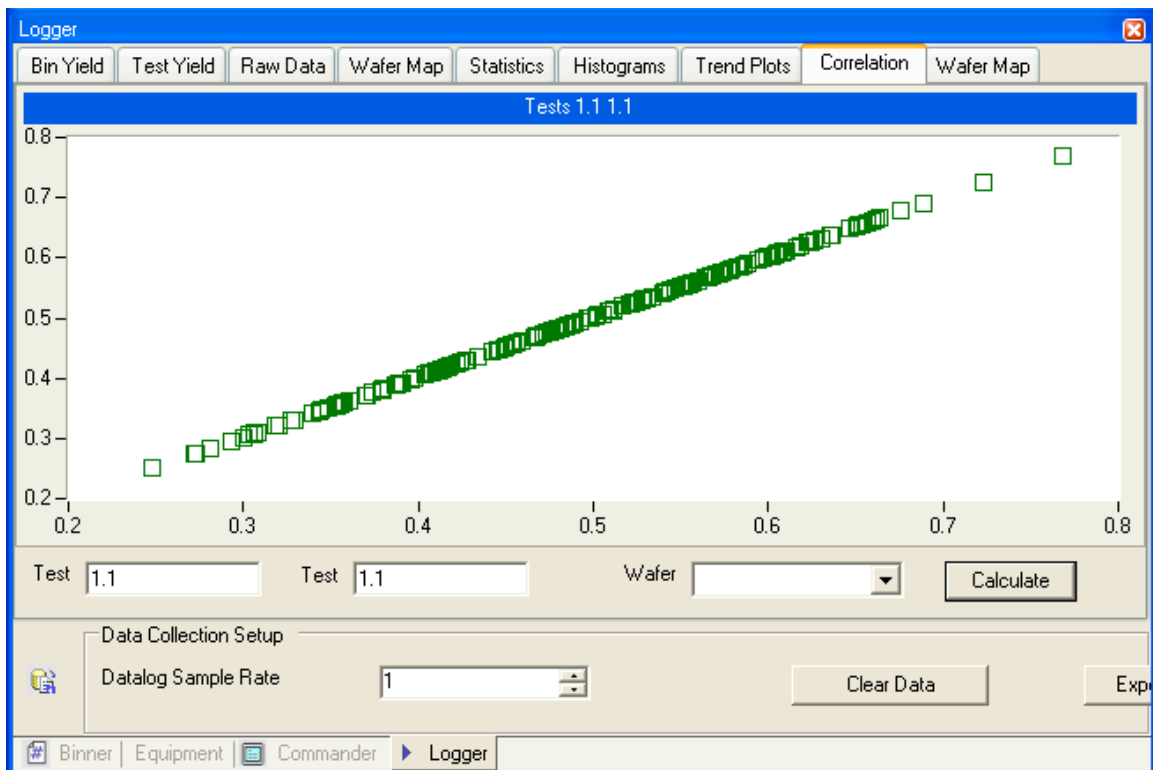


The Trend Plots tab displays trend data in scatter plots. It does not update dynamically. To plot data, enter a test number and press Calculate.

