Operators Manual
For

Pulse Instruments

PI-DATS Automated Test System Software

March 2016

1234 Francisco Street
Torrance, California 90502
(310) 515-5330
support@pulseinstruments.com
www.pulseinstruments.com

Table of Contents

Table of Figures

1. Introduction

PI-DATS (**P**ulse **I**nstruments **D**ata **A**cquisition and **T**est **S**oftware) is the automated test system software for use with Pulse Instruments imaging test systems. PI-DATS has a menu operated, object-oriented architecture which provides a flexible, low-maintenance software environment for the specification, real time control, data acquisition, data reduction/display and storage processes associated with testing infrared and visible imagers.

PI-DATS easily configures any test sequence and controls hardware. PI-DATS allows the operator to define test conditions prior to test execution using a Test Plan edit screen. The operator can enter or modify all information in a Test Plan pertinent to the type of testing being performed. Multiple Test Plans can be stored in the same file, ensuring consistency of testing procedures.

PI-DATS can operate either in Test Plan mode or in Controller mode. Test Plan mode allows creation, editing, and execution of automated test plans, while Controller mode allows immediate control of all hardware settings and functions. All hardware configuration and settings are preserved between the two modes, and switching between modes requires a single mouse-click, allowing the user to test and troubleshoot test plan parameters easily.

1.1. Reading This Manual

PI-DATS is closely related to PI-Controller and is integrated with PI-PAT. Setting up and running test plans assumes an operating knowledge of these applications as well as with the underlying hardware (e.g. Pulse Instruments mainframes, clock drivers, DC bias cards, pattern generators, and data acquisition systems).

The recommended approach for learning PI-DATS is the following:

- Read the System Overview section of this manual (next)
- Read the Operators Manuals for your individual instruments and their software applications
    - PI-PAT for Pulse Instruments pattern generators
    - PI-Controller for Pulse Instruments clock driver cards and DC bias cards
    - PI-Controller for the PI-3105 Data Acquisition subsystem
    - Work through the tutorials and sample files included with each software package
- Return to this PI-DATS manual to read the test planning and reference sections

1.2. Requirements

PI-DATS runs under Windows 7, with Microsoft Office 2013, and will typically be installed and configured on a computer purchased with the Pulse Instruments system. Minimum system requirements for customer installations:

- Core2Duo/2 GHz or better processor.
- 1 GB of RAM for systems without data acquisition subsystems, or 2 GB for systems with data acquisition (4 GB recommended).
- Windows 7 Professional Service Pack 1.
- Microsoft Office (any version) 2013 or newer
- National Instruments GPIB interface card or GPIB-USB adapter required for GPIB-capable instruments.
- Video display with minimum 1,920 x 1,080 resolution. 2 displays recommended for systems with data acquisition.

If PI-DATS will be run primarily to control instrumentation, with image analysis to be performed on another PC, then the minimum specs are sufficient. If the on-board CPU will be performing image analysis on large data sets, then a Quad-Core processor with 2+ GB of RAM is recommended.

Memory requirements will vary by test application. As a general rule, the minimum requirement will be 4 bytes/pixel for your largest single data acquisition operation, plus 1 GB of operating system “headroom.” For example, if you intend to capture 100 contiguous frames of data from a 1K x 1K imager, then your minimum RAM requirement would be approximately 4 Bytes x (1024 x 1024 x 100) + 1024 MB = 1,428 MB. Your system may acquire larger data sets, but they will be paged to disk, which can result in slow performance.

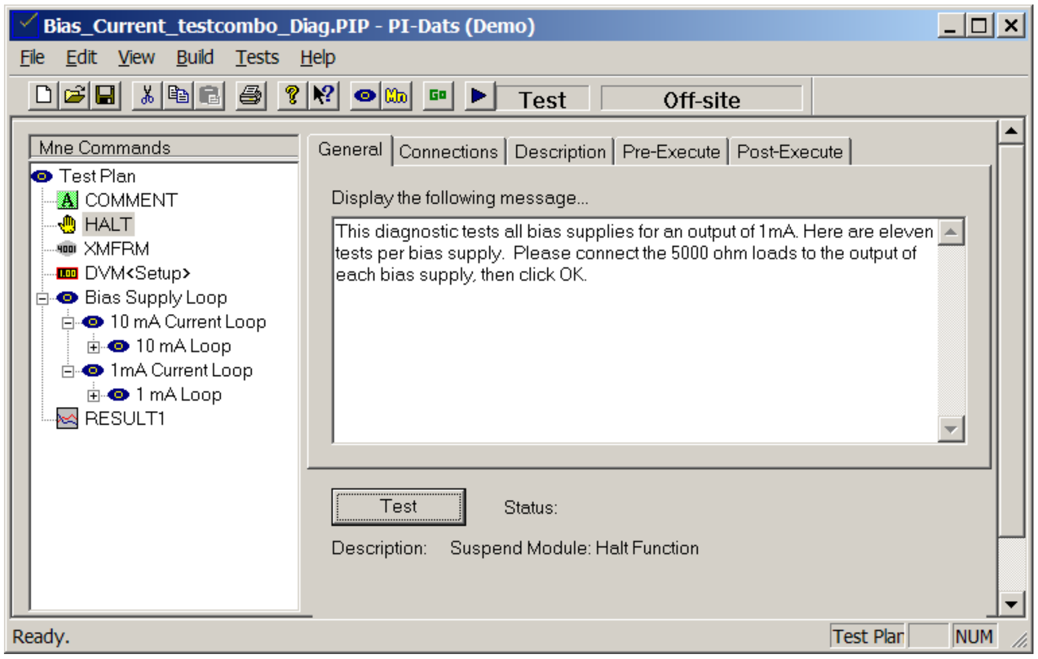
Note that your data acquisition card(s) must also have enough capacity to store the acquired frames.

1.3. Known Issues, PI-DATS 2.708

- The RESULT mnemonic does not properly return some meta-data parameters (e.g. Number of Frames, etc) if multiple masters are enabled.

2. System Overview

2.1. Overview



Display the following message...

This diagnostic tests all bias supplies for an output of 1mA. Here are eleven tests per bias supply. Please connect the 5000 ohm loads to the output of each bias supply, then click OK.

Description: Suspend Module: Halt Function

Figure 1: PI-DATS Test Plan

PI-DATS operates by creating and manipulating mnemonic representations of each object in your test system (instruments, procedures, and data sets). Mnemonics contain GUI-based controls for the programmable parameters of each object. Visual placement of mnemonics in the test sequence, along with looping and branching controls, allow you to construct sophisticated test plans without learning any language or syntax rules.

Test plans are built with groups and sequences of mnemonics. Each mnemonic represents the state of a piece of hardware, a set of data, or a procedure. As PI-DATS executes the test plan, it steps through and evaluates each mnemonic, executes it if appropriate, then loops and branches as necessary.

Furthermore, PI-DATS can be run in Controller mode (identical to PI-Controller), which allows immediate, real-time control over the hardware. This allows for rapid development and testing of test plan parameters. PI-DATS mnemonics are derived directly from the PI-Controller mnemonics, allowing the same files and settings used for engineering characterization to be used in production test. Any references to PI-Controller are synonymous with PI-DATS running in Controller mode, and vice-versa.

Analysis and data reduction are handled via a “Result” mnemonic that manipulates tables of data. Data can be generated from measurement devices (DMMs, oscilloscopes, and data acquisition systems) or it can be generated by the test plan itself (e.g. programmed settings and test plan environment parameters). Results can be displayed on-screen, saved to disk, or exported to MATLAB or Excel for reporting or additional processing.

Data processing in PI-DATS can be used to pass or fail devices in Production, to optimize device parameters in R&D and device characterization, or to post-process image data for visualization, display, and analysis.

2.2. Available Modules

PI-DATS is designed to work with the following modules. With the exception of the PI-Hardware Manager, each module controls a specific set of equipment, and therefore not all modules are required for your particular system.

- PI-Hardware manager (PIHWare.ocx, Required)
- PI-11000, or PI-2005 with PI-Bus card and PI-4002 mainframes (PIMnFrm.ocx)
- PI-Pattern Generator Controller (PIPat.ocx)
- PI-3105 Data Acquisition Controller (pidatacq.ocx)
- PI-Digital Voltmeter (PIDvm.ocx)
- PI-Oscilloscope (PIScope.ocx)
- PI-Function Generator (PIFunGen.ocx)
- PI-Blackbody Controller (PIBBody.ocx)
- PI-Motion Controller (PIMotion.ocx)
- PI-Generic GPIB (PIIEEE.ocx)
- PI-Connections (PIConn.ocx)
- PI-Result (PIResult.ocx)
- PI-Software Functions (PISWare.ocx)
- PI-Security (PISecure.ocx)

2.3. Mnemonics

A **Mnemonic** is a graphical representation of an object in your test system. Typically a mnemonic represents a hardware instrument, e.g. a DC Bias Channel. Mnemonics can be assigned user-defined names to facilitate test planning and maintenance. Some examples:

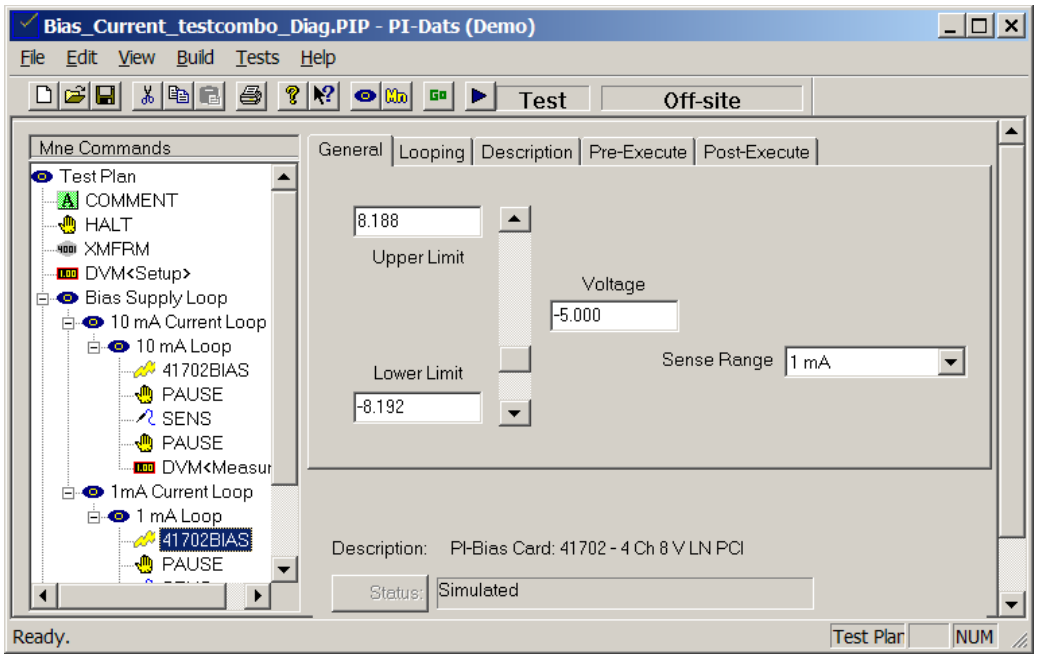
Mnemonic	Function or Instrument Module
BIAS1	Channel 1 of 40750 Bias card of PI-4001
CKDR2	Channel 1 of 40490 Clock Driver
RESULT1	Non-hardware component for managing measured data.

Mnemonics can also represent groups of mnemonics, sets of data, and operational controls for a Test Plan.

A set of property pages is associated with each instance of a mnemonic. Property pages will contain GUI-based controls for or information about the object represented by the mnemonic. For example, the **General** property page for a DC Bias channel will contain controls for the voltage settings, voltage limits, and current-sense ranges. The **Description** property page for that mnemonic would contain the DC Bias channel's model number and physical location in the system.

Other property pages may contain **Looping** parameters, and conditions to be evaluated before the mnemonic is executed (**Pre-Execute)** and actions to be taken after the mnemonic is executed **Post-Execute**).

2.3.1. Hardware Controls



Test Plan
- COMMENT
- HALT
- XMFRM
- DVM<Setup>
- Bias Supply Loop
  - 10 mA Current Loop
    - 10 mA Loop
      - 41702BIAS
      - PAUSE
      - SENS
      - PAUSE
      - DVM<Measur
  - 1mA Current Loop
    - 1 mA Loop
      - 41702BIAS
      - PAUSE

Upper Limit: 8.188

Lower Limit: -8.192

Voltage: -5.000

Sense Range: 1 mA

Description: PI-Bias Card: 41702 - 4 Ch 8 V LN PCI

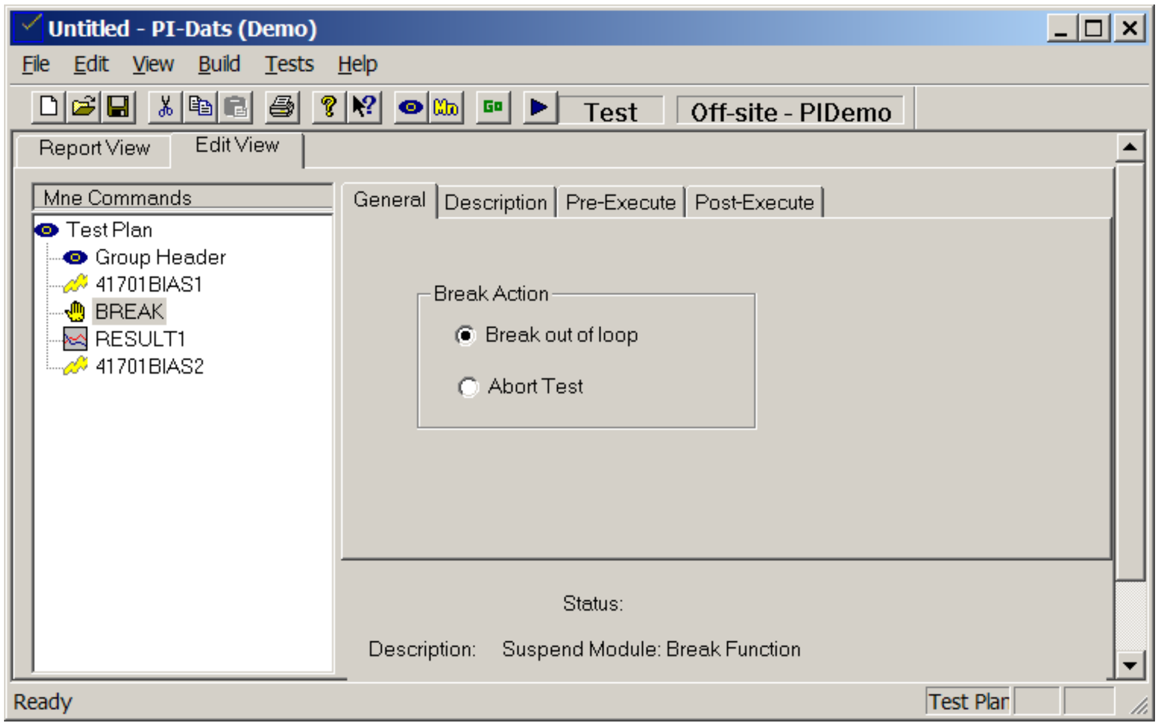
Status: Simulated

Figure 2: DC Bias mnemonic and General property page

Note that the **General** controls are derived from the real-time controls that appear in Controller mode (PI-Controller) and in PI-PAT. Familiarity with PI-Controller, PI-PAT, and the underlying hardware is essential to understanding test planning in PI-DATS. Operators should be proficient in controlling all instrumentation in PI-Controller, including acquiring data, before using PI-DATS.

2.3.2. Non-Hardware Controls

Mnemonics are also used to represent other system objects, such as measurement data and test plan controls.



Break Action

[✓ Break out of loop] Abort Test

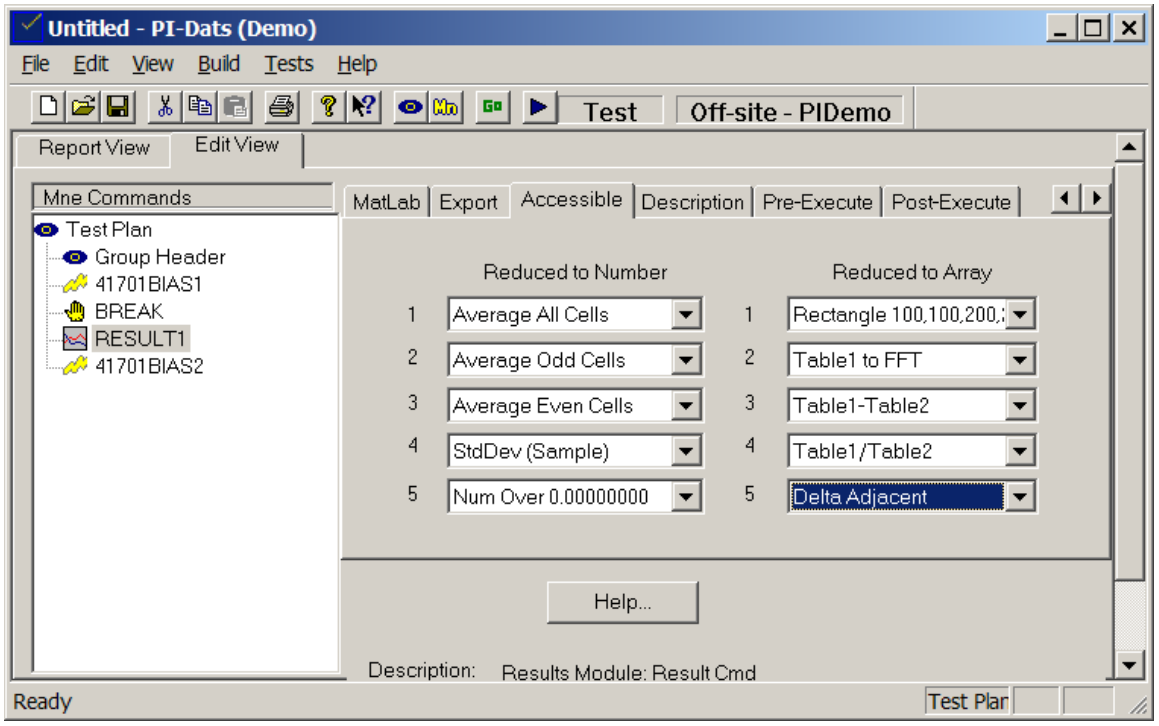
Status:

Description: Suspend Module: Break Function

Figure 3: Break mnemonic General property page

In **Figure 3**, a **Break** mnemonic can be used to break out of a loop when certain conditions are met. Other test plan controls can assign values to variables, ask for user input, branch to different sections of a test plan, etc.

2.3.3. Test Plan Results



	Reduced to Number		Reduced to Array
1	Average All Cells	1	Rectangle 100,100,200,:
2	Average Odd Cells	2	Table1 to FFT
3	Average Even Cells	3	Table1-Table2
4	StdDev (Sample)	4	Table1/Table2
5	Num Over 0.00000000	5	Delta Adjacent

Description: Results Module: Result Cmd

Figure 4: Result mnemonic and Accessible property page

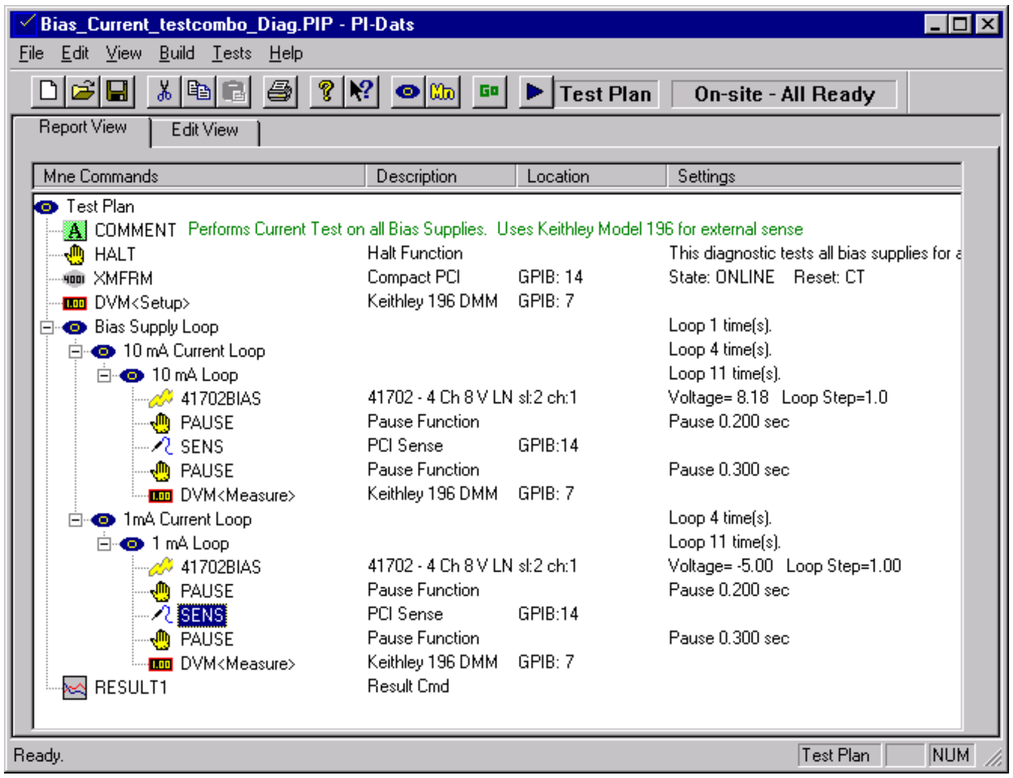
In **Figure 4**, measurement data can be made **Accessible** to other objects by reducing them to numbers (scalars) or by performing array functions on them. A sequence of Result mnemonics can be used to perform complex data analysis and/or post-processing of acquired data.

2.4. Test Plans and Group Headers

A **Test Plan** is a collection of mnemonics organized in tree-structured fashion. When executed, a Test Run will step through a **Test Plan's** mnemonics in sequence, looping and branching according to measurement results or programmed variables.

Group Headers are used to group one or more mnemonics for looping and branching.

2.4.1. Test Plan Example


Bias_Current_testcombo_Diag.PIP - PI-Dats
File Edit View Build Tests Help
Test Plan
On-site - All Ready
Report View
Edit View
Ready.
Test Plan
NUM

Mne Commands	Description	Location	Settings
Test Plan			
COMMENT Performs Current Test on all Bias Supplies. Uses Keithley Model 196 for external sense			
HALT	Halt Function		This diagnostic tests all bias supplies for a
XMFRM	Compact PCI	GPIB: 14	State: ONLINE Reset: CT
DVM<Setup>	Keithley 196 DMM	GPIB: 7	
Bias Supply Loop			Loop 1 time(s).
10 mA Current Loop			Loop 4 time(s).
10 mA Loop			Loop 11 time(s).
41702BIAS	41702 - 4 Ch 8 V LN sl:2 ch:1		Voltage= 8.18 Loop Step=1.0
PAUSE	Pause Function		Pause 0.200 sec
SENS	PCI Sense	GPIB:14	
PAUSE	Pause Function		Pause 0.300 sec
DVM<Measure>	Keithley 196 DMM	GPIB: 7	
1mA Current Loop			Loop 4 time(s).
1 mA Loop			Loop 11 time(s).
41702BIAS	41702 - 4 Ch 8 V LN sl:2 ch:1		Voltage= -5.00 Loop Step=1.00
PAUSE	Pause Function		Pause 0.200 sec
SENS	PCI Sense	GPIB:14	
PAUSE	Pause Function		Pause 0.300 sec
DVM<Measure>	Keithley 196 DMM	GPIB: 7	
RESULT1	Result Cmd		

Figure 5: PI-DATS Test Plan, Report View

Figure 5, above, shows a simple test plan to loop through bias voltage settings. In addition to the mnemonics representing instrumentation (Bias, Sense, and DVM), there are mnemonics for data (Result1), for Group Headers (Bias Supply Loop, 10 mA Current Loop, etc), test plan control (Halt, Pause), and for operator information (Comment).

This test plan was constructed by inserting mnemonics from a table of all available system objects, and then setting the programmable parameters for each mnemonic.

2.5. PIP File (Pulse Instruments Part file)

PI-DATS stores Test Plans in PIP files. A PIP file contains one or more test plans with all their mnemonics and their configurations and all hardware components with their connections and attributes. Multiple test plans can be stored within a single PIP file, to facilitate testing of parts or to control power-up/down sequencing of a device.

Except for hardware changes such as card changes, cabling and the physical interface to a device, switching test applications can be accomplished simply by loading a different PIP file.

All test plan information, with the exception of the pattern/timing files and image data sets, is contained within a single file to facilitate selection of a test plan and deployment of standard testing procedures in a production environment. Timing files for the Pulse Instruments pattern generators (PI-2005, PI-2000 and PI-5800A) are stored in separate files that are referenced from within PIP files.

PIP files may be stored anywhere, as well as emailed or shared on a network. References within a PIP file to pattern files, data files, or data export locations can be made with relative paths and filenames to facilitate copying of test plans among systems.

2.6. HWR File (Pulse Instruments Hardware file)

PI-DATS uses a special file (*.hwr) to store hardware configuration information. The default HWR file (**PIOnSite.hwr**) is updated every time PI-DATS detects a hardware change or is asked by the user to scan for new instrumentation. Under normal conditions, no direct interaction with the **PIOnSite.hwr** file is required.

Other HWR files may be created (e.g. the included **Demo.hwr**) to reflect available hardware on a different test station, or to edit test plans "offline" when there is no hardware available. HWR files should not be edited directly by the user, and should be stored in the **C:\Program Files\PULSE20** directory. See **Section 4. Hardware Configuration** for details on hardware configuration.

2.7. Test System Administration

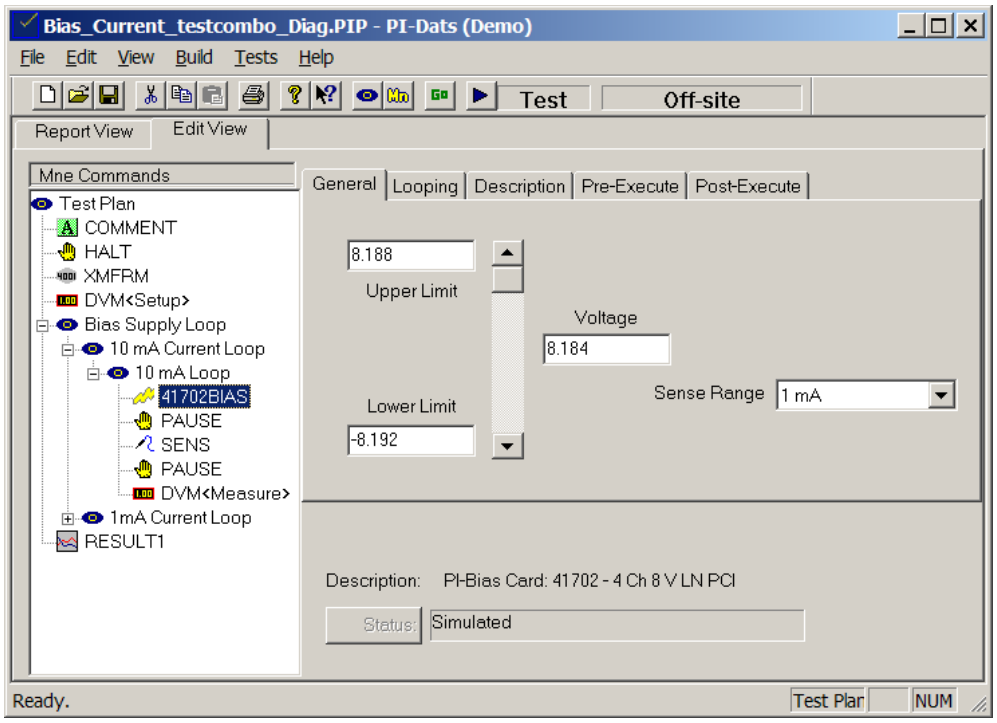
PI-DATS maintains a user and group database to protect the integrity of the testing environment. Test Plan files may then be assigned a list users and groups with passwords and varying levels of permissions to edit and run the test plan.

2.8. STOP HERE

At this point, it is highly recommended that the user read all the Operators Manuals for their Pulse Instruments equipment before resuming this manual.

3. User Interface

3.1. Main Window



Bias_Current_testcombo_Diag.PIP - PI-Dats (Demo)

Test   Off-site

Mne Commands

- Test Plan
  - COMMENT
  - HALT
  - XMFRM
  - DVM<Setup>
  - Bias Supply Loop
    - 10 mA Current Loop
      - 10 mA Loop
        - 41702BIAS
        - PAUSE
        - SENS
        - PAUSE
        - DVM<Measure>
    - 1mA Current Loop
  - RESULT1

Upper Limit: 8.188

Voltage: 8.184

Sense Range: 1 mA

Lower Limit: -8.192

Description: PI-Bias Card: 41702 - 4 Ch 8 V LN PCI

Status: Simulated

Ready.   Test Plar   NUM

Figure 6: PI-DATS Main Window, Edit View

Figure 6 shows the PI-DATS main window with a sample test plan loaded. Major user interface elements include:

- **Mnemonic Commands pane**
- **Property pages**
- **Toolbar**
- **Status Bar**
- **Controls**
- **Menus**

3.2. Mnemonic Commands Pane

The mnemonic commands pane (**Mne Commands**) shows the mnemonic commands in the current test plan. Mnemonics may be inserted, renamed, moved, copied and deleted here.

3.3. Property Pages

When selected each mnemonic displays a set of property pages containing information about and controls for the object represented by the mnemonic.

3.4. Toolbar

The toolbar provides quick mouse access to many tools used in PI-DATS,

To hide or display the Toolbar, choose **Toolbar** from the **View** menu.

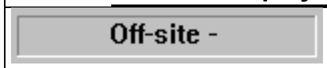
Click:	**To:**
🗋	Open a new document.
📂	Open an existing document (PIP file).
💾	Save the active document or template with its current name.
🖨	Print the active document.
✂	Remove selected data from the document and stores it on the clipboard.
⎘	Copy the selection to the clipboard.
📋	Insert the contents of the clipboard at the insertion point.
👁	Append a new header below the highlighted item. See **Section 3.10.1. Append Group** Header.
Mn	Append a new mnemonic below the highlighted item. See **Section 3.10.3. Append Mnemonic**.
Go	Starts the current Test Pan
▶ Test Plan	Switches between **Test Plan** and **Controller** modes. See **Section 3.11.7. Switch Mode**.

3.5. Status Bar

The status bar is displayed at the bottom of the PI-DATS window. To display or hide the status bar, use the **Status Bar** item in the **View** menu.

The left area of the status bar describes actions of menu items as you use the arrow keys to navigate through menus. This area similarly describes the actions of toolbar buttons when they are clicked, but before they are released.

Display:	**Indicates:**
On-site - All Ready	Indicates that all hardware is responding.
On-site - None Ready	Indicates that none of the hardware is responding. All the hardware will be simulated. Check connections and power to the hardware devices.
On-site - Some Ready	Indicates that some of the hardware is responding. The hardware that is not responding will be simulated. Check connections and power to the hardware devices.
On-site - Simulate	The operator has opted to simulate all hardware.

Display:	Indicates:
 Off-site -	The PIP file loaded does not comply with the hardware attached to your system. All hardware is simulated.

3.6. Controls

The area below the property pages contains controls, status, and identification information for the selected mnemonic.

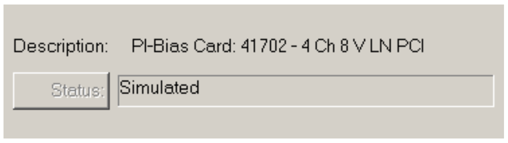

Description: PI-Bias Card: 41702 - 4 Ch 8 V LN PCI
Status: Simulated

Figure 7: PI-DATS Main Window, Controls area

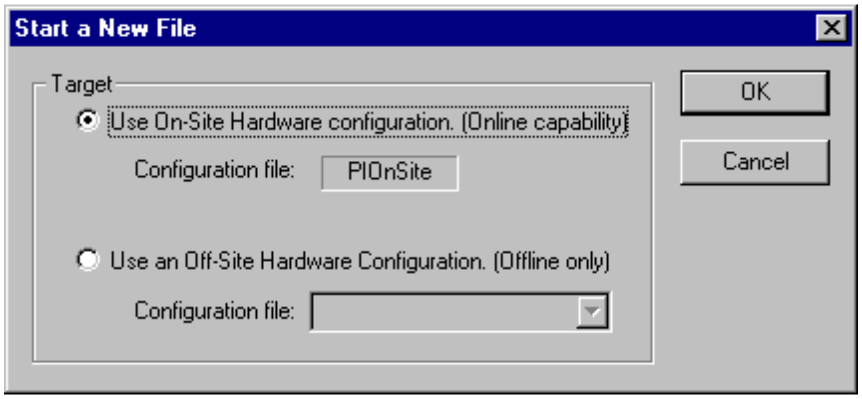
Controls are disabled in Test Plan mode. They are available in Controller mode. In Controller Mode, PI-DATS behaves like PI-Controller, and the controls become active, enabling real-time control over hardware. Please see the PI-Controller manual for more details on real-time control of your instrumentation.

3.7. File menu

The File menu offers the following items:

New	Creates a new document.
Open	Opens an existing document.
Save	Saves an opened document using the same file name.
Save As	Saves an opened document to a specified file name.
Properties	Displays the details about this PIP file.
Print	Prints a document.
Print Preview	Displays the document on the screen, as it would appear printed.
Print Setup	Selects a printer and printer connection.
Exit	Exits PI-DATS.

3.7.1. File:New


Start a New File

Target

(●) Use On-Site Hardware configuration. (Online capability)

Configuration file: PIOnSite

( ) Use an Off-Site Hardware Configuration. (Offline only)

Configuration file:

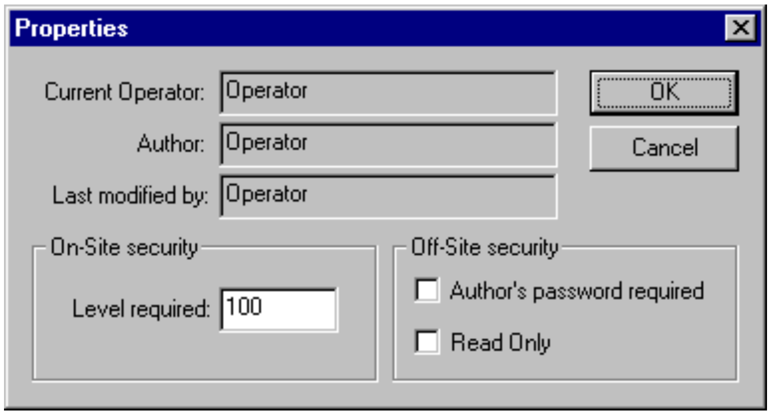
Figure 8: File New dialog box

When creating a New PIP file, select "**Use On-Site Hardware configuration**" to create a file that is compatible with your hardware. The name of the Configuration file is "**PIOnsite.hwr**".

Select "**Use an Off-Site Hardware Configuration**" to create a file compatible with an off-site hardware configuration. The drop-menu will display all available configuration files (.hwr) in the folder **C:\Program Files\PULSE20**.

3.7.2. Properties

Displays the details about this PIP file.


Properties

Current Operator: Operator

Author: Operator

Last modified by: Operator

On-Site security

Level required: 100

Off-Site security

☐ Author's password required

☐ Read Only

Figure 9: File: Properties dialog box

The **Current Operator** is the name that is entered on reports.

3.8. Edit menu

The **Edit** menu offers the following commands:

Cut	Removes the selected data/item from the document and places it in the clipboard.

Copy	Copies the selected data/item from the document and places it in the clipboard.
Paste	Pastes data from the clipboard into the document.
Delete	Removes the selected data/item
Quick Hardware Configure	Displays quick options for identifying hardware attached to your system.
Advanced Hardware Configure	Displays complete options for identifying hardware attached to your system and specifying cabling connections. See **Section 4.1. Advanced Hardware Configuration**.
Managerial . . .	Displays options for managing security via the User and Groups database.

Use the **Copy/Paste** commands to copy selected text to/from the clipboard. In Test Plan Edit mode or Controller mode, Copy can also be used to copy entire mnemonics or groups of mnemonics within a test plan. **Copy** is unavailable if there is no object currently selected, and **Paste** is unavailable if there is no data in the Clipboard.

3.9. View menu

The **View** menu offers the following commands:

Toolbar	Shows or hides the toolbar.
Status Bar	Shows or hides the status bar.

3.10. Build menu

Append Group	Append a new header below the highlighted item.
Insert Group within Group	Insert a new group header as the first item of this Group Header.
Append Mnemonic	Append a new Mnemonic below the highlighted item.
Insert Mnemonic within Group	Insert a new Mnemonic as the first item of this Group Header.

3.10.1. Append Group Header

Append a new header below the highlighted item. Group headers are used to organize your test plans for looping and branching. Each Group Header can contain mnemonics and/or other Group Headers.

3.10.2. Insert Group within Group

Insert a new group header as the first item of this group header. Use this menu item to build nested groups and nested loops.

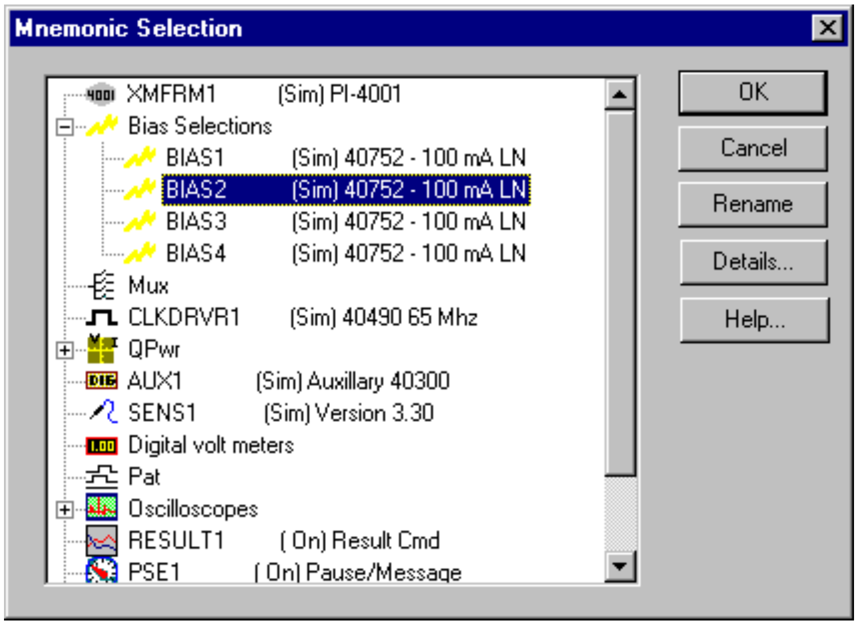
3.10.3. Append Mnemonic

Use this menu item to append a new below the currently selected item.

Selecting **Build: Append Mnemonic** displays a list of available mnemonics. Select the ⊞ symbol to view the multiple mnemonics of type.

3.10.3.1. Adding mnemonics

To add a mnemonic to your test plan, select it and click **OK**. Multiple mnemonics of the same type can be selected by using the Control key in conjunction with the ↓ and ↑ arrow keys.



Mnemonic Selection

XMFRM1	(Sim) PI-4001
Bias Selections	
BIAS1	(Sim) 40752 - 100 mA LN
BIAS2	(Sim) 40752 - 100 mA LN
BIAS3	(Sim) 40752 - 100 mA LN
BIAS4	(Sim) 40752 - 100 mA LN
Mux	
CLKDRVR1	(Sim) 40490 65 Mhz
QPwr	
AUX1	(Sim) Auxillary 40300
SENS1	(Sim) Version 3.30
Digital volt meters	
Pat	
Oscilloscopes	
RESULT1	(On) Result Cmd
PSE1	(On) Pause/Message

Figure 10: Mnemonic Selection dialog box

If a desired mnemonic does not appear at all, then the software module for that Instrument or function is not installed on your system. If a mnemonic does appear, but the OK button is grayed (not selectable), then the Instrument needs to be added to the hardware configuration. (See **Section 4. Hardware Configuration**).

3.10.3.2. Moving mnemonics

Mnemonics can be re-ordered in a test plan by dragging and dropping them into their new locations. Dragging a mnemonic onto a Group icon will place the mnemonic above or below the Group, but not in the Group. To move a mnemonic into a Group, drag it on top of a mnemonic that is already in the group. If the Group is currently empty, any mnemonic may be inserted into that group temporarily to facilitate the drag and drop function.

Mnemonics can also be cut, copied, and pasted from location to location. When a mnemonic is pasted it is placed below the currently selected mnemonic. Mnemonics retain all their settings when cut/copied and pasted, with the exception of Result mnemonics.

A RESULT mnemonic will "lose track" of its data source if that data source is moved into or out of a Group. For example, if a RESULT mnemonic has an array inserted from a DACQ Measure mnemonic that is subsequently moved into a Group, the RESULT mnemonic will show the table as "LOST." The array must be re-inserted into the table, but all other settings (data reduction, export, accessible, etc) are retained.

3.10.3.3. Simulated (Sim) vs. Active (On) Hardware

These symbols represent the current hardware state. If (Sim) or Simulation appears next to a mnemonic, this indicates that no hardware commands will be sent or received during a test run. If (On) appears next to the mnemonics, this indicates that the Instrument is active, and that hardware commands will be sent and/or received during a Test Run.

3.10.3.4. Rename

By default, each mnemonic is assigned a name that describes its model number and function, e.g. **41701BIAS**. The Rename button allows you to give your instrument channels names that are meaningful in your application, such as $\mathbf{V_{CC}}$, $\mathbf{V_{DD}}$ or **VertClk1**.

The new name will be used each time an instance of this mnemonic is inserted into your test plan.

You can also rename any instance of a mnemonic in your test plan by single-clicking on its name in the Main Window.

Mnemonic names are specific to each test plan, and new test plans are created with the default names.

3.10.3.5. Details

Press this button to display more details about the selected type of Instrument or Function, such as its model number and physical location. These details can also be viewed later, in each mnemonic's **Description** property page.

3.10.4. Insert Mnemonic within Group

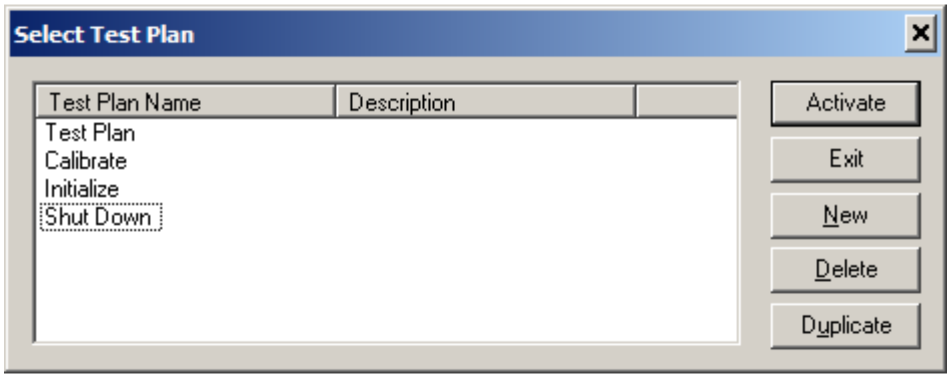
Insert a new mnemonic as the first item of the selected Group Header.

3.11. Tests menu

Select Test Plan	Select an existing test plan or create a new test plan within the same PIP file.
Run Test	Execute the currently selected test plan.
Previous Results	Display again the list of choices following a test run.
Results Options	Display choices to automate your test runs.
Results Log	Display previously run results.
Switch to . . .	Switch from PI-DATS to PI-Controller and visa-versa.

3.11.1. Select Test Plan

Select an existing or create a new Test Plan to work with. Existing test plans can also be selected by right-clicking the top level Group Header in the test plan window. The following is displayed after selecting **Select Test Plan** from the **Tests** menu:



Test Plan Name	Description
Test Plan	
Calibrate	
Initialize	
Shut Down	

Figure 11: Select Test Plan dialog box

All defined Test Plans within the current PIP file are displayed. Switch to a Test Plan by selecting it and clicking **Activate**.

Multiple test plans may be contained within a single PIP file to group and separate test functions. For example a test routine may require a calibration routine and an initialization routine before test data are collected, and many types of parts require specific shutdown procedures before they are disconnected.

To create a new Test Plan click **New**. Modify the default "Test Plan" name by clicking it. The name will highlight to allow typing a new name. A new, blank test plan will be displayed, and the old test plan will be hidden, but all of its mnemonics and parameters are preserved and will be saved with the current PIP file.

3.11.2. Run Test

This will initiate a Test Run. All Mnemonics within the Test Plan will be executed. Any defined looping will occur and the **Completed Results Options** display will appear upon completion.

3.11.3. Fast Mode

This executes the test run as fast a possible. Window painting and progress messages will be reduced to make execution faster. You should always run your test plan in normal mode first, to ensure that all functions are executing successfully, before switching to Fast Mode.

3.11.4. Previous Results/Completed Results Options

The following is displayed upon completion of a Test Run. The Result of the Test Run is displayed at the top. Any failure message or warning will be displayed in the area below. This same dialog can also be displayed by selecting **Tests: Previous Results**.

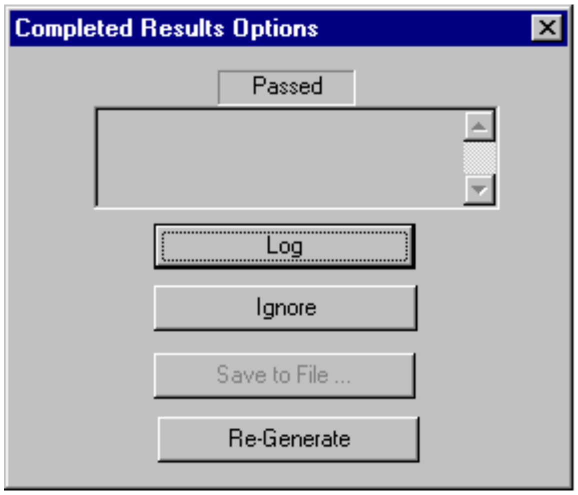


Figure 12: Completed Results/Previous Results dialog box.

3.11.4.1. Log

Select to preserve the test run in the Results Log.

3.11.4.2. Ignore

Dismiss dialog box without logging results.

3.11.4.3. Save to File

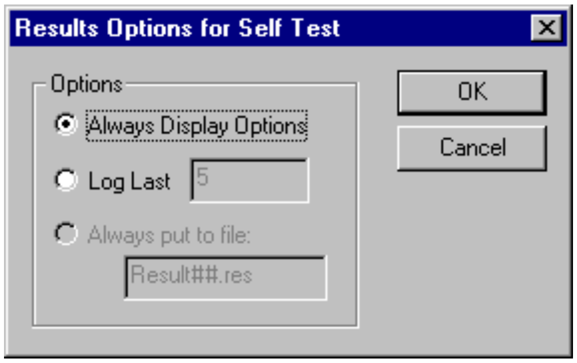
Not implemented.

3.11.4.4. Re-Generate

Re-generates the results of the test run without executing any hardware commands.

3.11.5. Results Options

This allows automation of the Test Run Results. There are separate Result Options stored for each Test Plan within a single PIP file. **Warning:** Logged results are stored within the PIP File. Therefore if PI-DATS is exited without a save, **all logged results will be lost**.



[✓ Always Display Options]
Log Last 5
Always put to file: Result##.res

Figure 13: Results Options

3.11.5.1. Always Display Options

Select this to always display the **Completed Results Options** at the end of each Test Run. This is the default selection.

3.11.5.2. Log Last *n*

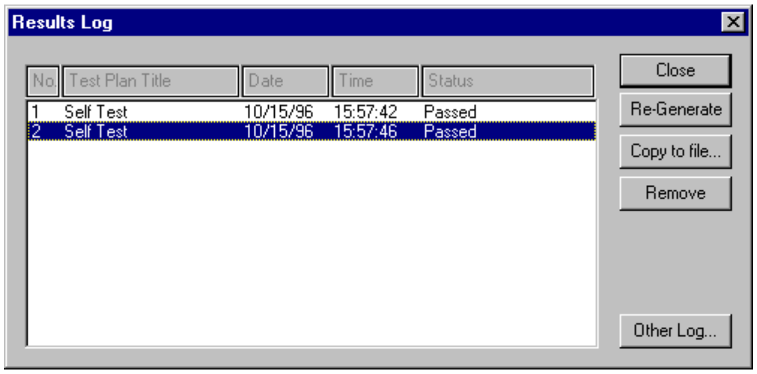
Causes the ***n*** most recent results to be logged. Older logged results will be deleted.

3.11.5.3. Always put to file:

Logged results are stored within the PIP file by default. This selection allows results to be written also to a separate file. Use the **#** symbol to enumerate your output files. The **#** will be replaced with sequential numbers.

3.11.6. Results Log

The Results Log display shows all logged results stored within the PIP File.



No	Test Plan Title	Date	Time	Status
1	Self Test	10/15/96	15:57:42	Passed
2	Self Test	10/15/96	15:57:46	Passed

Figure 14: Results Log dialog box

Re-Generate

Re-generates the results of the test run without executing any hardware commands.

3.11.6.1. Copy to file

Copies the selected logged result(s) to a separate file. Clicking this button will present the standard Windows **Save File** dialog box.

3.11.6.2. Remove

Deletes the selected logged result(s). The removal is permanent once the PIP File has been saved.

3.11.6.3. Other Log

Imports a log file previously saved by **Copy to file** or by **Always Put To File**. The display is replaced with the imported logged results.

3.11.7. Switch Mode

This button switches between **Test Plan** and **Controller** modes.

Test Plan mode (default) allows the user to build and edit test plans. Mnemonics will contain property pages for **General** controls, **Description**, **Looping**, **Pre-execute** and **Post-execute** parameters.

Controller mode allows the user to issue real-time hardware control commands via the General property page. The **Looping**, **Pre-execute** and **Post-execute** property pages are hidden in Controller mode.

The General property pages for each mnemonic may appear slightly differently in Test Plan and Controller modes because certain types of controls are not meaningful in the current context.

For example, the **General** property page for a hardware control will typically contain a **Write** button in **Controller** mode, but not in **Test Plan** mode. In **Controller** mode, the **Write** button enables the user to decide when to send the programmed parameters to the hardware. In **Test Plan** mode, the parameters are sent whenever the mnemonic is executed. See the section of this manual for each instrument mnemonic to obtain additional details.

4. Hardware Configuration

This section concerns identification and configuration of hardware connected to your system.

PI-DATS uses a special file (*.hwr) to store hardware configuration information. Your Pulse Instruments system was shipped with a configuration file that reflects the hardware that was installed when it left the factory.

The default HWR file (**PIOnSite.hwr**) is updated every time PI-DATS is asked by the user to scan for new instrumentation. All configured hardware will show up in the Add Mnemonic window. If your instrumentation does not show up in the Add Mnemonic window, then the hardware file must be updated. If third-party hardware is added to the system, changes to the hardware configuration may by implemented via the **Edit: Advanced Hardware Configuration** menu item.

Other HWR files may be created (e.g. the included **Demo.hwr**) to reflect available hardware on a different test station, or to edit test plans “offline” when there is no hardware available. HWR files should not be edited directly by the user, and should be stored in the **C:\Program Files\PULSE20** directory.

Note that it is not always necessary or desired to update the hardware file to match the actual configuration, as this allows the editing of test settings “offline”, even if all the desired hardware is not available or not currently installed.

4.1. Advanced Hardware Configuration

This module allows the operator to define the hardware that is installed on the system. The **Advanced Hardware Configure** dialog box is available from the **Edit** menu.

4.1.1. GPIB Address Assignments

GPIB-based instrumentation (including Pulse Instruments products) may be added via the **GPIB** property page.

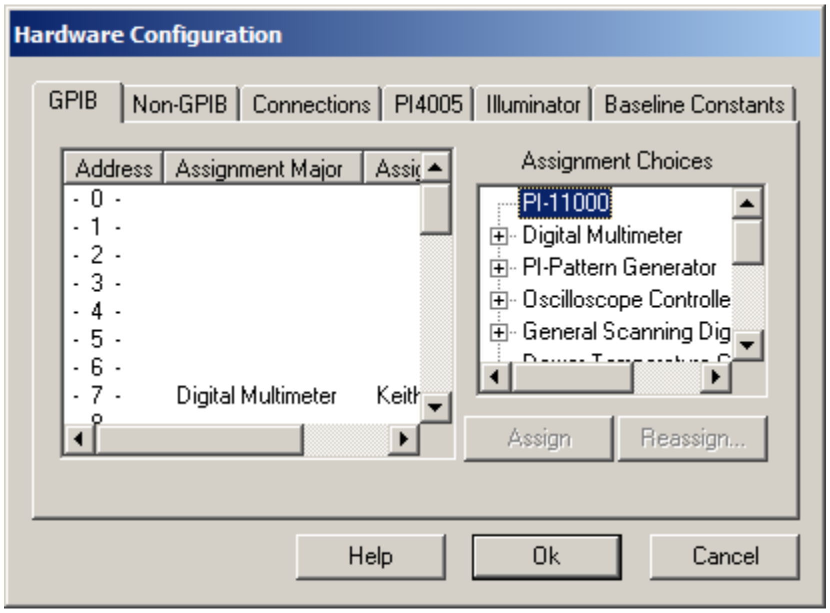


Figure 15: Advanced Hardware Configuration, GPIB property page

To assign instrumentation to a GPIB address, select the desired address in the left pane, select the attached instrument in the right pane, and then click **Assign**.

To re-assign an instrument to another address, select the GPIB address in the left pane and click **Reassign**. You will be prompted to select the new address.

To remove an instrument, select the GPIB address in the left page and click **Unassign**.

Please note that the PI-11000 Instrument Mainframe should only be added to the GPIB assignment table if PI-DATS is running on a remote PC. If PI-DATS is running on the local CompactPCI CPU board, then the PI-11000 Instrument Mainframe must be configured on the **Non-GPIB** property page, described **below**.

4.1.2. Recommended assignments:

The following table shows the default GPIB address assignments for Pulse Instrument test equipment. These assignments are optional, and any available GPIB address may be assigned to an instrument, but adhering to the following default assignments will minimize the amount of reconfiguration required when additional components are added.

Instrument	GPIB Address
DVM (second)	6
DVM (first)	7
PI-2005 or PI-11000	14
Oscilloscope	17

4.1.3. Non-GPIB Assignments

The Non-GPIB property page is used to declare certain types of non-GPIB hardware, including CompactPCI instrument cards from Pulse Instruments.

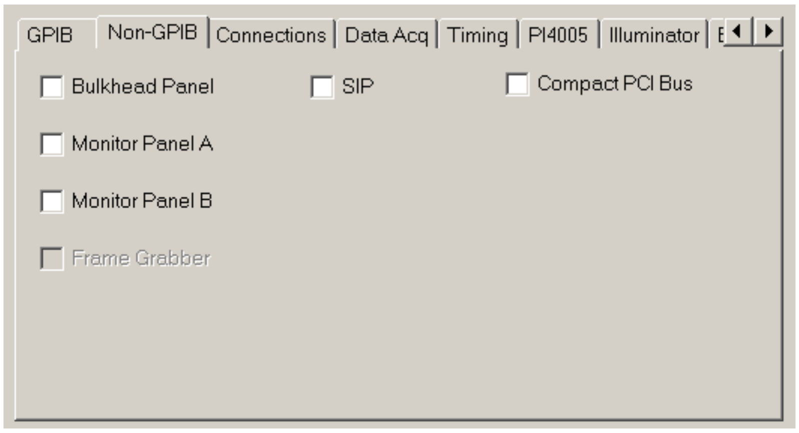


Figure 16: Advanced Hardware Configuration, non-GPIB property page

Use the checkboxes to enable the following types of hardware:

4.1.3.1. CompactPCI Bus

Use this checkbox to declare the installation of Pulse Instruments Clock Driver, DC Bias Cards, and Data Acquisition cards in CompactPCI. This box should be checked only for PI-DATS running on a CPU board in the CompactPCI chassis.

If PI-DATS is running on a remote PC communicating with clock driver and DC bias cards in an instrument mainframe over GPIB, then the mainframe should be declared on the GPIB property page. Clock driver and DC bias cards in that chassis will be detected automatically.

4.1.3.2. Bulkhead Panel

Use this checkbox to declare the installation of the standard Pulse Instruments Bulkhead Panel. (96 Connectors A-H, 1-12). This entry is primarily to facilitate accurate descriptions of connections in the Connections property pages. There is no software control of the bulkhead panel itself.

4.1.3.3. Monitor Panel

Use this checkbox to declare the installation of the standard Pulse Instruments Monitor Panel. (32 Connectors J 1-32). This entry is primarily to facilitate accurate descriptions of connections in the Connections property pages. There is no software control of the bulkhead panel itself.

4.1.3.4. Serial Interface Protocol (SIP)

Use this checkbox to declare a Serial Interface Protocol port, typically the I^2C port on the PI-21000 Clock Card.

4.1.4. Connections Property Page

4.1.4.1. Detecting Cards Automatically

Once instruments and mainframes have been declared on the GPIB or CompactPCI bus, instrument cards can be detected automatically using the **Read** button on the Connections property page:

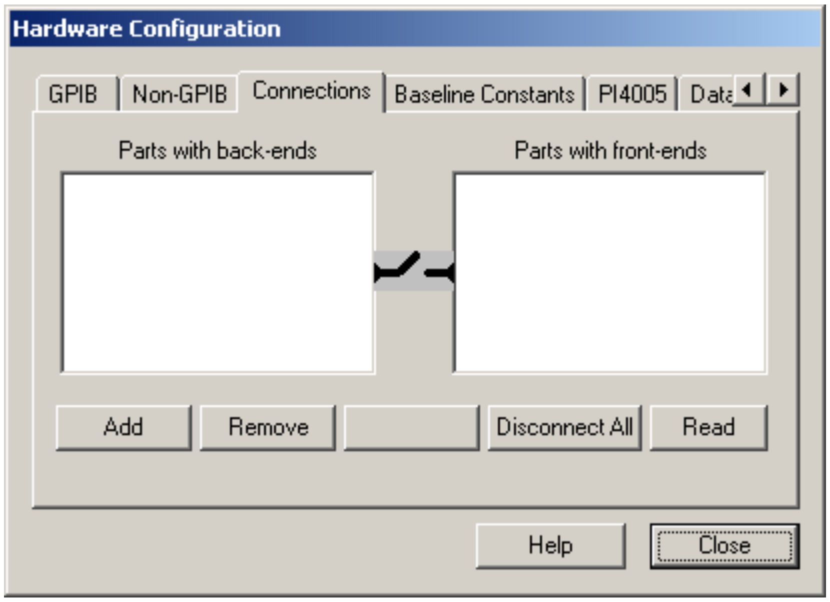


Figure 17: Connections Tab

Clicking the **Read** button will poll the PCI bus and/or the GPIB for any installed Pulse Instruments cards. Note that clicking the **Read** button will also remove any previously defined connections, including connections required for operation of the PI-3105 Data Acquisition System and its components (including the PI-41000 Digital Acquisition Card).

If clicking the **Read** button on your Remote PC does not properly detect all supported devices that are present in an instrument mainframe, please ensure that PI-PAT is running on the local instrument. PI-PAT must be running in order to process incoming GPIB commands.

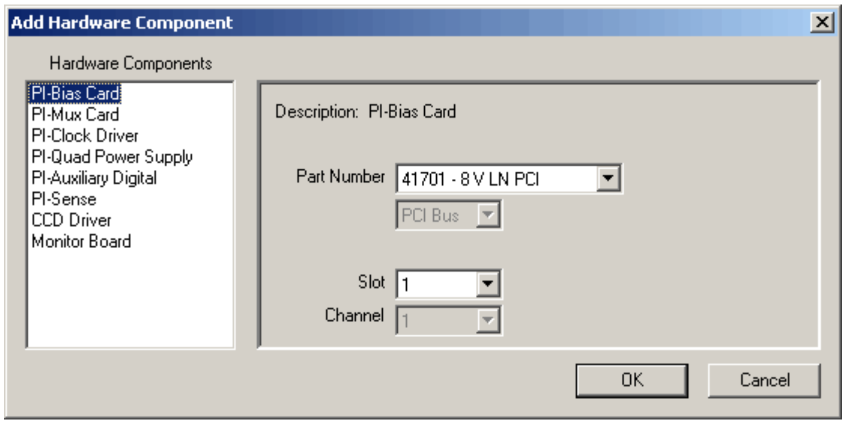
If your cards are still not detected, verify your hardware connections or else remove and re-seat the cards in the mainframe. Because Pulse Instruments CompactPCI instruments cards are not hot-swap compliant, it is necessary to power-down the unit before removing or inserting a card.

4.1.4.2. Manual Device Configuration

The **Connections** property page can also be used to manually define instruments that are not yet attached to the system. This feature is used for "offline" editing of test plans, or for editing of test plans in a system that have components temporarily removed or unavailable.

PI-DATS will scan all indicated busses and recognize all supported hardware when the **Read** button is clicked. In some cases, it may be desirable to add a device to the hardware configuration that is not currently available. For example, an instrument may be powered down, or the test plan might be being edited at a location where the instrument is not available. In these cases, hardware devices and their physical locations may be specified manually.

To add a device manually, click the **Add** button to display the **Add Hardware Component** dialog box:



Description: PI-Bias Card

Part Number 41701 - 8 V LN PCI

PCI Bus

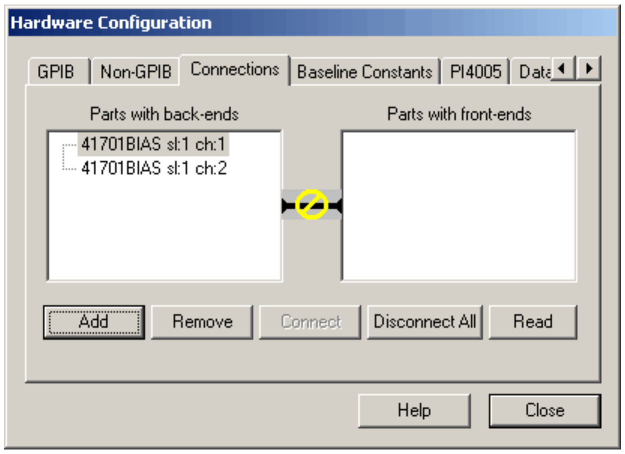
Slot 1

Channel 1

Figure 18: Add Hardware Component Dialog Box

Click on the type of device to add on the left, and then select the specific part number, bus type, and slot number before clicking **OK**. Click **Add** to repeat the process until you have added all your devices, then click **OK**.

Once you have added devices, either manually or automatically, they will appear in the **Connections** dialog box as well as become available in the mnemonic selection window.

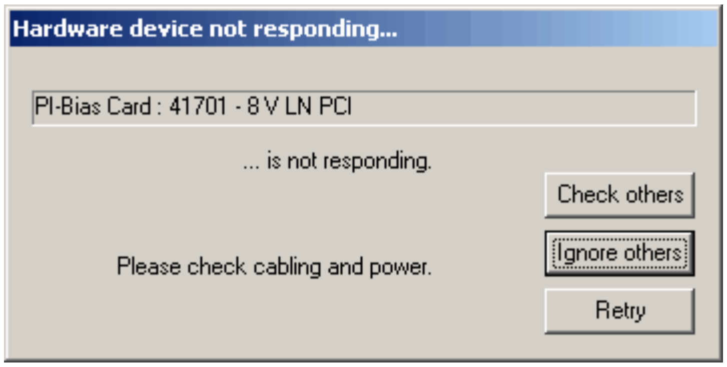


41701BIAS sl:1 ch:1
41701BIAS sl:1 ch:2

Figure 19: Connections Tab with Devices Added Manually

4.1.4.3. Off-Line/Simulated Devices

When you have finished specifying or adding devices, click **Close**. If any of the specified devices do not exist in the system, you will receive a **Hardware device not responding** warning:


Hardware device not responding...

PI-Bias Card : 41701 - 8 V LN PCI

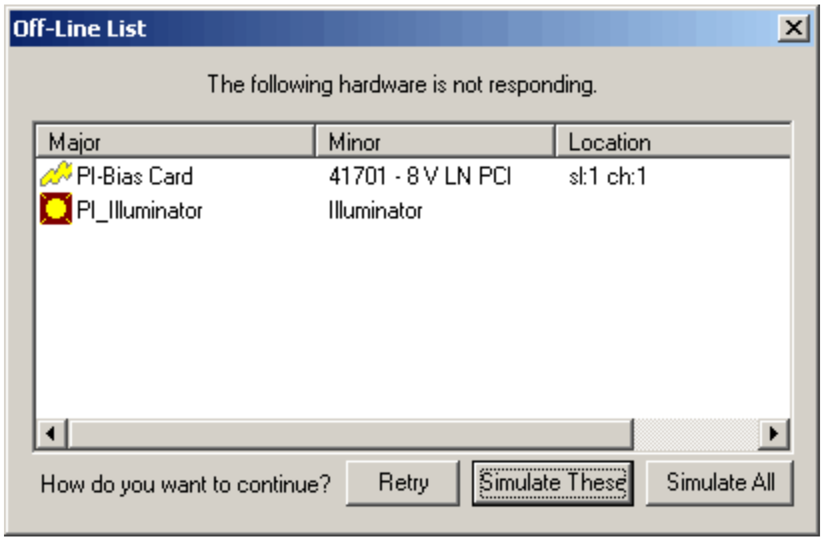
... is not responding.

Please check cabling and power.

Figure 20: Hardware Not Responding Warning

This is expected behavior if any of the devices you have specified are not currently installed, and you wish to simulate them. Click **Check others** or **Ignore others** if you wish to continue, or click **Retry** to try to connect to the indicated device again.

Once you have completed checking your hardware, any disconnected devices will appear in the **Off-Line list**:


Off-Line List

The following hardware is not responding.

Major	Minor	Location
PI-Bias Card	41701 - 8 V LN PCI	sl:1 ch:1
PI_Illuminator	Illuminator	

How do you want to continue?

Figure 21: Off-Line Devices List

You can then click **Simulate These** or **Simulate All** to add them to your hardware profile as simulated devices. Connected and simulated devices will then appear in the **Mnemonic Selection** dialog box.

4.1.4.4. Connections

The Connections property page is also used to define physical interconnections between instruments in the system, which is required for several functions of the PI-3105 Data Acquisition subsystem and for the Sense feature of the clock driver and DC bias cards.

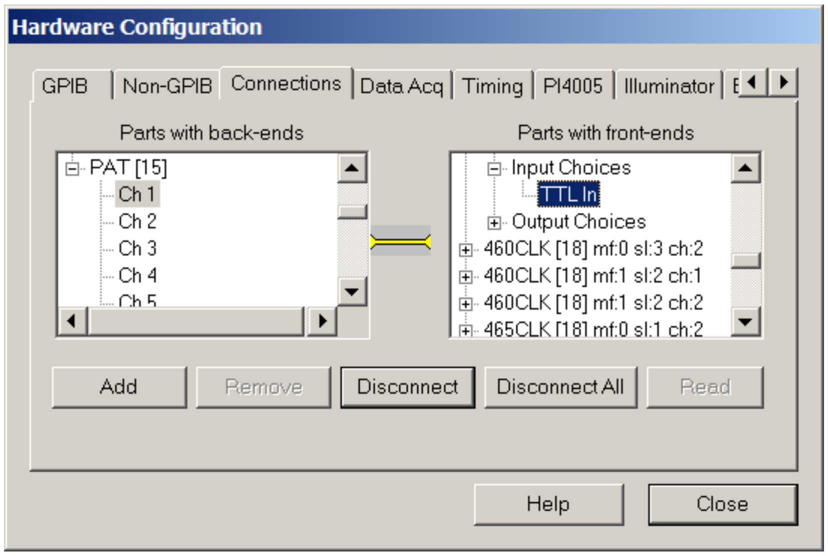


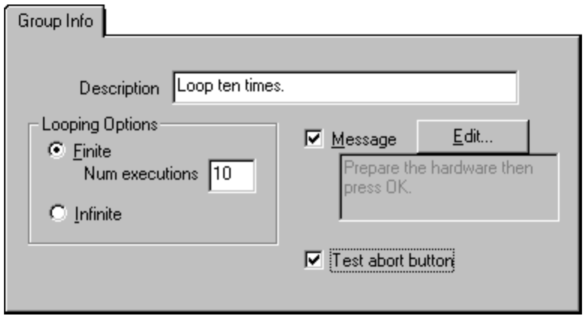
Figure 22: Connections Property Page

To define a connection select the output device from the **Parts with back-ends** pane, then click on the input device in the **Parts with front-ends** pane. Click **Connect** to establish the connection. To remove a connection, click on either end (e.g. either the connected **Front-end** or the connected **Back-end**) and click **Disconnect**. To remove all defined connections, click the **Disconnect All** button.

For specific information on connecting data acquisition components, please refer to the **PI-3105 Operators Manual**.

5. Group Headers and Mnemonics

As described in the System Overview, Group Headers are used to organize mnemonics for looping and branching. A Group Header may contain one or more mnemonics, and it may also contain other Group Headers. Mnemonics within a Group Header are said to “belong” to that Group Header, and they are displayed in an indented, hierarchical fashion as shown below. The containing Group Header is referred to as the “parent” for that mnemonic.



Description: Loop ten times.

Looping Options: [✓ Finite] Infinite

Num executions: 10

[✓] Message: Prepare the hardware then press OK.

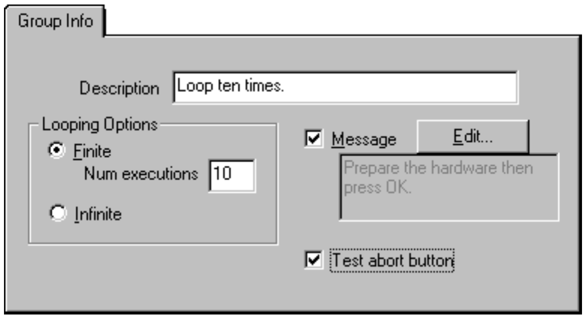
[✓] Test abort button

Figure 23: Group Header mnemonic and Group Info

All mnemonics and Groups are contained within the top-level Group Header, the Test Plan. As described below, the Test Plan header has certain parameters that are unique to it.

5.1. Group Info (Regular Group)

For any regular Group Header, the Group Info property page allows you to enter a description for the group, the looping parameters, and an optional message to be displayed before the group is executed.



Description: Loop ten times.

Looping Options: [✓ Finite] Infinite

Num executions: 10

[✓] Message: Prepare the hardware then press OK.

[✓] Test abort button

Figure 24: Build: Group Info item

5.1.1. Description

The **Description** box is for informational purposes only. You can type any desired text in this box.

5.1.2. Looping Options

When **Finite** looping is selected, the user can specify how many times to loop through the mnemonics in the selected Group. Either type an integer greater than 0 or choose from one of the available choices in the drop menu. If a 1 is entered as the finite looping parameter, then the mnemonics in the Group will execute exactly once.

The drop menu will contain special variables that are equal to the number of channels of certain instrument types in the system. For example, the selection NUM_750BIAS is equal to the number of PI-40750 Low Noise DC Bias channels currently installed in your system.

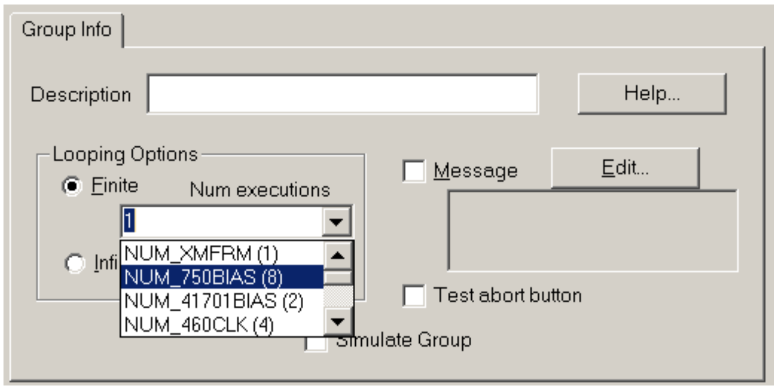


Figure 25: Finite Looping Options, drop menu selections

After a Group has completed its looping, PI-DATS will execute the next mnemonic or Group Header immediately below it.