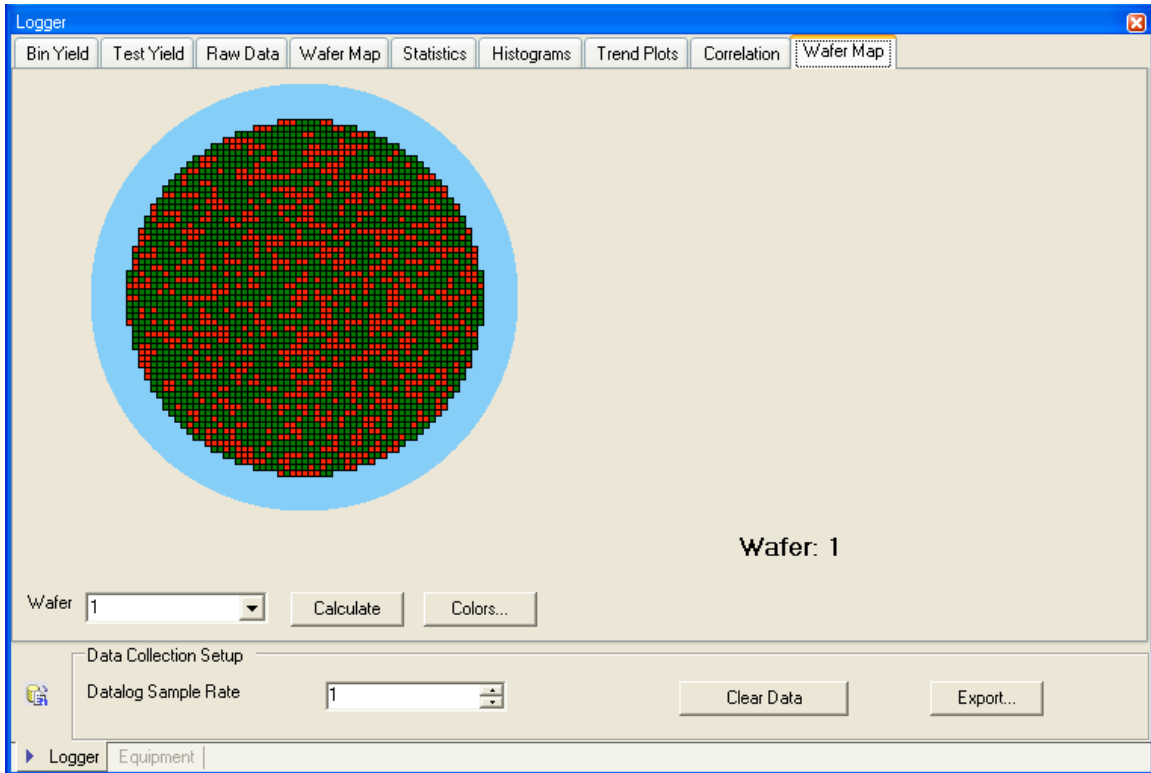
The correlation plot is an x-y plot. Here are two random parameters:



Here is perfectly correlated data:



The Wafer Map is used to look for pass fail patterns on wafers:

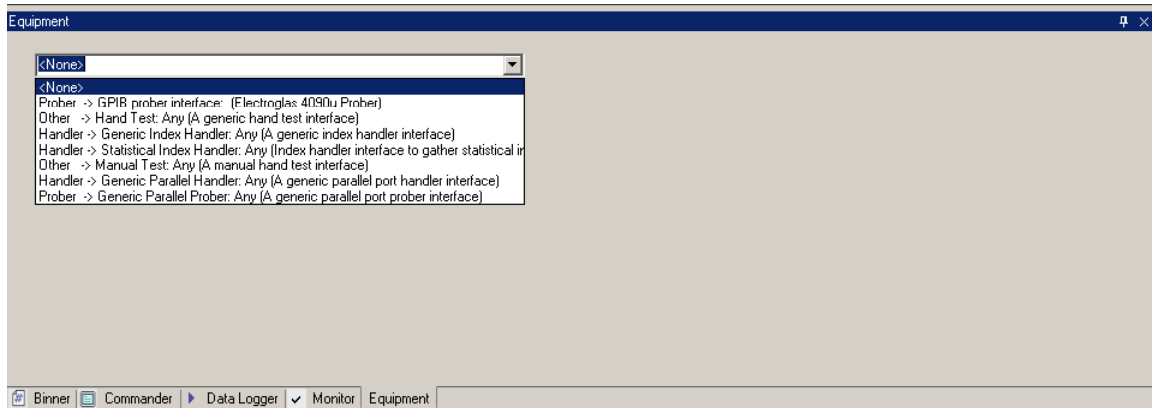


4.7.4 EQUIPMENT

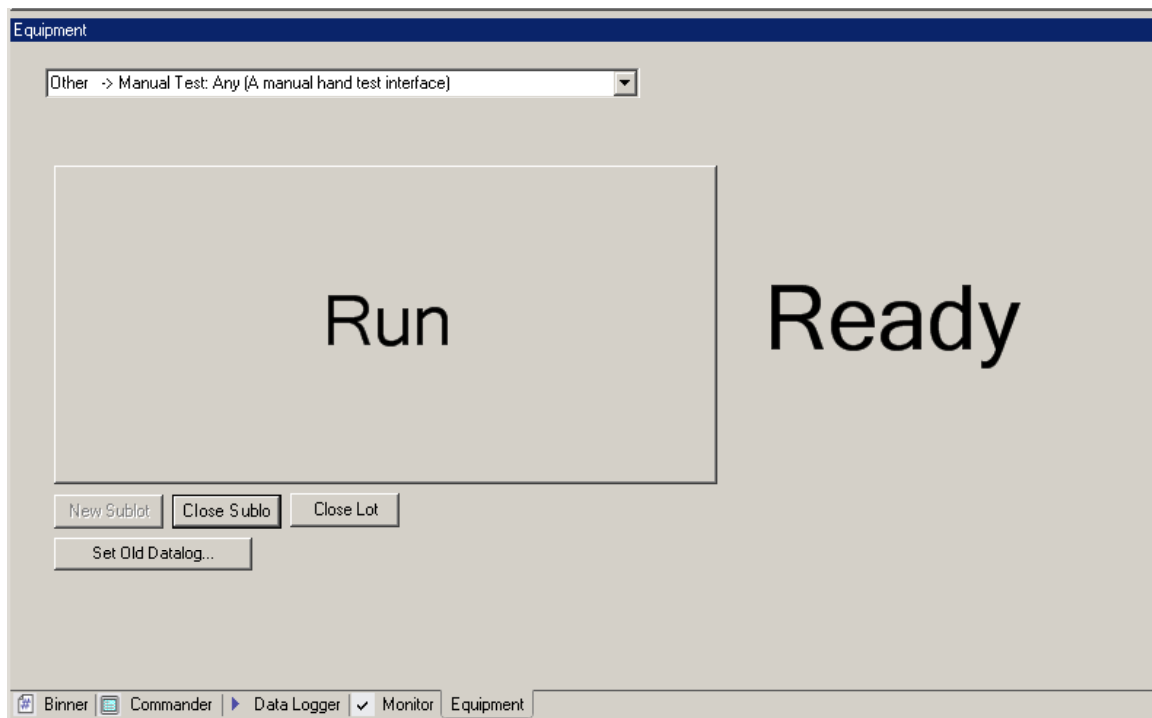
The Equipment tool is used to setup the handler or prober.

Handlers and probers are supported by plug-ins, like tools. They are placed in a well known directory and discovered when the system is run.

To select the equipment to be used, click on the drop down arrow of the equipment window:



This will list all the plug-ins available on the system. Select the equipment desired, and the window will update and display the plug-ins buttons and controls:



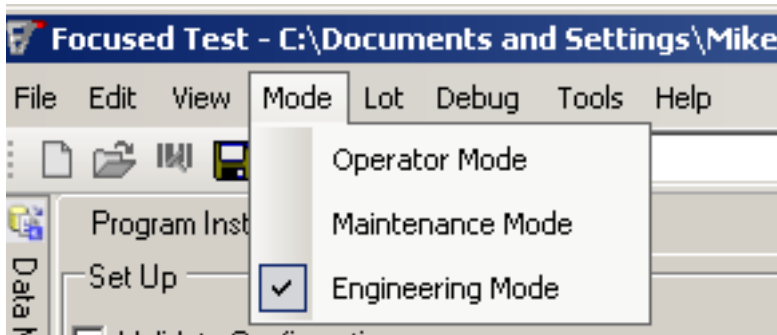
In this example, we have a generic hand teste interface. It has a simple run button to test.

5. USER INTERFACE MODES (PRODUCTION/ENGINEERING)

FTI Studio™ runs in three modes:

- Engineering Mode
- Maintenance Mode
- Operator Mode

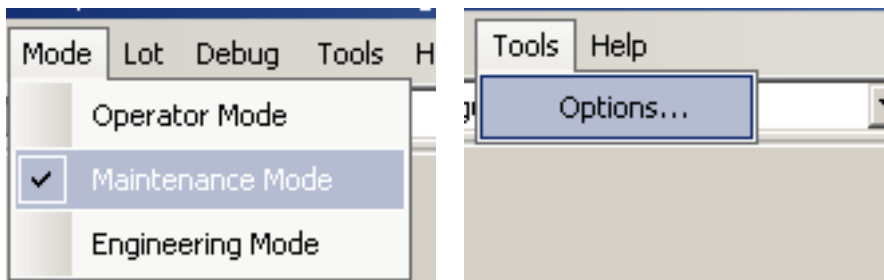
The Mode menu is used to change modes:



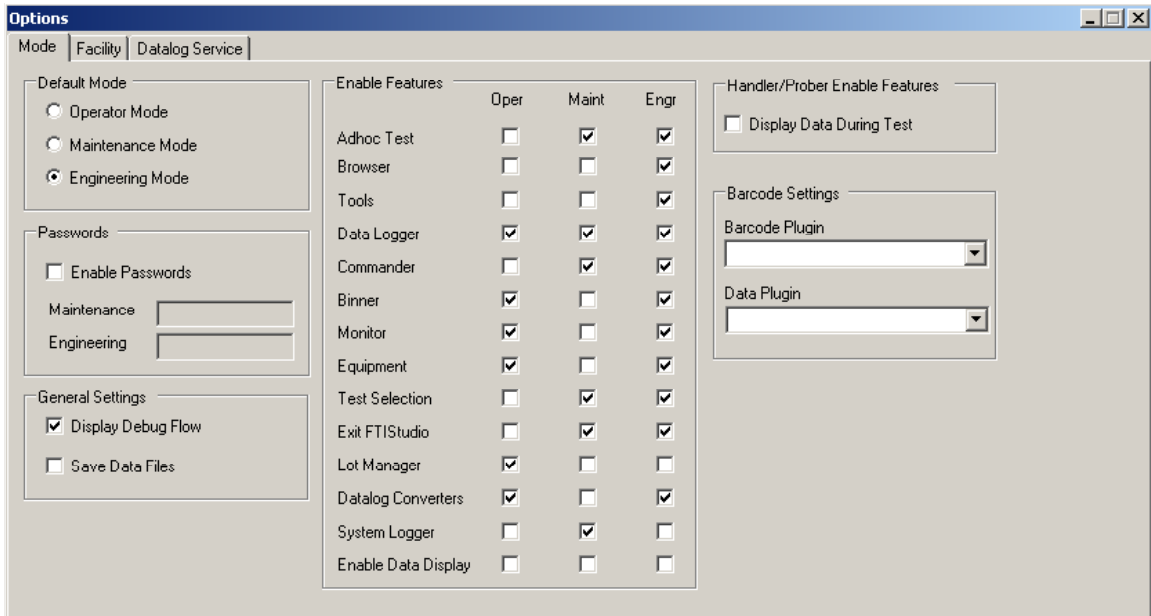
The default mode when FTI Studio™ is installed is Engineering Mode.

There are some rules about what a person can do in each mode. Some of these rules are fixed, and some are changeable. One fixed rule is that the operator can not change rules from any mode but Maintenance Mode.