5.1.3. Infinite Looping Option

Programs this Group Header to loop an infinite number of times, unless the loop is exited by a **Halt** mnemonic within the loop, an **Exit Loop** option in a Post-Execute property page, or by an **Abort Test** button being clicked between loops. See **Section 8.1.3. Halt Mnemonic**.

Please note that if a loop is halted with the **Abort Test** button, there will be no way to save the test results.

5.1.4. Message

You can have a Group Header display a message box on the screen before it executes each iteration of its loop, including the first loop. To enable the message, click the checkbox, and then click the Edit box to enter the desired text. The entry may consist of format specifiers according to the “printf” routine of the “C” programming language. For example, if there were a variable defined:

nBin (Integer) = 1

The entry “Please prepare bin %d then click OK., nBin” would expand to

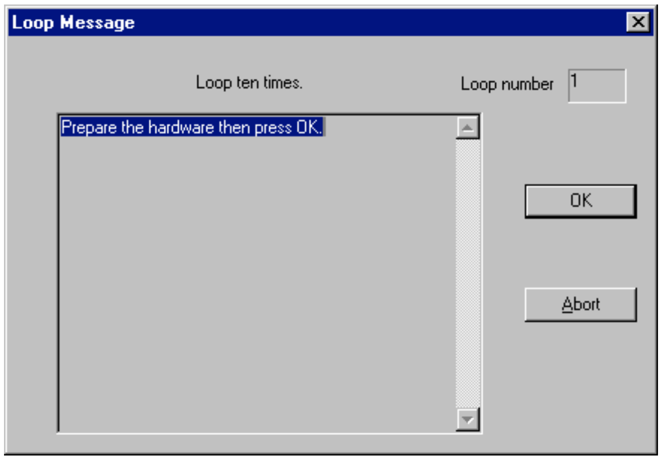
Please prepare bin 1 then click OK.

when the test is run.

5.1.5. Test abort button

If this checkbox is checked, then the message box will also have an **Abort Test** button when it appears, enabling the operator to halt the test.

This checkbox is enabled only if the Message box has been enabled.

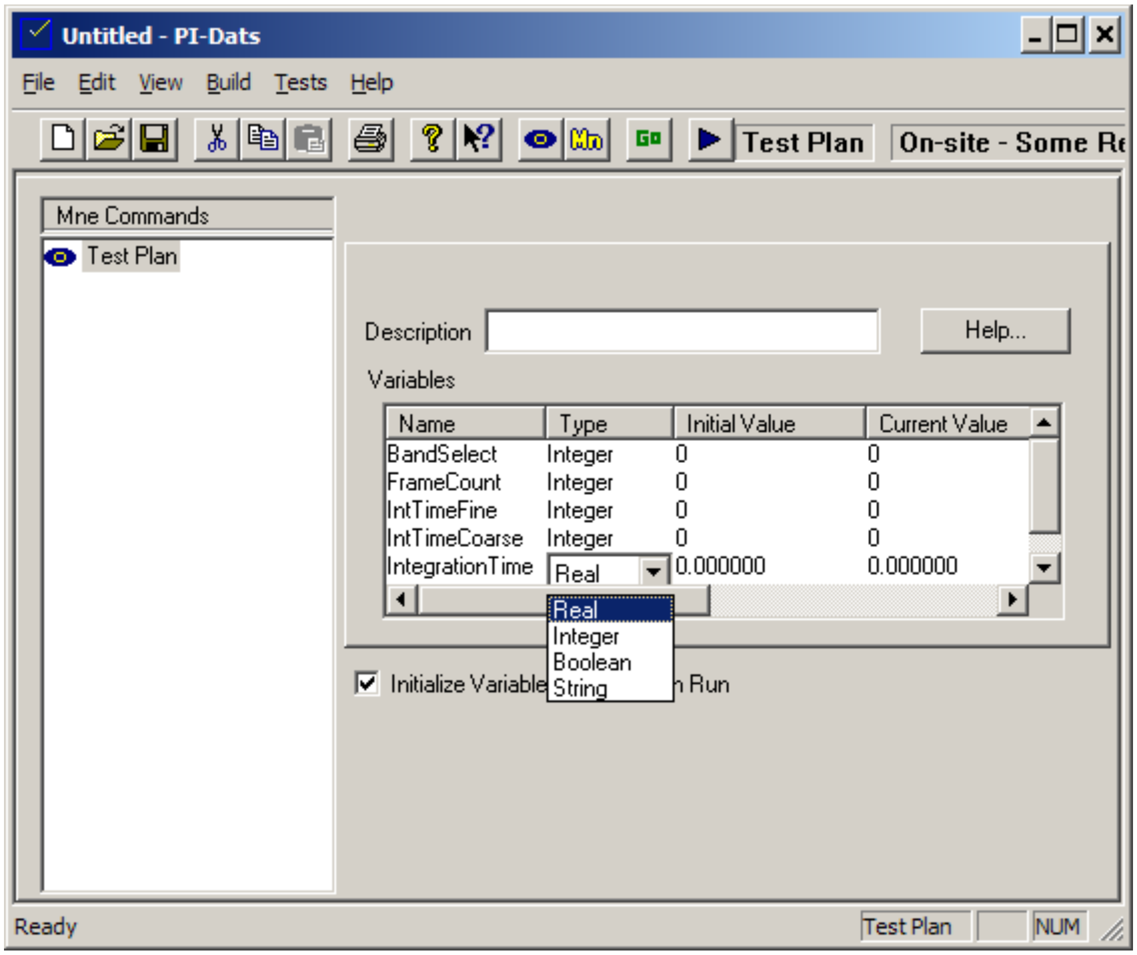


Prepare the hardware then press OK.

Figure 26: Loop Message with Abort enabled

5.2. *Group Info (Test Plan Header)*

5.2.1. Main Test Plan Group Info



Description

Variables

Name	Type	Initial Value	Current Value
BandSelect	Integer	0	0
FrameCount	Integer	0	0
IntTimeFine	Integer	0	0
IntTimeCoarse	Integer	0	0
IntegrationTime	Real	0.000000	0.000000

Real
Integer
Boolean
String

☑ Initialize Variable… …n Run

While Group Headers within the test plan contain looping parameters, the top-level Test Plan Group Header contains variable definitions for the entire test plan. Variables defined here are available to any mnemonic or Group Header within the test plan.

5.2.2. Variable Definitions

Add, delete, or modify variables in this list box. To add a variable, click below the last variable name and type a new variable name in the edit box. By convention, variables in this document will use C-style variable prefixes:

- n for an integer variable, e.g. nLoopCount
- b for a Boolean variable, bEnableScan
- s for a string, e.g. sOperatorName
- f for a floating-point number, e.g. fMaxVoltage

This convention is optional, and any non-reserved text can be used as a variable name.

Next, select a **Type** that defines the nature of the data to be contained in the variable:

- Integer – whole numbers from –2,147,483,648 to 2,147,483,647
- String – Any text string of any length.
- Real – floating decimal numbers like 3.14159
- Boolean – TRUE (1) or FALSE (0)

The **Initial Value** is the starting value assigned to the variable when initialized by a test plan.

The **Current Value** is the current value of the variable, based on the previous test run. This value will change depending on how test plan ran, whether it was aborted, what test results were returned, etc.

Modify a variable Name, Type, or Initial Value by clicking on the appropriate field until an edit cursor is obtained.

Delete a variable by selecting the variable then pressing the **Del** key.

5.2.3. Reserved Variables

The following reserved variables may also be used:

- CurDateTime
- CurDate
- CurTime
- TestRunDateTime
- TestRunDate
- TestRunTime
- TestRunTitle
- PIPFilename
- OperatorName

5.2.4. Global Variables

Because there are no true functions in PI-DATS test planning, all variables are global. There is only one instance of any named variable, and any mnemonic or control can change its value. To preserve the value of a variable at a given point in the test plan, insert a Variable Snapshot mnemonic into the test plan. This will capture a "snapshot" of values of all defined variables when executed. If executed in a loop, the Variable Snapshot will build a table of values for each variable.

5.2.5. Variable Initialization

Each defined Test Plan in a PIP file has a checkbox to **Initialize Variables**. If this box is checked, each variable will be set to its **Initial Value** when this test plan is executed. If this box is left unchecked each variable will retain its **Current Value** until modified. This feature allows variable values to be determined in one test plan (e.g. as the result of an optimization loop) and then used in another.

5.3. Mnemonics (Common Property Pages)

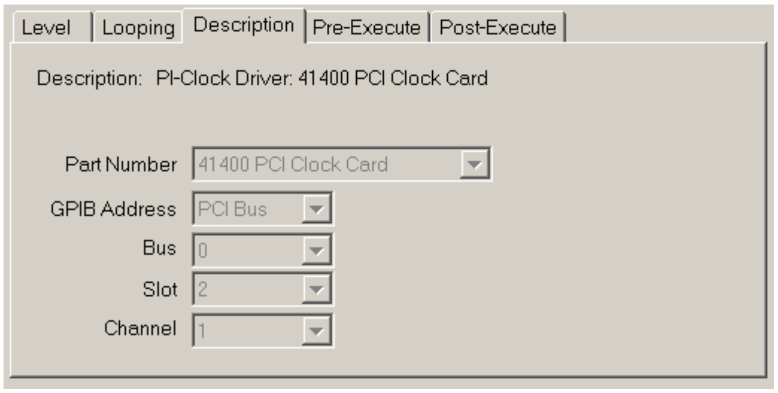
This section described property pages that are common to several different instrument mnemonics. Although individual values and labels may vary by instrument type, the general principals in this section are applicable to all relevant mnemonics.

All mnemonics, whether for hardware controls or for non-hardware objects, have at least three property pages:

- **Description**—basic information about the model number and physical location, or a description of the mnemonic's function.
- **Pre-execute**—time-out conditions and conditions upon which to execute or not execute the mnemonic
- **Post-Execute**—variables to update and/or actions to take based on the result of the mnemonic's execution

Hardware control mnemonics will also have **General/Level** property pages and **Looping** property pages, while **Result** (Data) mnemonics will have data manipulation and export property pages (See **Section 9. RESULT Mnemonic**). Operational control mnemonics also have several unique property pages. Property pages specific to each mnemonic class or instrument type will be described in the section of the manual for that instrument type.

5.3.1. Description Property Page



Description: PI-Clock Driver: 41400 PCI Clock Card

Part Number: 41400 PCI Clock Card
GPIB Address: PCI Bus
Bus: 0
Slot: 2
Channel: 1

Figure 27: Description property page for PI-41400 Clock Driver Card Channel

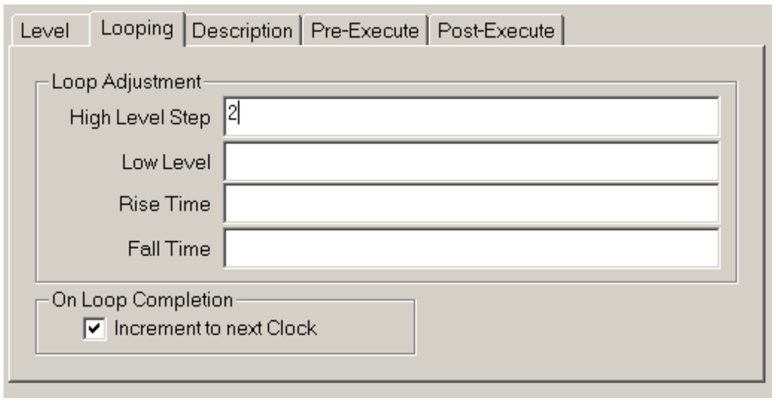
Figure 27 shows a typical **Description** property page for a hardware control mnemonic, in this case a PI-41400 Clock Driver Card channel. This **Description** shows the model number, the GPIB or CompactPCI address, bus number, slot number, and channel number.

The physical location of a hardware control cannot be edited. It is specified by the physical location or address of the instrument at the time it was added to the Test Plan via the **Add Mnemonic** window or to the location or address that was substituted for it if the test plan was converted to On-Site.

5.3.2. Looping property page

If the mnemonic representing a hardware channel is inserted within a Group Header that has loops, several of the programmable parameters can be programmed automatically on each loop iteration.

For example, the following diagram shows the Looping property page for a mnemonic representing a typical Clock Driver channel. Assume that this mnemonic is inserted within a Group Header with looping.



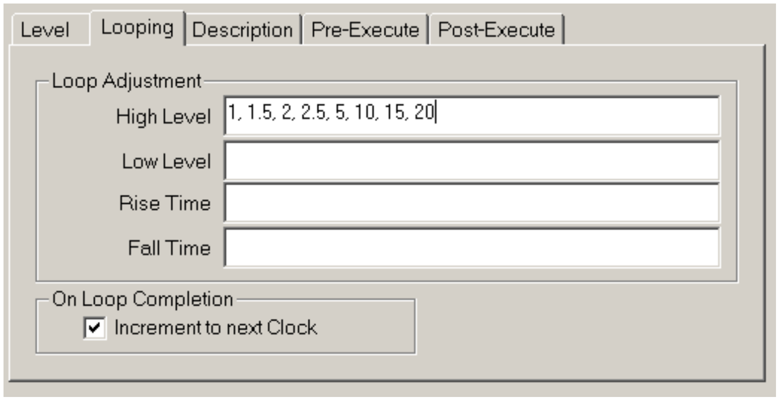
High Level Step: 2
Low Level:
Rise Time:
Fall Time:

☑ Increment to next Clock

Figure 28: Typical Clock Driver mnemonic, linear looping sequence

To increment the High Level value on each loop, enter a single value in the High Level Step entry of the Loop Adjustment section. The High Level will increment from its original value (on the Level page) on each loop iteration after the first. Enter a negative value to decrement a setting.

To loop through values that do not have evenly spaced values, enter multiple values (separated by a space, comma, or semi-colon) to define a custom loop sequence. The label on the Looping Property page will change from **High Level Step** to **High Level**.



High Level: 1, 1.5, 2, 2.5, 5, 10, 15, 20
Low Level:
Rise Time:
Fall Time:

☑ Increment to next Clock

Figure 29: Typical Clock Driver mnemonic, custom looping sequence

Note: When a custom sequence is entered, the value on the Level or General property page for that parameter is ignored, and the first value on the Looping page is used for the first loop iteration.

Use numbers (without units) for voltage values. For rise/fall times, use ns, ms, or us, but do not separate the units from the value with a space, e.g.:

- 20ns, 38ns, 44ns ←correct
- 20 ns, 38 ns, 44 ns←incorrect

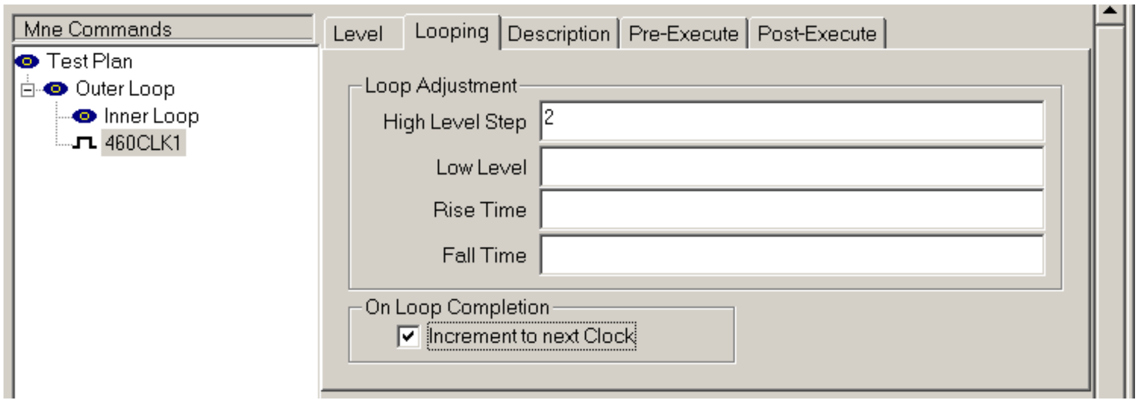
If the loop has more iterations than there are values in the custom sequence, the values will “wrap around” and start over at the beginning.

If value on a loop iteration causes the programmed value to exceed either a programmable Voltage limit or the programmable Amplitude limit, the mnemonic will fail execution on that loop.

5.3.3. Looping Through Channels

A single DC Bias or Clock Driver mnemonic may also be used to program different channels of the same type (e.g. model number) by placing the mnemonic in a nested loop.

For example, the following test plan has a PI-40460 Clock Driver mnemonic in a nested loop: When the **Increment to next Clock** box is checked, mnemonic will apply to the next channel of the same type when the next outer loop is executed.



High Level Step: 2
Low Level:
Rise Time:
Fall Time:

On Loop Completion
☑ Increment to next Clock

Figure 30: Nested looping

In this example, the mnemonic refers to a PI-40460 Clock Driver channel physically located in Mainframe 1, Slot 1, Channel 1. The **Inner Loop** is programmed for 10 iterations and the **Outer Loop** is programmed for 8 iterations. The **High Level** control on the **Level** property page is set for 2 V. There are three additional PI-40460 Clock Driver cards installed and recognized in Slots 2, 3 and 8 of Mainframe 1.

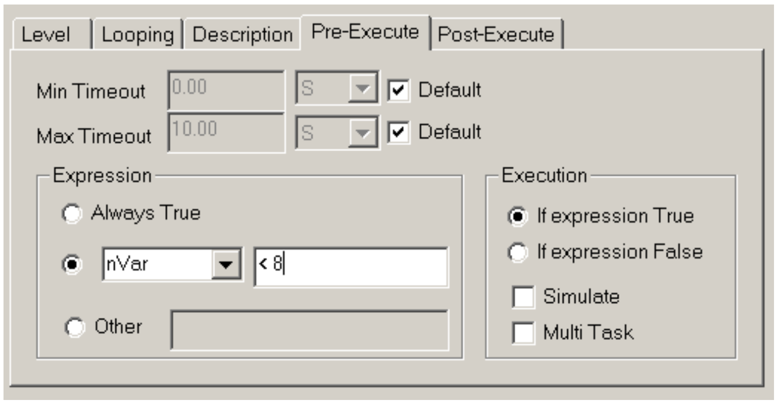
When this Test Plan is run, the first run of the **Outer Loop** will execute the **Inner Loop** 10 times, incrementing the **High Level** on this channel from 2 V to 20 V in 2 V increments. Because the **Increment to next Clock** box is checked, the second run of the **Outer Loop** will cause the Inner loop to operate on the next PI-40460 clock driver channel in the system.

For channel incrementing, channels are numbered in increasing order by channel number, slot number, and then mainframe number.

In the example above, the second **Outer Loop** would operate on Mainframe 1 Slot 1 Channel 2. The third **Outer Loop** would increment to the next available PI-40460 Clock Driver channel, in this case located in Mainframe 1 Slot 2 Channel 1. The fifth iteration would start with Mainframe 1 Slot 3 Channel 1, and the seventh iteration would start with Mainframe 1 Slot 8 Channel 1. Note that PI-DATS will increment to the next logical channel of the same model number, even if it is not physically adjacent.

5.3.4. Pre-Execute Property Page

The **Pre-Execute** property page allows the user to specify under what conditions to execute the mnemonic and to specify the time-out parameters.



Min Timeout 0.00 S [✓ Default]
Max Timeout 10.00 S [✓ Default]

Expression
- ○ Always True
- ● nVar < 8
- ○ Other

Execution
- ● If expression True
- ○ If expression False
- ☐ Simulate
- ☐ Multi Task

Figure 31: Pre-Execute property page

Pre-Execute controls can be used to activate differing sets of hardware on loop iterations and to control program execution, including conditional branching.

5.3.4.1. Execution Time

The minimum and maximum time-out conditions are 0 seconds and 10 seconds respectively. These can be changed by un-checking the **Default** checkboxes and entering new values.

The **Min Timeout** entry specifies the minimum time allowable to complete execution of the mnemonic. If the hardware completes execution in less than the minimum time, then PI-DATS will delay execution of the next mnemonic until the minimum delay has elapsed.

The **Max Timeout** entry specifies the maximum time allowable to complete execution of the mnemonic. If the hardware has not completed execution before the maximum time has elapsed, then PI-DATS will report a failure of execution for that mnemonic. The **Post-Execute** property page can be used to specify what action to take if the mnemonic fails to execute.

5.3.4.2. Conditional Execution

The **Expression** input allows the user to specify a variable or expression to evaluate in order to control execution of the mnemonic. The result is then evaluated against the controls in the **Execution** input.

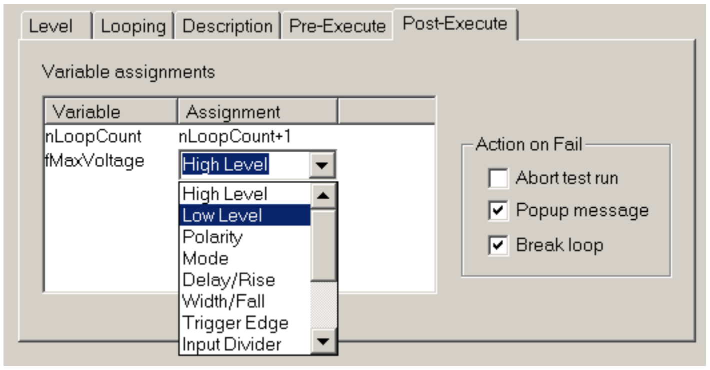
- When the **Always True** radio button is selected, then the **Expression** will always evaluate to **True**.
- When the variable radio button is selected, the user can select any defined variable from the drop-menu and then compare it against a user-defined expression entered in the text box to the right. In the example shown above, the mnemonic will be executed only when the integer variable **nVar** is less than 8.
- When the **Other** radio button is selected, the user may enter any mathematical expression for evaluation

The **Execution** input allows the user to specify whether the mnemonic is executed on a **True** result or **False** result from the **Expression**. If the **Simulate** checkbox is checked, then the mnemonic will be simulated. This allows test plans to be written and debugged even when all the desired hardware is not available.

If the **Multi Task** checkbox is checked, this allows the Test Plan to proceed and execute subsequent mnemonics even if the current mnemonic has not yet completed execution.

5.3.5. Post-Execute Property Page

The Post-Execute property page allows the user to assign new values to variables and specify what actions to take after successful or unsuccessful execution of the mnemonic



Variable	Assignment
nLoopCount	nLoopCount+1
fMaxVoltage	High Level

[ ] Abort test run
[✓] Popup message
[✓] Break loop

Figure 32: Post-Execute property page

Select any defined variable by clicking in the blank area below the **Variable** header. You can then select or type in a new value or expression into the **Assignment** area to the right. Available assignments include other variables, mathematical expressions, and programmable parameters of or calculated data from a RESULT mnemonic. In the example above, showing **Post-Execute** for a PI-40460 Clock Driver channel, the floating-point variable **fMaxVoltage** could be assigned to the value programmed for the **High Level**.

To delete a variable value assignment select the variable name and press Delete.

You can also specify what action to take upon failure to execute the mnemonic. A failure can result from an illegal programmed value, a time-out condition (see **Section 5.3.4.1. Execution Time**), a communications error, or a defined failure from a measurement device (see **Section 7.2.2. DVM Measure Mnemonic** and **Section 7.3.2. Scope Measure Mnemonic**).

If **Abort test run** is checked, then PI-DATS will interrupt the test plan on a failure and return control to the user. No test data will be logged.

If **Popup** message is checked, then PI-DATS will display a standard dialog on a failure. Execution will continue after the operator dismisses the dialog box.

If **Break loop** is checked, then PI-DATS will exit the parent Group Header's loop and begin executing the first mnemonic after the loop.

6. Pulse Instruments Hardware

6.1. General Notes

The following sections describe the use of each mnemonic used for hardware control. Although each property page described in this section is unique to a mnemonic, the following general principles apply to all mnemonics.

6.1.1. Controller mode vs. Test Plan mode

PI-DATS is closely related to PI-Controller. In fact, when PI-DATS is placed in Controller mode (via the Switch Mode button or menu item), the interface is identical to that of PI-Controller for hardware control mnemonics.

Please review the PI-Controller Operators Manual for a description of programmable parameters for each instrument and for a description of the Controller mode buttons. This manual will discuss only those items that relate to automated test planning. You should be familiar and comfortable with operation of all your hardware in Controller mode before starting to build automated test plans.

In Test Plan mode, the appearance of each mnemonic is different from what appears in Controller mode (or in PI-Controller). Controls that have meaning only in a run-time context (e.g. **Write**, **Read**, **Status**) are removed or grayed-out (disabled), and controls will appear that have meaning only in an automated test plan context (e.g. **Looping** controls, **Pre-Execute** and **Post-Execute** controls).

Mnemonics that have no meaning in a run-time context (e.g. **Result** mnemonics and operational controls such as **Goto** or **Halt**) will show only descriptive information while in Controller mode.

All programmable parameters, regardless of their context, are always preserved when switching between modes, or when switching among test plans within a single PIP file. All parameters are stored in the PIP file when saved.

6.1.2. Multiple instances

A mnemonic representing a given object can appear multiple times in a test plan. For example, the mnemonic representation of a DC Bias channel might appear once during an initialization routine, once during a parametric test routine, once during a dynamic test routine, and again during a shut-down routine while the tester is prepared for removal of the tested part.

Each instance of the mnemonic can have different values for the programmable parameters, looping controls, pre-execute and post-execute conditions, etc. All parameters in the General property pages are sent to the instrumentation when a mnemonic is executed, with the exceptions listed below.

To create multiple instances of a mnemonic that change only a few (or one) of the programmable parameters, use the Copy and Paste feature (buttons or menu items) to create duplicates of an existing instance of that mnemonic, then change only the parameters you wish to change when that mnemonic executes.

Different instances of a mnemonic can also be given different labels. The default name for each mnemonic in the Add Mnemonic window reflects the model number and the physical location of the hardware, but this information is always available in the Description property page. Therefore, you can give a mnemonic any label that is useful to the users or operator. For example, DC Bias channel mentioned above might be labeled Substrate_Bias_Init during the initialization phase of the test plan, and Substrate_Bias_Off during the shut-down phase.

6.1.3. Checkbox controls

While most programmable parameters are sent to the hardware when a mnemonic is executed, parameters marked with a checkbox are sent only if the checkbox is checked.

For example, in the Pattern Generator mnemonic for the PI-2005, the Set Period entry is used to set the clock period. If the checkbox is checked, then PI-DATS will send the value in the text box. If the checkbox is cleared, no Period command will be sent, and the Pattern Generator's period will remain unchanged.

6.1.4. Setup and Measure Mnemonics

For most measurement devices in PI-DATS, there are two paired mnemonics—a **Setup** mnemonic and a **Measure** mnemonic. The **Setup** mnemonic configures the instrument's measurement parameters, and is typically executed early in the test plan, or above a loop that takes multiple measurements. The **Measure** mnemonic acquires the reading from the instrument and optionally compares it against an expected value and/or assigns it to a PI-DATS variable.

At least one **Setup** mnemonic should have executed before its corresponding **Measure** mnemonic is executed. Otherwise, the instrument's state and settings will be unspecified, and the **Measure** may fail, or the readings may not be consistent from one test run to another.

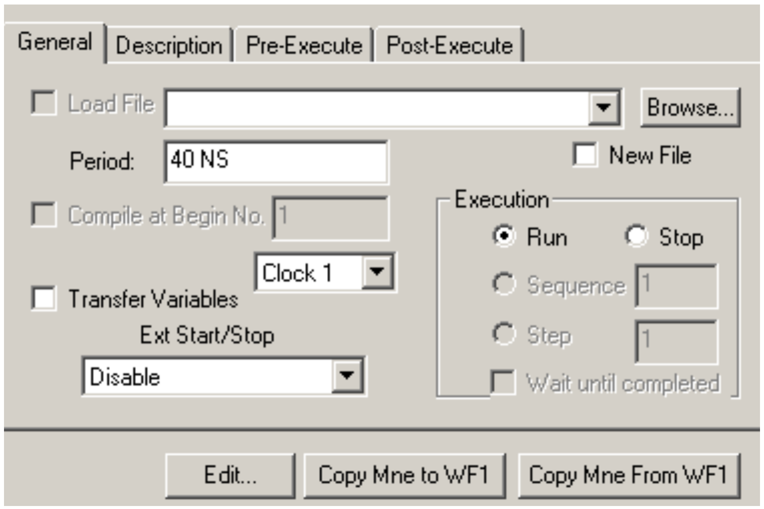
Most settings for a given instrument will be specified on its respective **Setup** mnemonic, and the **Measure** mnemonic will typically have few or no settings, except for the optional **Limits** checking and the standard **Pre-Execute** and **Post-Execute** controls.

6.2. Pattern Generator Mnemonic

This module controls a Pulse Instruments PI-2005 Pattern Generator.

Please note that there are different modules for different models of Pattern Generator. The PI-2005 uses a different control from that used by the PI-5800 and PI-2000, both of which are deprecated. Be sure to use the correct module for your model of pattern generator.

6.3. Pattern Generator General



- [ ] Load File:
- Period: 40 NS
- [ ] New File
- [ ] Compile at Begin No. 1
- Clock 1
- [ ] Transfer Variables
- Ext Start/Stop: Disable
- Execution: [✓ Run] Stop; Sequence 1; Step 1; [ ] Wait until completed

Figure 33: Pattern Generator mnemonic, General property page

6.3.1. Interaction Between PI-PAT and PI-Controller

PI-Controller and PI-DATS interact with PI-PAT by controlling it via OLE and Windows Automation. When either PI-DATS or PI-Controller launches, it first checks to see if PI-PAT is already running. If so, it establishes an OLE connection. If not, it will launch a hidden copy of PI-PAT and connect to it. A hidden copy of PI-PAT may be revealed either by double-clicking the PI-PAT icon or by clicking the **Edit** button on the Pat mnemonic in Controller mode.

If the operator closes the PI-PAT window after PI-Controller or PI-DATS has launched, PI-DATS and PI-Controller will re-open the window.

6.3.2. Load/Select Pattern File

To send a pattern file to the Pattern Generator when this mnemonic executes, check the **Load File** checkbox. Click **Select Pattern File** to browse for the pattern file (.wf1, .w20, 25, w58, .w65) to load.

A text file containing valid PI-2005 commands may also be imported using **Load File**. If the **New File** box is checked, then the text commands will be imported into a new, empty pattern file. If **New File** is not checked, the text commands will be imported into the last-opened file that was opened (.w25 or .txt).

If a filename is specified without any path information, PI-DATS will attempt to load the file from the same directory containing the current PIP file. This enables a PIP file and a pattern file to be copied or moved together from one system to another without “breaking” their links.

If the **Load File** checkbox is cleared, then the Pattern Generator will continue to operate on the last pattern file that was send to it.

Note that data loaded with a pattern file will not be reflected in the Pattern Generator output until patterns are updated, instructions are compiled, formats are written, and a period is set. See your **Pattern Generator Operators Manual** for details.

6.3.3. Period

To send a new clock period to the Pattern Generator, check the **Set Period** box and enter a new value in nanoseconds.

6.3.4. Compile

To compile a new set of program instructions, check the **Compile** checkbox and enter the **Begin** number of the desired program. If this checkbox is cleared, the Pattern Generator will run the last compiled program.

6.3.5. Transfer Variables

To transfer the values for variables defined in PI-DATS to the matching variable names in the pattern file, check the **Transfer Variables** checkbox. This enables PI-DATS variables to control loop and repeat counts in PI-PAT, enabling the automation of parameters such as integration time, dead time, and frame rate. The variable names and types must match exactly or else the values will not be transferred.

6.3.6. Execution

Choose the **Run** radio button to have the Pattern Generator run when the mnemonic is executed. Choose the **Stop** radio button to have the Pattern Generator stop.

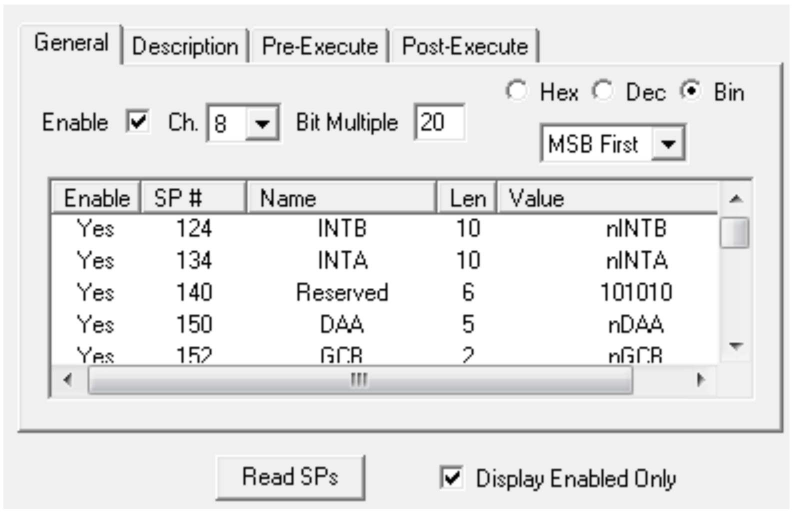
Use **Step** and **Sequence** to execute a finite run of the pattern. When **Step** or **Sequence** is chosen, the corresponding argument will control the number of subpatterns or program loops to execute.

When either **Step** or **Sequence** is selected, the **Wait until Completed** checkbox becomes enabled. When this box is checked the **Pat** mnemonic will wait until the Step or Sequence has completed before allowing PI-DATS to proceed to the next mnemonic.

6.3.7. Edit

Makes visible the PI-Pat application to permit editing of a Pattern file. PI-Pat will load the file specified in the **General** property page. Refer to your Pattern Generator Operators Manual for further information. Do not close PI-PAT after editing your file; otherwise PI-DATS will not be able to communicate with the pattern generator. Be sure to save any changes in PI-PAT before returning to PI-DATS, as execution of a PAT mnemonic will load the last-saved version of the specified pattern file.

6.4. Serial Command Word



Enable [✓] Ch. 8 Bit Multiple 20

Hex Dec [✓ Bin]

MSB First

Enable	SP #	Name	Len	Value
Yes	124	INTB	10	nINTB
Yes	134	INTA	10	nINTA
Yes	140	Reserved	6	101010
Yes	150	DAA	5	nDAA
Yes	152	GCR	2	nGCR

[✓] Display Enabled Only

Figure 34: Serial Command Word mnemonic, General property page

6.4.1. Introduction

The Serial Command Word (SCW) mnemonic permits the insertion of new data into pattern generator memory when the test plan runs. This data can program command registers used by FPAs and ROICs (hereafter "ROIC") to control features such as integration time, operating mode, windowing, internal bias generation, etc. Consult your ROIC documentation for a description of the control register.

6.4.2. Defining the Serial Word

Most ROIC control registers are programmed with a serial word containing a start bit, an optional preamble, and then a sequence of single- or multi-bit words that control various chip features. For example a ROIC may define a 25-bit serial word as follows:

- Bit 25: Start bit (always 1)
- Bits 24-23: Number of analog outputs enabled (00-11 binary)
- Bits 13-22: Integration time, in line periods (0-1023 decimal)
- Bits 9-12: On-chip bias generation factor, (0-15 decimal)
- Bits 5-8, Current adjustment factor, (0-15 decimal)
- Bits 1-4, Reserved (always 1010 binary)
- Bits 0: Skimming enable, (0 or 1)

While a serial word can be programmed directly into the pattern memory via the PI-PAT user interface, the SCW mnemonic provides the ability to automate the injection of serial data and use PI-DATS variables.

First, the pattern file must be written with a set of subpatterns of appropriate length to contain the serial data. The SCW mnemonic operates by writing a sequence of bits to the beginning of one or more subpatterns. Most FPAs and ROICs will receive one serial data bit per ROIC master clock.

The pattern file must be written to permit control of the shortest desirable word within the serial register. For example, in the example above there is at least one control defined as a single bit. If the user wishes to control all features discretely from PI-DATS, the pattern file must be written with each ROIC master clock period at least 20 pattern bits.

If several short words can be “ganged together” and programmed as a longer word, then the shortest subpattern can contain more ROIC clock periods. For example if “skimming” will never be programmed from within PI-DATS, then the “skimming” bit can be combined with the “reserved” bits, and the shortest programmable feature is now 2 bits. The shortest subpattern could now be written to contain two ROIC clocks and still provide control over all remaining features.

This clock multiplication factor, or the number of pattern bits per ROIC control bit, is called the “**Bit Multiple**,” and must be implemented in the pattern file before the SCW mnemonic can be used.

6.4.3. Reading Subpatterns

Once the timing file has been written with the appropriate subpatterns, insert a SCW mnemonic into your test plan. Select which pattern channel will contain the serial command word, based on the ROIC pinout and the system/DUT cabling, and then enter the **Bit Multiple** as defined above.

The subpattern definitions and their existing values can be read into the SCW by pressing the **Read SPs** button. PI-DATS will display a prompt to select a .w25 file. Once PI-DATS has read the pattern file, all subpatterns whose length is an integer multiple of the **Bit Multiple** will be displayed in the SCW list. Subpatterns whose length is not an integer of the **Bit Multiple** will be listed as errors in the status bar. This does not necessarily indicate an error in the pattern file, but any subpattern in the error list will not be programmable from the SCW mnemonic.

For each valid subpattern, 5 fields are displayed:

- **Enable**: Whether to write to this subpattern when the mnemonic is executed
- **SP#**: The subpattern number as read from the pattern file (read-only)
- **Name**: A user-editable name
- **Length**: The subpattern length divided by the bit multiple (read-only)
- **Value**: The value to be written to the pattern generator

6.4.4. Programming the Serial Word

When executed the SCW will write new data only to the subpatterns where **Enabled** is set to **Yes**. Values may only be programmed for enabled subpatterns. If a subpattern is set to **Enabled = No** in the SCW mnemonic, its value will be displayed as **?** and editing will be disabled. When a subpattern is **Enabled** for output, the **Value** field will display the last-programmed value/variable for the SCW mnemonic or the current value as read from the pattern file if no value has yet been programmed in the SCW mnemonic.

Use the **Name** field to identify which ROIC features are mapped to each subpattern.

The **Values** may be programmed as binary, hex, or decimal values, depending on the setting of the radio button, or using a PI-DATS variable. When the radio button selection is changed, all static values will be translated into the selected format. PI-DATS variables are always interpreted as decimal values.

When executed, each enabled **Value** in the SCW list is expanded by repeating each bit in the binary-equivalent value by the Bit Multiple. The bit stream is then written to pattern memory for the indicated channel, and then the pattern generator is updated. For example, if 20 were the Bit Multiple and 01 were programmed as the binary value for **Number of Outputs** in the example above, the following data would be written to pattern memory:

0000000000000000000011111111111111111111

Then the pattern memory would be written to hardware as follows:

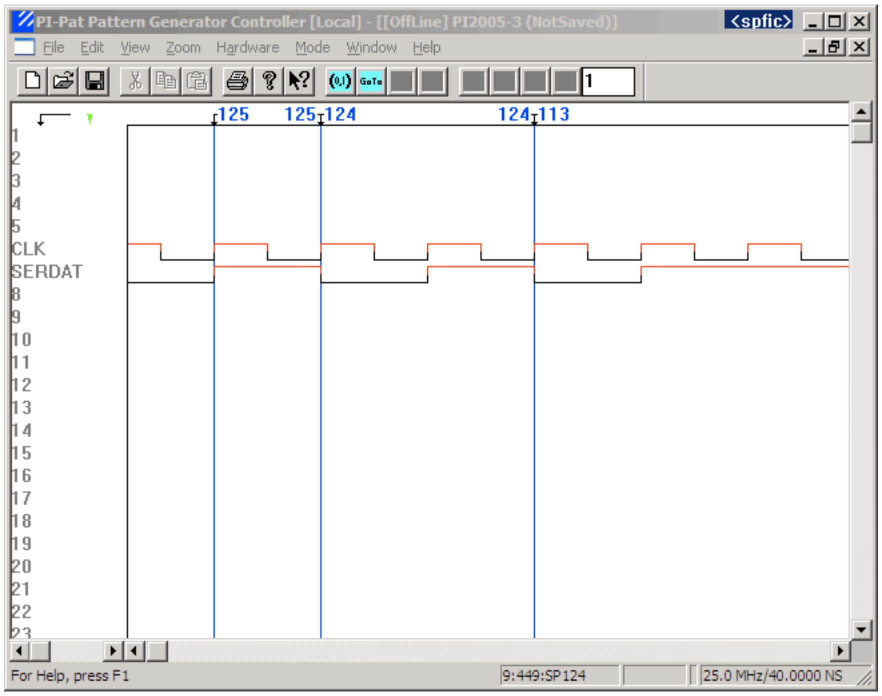

PI-Pat Pattern Generator Controller [Local] - [[OffLine] PI2005-3 (NotSaved)]
<spfic>
File Edit View Zoom Hardware Mode Window Help
125
125
124
124
113
1
2
3
4
5
CLK
SERDAT
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
For Help, press F1
9:449:SP124
25.0 MHz/40.0000 NS

Figure 35: Serial bits written to SPs 125, 124, and 113

6.4.5. Controlling The Serial Word

Disabled subpatterns can be hidden by checking the Display Enabled Only checkbox.

The entire SCW mnemonic can be disabled by unchecking the Enable checkbox.

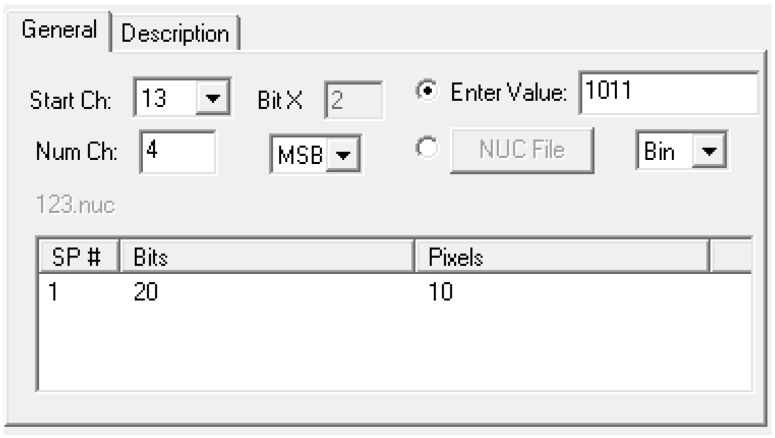
6.4.6. Best Practices

For convenience and readability, the user may want to use subpattern numbering based on the ROIC definition of the serial word. If the lower-numbered subpatterns have already been programmed, adding 100 or 200 may be a useful way to keep clear the relationship between serial register bits and pattern data. For example, defining subpatterns for the example ROIC as follows would make it easy to correlate subpattern data with ROIC register locations.

- Bit 25: SP 125
- Bits 24-23: SP 124
- Bits 13-22: SP 113
- Bits 9-12: SP 109
- Bits 5-8, SP 105
- Bits 1-4, SP 101
- Bits 0: SP 100

Similarly, the Name fields should be populated with names that match or map to the register names and/or the functions they are controlling. This is especially important if multiple control bits have been combined into a single subpattern.

6.5. NUC Pattern Injection



Start Ch: 13
Bit X: 2
(● Enter Value: 1011)
Num Ch: 4
MSB
Bin
123.nuc

SP #	Bits	Pixels
1	20	10

Figure 36: NUC Pattern Injection mnemonic, General property page

6.5.1. Introduction

The **NUC Pattern** mnemonic permits the insertion of new data into pattern generator memory when the test plan runs. This data is used primarily for on-chip non-uniformity correction (NUC), on ROICs specifically designed with this feature. In these applications the ROIC hardware requires an N-bit correction word to be clocked into the readout for each pixel on the array. Consult your ROIC documentation for a description of its NUC correction feature.

The **NUC Pattern** mnemonic will inject into an existing pattern file one of the following:

- NUC data from a file (see **Section** 9.6.2.2. NUC Export)
- An **Entered Value**, repeated for every pixel (e.g. 0000 or 1011)
    - Entered values may be static or variable
    - Entered values may be interpreted as hex, decimal, or binary
    - Entered values may be converted to binary MSB-first or LSB-first
    - **Entered Values** are typically used for calibration

Each pixel's data will be "rotated" and entered across N pattern channels, where N is the bit-width of the data, so that the bits of a correction word are output to the ROIC in parallel, synchronously with the ROIC timing.

The user must specify the channel number(s) and subpattern number into which to inject the data.

If the data to be entered is from a previously-created **NUC File**, the user may also enter an “offset” number of pixels to advance (positive) or retard (negative) the NUC data relative to the start of the subpattern. The data will be "wrapped around" in the data array before all other processing is done, e.g. if the data array has the following 8 pixels of NUC data:

ABCDEFGH

Advancing it by +2 would result in:

CDEFGHAB

and Retarding it by -2 would result in:

GHABCDEF

This will occur before bit multiplying and other processing.

This feature accommodates the processing pipeline in the ROIC’s hardware, in order to align each correction word with its pixel.

6.5.2. Entered Value example

An **Entered Value** will be converted to binary (if necessary) and then injected across the specified channels as constant bit values. If **MSB** is specified, then the MSB of the binary word will be injected into the first (lowest-numbered) channel.

For example the **Entered Value** “1011”, injected into Supattern 1, MSB first, across Channels 13-16, will appear as follows:

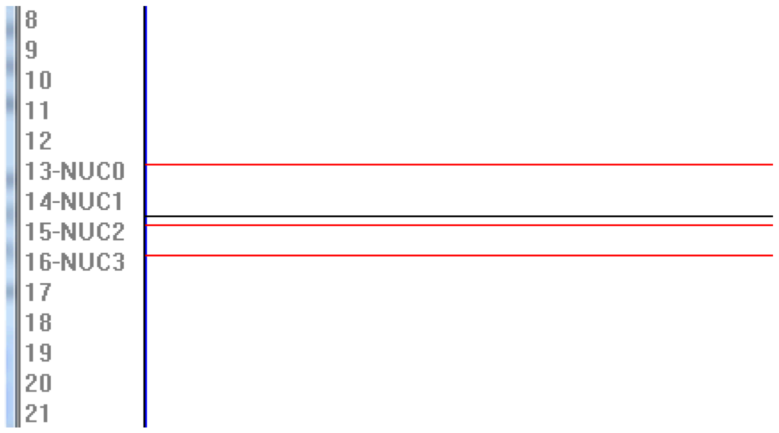
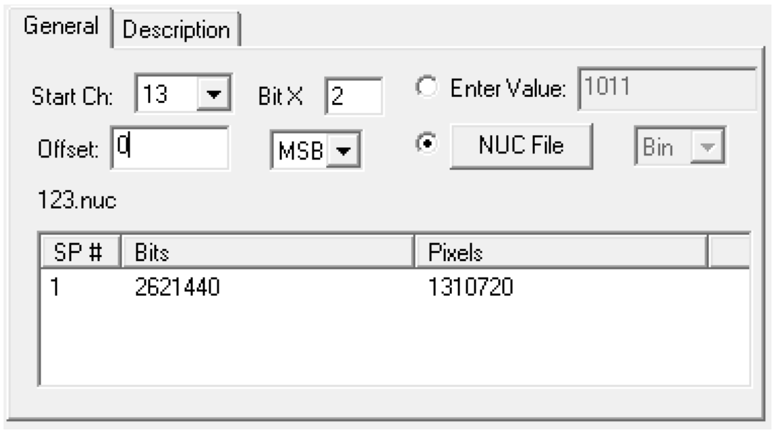


Figure 37: NUC Pattern Injection data in PI-PAT (Fixed value)

The magnitude of the injected word must be less than or equal to 2^n-1, where n = the Number of Channels.

6.5.3. NUC File example



Start Ch: 13

Bit X 2

Enter Value: 1011

Offset: 0

MSB

[✓ NUC File]

Bin

123.nuc

SP #	Bits	Pixels
1	2621440	1310720

Figure 38: NUC Pattern Injection, NUC File

Data from a **NUC File** will be injected into the specified channels, in similar fashion to the **Entered Value**, but with an individual vector for each pixel, as read from the NUC file.

The bit multiplier (**Bit X**) setting controls how many times to repeat each datum in pattern generator memory, e.g. 0110 expands to 000011110000 if **Bit X** is set to 2, for a case of two pattern bits per FPA master clock.

The **Offset** controls the shifting of data in the NUC file relative to the readout timing.

The number of pixels in the NUC file must be less than or equal to the number of pixels in the target subpattern.

In the example below, a NUC file containing 1280 x 1024 = 1,310,720 pixels of 4-bit NUC data is injected into Channels 13-16, MSB first, with a Bit X of 2, offset of zero, and the following data values:

Pixel Number:	**Floating Point Value:**	**Rounded Value:**	**Binary Value:**
0	6.24	6	0110
1	5.87	6	0110
2	4.89	5	0101
3	6.63	7	0111
4	3.15	3	0011
5	3.39	3	0011
6	3.54	4	0100
7	4.89	5	0101
8	2.99	3	0011
9	3.14	3	0011
. . .			
1310718	26.28	15	1111
1310719	26.75	15	1111

The data is “rotated” into the data channels as follows:

	Pixel Number:											
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**…**	**1310718**	**1310719**
NUC0	0	0	0	0	0	0	0	0	0	0		1	1
NUC1	1	1	1	1	0	0	1	1	0	0		1	1
NUC2	1	1	0	1	1	1	0	0	1	1		1	1
NUC3	0	0	1	1	1	1	0	1	1	1		1	1

The data is then repeated by the **Bit X** factor:

	Pixel Number:												
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**…**	**1310718**	**1310719**
NUC0	00	00	00	00	00	00	00	00	00	00		11	11
NUC1	11	11	11	11	00	00	11	11	00	00		11	11
NUC2	11	11	00	11	11	11	00	00	11	11		11	11
NUC3	00	00	11	11	11	11	00	11	11	11		11	11

The resulting data in PI-PAT appears below (2X horizontal zoom and green gridlines added for clarity):

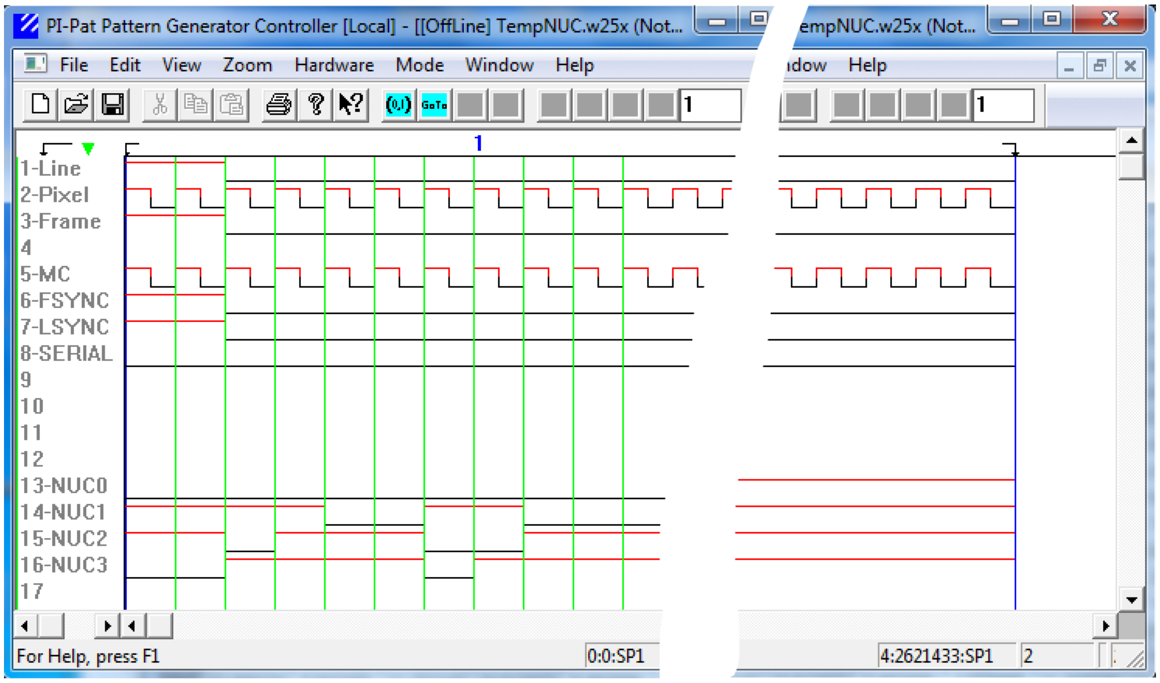


Figure 39: NUC Pattern Injection data in PI-PAT (NUC File), Offset = 0

If the Offset were set to a value of 2, the data would be advanced by 2 pixels, with first two pixels “wrapped around” to the end of the subpattern:

	Pixel Number:												
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**…**	**1310718**	**1310719**
NUC0	00	00	00	00	00	00	00	00	00	00		00	00
NUC1	11	11	00	00	11	11	00	00	00	00		11	11
NUC2	00	11	11	11	00	00	11	11	11	11		11	11
NUC3	11	11	11	11	00	11	11	11	11	11		00	00

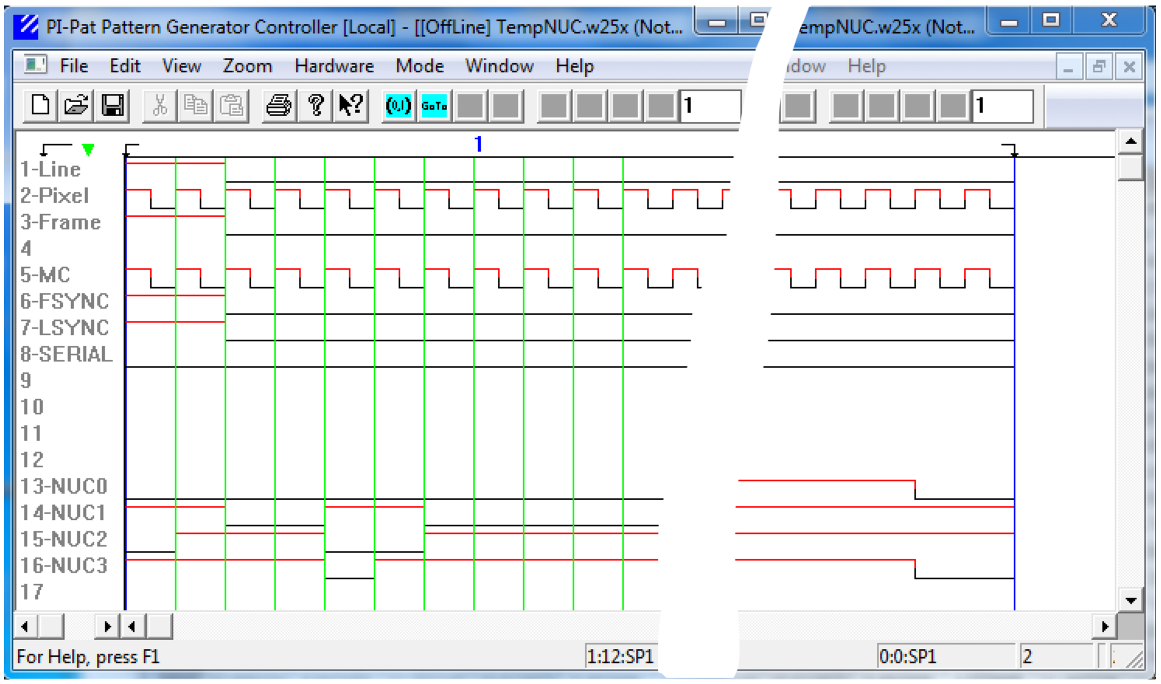


Figure 40: NUC Pattern Injection data in PI-PAT (NUC File), Offset = +2

If the Offset were set to a value of 2.5, the data would be advanced by 2.5 pixels, with first two pixels "wrapped around" to the end of the subpattern and shifted by an additional half pixel:

	Pixel Number:												
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**…**	**1310718**	**1310719**
NUC0	00	00	00	00	00	00	00	00	00	00		00	00
NUC1	11	10	00	01	11	10	00	00	00	00		11	11
NUC2	01	11	11	10	00	01	11	11	11	11		11	10
NUC3	11	11	11	10	01	11	11	11	11	11		00	01

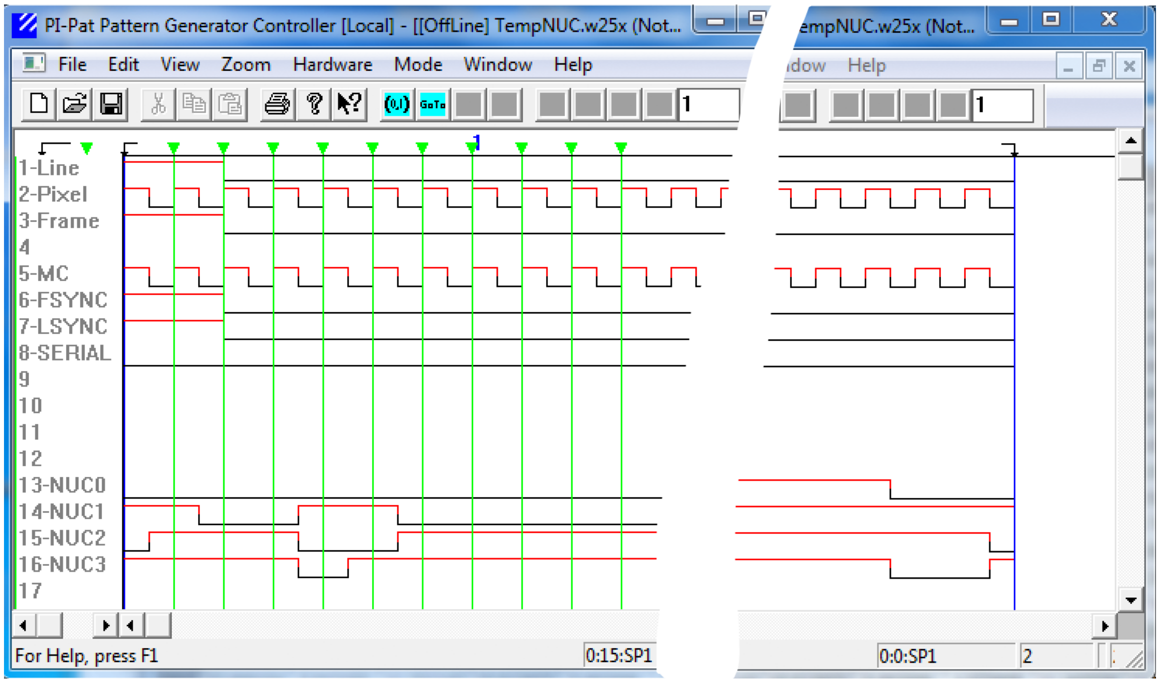


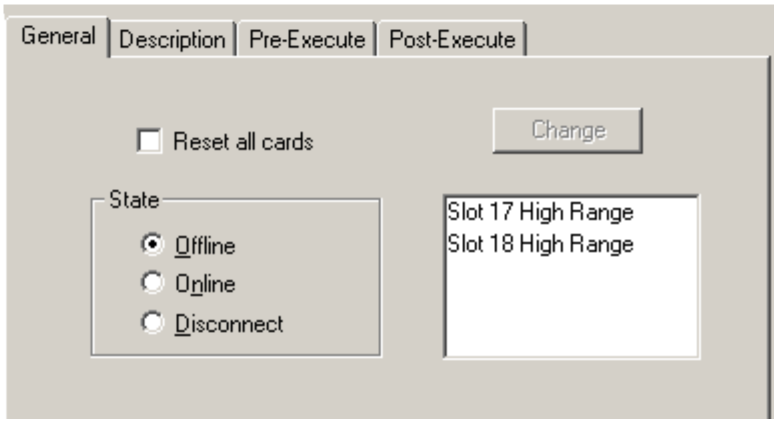
Figure 41: NUC Pattern Injection data in PI-PAT (NUC File), Offset = +2.5

6.6. *Mainframe Mnemonic*

This module controls a Pulse Instruments Control Mainframe or PI-Bus Interface card for the 4000 Series Low Noise Clock Driver and DC Bias instruments. The following models are supported:

- PI-11000 Instrument Mainframe (11006, 11006S, 11008, 11008S), with or without a PI-31001 PI-Bus Interface Card

For information on programming the instrument cards, please the section for each type of card (Clock Driver, DC Bias, Multiplexer, etc)



☐ Reset all cards

State: [✓ Offline] Online Disconnect

Slot 17 High Range
Slot 18 High Range

Figure 42: Control Mainframe, General property page

If the **Reset all cards** box is checked, the all cards will be initialized and set to their default values when this mnemonic is executed.

Choose the desired mainframe state with the **State** radio buttons. In **Offline** mode, programmed changes to channel voltages and settings will not be reflected at the channel outputs until the mainframe is set to **Online**. While **Offline**, all channel outputs will remain at their last programmed value.

In **Online** mode, all changes written to instrument cards are reflected immediately at the outputs.

In **Disconnect** mode, outputs from all instrument cards are connected to ground through a resistor. This effectively disconnects them from the DUT, allowing the DUT to be physically disconnected safely.

In **Disconnect** mode, selected models of clock driver or DC bias card can also be switched to different operating ranges.

For clock driver and DC bias card models that have multiple voltage ranges, the Range Selection box will contain the programmed range for all channels on each card, as appropriate. To change the range for a card, program the mainframe to **Disconnect** mode, then select the desired card and click **Change**. The ranges can be changed only in **Disconnect** mode, and only when all programmed voltages and limits are within the allowable values for the desired range. Please see the **PI-11000 Operators Manual** for more information.

6.7. *Clock Driver Mnemonic*

This module controls a Pulse Instruments Clock Driver card. The following models are supported:

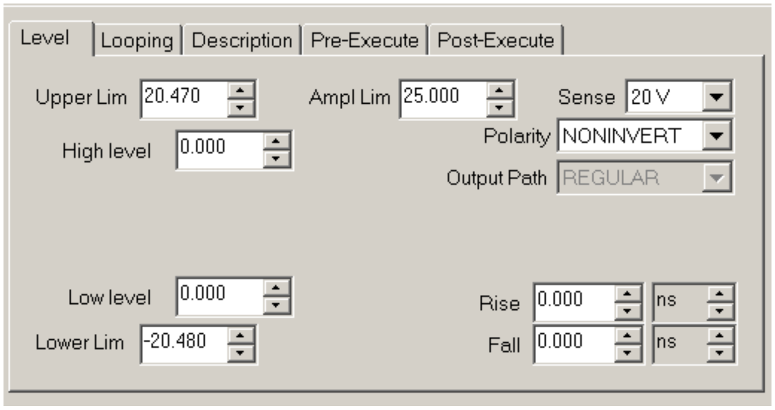
- 4000 Series
  - PI-40460
  - PI-40461
  - PI-40462
  - PI-40465
  - PI-40480
  - PI-40490
  - PI-42460
  - PI-42490
- CompactPCI Series
  - PI-41400
  - PI-41401

A basic Clock Driver channel mnemonic (PI-40460) will be described first, followed by additional features specific to other card models.

6.7.1. Clock Driver Level property page

The Clock Driver Level property page in PI-DATS is identical to the property page for a clock driver in PI-Controller. It permits the setting of rail voltages, voltage limits, amplitude limit, sense range, and rise/fall or slew times. Selected models will also have controls for polarity, output path, input path and/or impedance, tri-state control, delay, width, width mode, input divider, and mid-level controls.

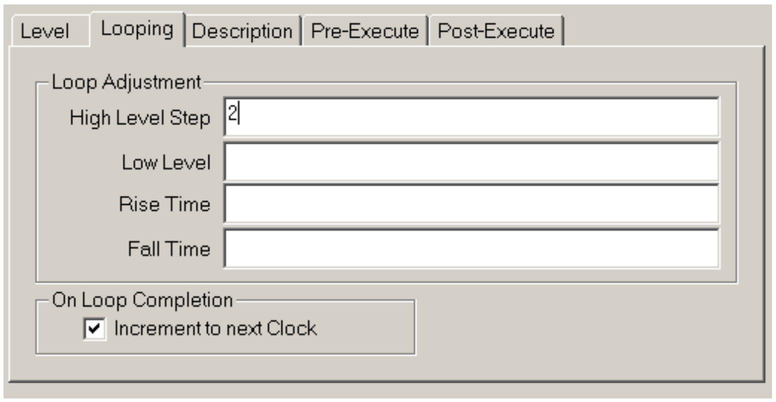
Please review your PI-Controller manual for details on the programmable parameters for each card model.



Upper Lim 20.470
High level 0.000
Ampl Lim 25.000
Sense 20 V
Polarity NONINVERT
Output Path REGULAR
Low level 0.000
Lower Lim -20.480
Rise 0.000 ns
Fall 0.000 ns

Figure 43: Typical Clock Driver mnemonic, Level property page

6.7.2. Clock Driver Looping property page



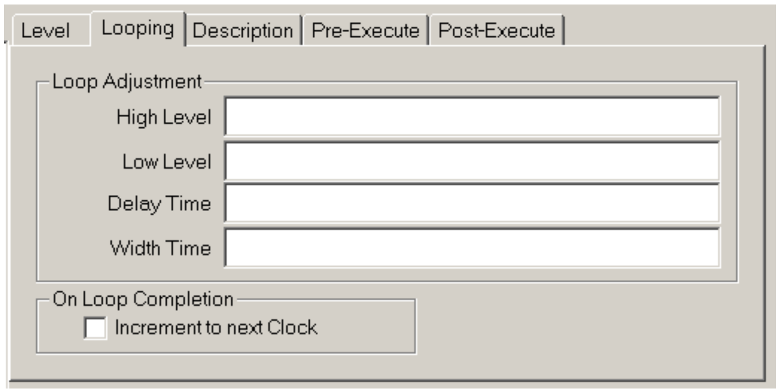
High Level Step: 2
Low Level:
Rise Time:
Fall Time:

[✓] Increment to next Clock

Figure 44: Typical Clock Driver mnemonic, Looping property page

All models of Clock Driver card support looping and nested looping. Please see **Section 5.1.2. Looping Options** for details on implementing looping.

High Level, Low Level, Rise Time, and Fall Time may be adjusted on each loop. **Figure 44** shows the looping controls for all models of Clock Driver card except for the PI-41401, PI-40480 and PI-40490.



High Level:
Low Level:
Delay Time:
Width Time:

[ ] Increment to next Clock

Figure 45: Looping property page for PI-40480 and PI-40490

The PI-40480 and PI-40490 have adjustable delay and width, but do not have adjustable rise/fall times. Therefore, the **Looping** property page is different for these two models. **Delay Time** and **Width Time** controls replace the **Rise Time** and **Fall Time** controls for the other Clock Driver cards.

The PI-41401 has a common slew rate control instead of separate rise and fall times.

6.8. DC Bias Mnemonic

This module controls a Pulse Instruments DC Bias card. The following models are supported:

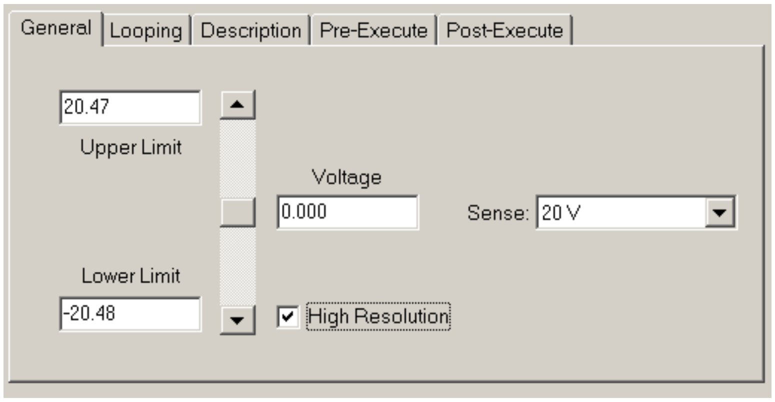
- 4000 Series
  - PI-40740
  - PI-40741
  - PI-40750
  - PI-40751
  - PI-40752
  - PI-40753
  - PI-40754
  - PI-40755
  - PI-40756
  - PI-40757
  - PI-40758
  - PI-40759
- CompactPCI Series
  - PI-41700
  - PI-41701
  - PI-41702
  - PI-41703

A basic DC Bias channel mnemonic (PI-40750) will be described first, followed by additional features specific to other card models.

6.8.1. DC Bias General property page

The DC Bias General property page in PI-DATS is identical to the property page for a DC Bias in PI-Controller. It permits the setting of voltage, voltage limits, and sense range. Selected models will also have controls for current protection.

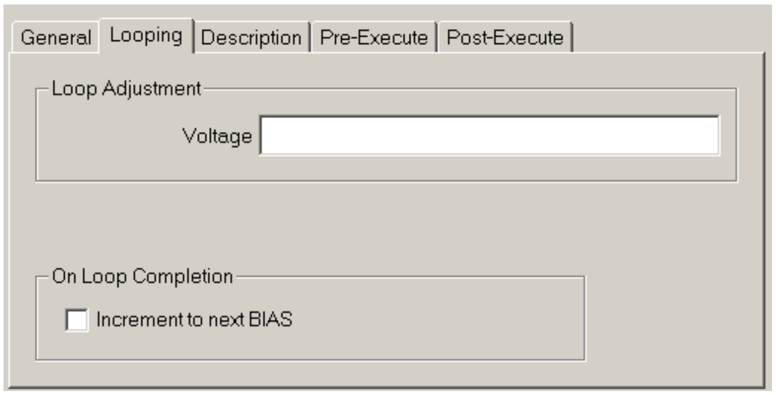
Please review your PI-Controller manual for details on the programmable parameters for each card model.



Upper Limit: 20.47
Voltage: 0.000
Sense: 20 V
Lower Limit: -20.48
[✓] High Resolution

Figure 46: Typical DC Bias mnemonic, General property page

6.8.2. DC Bias Looping property page



Loop Adjustment

Voltage

On Loop Completion

☐ Increment to next BIAS

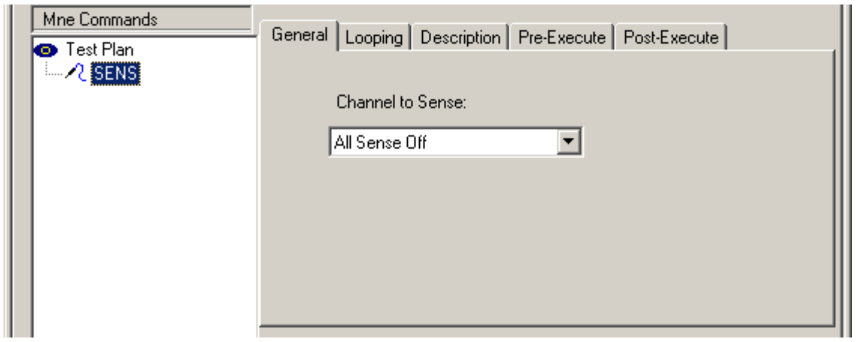
Figure 47: Typical DC Bias mnemonic, Looping property page

All models of DC Bias card support looping and nested looping. Please see **Section 5.1.2. Looping Options** for details on implementing looping.

The voltage level may be adjusted on each loop iteration.

6.9. Sense Mnemonic

6.9.1. Sense mnemonic General property page



Channel to Sense:

All Sense Off

Figure 48: Sense mnemonic, General property page

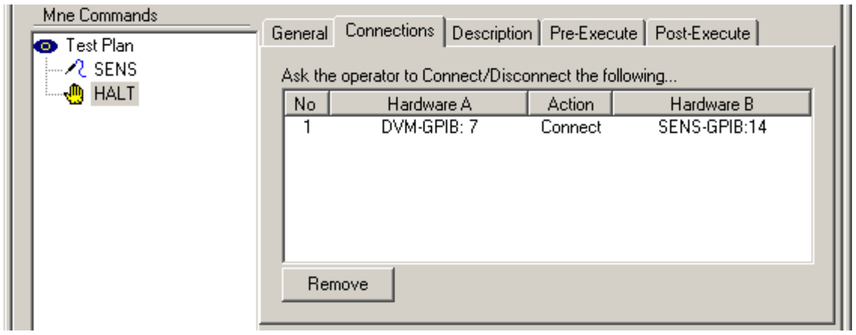
The General property page selects which DC Bias or Clock Driver channel to select for voltage or current sensing. The drop-menu will show all available channels in the system, plus an option for “All Sense Off.” When the Sense mnemonic is executed, the selected channel will have its Sense circuit routed to the sense port, and all other channels will have their Sense outputs shut off. If **All Sense Off** is chosen, there will be no output at the sense ports.