Once in Maintenance Mode, one can use the options to change the other rules:



The options dialog is used to set the rules:

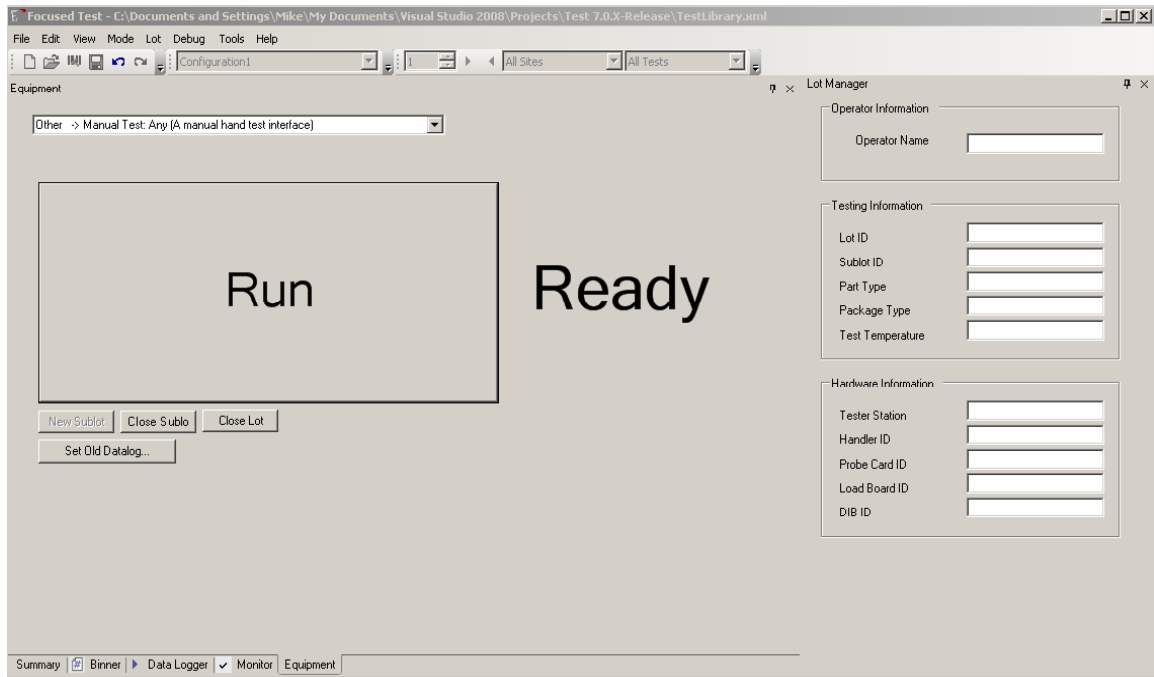


The Default Mode is used to set the mode that FTI Studio™ is in when it is run.

If Enable Passwords is checked, you can enter passwords for Maintenance Mode and Engineering Mode. However, once passwords are enabled, Test Studio™ always starts in Operator Mode. This means anyone can operate the system in Operator Mode, but passwords are required for other modes. This prevents manufacturing personnel from performing some functions.

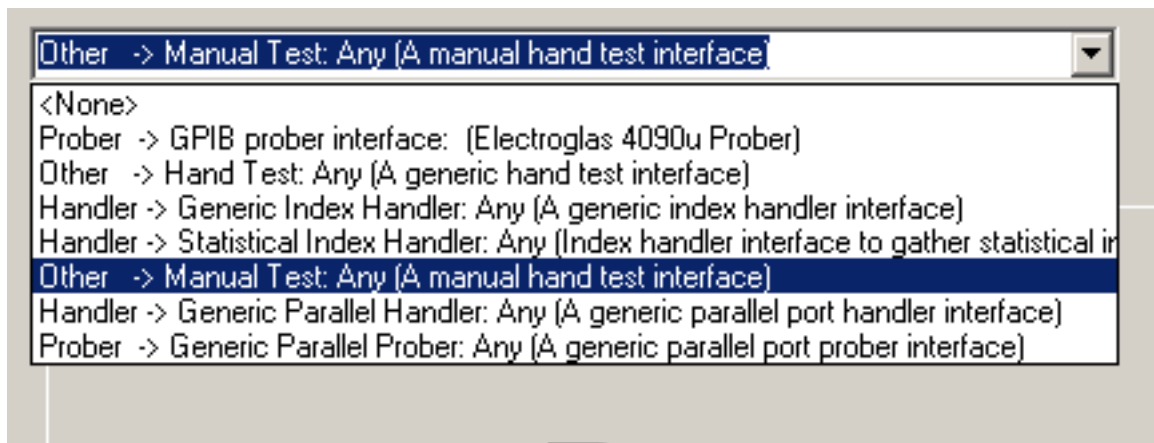
The Enable Features area is used to determine what features are available in each mode.