<u>6.9.1.1. Sense mnemonic and Connections</u>

The sensing type (voltage or current) and the sense range will be controlled by the mnemonic for the desired clock driver or bias channel. If the sense measurement is to be made by a DMM or DVM in PI-DATS, you will need to ensure that PI-DATS is aware that it is measuring a sense output; otherwise it will not know how to scale the measurement properly.

For example, if a DC bias channel is programmed for 1 V, a DVM hooked to the sense port will read 1.0 V if the bias channel is set to the 2 V range, but only 0.1 V if the channel is set to the 20 V range. If the 1 V output is loaded with 50 Ohms and the bias channel set for 100 mA current sensing, the DVM will read 0.2 V.

To inform PI-DATS that it is measuring a **Sense** output, use either the **Connections** property page in the **Advanced Hardware Configuration** dialog box (for permanent connections) or the **Connections** property page of a **Halt** mnemonic in the test plan itself. Once the connection has been established PI-DATS will use the Sense control to determine which channel is being sensed and the channel's mnemonic to determine what sense range is being used.



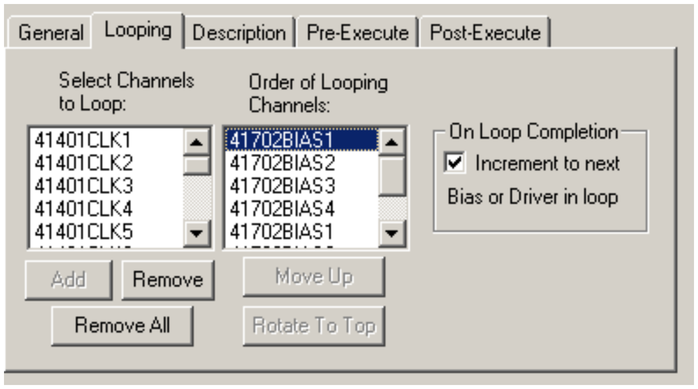
Ask the operator to Connect/Disconnect the following...

No	Hardware A	Action	Hardware B
1	DVM-GPIB: 7	Connect	SENS-GPIB:14

Figure 49: Sense connection specified in a Halt mnemonic

For more information on how PI-DATS scales readings from the DVM, please see **Section 7.2. DVM Mnemonic**.

6.9.2. Sense mnemonic Looping property page



Select Channels to Loop:

41401CLK1
41401CLK2
41401CLK3
41401CLK4
41401CLK5

Order of Looping Channels:

41702BIAS1
41702BIAS2
41702BIAS3
41702BIAS4
41702BIAS1

On Loop Completion

☑ Increment to next Bias or Driver in loop

Figure 50: Sense mnemonic, Looping property page

If your Sense mnemonic is placed in a loop, you may select a different channel to sense on each iteration of the loop.

To sense channels in a specific order, select them in the left pane of the **Looping** property page and click **Add** to add them to the looping order. To sense channels sequentially, add the first channel to sense and then check the Increment box.

If the **Order of Looping** contains channels, the **General** property page will be disabled.

6.10. Data Acquisition Mnemonics

There are six mnemonic controls for the PI-3105 Data Acquisition System in PI-DATS:

- AIM
- TIMING
- DEFINE
- DACQ Setup
- DACQ Measure
- BLOCK mnemonic

These mnemonics correspond directly to the mnemonics of the same name in PI-Controller, with the exception of the DACQ Measure and BLOCK mnemonics that occurs only in PI-DATS.

In general, each of the first three mnemonics will execute exactly as if the **Write** button had been clicked in PI-Controller or Controller mode. If Looping parameters have been entered, then the loop values will be used instead.

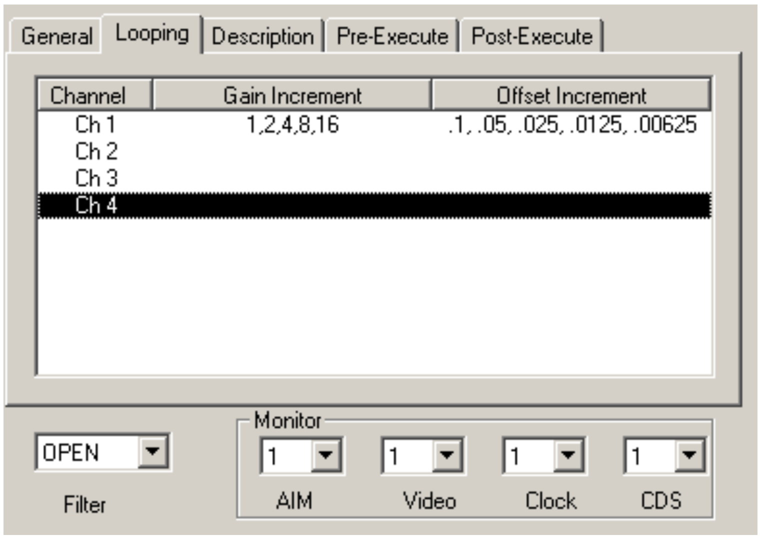
The **DACQ Setup** mnemonic is derived from the **DACQ** mnemonic in PI-Controller, but lacks the **Start Acq** button. The function that triggers the data acquisition is contained in the **DACQ Measure** mnemonic, which has only that one function.

6.10.1. AIM Mnemonic

This mnemonic contains controls for the Gain and Offset for each A/D channel, as well as the Filter and Monitor settings. When this mnemonic is executed in non-looped mode, it will behave as though the **Write** button were clicked in PI-Controller, using all the settings that are populated on the **General** property page.

The **Set All** checkbox is for programming convenience only, and has no effect during the test plan run. When **Set All** is checked, the user may use the slider controls to populate the settings for all channels simultaneously, or the user may select individual channels and populate them by typing in values. In either case, the settings that are present on the property page when the mnemonic is executed will be applied, regardless of the state of the **Set All** checkbox.

6.10.1.1. Looping



Channel	Gain Increment	Offset Increment
Ch 1	1,2,4,8,16	.1, .05, .025, .0125, .00625
Ch 2		
Ch 3		
Ch 4		

Filter: OPEN

Monitor: AIM 1, Video 1, Clock 1, CDS 1

Figure 51: AIM Mnemonic, Looping Property Page

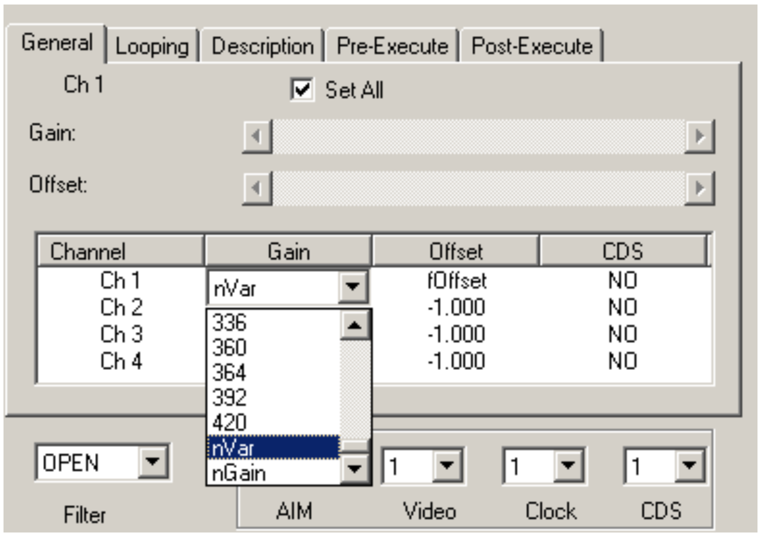
Gain and Offset values may be changed on each iteration of a loop. If a single value is placed in the Gain Increment or Offset Increment box, the Gain or Offset will be incremented by that amount on each subsequent loop following Loop 1. If multiple values are entered in a comma-delimited list, those values will be applied in order on each loop, beginning with Loop 1.

If the AIM mnemonic attempts to program a Gain value that lies between two allowable values, PI-DATS will enter the nearest value. If the AIM mnemonic attempts to program a Gain value outside the allowable range, the mnemonic will fail execution on that loop.

If the AIM mnemonic attempts to program an Offset value outside the allowable range, the mnemonic will fail execution on that loop.

6.10.1.2. Variables

The Gain and Offset can also be controlled via PI-DATS variables. In Test Plan mode, variable names may be used instead of numeric values. On the General property page, the Gain control for each channel will contain a list of all possible gain values, plus a list of all defined PI-DATS integer variables. The Offset controls do not display a list, but they will accept any integer or real variable names as valid entries.



Channel	Gain	Offset	CDS
Ch 1	nVar	fOffset	NO
Ch 2		-1.000	NO
Ch 3		-1.000	NO
Ch 4		-1.000	NO

336
360
364
392
420
nVar
nGain

OPEN | 1 | 1 | 1

Figure 52: AIM Mnemonic, Variable Support

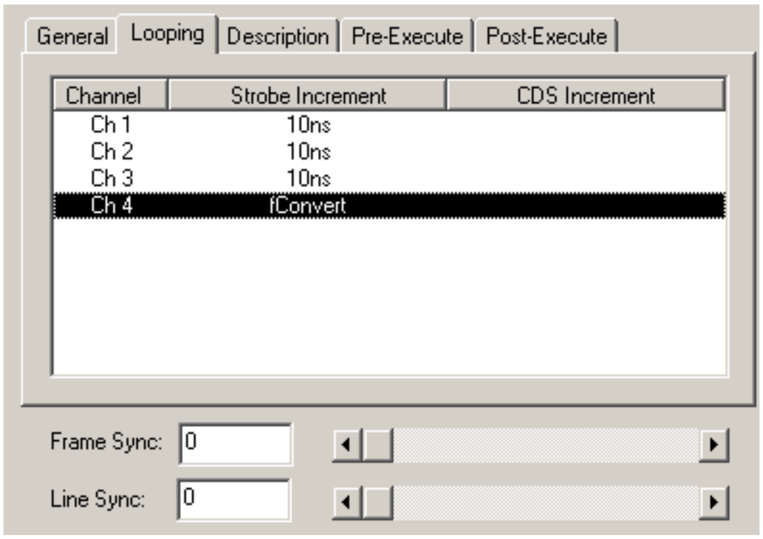
Variable names can also be used as entries on the Looping property page.

6.10.2. TIMING Mnemonic

This mnemonic contains controls for the Convert Strobe and CDS Strobe for each A/D channel, as well as the Line Sync and Frame Sync for each AIM. When this mnemonic is executed in non-looped mode, it will behave as though the **Write** button were clicked in PI-Controller, using all the settings that are populated on the **General** property page.

The **Set All** checkbox is for programming convenience only, and has no effect during the test plan run. When **Set All** is checked, the user may use the slider controls to populate the settings for all channels simultaneously, or the user may select individual channels and populate them by typing in values. In either case, the settings that are present on the property page when the mnemonic is executed will be applied, regardless of the state of the **Set All** checkbox.

6.10.2.1. Looping



Channel	Strobe Increment	CDS Increment
Ch 1	10ns	
Ch 2	10ns	
Ch 3	10ns	
Ch 4	fConvert	

Frame Sync: 0

Line Sync: 0

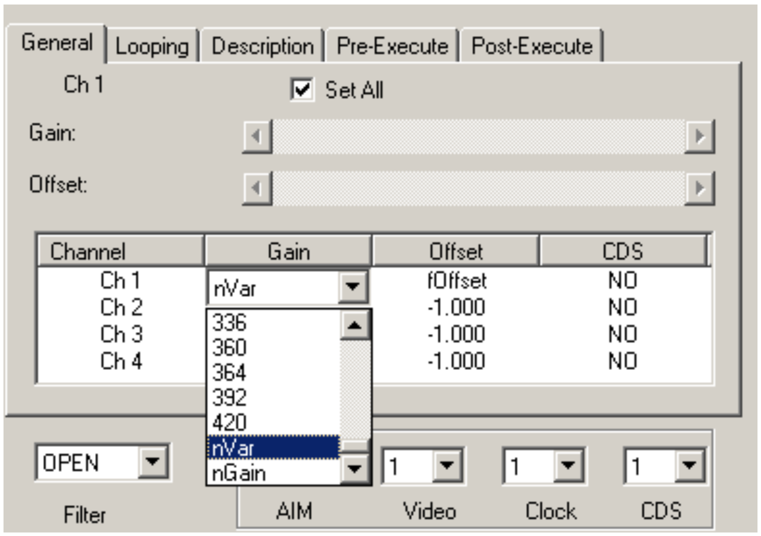
Figure 53: PI-3105 TIMING Mnemonic, Looping Property Page

The Convert Strobe and CDS Strobe values may be changed on each iteration of a loop. If a single value is placed in the Strobe Increment or CDS Increment box, the Convert Strobe or CDS Strobe will be incremented by that amount on each subsequent loop following Loop 1. If multiple values are entered in a comma-delimited list, those values will be applied in order on each loop, beginning with Loop 1.

If the TIMING mnemonic attempts to program a value outside the allowable range, the mnemonic will fail execution on that loop.

6.10.2.2. Variables

The Convert Strobe and CDS Strobe can also be controlled via PI-DATS variables. In Test Plan mode, variable names may be used instead of numeric values. On the General property page, the Convert Strobe and CDS Strobe controls for each channel will accept any integer or real variable names as valid entries.



General | Looping | Description | Pre-Execute | Post-Execute

Ch 1 ☑ Set All

Gain:

Offset:

Channel	Gain	Offset	CDS
Ch 1	nVar	fOffset	NO
Ch 2		-1.000	NO
Ch 3		-1.000	NO
Ch 4		-1.000	NO

Gain drop-down list: 336, 360, 364, 392, 420, nVar, nGain

Filter	AIM	Video	Clock	CDS
OPEN		1	1	1

Figure 54: TIMING Mnemonic, Variable Support

Variable names can also be used as entries on the Looping property page.

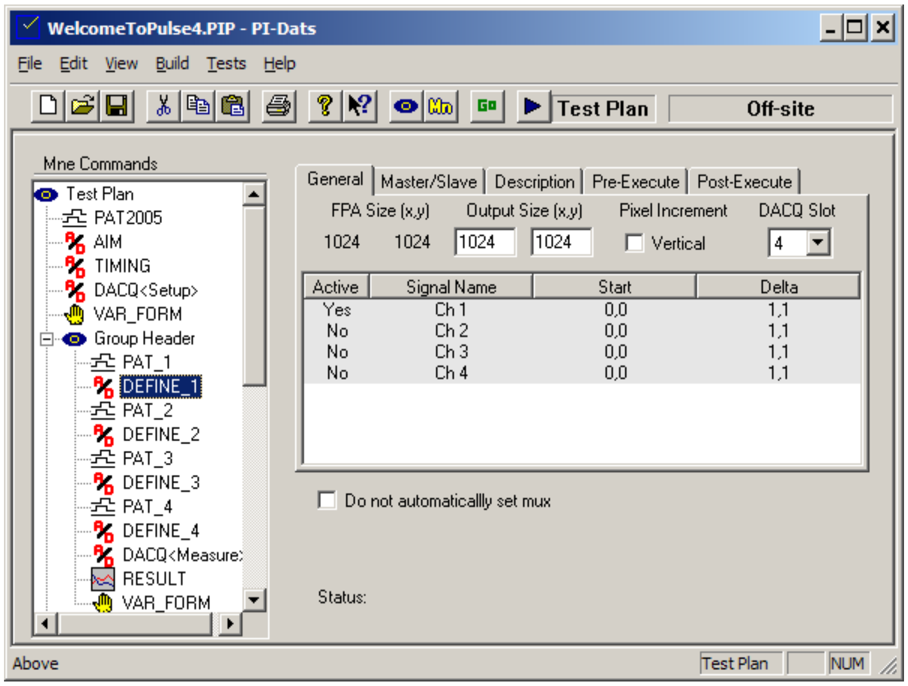
6.10.3. DEFINE Mnemonic

This mnemonic contains controls for the activating A/D channels and defining their locations within your device. When this mnemonic is executed, it will behave as though the **Write** button were clicked in PI-Controller, using all the settings that are populated on the **General** and **Master/Slave** property pages.

There is no looping or variable entry support in the DEFINE mnemonic. If you wish to change the FPA definition within a PI-DATS loop, this can be done via multiple DEFINE mnemonics that are conditionally executed.

For example, if your device can read out from any one of 4 output ports, it may be desirable to set up a test plan that collects data from each of the 4 output modes in a loop.

To accomplish this, set up a loop with 4 iterations, and insert 4 different DEFINE mnemonics inside the loop, where each DEFINE sets up the device and the A/D system for a different device output. The device output mode will typically be controlled either by the timing pattern used or a set of configuration registers programmed by a serial command word or voltage levels. The example test plan below assumes that output modes can be switched by using a different file or BEGIN number in a PAT mnemonic.


WelcomeToPulse4.PIP - PI-Dats

File Edit View Build Tests Help

Test Plan | Off-site

Mne Commands

- Test Plan
  - PAT2005
  - AIM
  - TIMING
  - DACQ<Setup>
  - VAR_FORM
  - Group Header
    - PAT_1
    - DEFINE_1
    - PAT_2
    - DEFINE_2
    - PAT_3
    - DEFINE_3
    - PAT_4
    - DEFINE_4
    - DACQ<Measure>
    - RESULT
    - VAR_FORM

General | Master/Slave | Description | Pre-Execute | Post-Execute

FPA Size (x,y)		Output Size (x,y)		Pixel Increment	DACQ Slot
1024	1024	1024	1024	☐ Vertical	4

Active	Signal Name	Start	Delta
Yes	Ch 1	0,0	1,1
No	Ch 2	0,0	1,1
No	Ch 3	0,0	1,1
No	Ch 4	0,0	1,1

☐ Do not automaticallly set mux

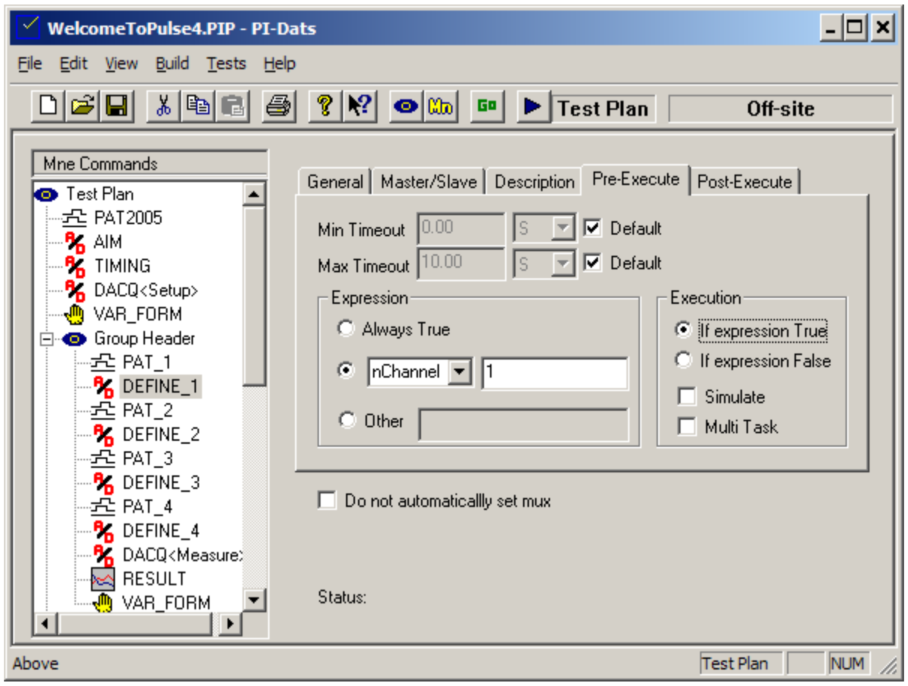
Status:

Above | Test Plan | NUM

Figure 55: DEFINE Mnemonic inside a loop

Because there is no explicit looping support for the **DEFINE** mnemonic, the execution of each mnemonic in the loop should be controlled by a **Pre-Execute** test against a loop counter. For example, this test plan contains an integer variable named **nChannel**, which is initialized to 1 immediately before the loop begins (via the VAR_FORM mnemonic).

Both the **PAT_1** mnemonic and the **DEFINE_1** mnemonic are conditionally executed only if **nChannel** is equal to 1. The **Pre-Execute** property page for the **DEFINE_1** is shown below; the **Pre-Execute** property page for the **PAT_1** mnemonic is similar.



Min Timeout 0.00 s [✓ Default]
Max Timeout 10.00 s [✓ Default]

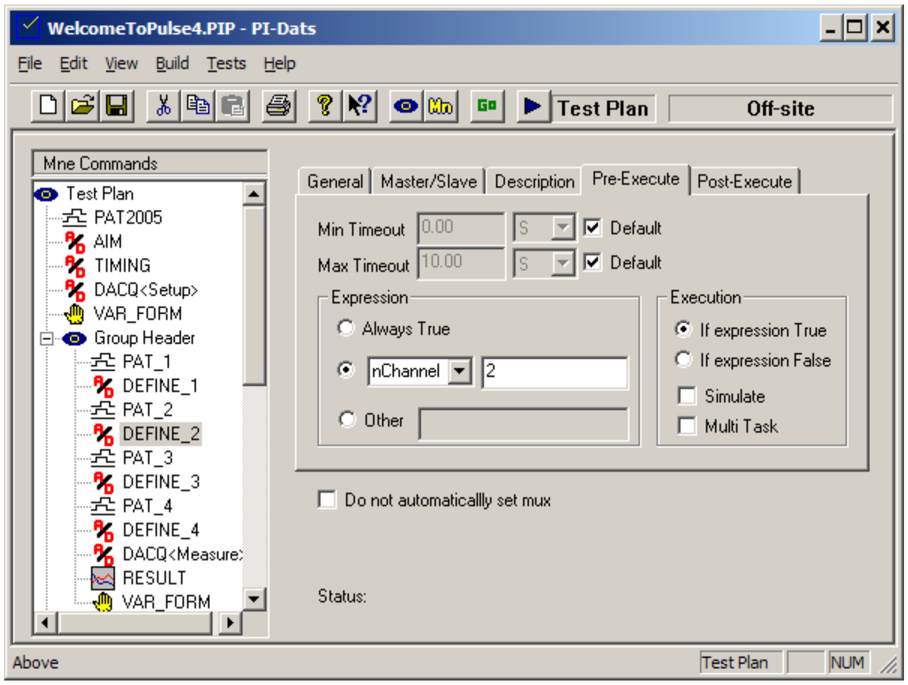
Expression: Always True [✓ nChannel 1] Other

Execution: [✓ If expression True] If expression False; Simulate [ ]; Multi Task [ ]

Do not automaticallly set mux [ ]

Figure 56: DEFINE Mnemonic, conditionally executed (1)

Pat_2 and DEFINE_2 execute only if **nChannel** equals 2, and so on.



Min Timeout 0.00 s [✓ Default]
Max Timeout 10.00 s [✓ Default]

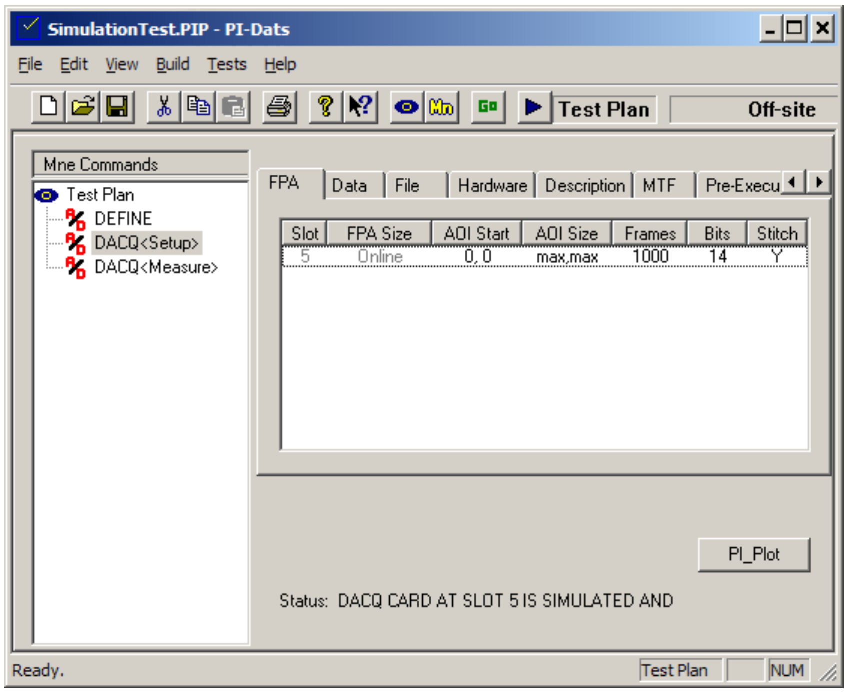
Expression: Always True [✓ nChannel 2] Other

Execution: [✓ If expression True] If expression False; Simulate [ ]; Multi Task [ ]

Do not automaticallly set mux [ ]

Figure 57: DEFINE Mnemonic, conditionally executed (2)

6.10.4. DACQ Setup Mnemonic



Slot	FPA Size	AOI Start	AOI Size	Frames	Bits	Stitch
5	Online	0, 0	max,max	1000	14	Y

Status: DACQ CARD AT SLOT 5 IS SIMULATED AND

Figure 58: DACQ Setup Mnemonic

The **DACQ Setup** mnemonic is used to set up the data acquisition cycle, much as the **DACQ** mnemonic operates in PI-Controller. In fact, when PI-DATS is switched to Controller mode, the **DACQ Setup** mnemonic becomes the **DACQ** mnemonic. Key differences between the **DACQ Setup** mnemonic in **Test Plan** vs. **Controller** mode:

- Because the number of active channels (and therefore cards and ports) and the FPA size(s) are not known at programming time, in Test Plan mode all settings for all cards and ports are editable. When the Test Plan is run, settings are applied only to the cards and ports that are active when the DACQ Setup mnemonic is executed.
- There is no **Save File** button in **Test Plan** mode, as file saving is handled by the settings in the **DACQ Setup**'s **File** property page.
- As with all hardware control mnemonics, there are **Pre-Execute** and **Post-Execute** property pages in Test Plan mode.
- There is no **Start Acq** button in **Test Plan** mode, as this function is moved to a separate mnemonic (**DACQ Measure**), described below.

6.10.4.1. Variables

The **Frames**, **AOI**, **Average**, and **FileName** settings can be controlled via PI-DATS variables. In Test Plan mode, variable names may be used instead of numeric values.

6.10.4.2. Automatic File Saving and Auto-Incrementing File Names

Although the automatic file saving and incrementing file name features are set up during execution of the **DACQ Setup** mnemonic, the saving and incrementing of the file name occur after the execution of the **DACQ Measure** mnemonic, when the data are actually captured. When the **DACQ Setup** mnemonic is executed, the filename will be set according to the value of string in the Filename box at the time the mnemonic is executed. For example, in the test plan shown in **Figure 55: DEFINE Mnemonic inside a loop**, the DACQ Setup mnemonic is in the Outer Loop, programmed with a variable Filename argument, and the DACQ Measure mnemonic is in the Inner Loop.

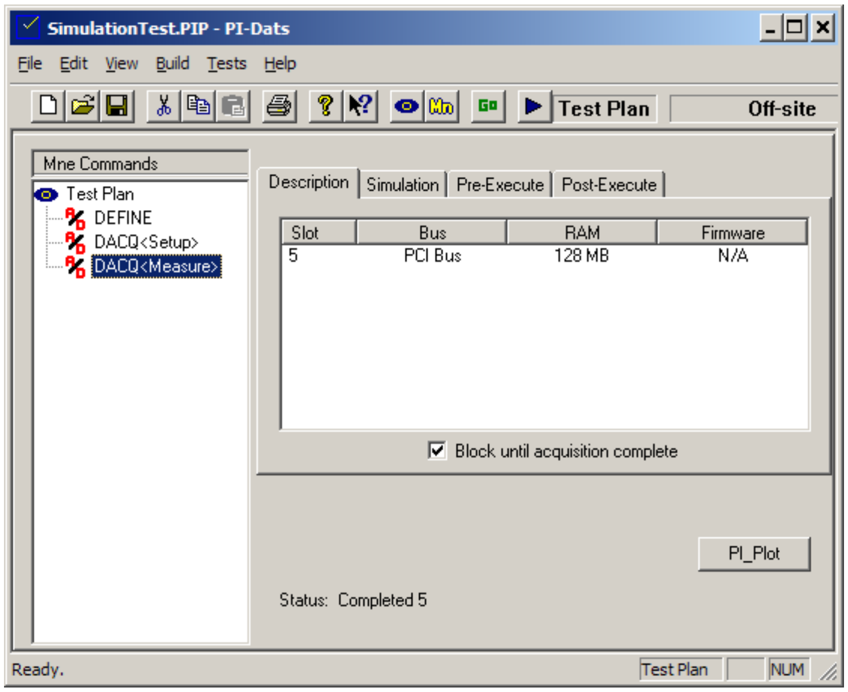
In this particular test plan, string variables are manipulated so as to cause the Filename variable to take the following values on each iteration of the outer loop:

- C:\Data\TEMP01_FRAME###.dta
- C:\Data\TEMP02_FRAME###.dta
- C:\Data\TEMP03_FRAME###.dta
- etc.

The Inner Loop is programmed to execute 3 times. Therefore the test plan will automatically generate the following data files:

- C:\Data\TEMP01_FRAME001.dta
- C:\Data\TEMP01_FRAME002.dta
- C:\Data\TEMP01_FRAME003.dta
- C:\Data\TEMP02_FRAME001.dta
- C:\Data\TEMP02_FRAME002.dta
- C:\Data\TEMP02_FRAME003.dta
- C:\Data\TEMP03_FRAME001.dta
- C:\Data\TEMP03_FRAME002.dta
- C:\Data\TEMP03_FRAME003.dta
- etc.

6.10.5. DACQ Measure Mnemonic



Slot	Bus	RAM	Firmware
5	PCI Bus	128 MB	N/A

☑ Block until acquisition complete

Status: Completed 5

Figure 59: DACQ Measure Mnemonic

The DACQ Measure mnemonic is used to trigger the data acquisition cycle, similar to clicking the **Start Acq** button on the DACQ mnemonic in PI-Controller.

When executed, the DACQ Measure mnemonic will use the settings from the last DACQ Setup that was executed. The only settable parameters for DACQ Measure is the **Block Until Complete** checkbox.

6.10.5.1. Blocking and Non-blocking Acquisitions

Depending on your application, it may or may not be desirable to block continuation of the test plan until the DACQ Measure mnemonic has finished acquiring and processing its data. By default, the DACQ Measure mnemonic will **Block until acquisition complete**.

For example, if your pattern generator or device is free running, it may be acceptable to acquire frames as soon as the DACQ Measure is ready to execute, and then move on to the next step only when the frame(s) have been collected. In this case, the default behavior is desired.

In some cases, it may be desirable to arm the data acquisition hardware, but then execute other mnemonics immediately. For example, some applications may require the pattern generator or device to be started only after the data acquisition system has been armed, so that the acquisition system collects the first frame output by the device. Some applications that may indicate this behavior include the following:

- Very long integration or readout cycles, where a free-running pattern would make the time until the next frame sync indeterminate
- External address generation for a CMOS readout, where the pattern generator must be started from a known state after the data acquisition is armed
- Applications where the timing between an external event and the frame sync must be definite
- When a free running device is not desirable, e.g. to measure accumulated dark current from a device in its quiescent state
- When changes to other instrument parameters (e.g. clock driver rise-time or bias voltage) must be made during data acquisition

In these types of applications, a **DACQ Measure** may be set with the **Block** checkbox turned off so that subsequent mnemonics can then be used to start the pattern generator or device. If the **Block** checkbox were enabled, the **DACQ Measure** would not allow the next mnemonic to execute.

In these cases, it may be necessary at some later point to block the test plan until data acquisition is complete before proceeding. For example, if a **Result** mnemonic is set up to process the results of a non-blocking DACQ Measure, a **Block** needs to be inserted above it in the test plan, to ensure that the Result mnemonic will have valid data when it executes.

The **Block** mnemonic can be set up to block execution until some or all of the DACQ cards have completed their execution.

6.10.5.2. Passing Data to the Result mmenonic

DACQ Measure also makes acquired data available to the **RESULT** mnemonic for processing. When viewed in the **Field Selection** dialog box of the **Result** mnemonic's **Insert Field** feature, **DACQ Measure** will present each connected data acquisition card in the system as an insertable array. Upon execution, only cards that are Active Masters or Active Independents will contain acquisition data. Inactive cards will have no data, nor will Slave cards. Because the state of each card is not known at program time the test plan cannot be automatically checked for correctness, and it is the users' responsibility to ensure that the **Result** mnemonic is set up correctly.

6.10.5.3. Looping

Because PI-DATS operates only on 2-D arrays, acquired data from a **DACQ Measure** are available only to **Result** mnemonics within the same loop. For example, consider the test plan shown in **Figure 58: DACQ**.

The **DACQ Measure** mnemonic will be visible to **RESULT1**, but not to **RESULT2**. In order to process 1000 frames, it must be reduced to a single array of data. It can be reduced either to a single frame of data or it may be processed into a single scalar per frame. This must be done in **RESULT1**, which can then make the data available to **RESULT2** by making it accessible as an array.

6.10.5.4. Analyzing Each Frame of Multi-Frame Acquisitions

A common application for DACQ Measure in a loop is to analyze each frame of a multi-frame acquisition. Consider again the test plan in **Figure 58: DACQ**. If we wish to analyze the data returned from the DACQ mnemonic, we must use a RESULT mnemonic within the same loop.

We only want the 1000 frames to be acquired once, so that we analyze a single contiguous data set. Therefore, we use the Pre-Execute control to execute the DACQ Measure only on the first iteration of the loop.

Then, set the loop count to 1000 iterations (either explicitly or via a variable), and use the Nth Frame reduction in RESULT1 with the loop counter variable to extract the Nth frame on the Nth loop. **See 9.2.11. Nth Frame**

RESULT1 (or additional RESULT mnemonics) can then further reduce the data by performing data operations on each frame.

7. 3rd-Party Hardware

7.1. General Notes

Pulse Instruments specifically supports several types of 3rd-party test instrumentation via GPIB.

- **DVM/DMM**: selected models from Fluke, HP/Agilent, and Keithley
- **Oscilloscope**: selected Tektronix models
- **Function Generator**: selected HP/Agilent models

For 3rd-party GPIB instrumentation that is not specifically supported by Pulse Instruments, PI-DATS provides a **Generic GPIB mnemonic** with sophisticated controls involving bi-directional data exchange.

7.2. DVM Mnemonic

This module is designed to control selected Keithley and Fluke Digital Multi-meters. The following models are supported:

- Fluke
  - Fluke 8520A
  - Fluke 8840A
  - Fluke 8860A
- HP/Agilent
  - HP 34970A (with software unit)
- Keithley
  - Keithley 2000
  - Keithley 195A
  - Keithley 196

The figures below are for the Keithley 196, but the controls for all supported DVMs (except the HP 34970A) are highly similar. Significant differences will be noted where appropriate. Please review your DVM Operators Manual before using your DVM with PI-DATS.

As with other measurement devices in PI-DATS, there are two mnemonics for DMMs—a Setup mnemonic and a Measure mnemonic. The Setup mnemonic configures the instrument's measurement parameters, and the Measure mnemonic acquires the reading from the instrument.

The reading from the instrument can be evaluated against an expected value immediately and/or it can be evaluated by passing the value to the Result mnemonic. Please see **Section 9. RESULT Mnemonic** for details on using the Result mnemonic.