6. PRODUCTION OPERATION

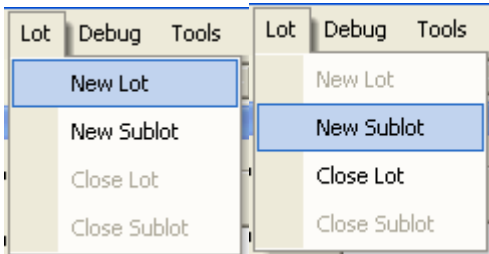


In operator mode the Lot Manager is displayed on the right. And the Equipment tool is shown to the right.

The Lot Manager is used to input information about a lot and subplot that will be saved with the datalog information when the lot is closed. The Equipment tool is used to setup the handler or prober, which you can select by using the drop down box.

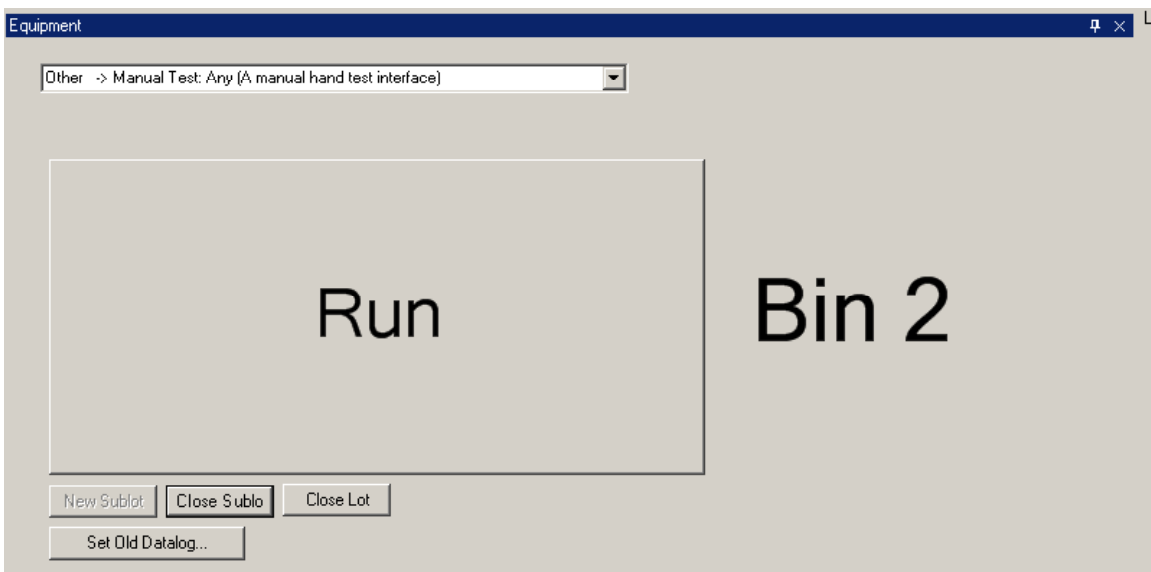


Once the equipment is selected and setup, and a few setup devices have been checked, a lot and subplot should be created with the lot menu:

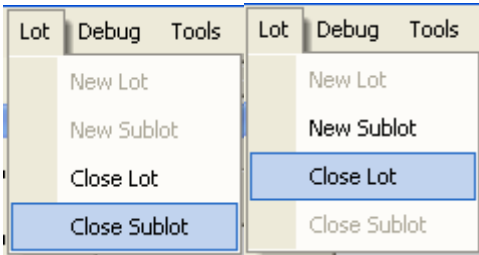


Note: Some equipment plug-ins take control over lot and subplot functions.

Once the lot is open, begin the production test. In this example, pressing the Run button:



When testing is complete, it is time to close the subplot and/or lot:



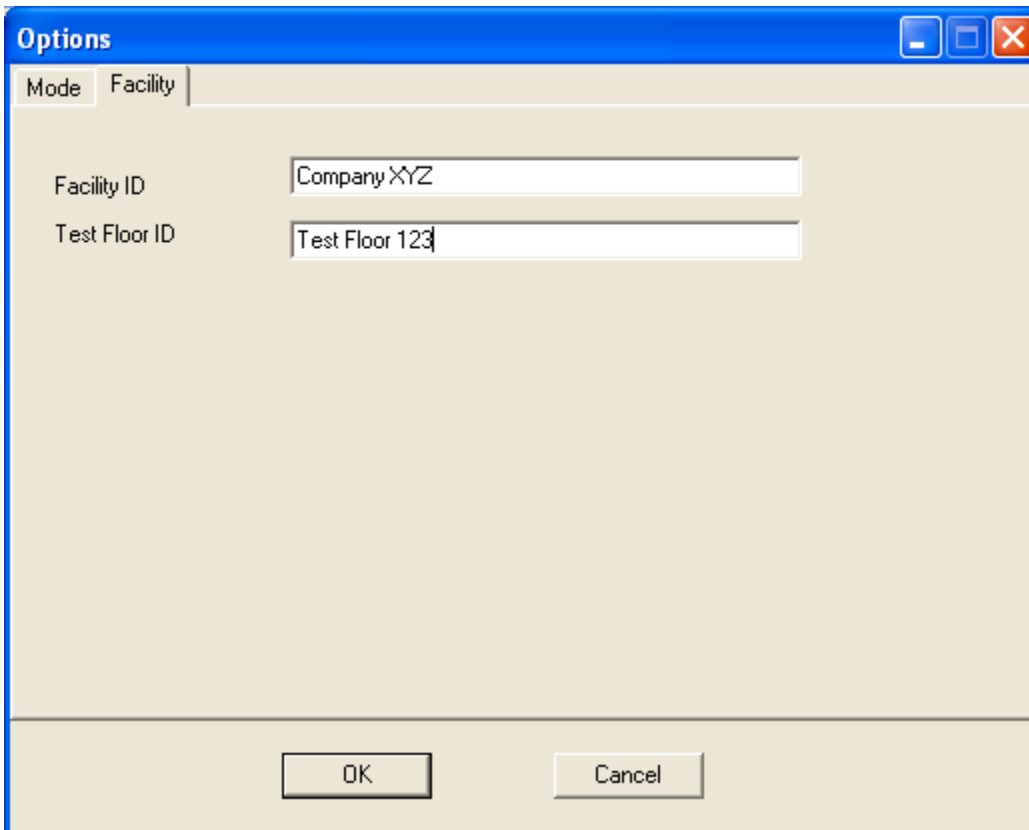
If you close the lot while the subplot is open, the subplot is automatically closed.

Note: Some equipment plugins close sublots and lots for you.

When a subplot is closed, the STDF file is copied to the data directory where the Datalog Service can process it, normally C:\Program Files\Focused Test Inc\FTI Studio\ProductionData

The Lot Manager data is stored in the datalog when a subplot or lot is closed. Therefore, the operator must fill in the data before closing.