7.2.1. DVM Setup Mnemonic

The Setup mnemonic is used to select the measurement mode, the measurement range, and the trigger mode for the instrument:

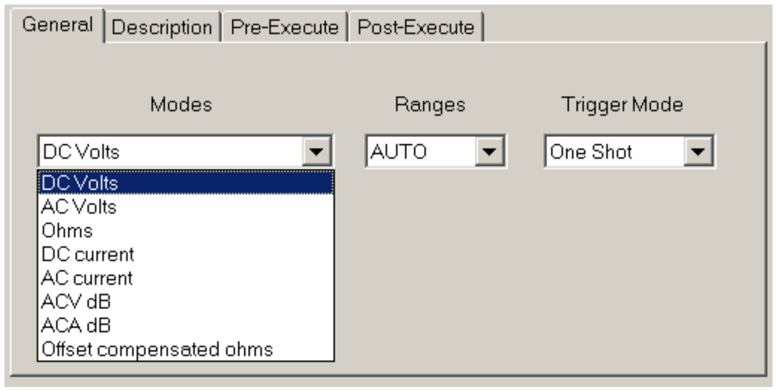


Figure 60: Keithley 196 Setup Mnemonic, General property page, Mode selection

Use the **Modes** drop-menu to select from all measurement modes supported by your DVM. **Figure 60** shows the modes supported by the Keithley 196; your DVM may have different modes available.

Next, use the **Ranges** drop menu to select a measurement range.

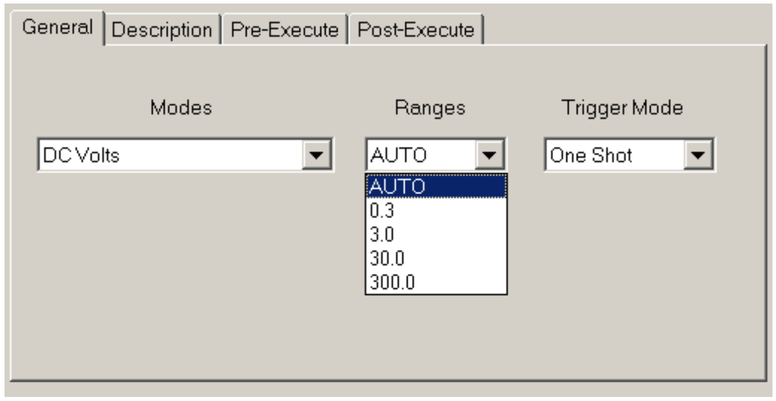
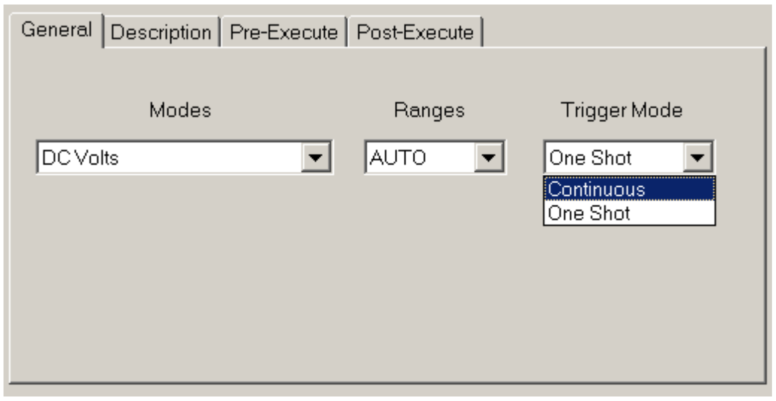


Figure 61: Keithley 196 Setup Mnemonic, General property page, Range selection

All supported DVMs offer a choice of distinct ranges or an AUTO mode. Finally, use the **Trigger Modes** drop-menu to select a triggering mode:



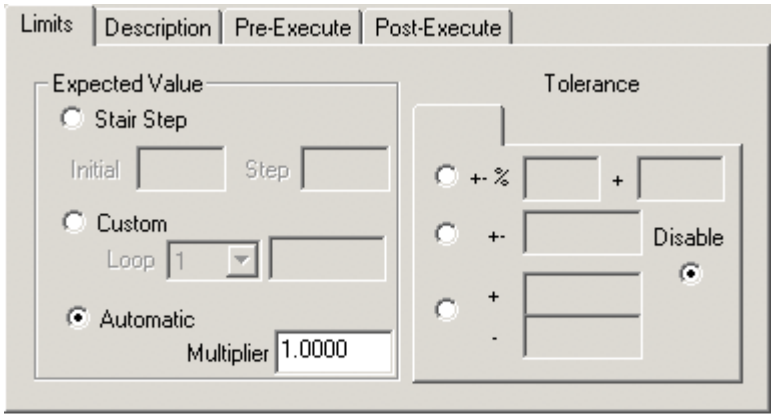
Modes: DC Volts
Ranges: AUTO
Trigger Mode: One Shot
Continuous
One Shot

Figure 62: Keithley 196 Setup Mnemonic, General property page, Trigger mode selection

Select Continuous to have the DVM run continuously, or select One Shot to have the DVM take a measurement only when strobed by the Measure mnemonic.

7.2.2. DVM Measure Mnemonic

Once you have used the Setup mnemonic to define the measurement, range, and trigger for your DVM, use the **Measure** mnemonic to make the measurement and compare it to an expected result.



Expected Value
Stair Step
Initial
Step
Custom
Loop 1
[✓ Automatic]
Multiplier 1.0000

Tolerance
+- %
+
+-
+
-
[✓ Disable]

Figure 63: Keithley 196 Measure Mnemonic, Limits property page

After every measurement the measured value can be compared against an expected value with specified tolerances. In the default case the tolerance test is set to **Disable**, so no comparison is made. To enable automatic limits testing choose one of the other three radio buttons to select a tolerance model.

7.2.2.1. Stair Step

Select the **Stair Step** radio button if you want to compare the DVM measurement against a series of incrementing values. This option is meaningful only when the **DVM Measure** mnemonic is in a loop. The **Initial** value will be used for the comparison on the first loop iteration, and the comparison value will be incremented by the **Step** value on each subsequent loop.

To decrement the value on each loop, enter a negative value for the **Step**.

7.2.2.2. Custom

Instead of using a linear stair step of values, you can also define a series of custom values. Select the **Custom** radio button, use the **Loop** drop menu to choose a loop iteration, and then enter a value in the text box for that loop. To add additional loop counts to the menu, select the **More** item. To reduce the number of loop counts in the menu, select the **Less** item.

7.2.2.3. Automatic (Not implemented)

This feature is not currently implemented. It may be implemented in a future update to PI-DATS.

7.2.2.4. Tolerance

Enter the tolerance that determines a successful measurement. There are three options for defining a measurement tolerance.

- **+-%** Plus or minus a percentage of programmed value, plus or minus a constant tolerance.
- **+-** Plus or Minus a fixed tolerance about the expected value.
- **+/-** Plus a fixed tolerance or Minus a fixed tolerance about the expected value.
- **Disable** Limits testing is disabled

Different tolerance parameters can be specified for each loop iteration in a **Stair Step** and in a **Custom** sequence of expected values. If the expected value is set for **Automatic**, then the appropriate programmed value will be used on each loop iteration.

If the measured value does not meet the tolerance criteria a failure condition will be set. This will cause a **Fail** message at the end of the Test Run. The **Fail** message can also be accessed as a field using a **Result** mnemonic.

7.2.2.5. Voltage Scaling

The DVM mnemonic will automatically scale the voltage readings into the appropriate Voltage, Voltage Sense or Current Sense measurements if the following conditions are met:

1) A DC Bias or Clock Driver channel must be selected for sensing at the time the **DVM Measure** mnemonic executes. A **Sense** mnemonic must precede the **DVM Setup** mnemonic in the test plan, and it must execute successfully.

2) The Sense output port must be **Connected** to the DVM input (either directly or through a bulkhead panel connection) in the **Connections** table in the **Advanced Hardware Configuration** or via the **Connections** table in a **Halt** mnemonic. Please see **Section 4.1.4. Connections Property Page** and **Section 8.1.3. Halt Mnemonic** for further information on specifying connections.

If both of these conditions are true, the DVM mnemonic will scale the voltage reading using the scaling factors from the DC Bias or Clock Driver model currently selected in the Sense mnemonic and the **Sense Range** selected for that channel.

If one or both of these conditions is not met, the DVM mnemonic will perform no conversion, and the returned value from the instrument will be used.

If you are using the DVM Measure to measure a voltage other than a Sense reading, then you must Disconnect the DVM from the Sense, using either the **Connections** table in the **Advanced Hardware Configuration** or via the **Connections** table in a **Halt** mnemonic.

When using the **Limits** property page to define the expected value and the tolerance against which the measurement should be compared, be sure to correct for the scaling factor if the Sense feature is being used.

7.3. Scope Mnemonic

This module is designed to control selected Tektronix Digitizing Oscilloscopes. The following models are supported:

- TDS 460A
- TEK 11402
- TDS 754D
- TDS 2024B
- TDS 3054C
- DPO 3054

The figures below are for the Tektronix TDS 460A, but the controls for all supported oscilloscopes are highly similar. Significant differences will be noted where appropriate. Please review your Oscilloscope Operators Manual before using your oscilloscopes with PI-DATS.

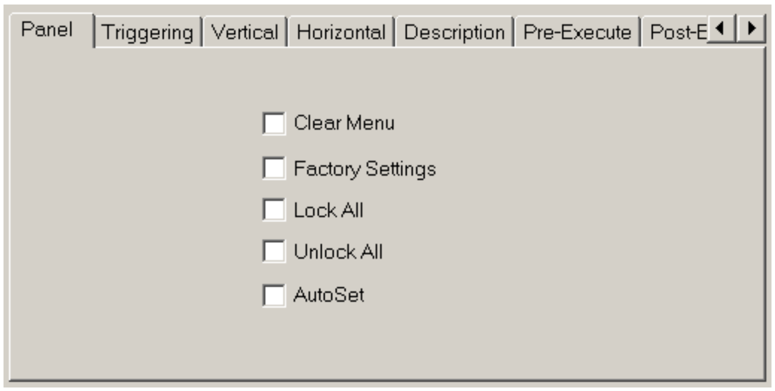
As with other measurement devices in PI-DATS, there are two mnemonics for oscilloscopes—a Setup mnemonic and a Measure mnemonic. The Setup mnemonic configures the instrument's measurement parameters, and the Measure mnemonic acquires the reading from the instrument.

The reading from the instrument can be evaluated against an expected value immediately and/or it can be evaluated by passing the value to the Result mnemonic. Please see **Section 9. RESULT Mnemonic** for details on using the Result mnemonic.

7.3.1. Scope Setup Mnemonic

The Setup mnemonic is used to select the triggering mode, the horizontal parameters, and the vertical parameters for the scope. An additional Panel property page allows PI-DATS to emulate front-panel controls.

7.3.1.1. Panel Property Page



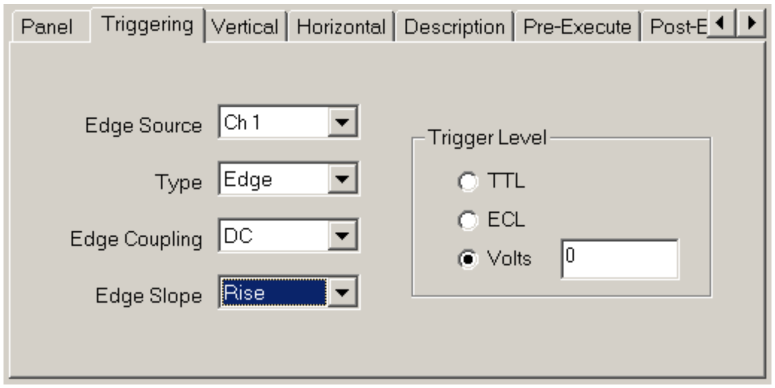
- [ ] Clear Menu
- [ ] Factory Settings
- [ ] Lock All
- [ ] Unlock All
- [ ] AutoSet

Figure 64: Tektronix TDS 460A Setup Mnemonic, Panel property page

Check a checkbox to emulate pressing of the corresponding front panel on the oscilloscope when the Setup mnemonic is executed.

Note: The **AutoSet** and the **Factory** checkboxes are currently disabled. They will be enabled in a future version of PI-DATS software.

7.3.1.2. Triggering Property Page



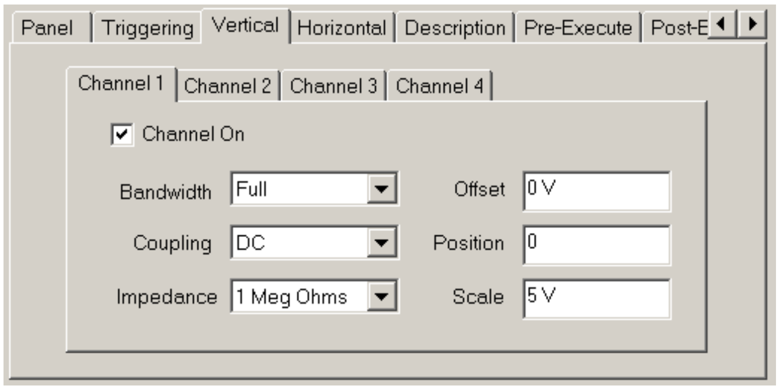
Edge Source: Ch 1
Type: Edge
Edge Coupling: DC
Edge Slope: Rise

Trigger Level
TTL
ECL
[✓ Volts] 0

Figure 65: Tektronix TDS 460A Setup Mnemonic, Triggering property page

The **Triggering** property page duplicates most of the functions of the Triggering menu shown on the TDS 460A when the Trigger Menu button is pressed. You can specify the edge source, edge type, edge coupling, edge slope, and voltage level on which to trigger.

7.3.1.3. Vertical Property Page



☑ Channel On

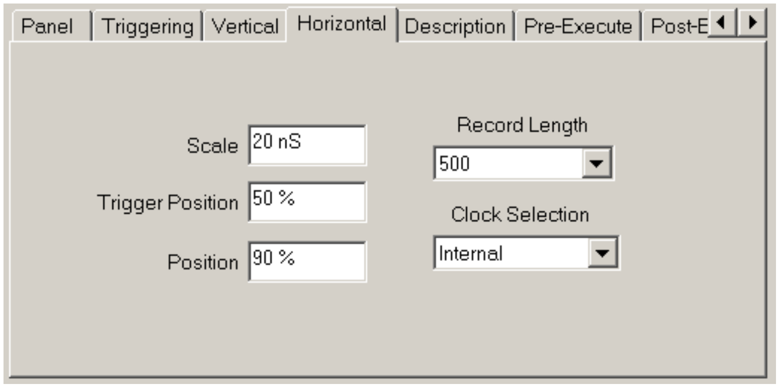
Bandwidth: Full
Coupling: DC
Impedance: 1 Meg Ohms
Offset: 0 V
Position: 0
Scale: 5 V

Figure 66: Tektronix TDS 460A Setup Mnemonic, Vertical property page

The **Vertical** property page duplicates most of the functions of the Vertical menu shown on the TDS 460A when the Vertical Menu button is pressed. You can specify the bandwidth, coupling, input impedance, DC offset, vertical position, and the vertical scale for each channel.

Use the **Channel 1**, **Channel 2**, **Channel 3**, and **Channel 4** sub-property page tabs to cycle through the settings for each channel. Check the **Channel On** checkbox to make the current channel active.

7.3.1.4. Horizontal Property Page



Scale: 20 nS
Trigger Position: 50 %
Position: 90 %
Record Length: 500
Clock Selection: Internal

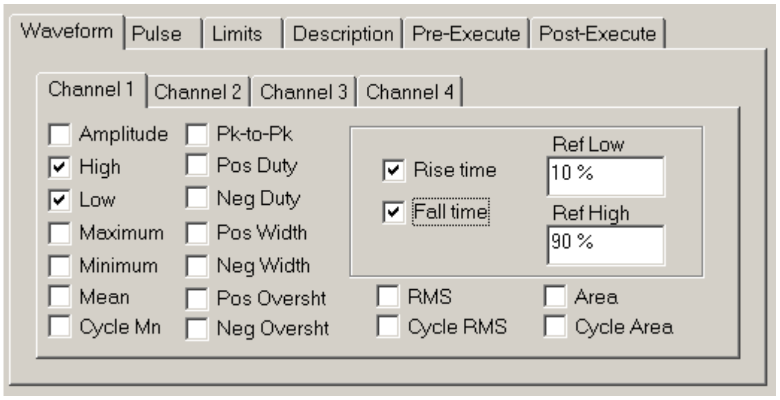
Figure 67: Tektronix TDS 460A Setup Mnemonic, Horizontal property page

The **Horizontal** property page duplicates most of the functions of the Horizontal menu shown on the TDS 460A when the Horizontal Menu button is pressed. It specifies the horizontal scale, trigger position (in percent), vertical position (in percent), record length to capture, and the clock selection.

7.3.2. Scope Measure Mnemonic

Once you have used the Setup mnemonic to define the measurement, range, and trigger for your scope, use the Measure mnemonic to make measurements and compare them to expected results.

7.3.2.1. Waveform Measurements

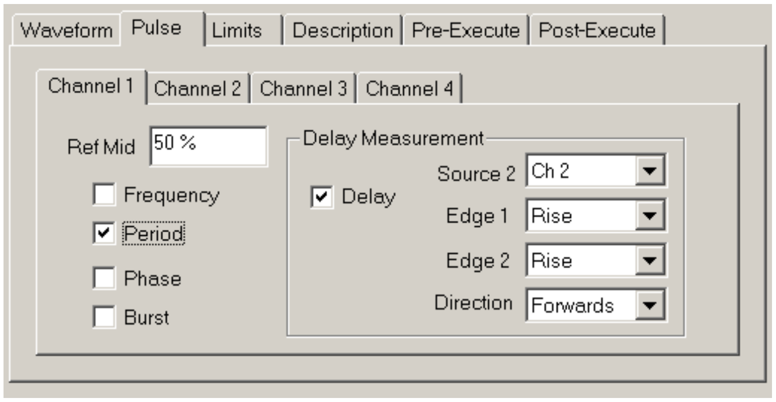


- [ ] Amplitude
- [✓] High
- [✓] Low
- [ ] Maximum
- [ ] Minimum
- [ ] Mean
- [ ] Cycle Mn
- [ ] Pk-to-Pk
- [ ] Pos Duty
- [ ] Neg Duty
- [ ] Pos Width
- [ ] Neg Width
- [ ] Pos Oversht
- [ ] Neg Oversht
- [✓] Rise time
- [✓] Fall time
- Ref Low: 10 %
- Ref High: 90 %
- [ ] RMS
- [ ] Cycle RMS
- [ ] Area
- [ ] Cycle Area

Figure 68: Tektronix TDS 460A Measure Mnemonic, Waveform property page

Use the **Waveform** property page to select waveform measurements to read for each channel. Check the box for each desired reading, and enter any required parameters in the text entry boxes. Use the **Channel 1**, **Channel 2**, **Channel 3**, and **Channel 4** sub-property page tabs to cycle through the settings for each channel

7.3.2.2. Pulse Measurements

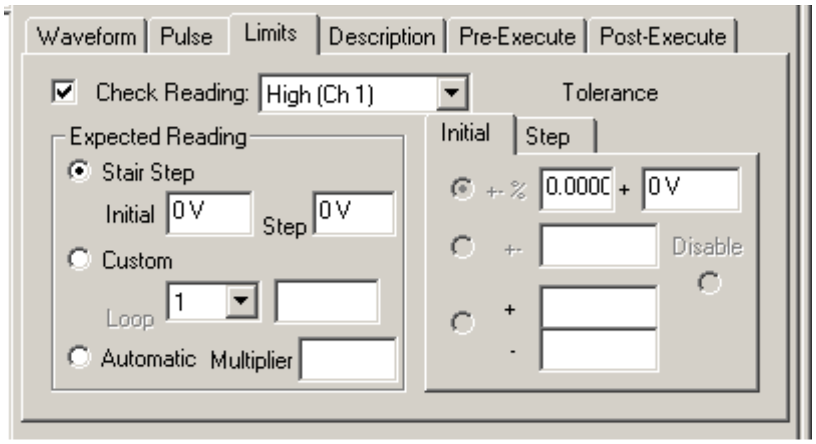


- Ref Mid: 50 %
- [ ] Frequency
- [✓] Period
- [ ] Phase
- [ ] Burst
- [✓] Delay
- Source 2: Ch 2
- Edge 1: Rise
- Edge 2: Rise
- Direction: Forwards

Figure 69: Tektronix TDS 460A Measure Mnemonic, Pulse property page

Use the **Pulse** property page to select pulse measurements to read for each channel. Check the box for each desired reading, and enter or select any required parameters in the text entry boxes and drop menus. Use the **Channel 1**, **Channel 2**, **Channel 3**, and **Channel 4** sub-property page tabs to cycle through the settings for each channel

7.3.3. Scope Measurement, Limits Property Page



Waveform | Pulse | Limits | Description | Pre-Execute | Post-Execute

[✓] Check Reading: High (Ch 1)

Expected Reading
- [● Stair Step] Initial: 0 V Step: 0 V
- Custom Loop: 1
- Automatic Multiplier:

Tolerance (Initial | Step)
- [● +- %] 0.0000 + 0 V
- +-
- Disable
- + / -

Figure 70: Tektronix TDS 460A Measure Mnemonic, Limits property page

The **Limits** property page can be used to define the expected value and the tolerance against which the measurement should be compared. Use of the **Limits** comparison feature can be turned on or off as needed.

To enable the **Limits** feature for a particular **Scope Measure** mnemonic, check the **Check Readings** box. This will be enabled only if there is at least one **Waveform** or **Pulse Measurement** selected for this **Scope Measure** mnemonic.

Each measurement selected from the **Waveform** and **Pulse** property pages will be available from the **Reading** drop-menu. Each measured reading can be assigned its own measurement limits.

After every measurement the measured value will be compared against the specified tolerances.

7.3.3.1. Stair Step

Select the **Stair Step** radio button if you want to compare the DVM measurement against a series of incrementing values. This option is meaningful only when the DVM Measure mnemonic is in a loop. The **Initial** value will be used for the comparison on the first loop iteration, and the comparison value will be incremented by the **Step** value on each subsequent loop.

To decrement the value on each loop, enter a negative value for the **Step**.

7.3.3.2. Custom

Instead of using a linear stair step of values, you can also define a series of custom values. Select the **Custom** radio button and use the **Loop** drop menu to choose a loop iteration, and then enter a value in the text box for that loop. To add additional loop counts to the menu, select the **More** item. To reduce the number of loop counts in the menu, select the **Less** item.

7.3.3.3. Automatic (Not implemented)

This feature is not currently implemented. It may be implemented in a future update to PI-DATS.

7.3.3.4. Tolerance

Enter the tolerance that determines a successful measurement. There are four options for defining a measurement tolerance.

- **+-%** Plus or minus a percentage of programmed value, plus or minus a constant tolerance.
- **+-** Plus or Minus a fixed tolerance about the expected value.
- **+/-** Plus a fixed tolerance or Minus a fixed tolerance about the expected value.
- **Disable** Choosing Disable turns off **Limits** checking for this particular measurement

Different tolerance parameters can be specified for each loop iteration in a **Stair Step** and in a **Custom** sequence of expected values. If the expected value is set for **Automatic**, then the appropriate programmed value will be used on each loop iteration.

If the measured value does not meet the tolerance criteria a failure condition will be set. This will cause a **Fail** message at the end of the Test Run. The **Fail** message can also be accessed as a field using a **Result** mnemonic.

In the default case the tolerance is **Disable**.

7.4. Blackbody Mnemonic

This module is designed to control selected Blackbody Controllers. The following models are supported:

- SBIR 2000
- SBIR Infinity
- SBIR 4000
- Electro Optical Industries 28 Series
- CI System SR-200
- CI System SR-800

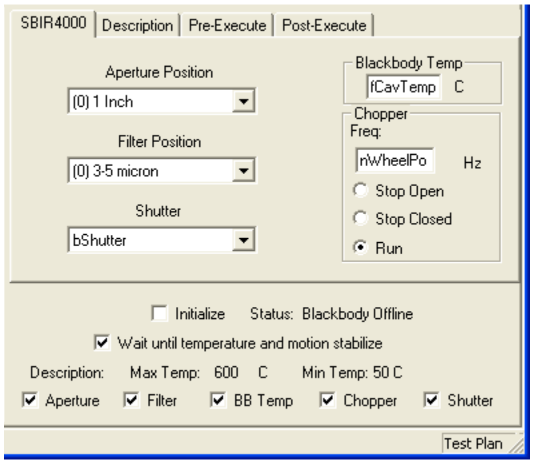
The figures below are for the SBIR 4000, but the controls for all supported blackbody controllers are highly similar. Significant differences will be noted where appropriate. Please review your Blackbody Controller Operators Manual from the manufacturer before using your blackbodies with PI-DATS.

The instrument settings and temperature readback can be passed into the Result mnemonic for use in your test reports. Please see **Section 9. RESULT Mnemonic** for details on using the Result mnemonic.

7.4.1. Blackbody Mnemonic

The blackbody mnemonic is used to select the temperature, temperature mode, and any filter, target, aperture, chopper or shutter settings available on your particular model of blackbody.

7.4.1.1. General Controls

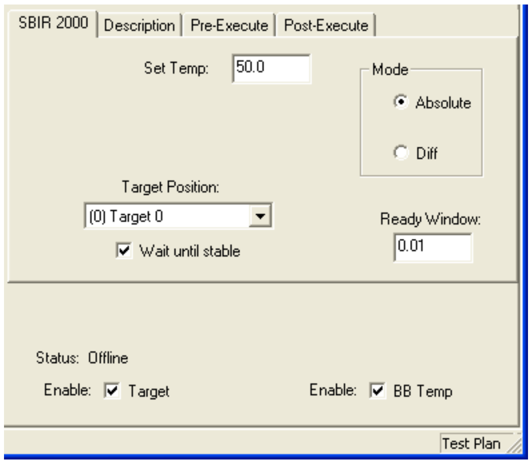

SBIR4000 | Description | Pre-Execute | Post-Execute

Aperture Position: (0) 1 Inch
Filter Position: (0) 3-5 micron
Shutter: bShutter
Blackbody Temp: fCavTemp C
Chopper Freq: nWheelPo Hz
Stop Open / Stop Closed / [✓ Run]

☐ Initialize   Status: Blackbody Offline
☑ Wait until temperature and motion stabilize
Description: Max Temp: 600 C   Min Temp: 50 C
☑ Aperture ☑ Filter ☑ BB Temp ☑ Chopper ☑ Shutter

Test Plan

Figure 71: SBIR 4000 Mnemonic, General property page


SBIR 2000 | Description | Pre-Execute | Post-Execute

Set Temp: 50.0
Mode: [✓ Absolute] / Diff
Target Position: (0) Target 0
☑ Wait until stable
Ready Window: 0.01

Status: Offline
Enable: ☑ Target   Enable: ☑ BB Temp

Test Plan

Figure 72: SBIR 2000 Mnemonic, General property page

The blackbody **Temp** may be set using a value or a PI-DATS variable. Temperature variables may be integer or real values. If your blackbody supports differential mode, the mode may be selected via the **Mode** radio button. In differential mode, the temperature setting will the blackbody setpoint relative to the ambient/background probe temperature.

If your blackbody has a programmable **Ready Window** (temperature tolerance) this may be set with a value or a real variable.

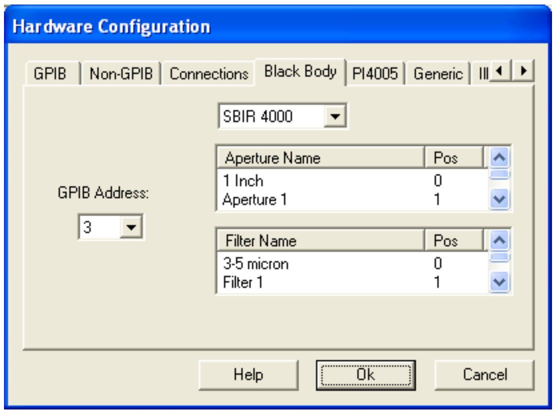
If your blackbody has a **Shutter**, it may be set to **Open** or **Closed**, or it may be set with a PI-DATS Boolean variable, which will appear at the bottom of the drop-down menu. The Booelan values corresponding to Open or Closed depend on the blackbody manufacturer and model.

If you blackbody controller has a programmable chopper, the chopper may be programmed to **Stop Open**, **Stop Closed**, or to **Run** at a programmed frequency. The frequency may be programmed by a value or a by a PI-DATS integer variable, in Hz.

Wheel positions for aperture, target, or filter wheels may be set via the drop-down menu choices or via an integer variable. All defined integer variables will appear at the bottom of the drop-down menu.

By default wheel positions are named Target 0, Target 1, Target n, etc., but these may be edited to display the actual name, size, or value of the wheel position.

To apply a custom name to a wheel position, open the **Edit: Advanced Hardware Configuration** dialog box, **Blackbody** property page, then select the manufacturer and model of the blackbody from the top drop-down menu and the corresponding GPIB address from the side drop-down menu. Once the desired blackbody unit/address is displayed, the wheel names can be edited:



SBIR 4000

GPIB Address: 3

Aperture Name	Pos
1 Inch	0
Aperture 1	1

Filter Name	Pos
3-5 micron	0
Filter 1	1

Figure 73: Blackbody hardware configuration page

7.4.1.2. Selective Control

By default, all programmable parameters are programmed to the blackbody controller every time **Write** is clicked in **Controller** mode or any time the mnemonic is executed in **Test Plan** mode.

In some applications it may be desirable to program only a subset of features at one time. For example a test plan might set the temperature to an optimized value by iterating in a temperature loop, but then iterate through the wheel positions after the temperature loop has finished.

Since the desired temperature value for the wheel loop is not known at programming time, the mnemonic can be set up to program only the wheel position, but the leave the temperature at its last programmed value. This can be accomplished by un-checking any features that you do not want to program for this instance of the mnemonic.

The **Enable** checkboxes can be used to program all, some, or none of the parameters when the mnemonic executes.

7.4.1.3. Test Plan Flow Control

By default, PI-DATS will execute the Blackbody mnemonic and then wait until all programmed settings have reached their programmed values before proceeding to the next mnemonic.

Because blackbody settling times can be very long (more than an hour for large delta-T values on a cavity model), it may be desirable to program the device and then move on to the next mnemonic. This can be accomplished by un-checking the **Wait Until Stable/Stabilized** checkbox.

7.5. Motion Controller Mnemonic

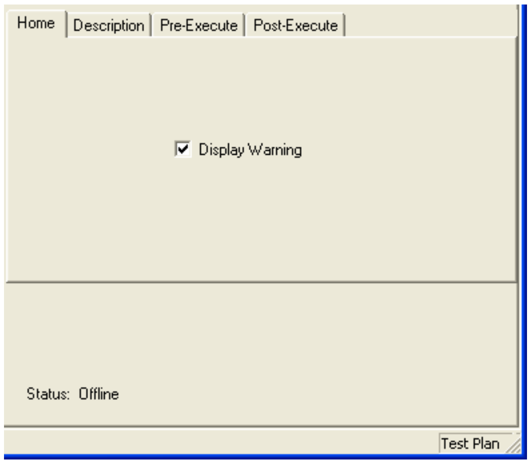
This module is designed to control selected Motion Controllers. The following models are supported:

- Newport ESP 300

The instrument settings and position readback can be passed into the Result mnemonic for use in your test reports. Please see **Section 9. RESULT Mnemonic** for details on using the Result mnemonic.

7.5.1. Home Mnemonic

The Motion Controller **Home** mnemonic is used to home the controller. Motion controllers should be homed at the beginning of each test session or after power-cycling the controller to ensure accurate positioning.



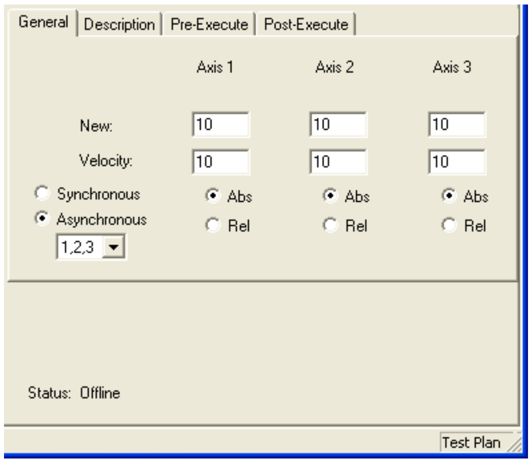
[✓] Display Warning

Status: Offline

Figure 74: ESP 300 Home Mnemonic

In Test Plan mode, the **Home** mnemonic has a checkbox to display an optional warning to the operator. If checked, the test plan will Halt and request instruction to **Home**, **Don't Home** or **Abort**.

If the operator clicks **Home**, the controller will attempt its built-in homing procedure. If the operator clicks **Don’t Home**, PI-DATS will skip the homing procedure and proceed to the next mnemonic. If the operator clicks **Abort**, the test plan will stop.



	Axis 1	Axis 2	Axis 3
New:	10	10	10
Velocity:	10	10	10
Synchronous [✓ Asynchronous] 1,2,3	[✓ Abs] Rel	[✓ Abs] Rel	[✓ Abs] Rel

Status: Offline

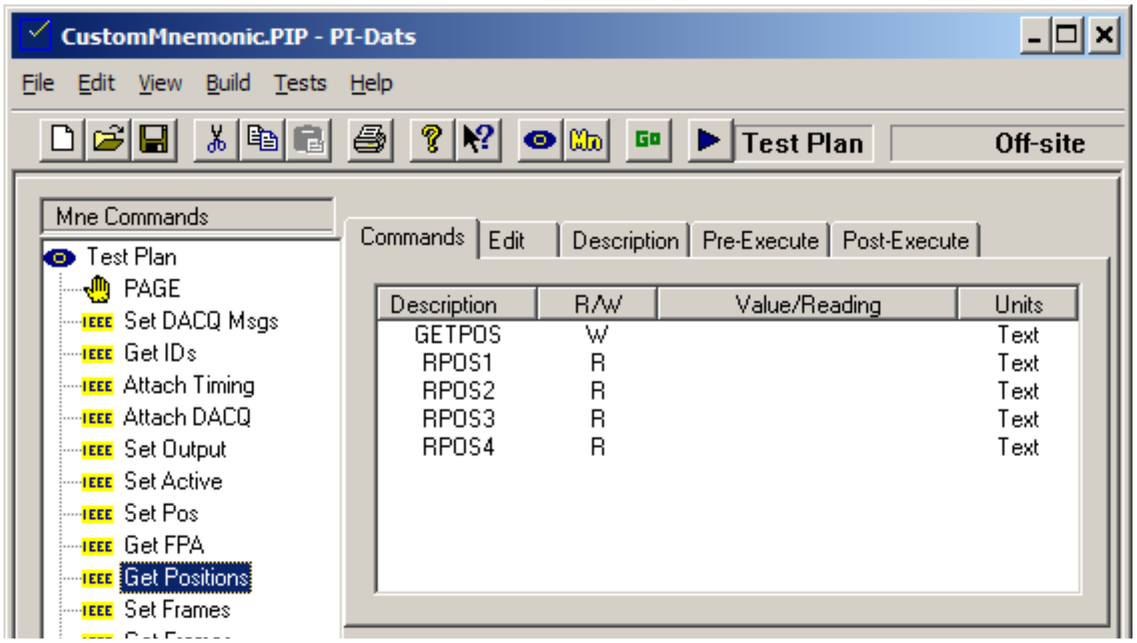
Figure 75: ESP 300 Mnemonic

Each axis of the motion controller can be programmed for absolute or relative motion, and the motion of the 3 axes can be synchronous or asynchronous.

Positions and velocities can be programmed by value or by integer variable. Consult your motion controller operator manual for appropriate velocity and position values.

When programmed for **Synchronous** motion, all axes will begin motion simultaneously, and each axis will continue motion until it has reached its programmed position. When programmed for **Asynchronous** motion, each axis will move and finish in the programmed order before the next axis starts. The **Asynchronous** drop-down menu can be used to select the desired order of movement.

7.6. Generic mnemonic



Description	R/W	Value/Reading	Units
GETPOS	W		Text
RPOS1	R		Text
RPOS2	R		Text
RPOS3	R		Text
RPOS4	R		Text

Figure 76: Generic mnemonic

The **Generic** mnemonic can be defined by the user to provide interactive control over GPIB instrumentation not specifically supported by Pulse Instruments. The property pages and command strings for a **Generic** mnemonic are defined by the user, and they can use test plan variables like other mnemonics in PI--DATS. **Generic** mnemonics can query GPIB instrumentation and pass data to PI-DATS variables. **Generic** GPIB mnemonics can be constructed to control either a limited subset or a large complement of an instrument's functionality, depending on your needs.

The behavior of a **Generic** mnemonic is defined by a text script. Each line of the script contains:

- A text label for user identification of the command string(s)
- A short name for internal PI-DATS identification
- Commands to **Write** to the instrument or to **Read** from the instrument
- Additional parameters to modify the **Write** string or to parse the **Read** string

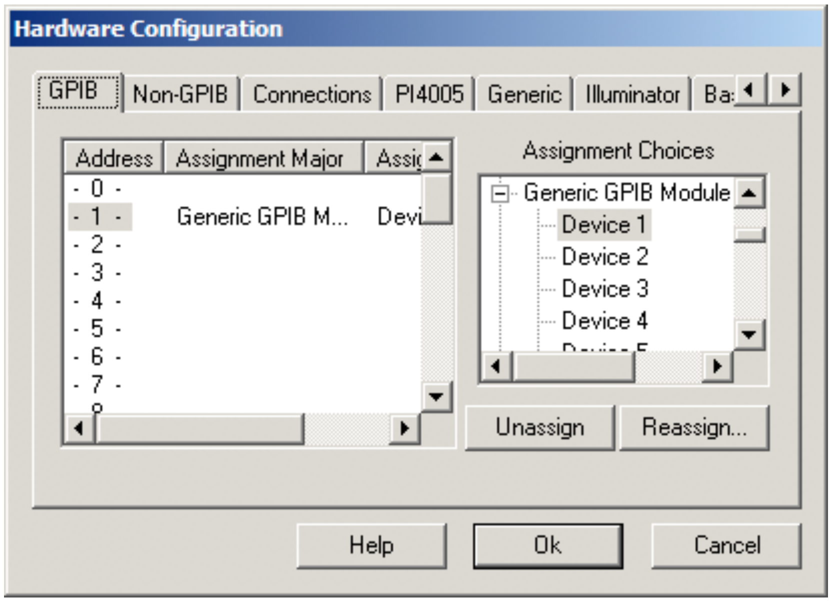
The script is stored in three ways in increasing order of precedence:

- Default scripts can be stored in a set of INI files. These INI files can then be read into the **Generic** definitions in the **Hardware Configuration**.

- The definitions in the **Hardware Configuration** can then be further edited by the user, if desired. When a new **Generic** mnemonic is inserted into a test plan, its script is populated from this definition.

- Once a **Generic** mnemonic has been inserted into a PIP file its script is self-contained. It can be edited directly in the **Edit** property page of the mnemonic, and it is saved in the test plan. It is now independent of the default script in the hardware configuration file. Different instances of the same instrument's mnemonic can have different scripts.

To declare the instrument, select it in the **Assignment Choices**, select its GPIB address, and click **Assign**. Each defined instrument may be assigned to one or more GPIB addresses, but each GPIB address may only be assigned to one instrument definition.



Address	Assignment Major	Assi
- 0 -		
- 1 -	Generic GPIB M...	Devi
- 2 -		
- 3 -		
- 4 -		
- 5 -		
- 6 -		
- 7 -		

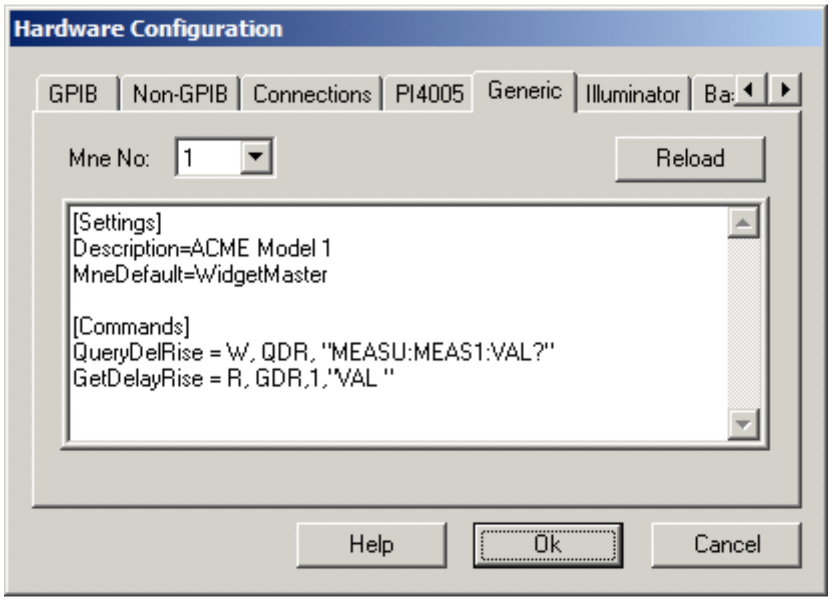
Figure 77: Assigning a Generic GPIB Device

If no custom device definitions have been loaded the choices will be listed as **Device 1**, **Device 2**, etc., and the mnemonic will not have a default script or any default behaviors.

Such a mnemonic can still be used, by editing its definition manually in each instance of the mnemonic in the test plan, but in most cases it is desirable to have a default script. There also are features of the **Generic** mnemonic, such as EnableRemote() and Limits, that are available only if a definition has been loaded in the **Hardware Configuration**.

7.6.1. Default Script

Default scripts for custom instruments are defined in the **Generic** property page of the **Hardware Configuration** dialog box. The contents of these scripts can be imported from INI files and/or manually edited.



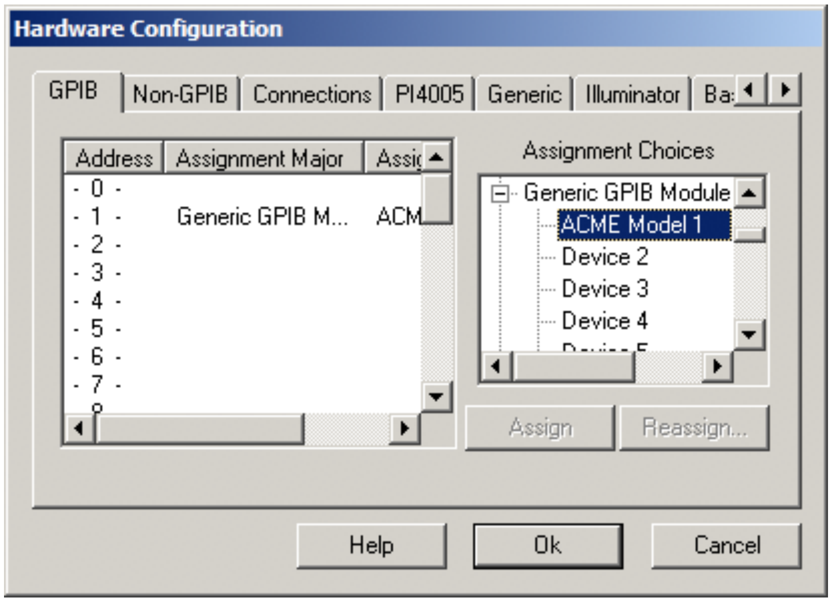
```
[Settings]
Description=ACME Model 1
MneDefault=WidgetMaster

[Commands]
QueryDelRise = W, QDR, "MEASU:MEAS1:VAL?"
GetDelayRise = R, GDR,1,"VAL "
```

Figure 78: Default Script for Device 1

Select the definition to edit by selecting it from the **Device No:** drop-menu. To reload the definitions from INI files click the **Reload** button. Please note that existing instrument definitions will be over-written by the correspondingly-numbered INI files.

Defined **Generic** mnemonics will appear in the **Assignment Choices** pane of the **GPIB** property page, in order as defined by their **Device No**. arguments and with names defined by their script contents. If no **Description** exists (see below) the instrument will be listed simply as **Device 1**, **Device 2**, etc.



Address	Assignment Major	Assig
- 0 -		
- 1 -	Generic GPIB M...	ACM
- 2 -		
- 3 -		
- 4 -		
- 5 -		
- 6 -		
- 7 -		

Figure 79: Assigning an Instrument with a Description

To create re-usable defaults that can shared among different systems use any text editor to create an INI file named **PIUserXX.ini**, where XX can be any two-digit number from 00 to 31 (e.g. PIUser00.ini to PIUser31.ini). Generic mnemonic INI files must be placed in the same directory as the PIIEEE.ocx file, typically C:\Program Files\Pulse20. The number in the filename (01 to 31) determines the order in which the generic mnemonics are presented in the **Hardware Configuration** dialog box.

Each INI file must contain a set of scripts that define the appearance and operation of its corresponding **Generic** mnemonic. There are two types of script commands—**Settings** and **Commands**.

7.6.2. Settings Script

The **[Settings]** section of the INI file or the Default script in the Hardware Configuration contains the following parameters:

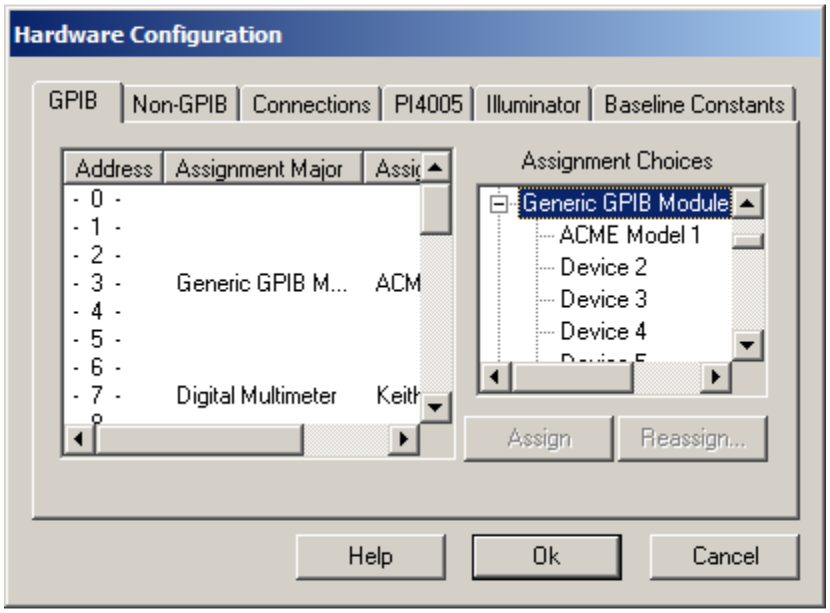
- The name and default mnemonic label of instrument to be controlled.
- Whether to send a GPIB **EnableRemote()** command upon initialization
- Whether to display a **Limits** page with each mnemonic instance
- ***[function to be enabled in a future release]*** An external DLL or Automation interface to call instead of a GPIB instrument

Example:

```
[Settings]
Description=ACME Model 1
MneDefault=WidgetMaster
EnableRemote=Y
Limits=N
```

7.6.2.1. Description (optional)

This **Description** entry defines the text label displayed in the **GPIB** assignment property page in the **Hardware Configuration**. The example **Settings** script for **Device No. 1** above will display as follows in the **GPIB** assignment window:


Hardware Configuration

GPIB | Non-GPIB | Connections | PI4005 | Illuminator | Baseline Constants

Address	Assignment Major	Assig
- 0 -		
- 1 -		
- 2 -		
- 3 -	Generic GPIB M...	ACM
- 4 -		
- 5 -		
- 6 -		
- 7 -	Digital Multimeter	Keith

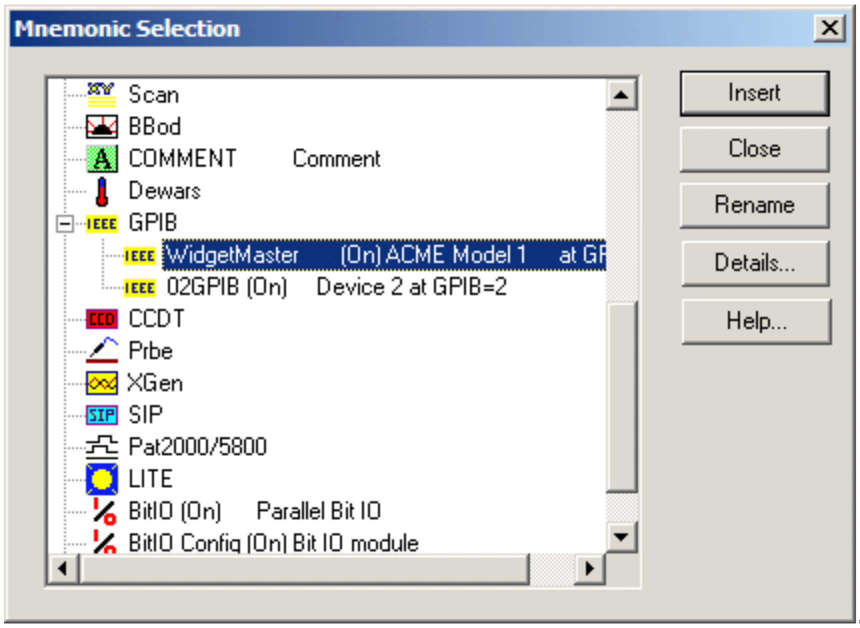
Assignment Choices
- Generic GPIB Module
  - ACME Model 1
  - Device 2
  - Device 3
  - Device 4

Figure 80: Description parameter

If **Description** is omitted the instrument will be displayed as **Device 1**, **Device 2**, etc.

7.6.2.2. MneDefault (optional)

The **MneDefault** entry defines the default mnemonic name displayed in the Mnemonic Selection dialog box:


Mnemonic Selection

- Scan
- BBod
- COMMENT Comment
- Dewars
- GPIB
  - WidgetMaster (On) ACME Model 1 at GP
  - 02GPIB (On) Device 2 at GPIB=2
- CCDT
- Prbe
- XGen
- SIP
- Pat2000/5800
- LITE
- BitIO (On) Parallel Bit IO
- BitIO Config (On) Bit IO module

.

Figure 81: MneDefault Parameter

If the **MneDefault** is omitted the instrument will be described as **01GPIB**, **02GPIB**, etc., in the **Mnemonic Selection** window.

7.6.2.3. EnableRemote (optional)

If **EnableRemote** is entered as “**Y**” the instrument will issue a GPIB **EnableRemote()** command upon initialization when PI-DATS is launched or when the **Hardware Configuration** dialog box is closed. If this line is omitted no **EnableRemote()** will be issued. Many GPIB instruments require a **EnableRemote()** before they will respond to any commands.

7.6.3. Commands Script

The **Commands** script defines the default set of commands available to each instance of the mnemonic when it is inserted into the test plan. There are two types of commands: **Write** commands and **Read** commands.

Write commands may be defined several different ways:

- Fixed strings, e.g. “SET TEMP 30”, that are output exactly as entered.
- Strings with an editable parameter, e.g. “SET TEMP ***x***” where ***x*** can be specified by the test plan operator or with a test plan variable.
- Strings with multiple editable parameters, e.g. “OUTPUT ***x***, ***y***” where both ***x*** and ***y*** can be specified by the test plan operator or with test plan variables.
- Variable strings, where the entire command string can be specified by the test plan operator or a test plan variable. Note that this is just a special case of the “editable parameter” case listed above.

Read commands read a single string from the specified GPIB instrument address into PI-DATS. This string can then be parsed and assigned to one or more PI-DATS objects and/or variables. For example:

- A returned string such as “NDCV4.28” can be assigned to a string variable.
- “NDCV4.28” can be parsed to assign only “4.28” to a real number variable.
- “POSITION 255, 0, -1, 1” can be parsed and the four values 255, 0, -1, and 1 assigned to four separate integer or real number variables.

7.6.3.1. Definition of Write Commands

The **Write** command format is:

LongDesc= W, [ShortDesc], StringFormat, [UserEntry] [, &]

LongDesc and **ShortDesc** are limited to alphanumeric characters and the underscore. Space characters are not permitted. The **ShortDesc** must be 3 characters or fewer. All **LongDesc** and **ShortDesc** entries within one instance of a Generic mnemonic must be unique (except when the **ShortDesc** is blank). If the **ShortDesc** is blank, the command’s execution parameters will not be available to the **Result** mnemonic or in the **Post-Execute** property page. This may be acceptable for command strings that do not contain variable parameter values, such as “START”, “ENABLE”, etc.

StringFormat contains the command to be sent to the instrument. If **StringFormat** contains the optional "%%" placeholder, it will be replaced at runtime with the **UserEntry** string. For example the **StringFormat** "SET TEMP %%" and the **UserEntry** "30.4" will resolve to the command string "SET TEMP 30.4" at runtime.

Alternatively, the **StringFormat** could contain only "%%" and the **UserEntry** could be the entire string "SET TEMP 30.4", which would result in the same output to the instrument. If **StringFormat** does not contain "%%" the **UserEntry** is ignored and can be omitted. **UserEntry** is required if the "%%" is present in the **StringFormat**. "%%" can occur anywhere in the **StringFormat** argument, but can occur only once. Use the continuation symbol (see below) for specifying multiple **UserEntry** items in a single GPIB command string.

& is an optional continuation symbol indicating that the command string will be concatenated with the command string defined on the next line(s) before it is output to the instrument.

All commas are required except for the final comma when the "&" is not used.

Commas, quotation marks, and non-printable control characters in the actual command string must be escaped with a backslash "\" character. Use "\r" for a carriage return and use "\n" for the newline character. For example to send the following literal command to a Tektronix sampling scope:

```
:MEASU:SAVE:"\\Fileserver\ScopeCaptures\Capture1.bmp"
```

it must be entered as:

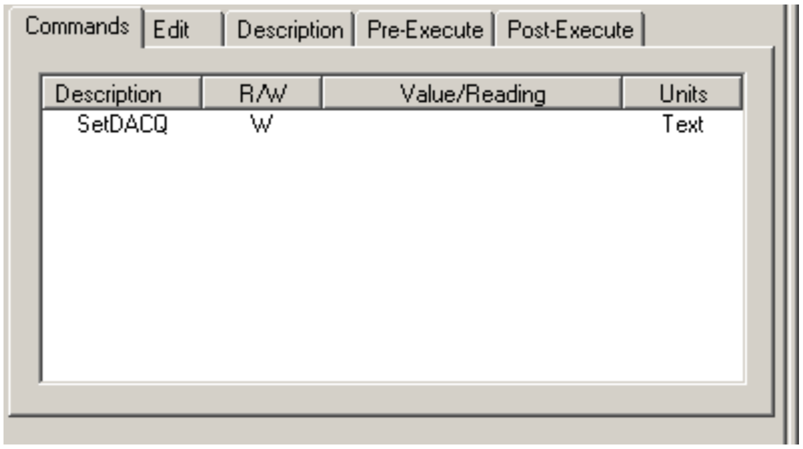
```
Location=W, LOC, ":MEASU:SAVE:\"\\Fileserver\ScopeCaptures\Capture1.bmp\"\r\n"
```

7.6.3.2. Write Command Example 1

The entry:

```
SetDACQ = W, DAQ, "DACQMSGS"
```

displays as:



Description	R/W	Value/Reading	Units
SetDACQ	W		Text

Figure 82: Write Example 1

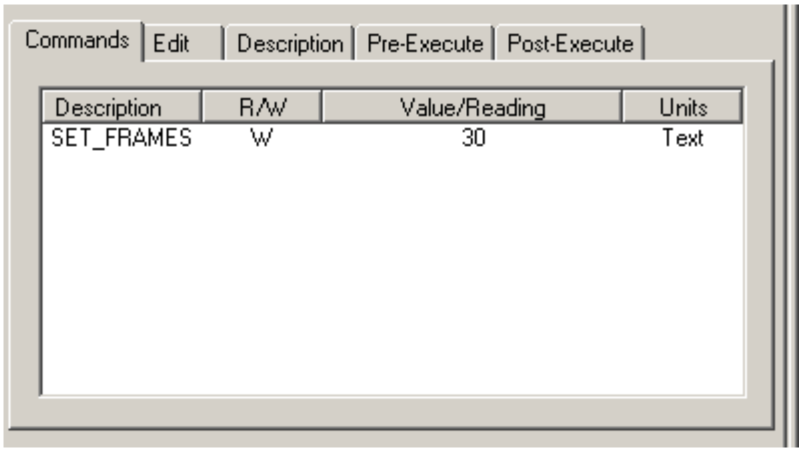
and sends the command "DACQMSGS". The only operator-editable setting is the "**W**", which may also be set to "**Off**" to disable this line.

7.6.3.3. Write Command Example 2

The entry:

```
SET_FRAMES = W, FRM, "FRAME COUNT %%", "30"
```

displays as:



Description	R/W	Value/Reading	Units
SET_FRAMES	W	30	Text

Figure 83: Write Example 2

and sends the command "FRAME COUNT 30". The value “30” may be edited by the test plan operator on the **Commands** property page, or it may specified with a test plan variable.

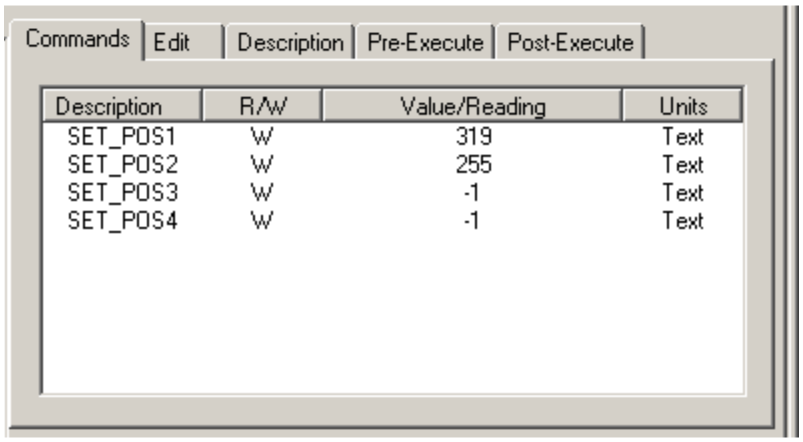
7.6.3.4. Write Command Example 3

The following is an example of a **Write** command that takes 4 parameters:

The entry:

```
SET_POS1 = W, SP1, "POSITION %%, ", "319", &
SET_POS2 = W, SP2, "%%, ", "255", &
SET_POS3 = W, SP3, "%%, ", "-1", &
SET_POS4 = W, SP4, "%%", "-1"
```

Displays as:



Description	R/W	Value/Reading	Units
SET_POS1	W	319	Text
SET_POS2	W	255	Text
SET_POS3	W	-1	Text
SET_POS4	W	-1	Text

Figure 84: Write Example 3

and sends the command "POSITION 319, 255, -1, -1" by default. Some or all of the four arguments can be over-ridden by the test plan operator or replaced by test plan variables.

7.6.3.5. Definition of Read Commands

A **Read** commands queries the instrument for a **returned string**, then optionally parses it into a **retained string** or strings. The **Read** command format is:

LongDesc = R, ShortDesc, [StartPos[, StartString[, EndPos[, EndString] [,&]]]]

where

StartPos: A one-based number that specifies the minimum position number of the character, reading from the left, of the returned string to retain (or where to start searching for the **StartString**). If the returmed string is "NDCV6.0", and the **StartPos** is 5 (with no **StartString**), the retained string will be "6.0"

StartString: A string to search for in the returned string. The search starts at the **StartPos.** This string is optional; if it is empty it is not used. The retained string starts at the first character following **StartString**, which must be in quotes. A **StartString** of "NDCV" with a returned string of "XXXXXNDCV6.0" would result in the retained string "6.0".

EndPos: Characters at this position and beyond are ignored, unless there is an **EndString**. If the returned string is "XXXXXNDCV6.0C" the **StartString** “NDCV, and the **EndPos** 13, the retained string will be "6.0".

EndString: A string to search for from the **EndPos** onward. The retained string ends before **EndString**, which must be in quotes. If the returned string were "XXXXXNDCV6.0e-1C", the **StartString** “NDCV”, and the **EndString** "C", the retained string would be "6.0e-1". If **EndString** is supplied it cannot be null.

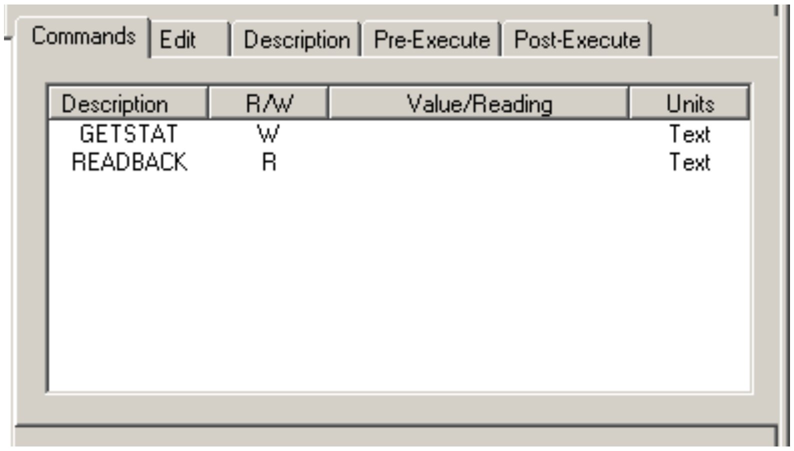
& is an optional continuation symbol indicating that the next line continues parsing the remainder of the read string. The **StartPos** and **StartString** for the next line will begin counting and comparing after the last character of the previous retained string. If the **&** continuation symbol is supplied there must be a valid **Read** command on the following line.

7.6.3.6. Read Command Example 1

This **Write** and **Read** script:

```
GETSTAT= W,GST,"STATUS\r\n"
READBACK = R, RB
```

displays as:



Description	R/W	Value/Reading	Units
GETSTAT	W		Text
READBACK	R		Text

Figure 85: Read Example 1

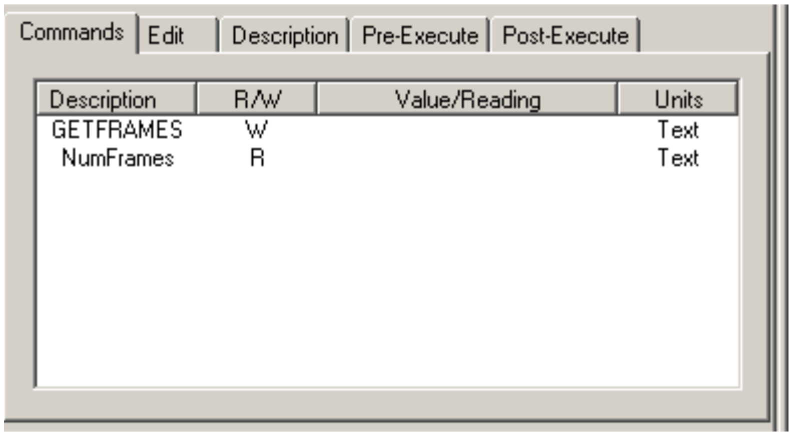
This would send the command "STATUS" and then call a GPIB **Receive()**, and put the contents in the object named READBACK.

7.6.3.7. Read Command Example 2

The entry:

```
GETFRAMES = W,GFR ,"GET FRAMECOUNT"
NumFrames = R, NFR, 1, "FRAMECOUNT "
```

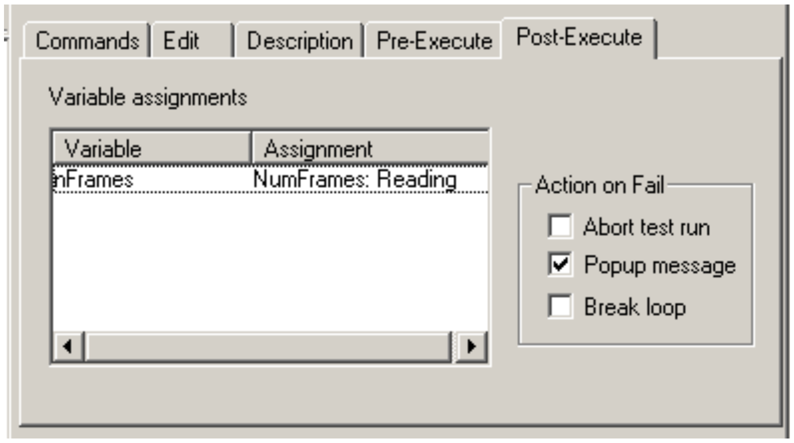
displays as:



Description	R/W	Value/Reading	Units
GETFRAMES	W		Text
NumFrames	R		Text

Figure 86: Read Example 2

This would send the command "GET FRAMECOUNT", then execute a GPIB **Receive().** Once the returned string (e.g. "FRAMECOUNT 30") has been read back from the instrument, PI-DATS will parse the string for "FRAMECOUNT " and save the remainder of the string (i.e. "30") in the object named **NumFrames**. The object **NumFrames** can then be assigned to a test plan variable in the **Post-Execute** property page as follows:



Variable assignments

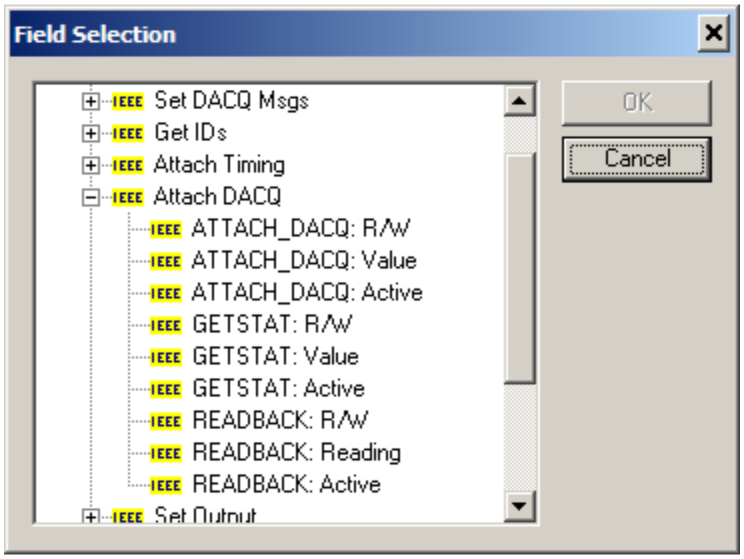
Variable	Assignment
nFrames	NumFrames: Reading

Action on Fail
- [ ] Abort test run
- [✓] Popup message
- [ ] Break loop

Figure 87: Assignment to Variable in Post-Execute

NumFrames can also be picked up in a **Result** mnemonic later in the test plan. Each entry in the Generic Mnemonic's definition makes three settings available to PI-DATS, either in **Post-Execute** or in a later **Result** mnemonic:

- Whether the command is a Read or Write command (**R/W**)
- The "**Value**" (the **UserEntry** at run-time for **Write** commands) or the "**Reading**" (e.g. the retained string for **Read** commands)
- Whether the command was **Active** at or not at runtime.



```
Set DACQ Msgs
Get IDs
Attach Timing
Attach DACQ
    ATTACH_DACQ: R/W
    ATTACH_DACQ: Value
    ATTACH_DACQ: Active
    GETSTAT: R/W
    GETSTAT: Value
    GETSTAT: Active
    READBACK: R/W
    READBACK: Reading
    READBACK: Active
Set Output
```

Figure 88: Assignment in Result mnemonic

7.6.3.8. Read Command Example 3

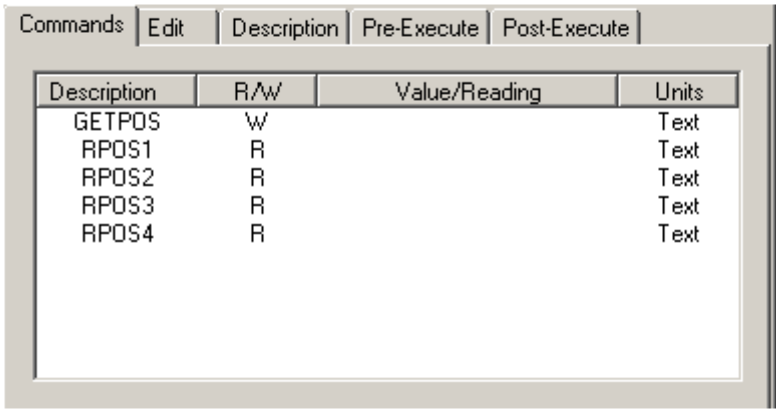
The following example parses for multiple values:

Suppose the command “GET POSITION” were sent to a device, and the expected reply is in the form of "POSITION 319, 255, -1, -1"

The following script:

```
GETPOS= W,POS,"GET POSITION"
RPOS1 = R, RP1, 1, "POSITION", 1, ",", &
RPOS2 = R, RP2, 1, ",", 1, ",", &
RPOS3 = R, RP3, 1, ",", 1, ",", &
RPOS4 = R, RP4, 1, ","
```

displays as:



Description	R/W	Value/Reading	Units
GETPOS	W		Text
RPOS1	R		Text
RPOS2	R		Text
RPOS3	R		Text
RPOS4	R		Text

Figure 89: Read Example 3

and returns the values “320”, “255”, “-1”, and “-1” into the objects RPOS1 through RPOS4 respectively. Note that spaces do not affect the conversion of a string to a numeric value. A string " 40" will have the same value as "40" when assigned to an integer or real number variable.

7.6.4. Inserting a Generic mnemonic

Once declared, the **Generic** mnemonic will be available in the **Mnemonic Selection** dialog box with its name and description as defined in the default script:

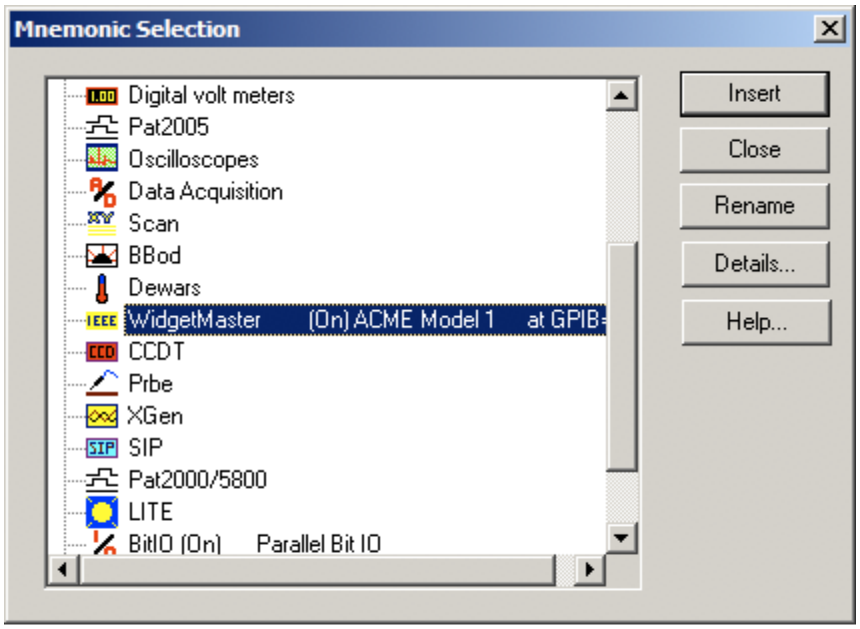


Figure 90: Generic mnemonic in the Insert Mnemonic dialog box

Once inserted into a test plan, a Generic mnemonic behaves like any other PI-DATS mnemonic. It supports PI-DATS features such as **Pre-execute** and **Post-execute**, and its settings are available to **Result** mnemonics.

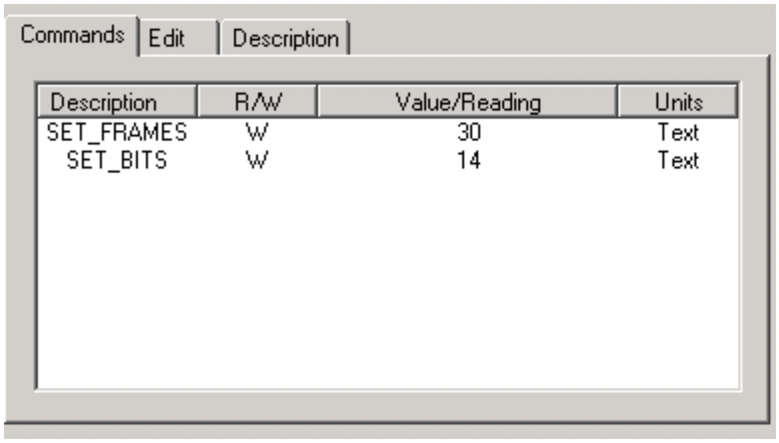
7.6.5. Editing a Generic mnemonic

Once inserted into a test plan each instance of the **Generic** mnemonic can be edited in two fundamentally different ways:

- Where the mnemonic's definition allows it via command placeholders, the user can change individual parameters sent to the instrument, such as a programmed voltage, temperature, delay, etc. The user can also enable or disable any of the defined commands. This level of editing is intended for the test plan operator.

- The test plan programmer can also edit the script and completely redefine the behavior of a particular instance of a mnemonic from within that mnemonic. Commands can be edited, added, or removed completely.

7.6.5.1. Editing Parameters



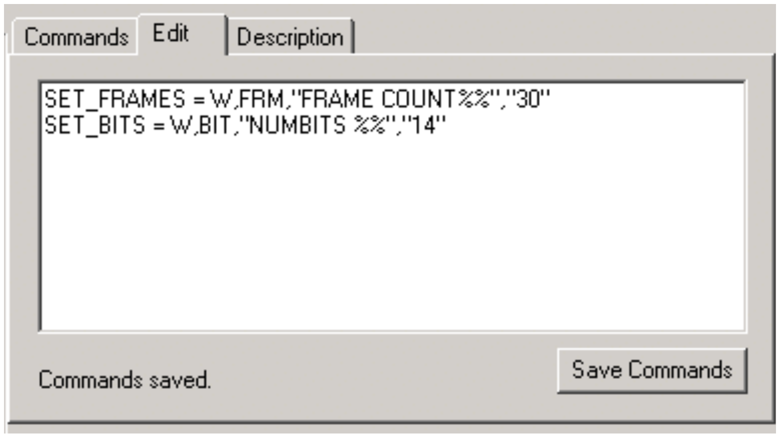
Description	R/W	Value/Reading	Units
SET_FRAMES	W	30	Text
SET_BITS	W	14	Text

Figure 91: Editing Parameters

Each line in the **Commands** property page may be disabled by toggling the entry in the **R/W** column to "**N**". Only enabled commands will be sent to the instrument when the mnemonic is executed.

If provided for via the placeholder syntax, the **Value** may be edited for **Write** commands. For **Read** commands, the **Value/Reading** column will display the **retained string** returned from the instrument, following any parsing specified in the script.

7.6.5.2. Editing Scripts



```
SET_FRAMES = W,FRM,"FRAME COUNT%%","30"
SET_BITS = W,BIT,"NUMBITS %%","14"
```

Commands saved.

Figure 92: Editing Scripts

The **Edit** property page contains the script that generates the **Commands** property page. The script was populated with the contents of the **Generic** property page in the **Hardware Configuration** when the mnemonic was inserted into the test plan, but it can be edited within each instance of the mnemonic.

The syntax for the script in the mnemonic is identical to that of the **[Commands]** section of the default script in the **Hardware Configuration**. The **[Settings]** section of the default script cannot be over-ridden in the mnemonic.

Please note that the Save Commands must be clicked before de-selecting this mnemonic. Otherwise the edits will be lost, and the definition will be read from the PIP file when this mnemonic is selected again. When **Save Commands** is clicked the script will also be checked for errors.

7.6.5.3. Limits Property Page

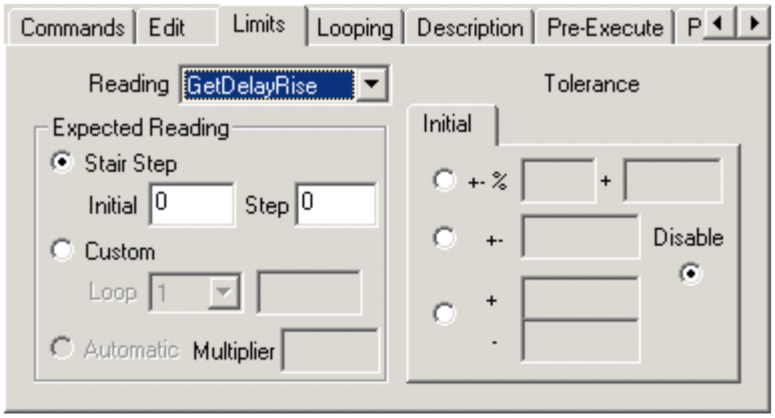


Figure 93: Limits Property Page

If the **[Settings]** section of the default script contains **Limits=Y**, then each instance of the mnemonic will contain a Limits property page. This page can be used to automatically compare each **Read** command's value (after parsing) to a set of defined values with specified tolerances. In the default case the tolerance test is set to **Disable**, so no comparison is made. To enable automatic limits testing choose one of the other three radio buttons to select a tolerance model.

7.6.5.4. Stair Step

Select the **Stair Step** radio button if you want to compare the selected measurement against a series of incrementing values. This option is meaningful only when the mnemonic is in a loop. The **Initial** value will be used for the comparison on the first loop iteration, and the comparison value will be incremented by the **Step** value on each subsequent loop.

To decrement the value on each loop, enter a negative value for the **Step**.

7.6.5.5. Custom

Instead of using a linear stair step of values, you can also define a series of custom values. Select the **Custom** radio button, use the **Loop** drop menu to choose a loop iteration, and then enter a value in the text box for that loop. To add additional loop counts to the menu, select the **More** item. To reduce the number of loop counts in the menu, select the **Less** item.

7.6.5.6. Tolerance

Enter the tolerance that determines a successful measurement. There are three options for defining a measurement tolerance.

- **+-%** Plus or minus a percentage of programmed value, plus or minus a constant tolerance.
- **+-** Plus or Minus a fixed tolerance about the expected value.
- **+/-** Plus a fixed tolerance or Minus a fixed tolerance about the expected value.
- **Disable** Limits testing is disabled

Different tolerance parameters can be specified for each loop iteration in a **Stair Step** and in a **Custom** sequence of expected values.

If the measured value does not meet the tolerance criteria a failure condition will be set. This will cause a **Fail** message at the end of the Test Run. The **Fail** message can also be accessed as a field using a **Result** mnemonic.

8. Control Mnemonics

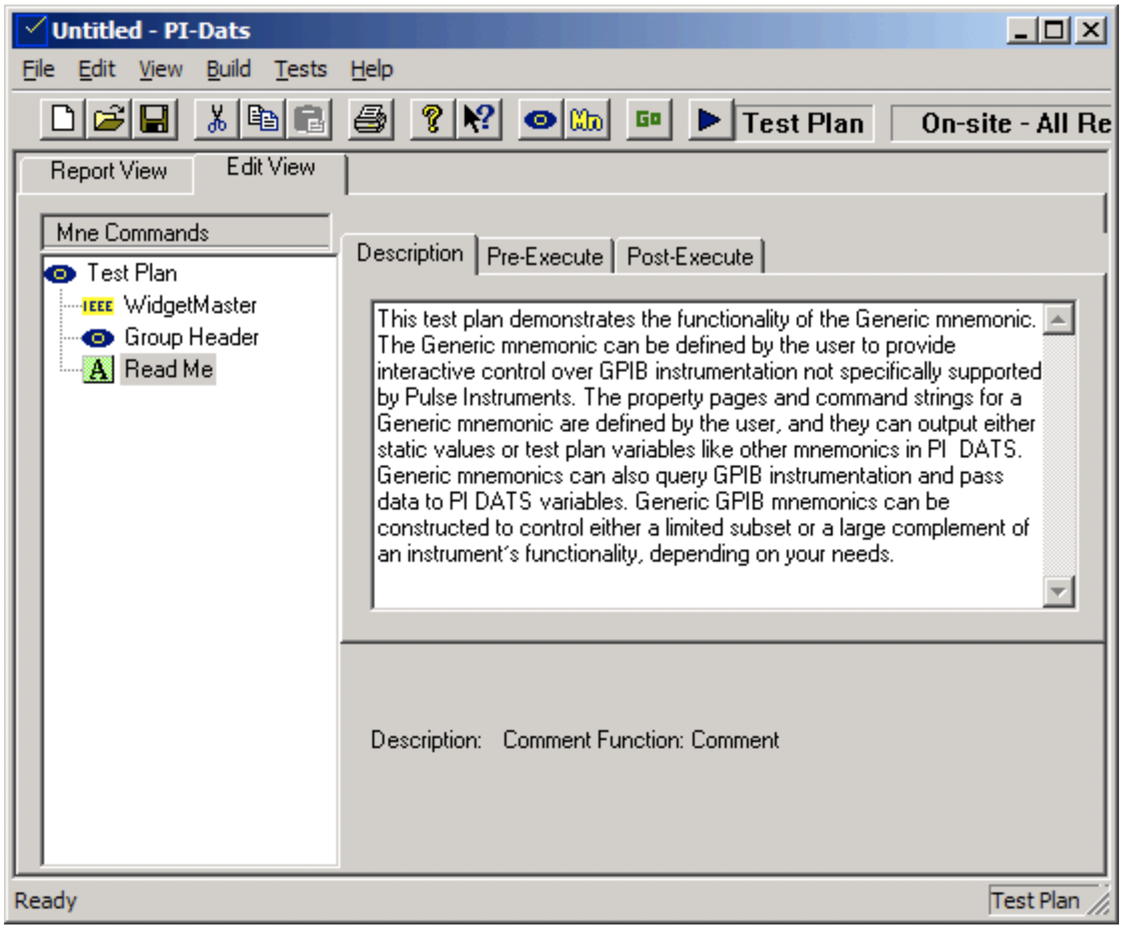
PI-DATS provides several functions designed to control execution of a test run. There is also a comment function to aid the designer.

8.1. *Suspend Functions*

PI-DATS provides several functions designed to suspend sequential execution of mnemonics. These mnemonic controls permit the test plan to alert the operator, obtain operator input, modify variables, break out of loops, branch to other Group Headers, and to and pause execution of a test run. There is also a comment function to aid the designer.

8.1.1.Comment Mnemonic

This function allows text comments to be stored and displayed in a test plan. To enter a comment, select the A mnemonic.



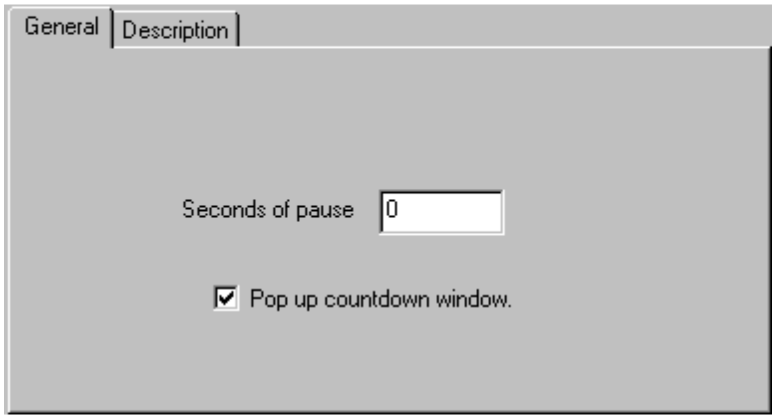
This test plan demonstrates the functionality of the Generic mnemonic. The Generic mnemonic can be defined by the user to provide interactive control over GPIB instrumentation not specifically supported by Pulse Instruments. The property pages and command strings for a Generic mnemonic are defined by the user, and they can output either static values or test plan variables like other mnemonics in PI DATS. Generic mnemonics can also query GPIB instrumentation and pass data to PI DATS variables. Generic GPIB mnemonics can be constructed to control either a limited subset or a large complement of an instrument's functionality, depending on your needs.

Description: Comment Function: Comment

Figure 94: PI-DATS Test Plan, Report View

8.1.2.Pause Mnemonic

This mnemonic will pause the execution of a Test Run.

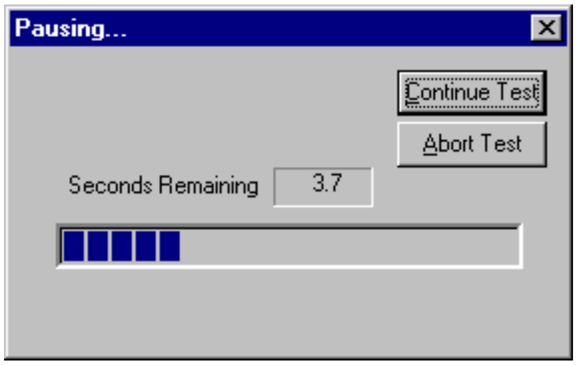


Seconds of pause: 0

☑ Pop up countdown window.

Figure 95: Pause Mnemonic

Enter the number of seconds to pause the Test Run execution. If the Pop up countdown window box is checked, a dialog box will be displayed during the pause:



Seconds Remaining: 3.7

Figure 96: Countdown Window displayed during Test Plan Pause

While the **Countdown Windows** is displayed during a test run, the Test Plan may be aborted by clicking the **Abort Test** button. The remaining Pause time can be bypassed by clicking the **Continue Test** button.

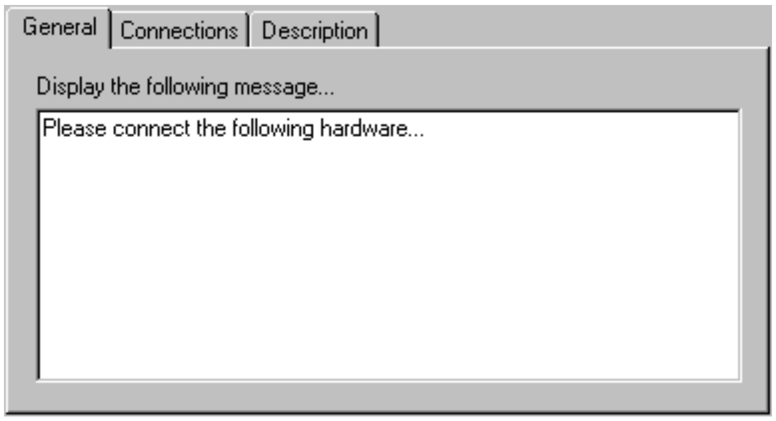
Click on the **Test** button to demonstrate how the pop up window will look during a normal Test Run.

8.1.3. Halt Mnemonic

This mnemonic causes the Test Run execution to Halt until started again by the operator. A message will be displayed to inform the operator of any instructions. There is also a mechanism to tell PI-DATS of any connections the operator should be making.

8.1.3.1. General Property Page

The General property page defines the message to be displayed when execution is halted.



Display the following message...

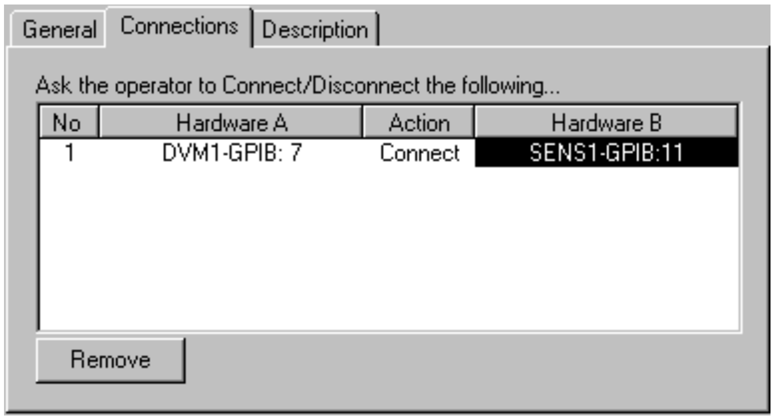
Please connect the following hardware...

Figure 97: Halt Mnemonic, General property page

Enter any text to be displayed when the execution is halted. For example, the operator may be instructed to connect specific hardware.

8.1.3.2. Connections Property Page

The **Connection** property page defines the connections assumed to take place. This informs PI-DATS of new connections to assist in automation and to allow Voltage and Current Sense readings to be scaled properly.



Ask the operator to Connect/Disconnect the following...

No	Hardware A	Action	Hardware B
1	DVM1-GPIB: 7	Connect	SENS1-GPIB:11

Figure 98: Halt Mnemonic, Connections property page

To define a new connection or connections, click in the space below the **Hardware A** column header. Select a hardware **Input** or **Output** port from the list of available devices. Do the same below the **Hardware B** column header, and then click below the **Action** column header to toggle the **Connect/Disconnect** instruction.

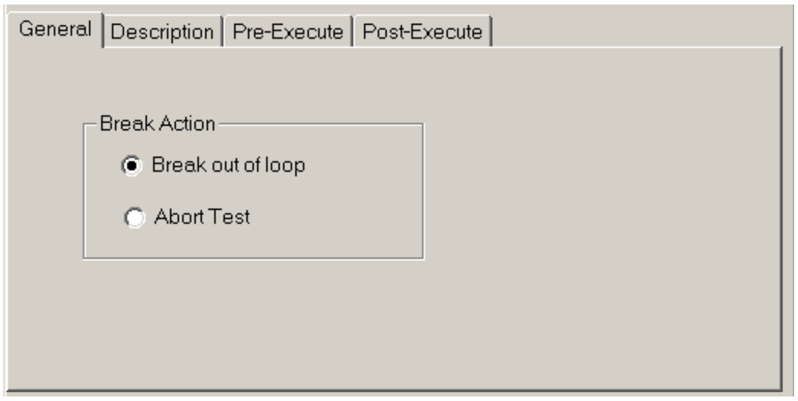
For example, to enable scaling for Voltage and Current sense readings, connect the **DVM** input to the **Sense** output, as shown above. For more information about voltage and current sense, please see **Section 6.9. Sense Mnemonic**. To disable scaling, e.g. for normal voltage output measurements, enter the above components with a **Disconnect** action.

Click the **Remove** button to remove a single connection entry.

Click the **Test** button to demonstrate how the pop up window will look during a normal Test Run.

8.1.4. Break Mnemonic

This mnemonic causes the Test Run execution to break out of a loop or to abort the test run completely.



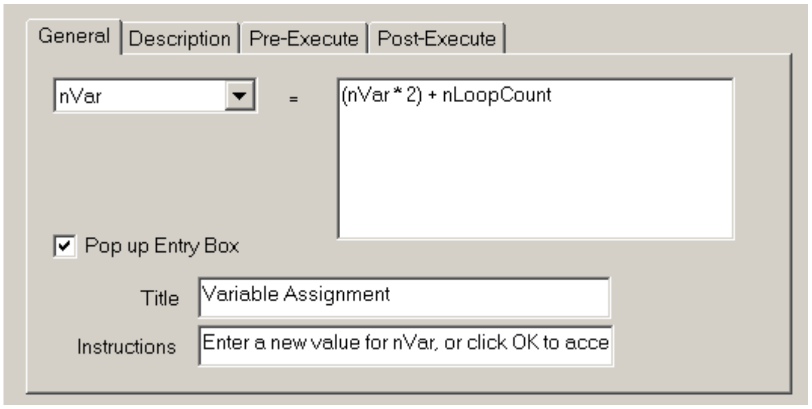
● Break out of loop
○ Abort Test

Figure 99: Break Mnemonic

Use the **Break Action** radio button to select the desired behavior. Use the **Pre-Execute** property page to set parameters for conditional execution.

8.1.5. VAR_FORM Mnemonic

The **Variable Formula** (VAR_FORM) mnemonic allows the Test Plan to assign new values to the user-defined variables.



nVar = (nVar * 2) + nLoopCount

[✓] Pop up Entry Box

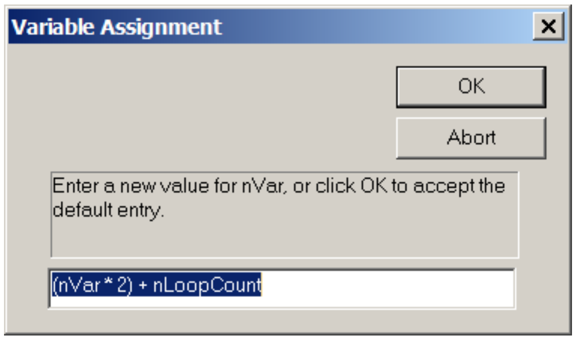
Title: Variable Assignment

Instructions: Enter a new value for nVar, or click OK to acce

Figure 100: VAR_FORM Mnemonic

Select a variable to modify from the drop-down menu. Then, enter an expression for the desired value of the selected variable. The expression may contain constants or other variables.

If the **Pop up Entry** box is checked, PI-DATS will display a dialog box when this mnemonic is executed. The dialog box will contain the title and instructions specified here, plus a text entry box for the operator. The text box will contain, as its default contents, the expression entered in the variable assignment box:



Enter a new value for nVar, or click OK to accept the default entry.

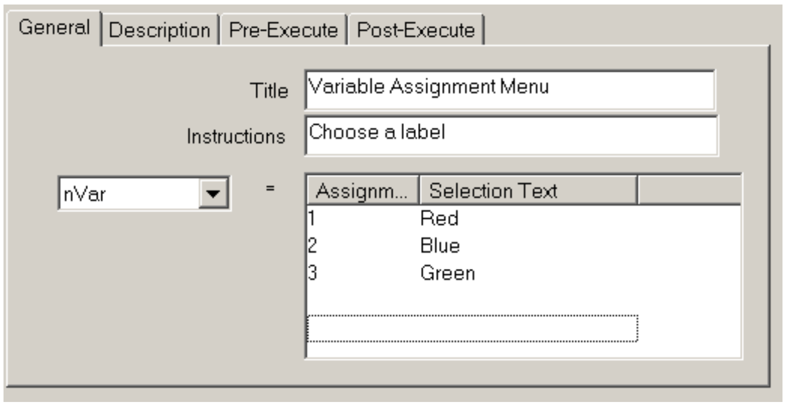
(nVar * 2) + nLoopCount

Figure 101: VAR_FORM Pop-Up Entry box

Click the **Test** button to demonstrate how the pop up window will look during a normal Test Run.

8.1.6. VAR_MENU Mnemonic

The **Variable Menu** (VAR_MENU) mnemonic will display a dialog box allowing the Test Plan operator to assign to a variable a new value from a set of predefined choices.



Title: Variable Assignment Menu

Instructions: Choose a label

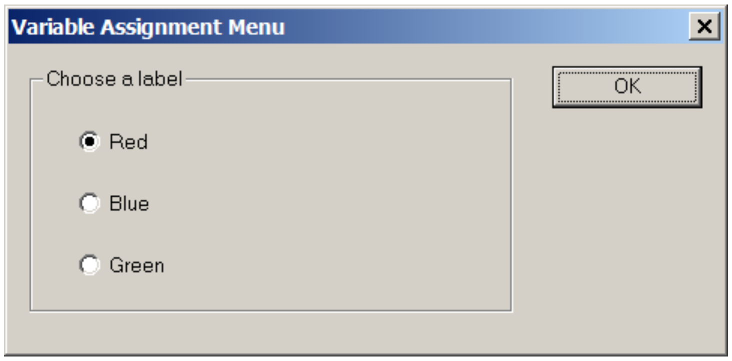
nVar =

Assignm...	Selection Text
1	Red
2	Blue
3	Green

Figure 102: VAR_MENU Mnemonic

Select a variable to modify from the drop-down menu. Then, enter a set of potential new values for the variable in the Assignments column header, and the **Selection Text** for the radio buttons to be displayed for the operator.

The dialog box will contain the title and instructions specified here, plus a radio button control for the operator. For example, the settings above would result in the following dialog box being displayed:



Red [✓]
Blue
Green

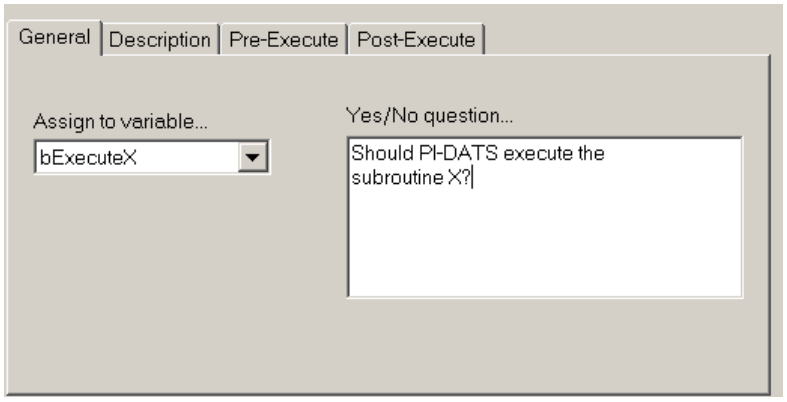
Figure 103: VAR_MENU Pop-Up Entry box

If the Operator were to click **Blue** and then **OK**, the variable nVar would be assigned a value of 2.

Click the **Test** button to demonstrate how the pop up window will look during a normal Test Run.

8.1.7. VAR_BOOL Mnemonic

The Variable Boolean (VAR_BOOL) mnemonic will display a dialog box allowing the Test Plan operator to assign **True** or **False** to a Boolean variable.



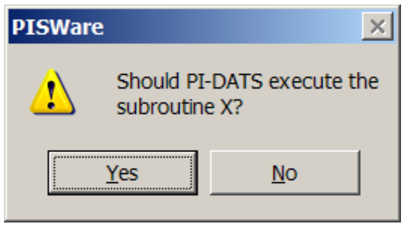
Assign to variable... bExecuteX

Yes/No question... Should PI-DATS execute the subroutine X?

Figure 104: VAR_BOOL Mnemonic

Select a variable to modify from the drop-down menu. Then, enter text for the prompt to display for the operator.

The dialog box will contain the text entered here with Yes and No buttons, corresponding to True and False values. For example, the settings above would result in the following dialog box being displayed:



Should PI-DATS execute the subroutine X?

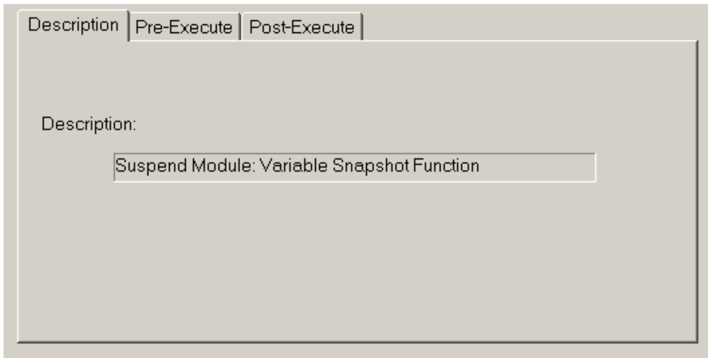
Figure 105: VAR_BOOL Pop-Up Entry box

If the Operator were to click **Yes**, the variable **bExecuteX** would be assigned a value of **True**.

Click the **Test** button to demonstrate how the pop up window will look during a normal Test Run.

8.1.8. VAR_SNAPSHOT Mnemonic

The Variable Snapshot (VAR_SNAPSHOT) mnemonic captures the value of all defined variables at the time at which this mnemonic is executed.



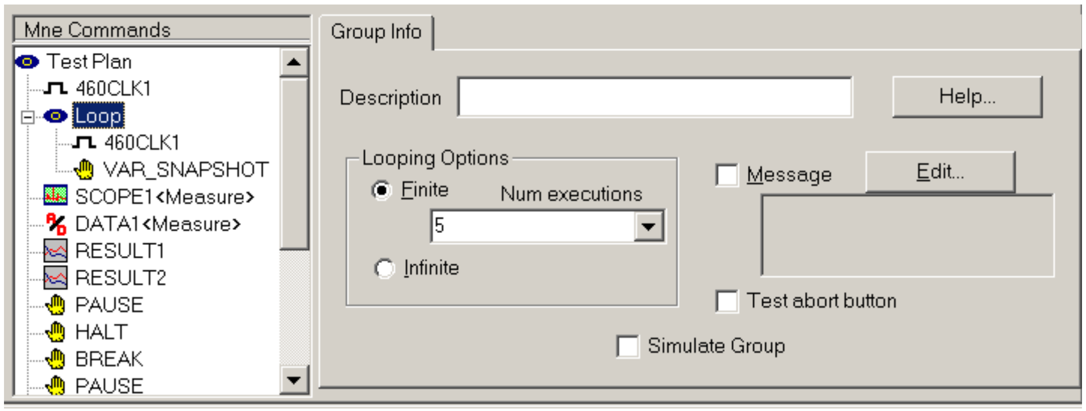
Description:

Suspend Module: Variable Snapshot Function

Figure 106: VAR_SNAPSHOT property page

There are no programmable parameters for this mnemonic, other than the standard **Pre-Execute** and **Post-Execute** settings.

The values captured in the snapshot can be retrieved by a Result mnemonic inserted later in the test plan. For example, assume the following test plan has a user-defined variable ***nLoopCount*** with an initial value of 1, and that its value gets incremented by 1 on each pass through the **Loop** Group Header.



Description

Looping Options

● Finite Num executions 5

○ Infinite

☐ Message

☐ Test abort button

☐ Simulate Group

Figure 107: Sample Loop with VAR_SNAPSHOT

The VAR_SNAPSHOT mnemonic will capture the value of ***nLoopCount*** during each iteration of the loop, and make that available to a Result mnemonic. A **Result** mnemonic placed below this loop would show the VAR_SNAPSHOT available fields as follows:

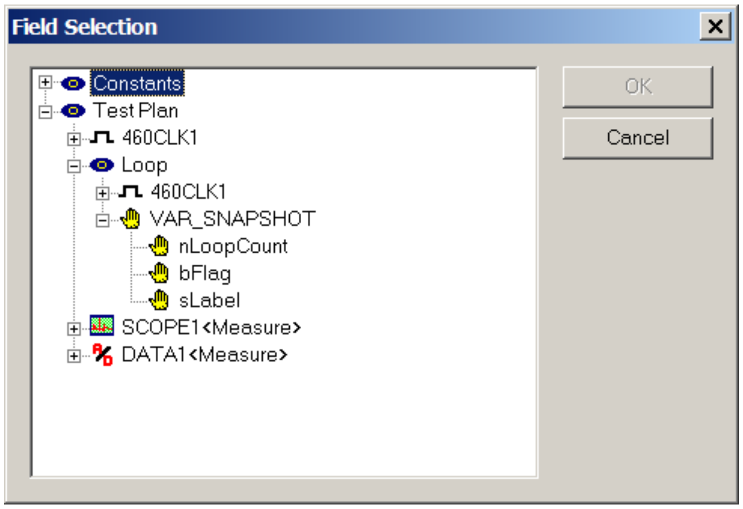
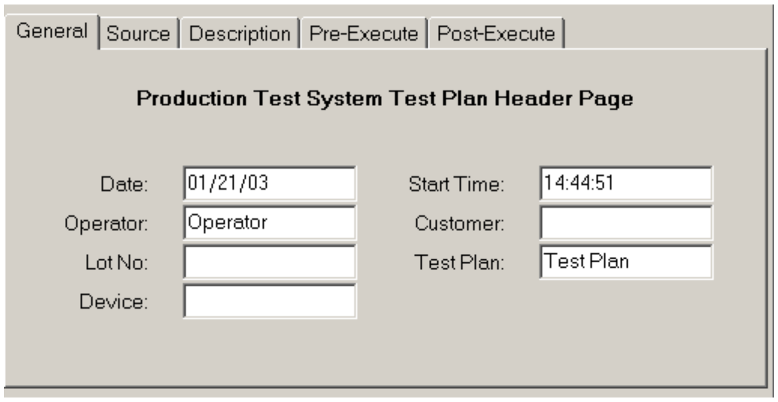


Figure 108: VAR_SNAPSHOT values passed to Result mnemonic

8.1.9. PAGE Mnemonic

This mnemonic provides a simple dialog box allowing the user to enter production test information.



Production Test System Test Plan Header Page

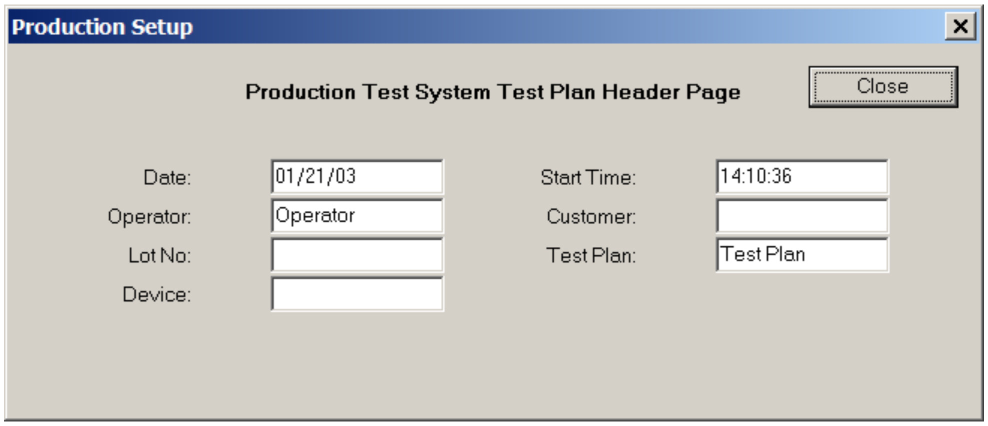
Date:	01/21/03	Start Time:	14:44:51
Operator:	Operator	Customer:	
Lot No:		Test Plan:	Test Plan
Device:			

Figure 109: PAGE Mnemonic, General property page

By default, the dialog box contains the following entries. The string in parentheses identifies the PI-DATS variable name that will receive each entry:

- **System Constants**
  - **Date** (CurDate)
  - **Start Time** (CurTime)
  - **Operator** (OperatorName)
  - **Test Plan** (TestRunTitle)
- **Variables**
  - **Customer** (Customer)
  - **Lot No.** (LotNo)
  - **Device** (Device)

The System Constants are not editable at run time, but the Variables can be entered by the user. Note that the variables used by this mnemonic are not automatically defined. Variables matching these entries must already have been defined in the Test Plan header.

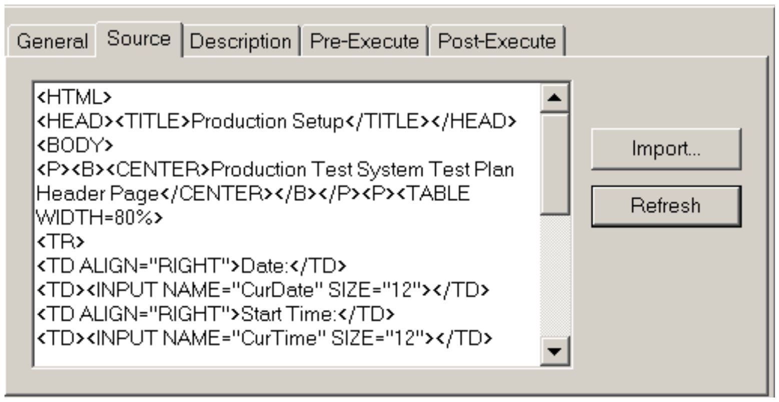


Production Test System Test Plan Header Page

Date:	01/21/03	Start Time:	14:10:36
Operator:	Operator	Customer:	
Lot No:		Test Plan:	Test Plan
Device:			

Figure 110: Dialog Box generated by PAGE mnemonic

When executed, this mnemonic will cause the test plan to suspend execution temporarily until the operator clicks the **Close** button.



```
<HTML>
<HEAD><TITLE>Production Setup</TITLE></HEAD>
<BODY>
<P><B><CENTER>Production Test System Test Plan
Header Page</CENTER></B></P><P><TABLE
WIDTH=80%>
<TR>
<TD ALIGN="RIGHT">Date:</TD>
<TD><INPUT NAME="CurDate" SIZE="12"></TD>
<TD ALIGN="RIGHT">Start Time:</TD>
<TD><INPUT NAME="CurTime" SIZE="12"></TD>
```

Figure 111: PAGE Mnemonic, Source property page

The **Source** property page shows the HTML used to define the dialog box. The HTML may be edited to create a custom page. The page takes the form of an HTML table with standard HTML text-input elements. SELECT elements are not supported at this time.

For each INPUT element, the NAME must match the name of a defined PI-DATS variable. No VALUE item should be specified, as it will be populated by PI-DATS at run-time with the current variable value.

Due to the manner in which PI-DATS captures keystrokes, common keyboard shortcuts will act upon the mnemonic itself instead of on the edit box contents. To enter a line-break within the HTML edit box, hold down the Ctrl key while pressing Enter. To copy and paste text, right-click and use the contextual menu.

You can also import HTML from a file using the **