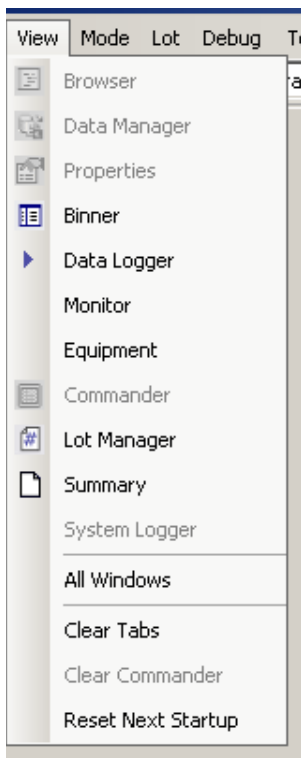
The options dialog has a couple of settings that pertain to lot information:



The two id values are stored in the datalog when a subplot or lot is closed.

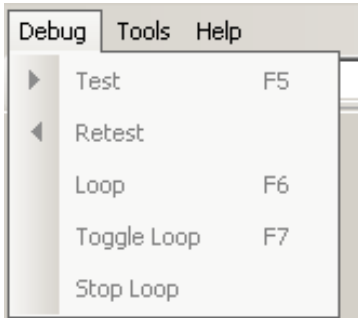
7. MISCELLANEOUS

7.1 VIEW MENU

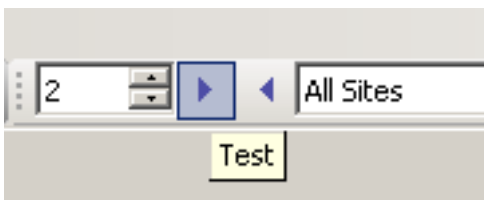


The view menu is used to force a window to display if it has been closed. If the window is not allowed because of settings in options, All Windows will not cause it to be displayed.

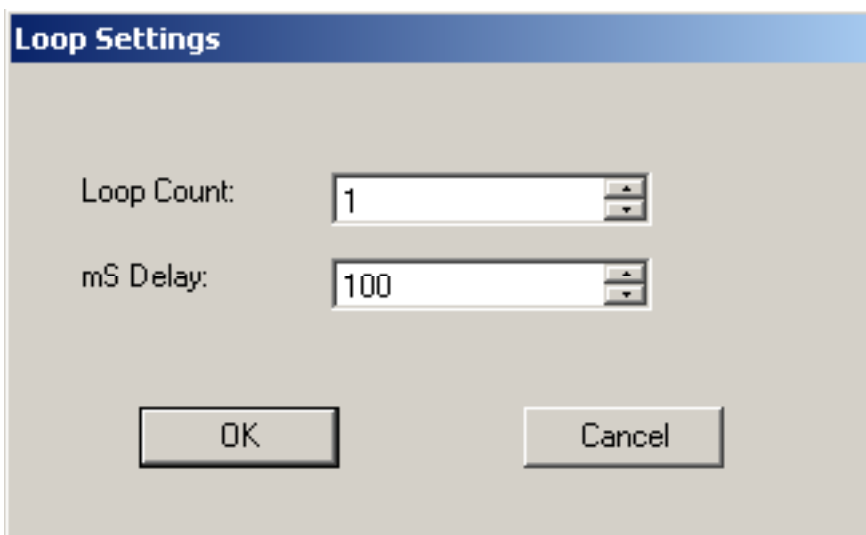
7.2 DEBUG MENU



The debug menu allows you to run or loop the test program. You can also run from the toolbar:

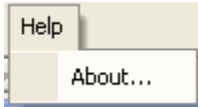


The Loop menu will display a dialog that will loop run a test program:



7.3 HELP MENU

The help menu does not contain any manuals, but it does contain an About... box.



The About... box displays copyright information for all subsystems, tools, and test methods:

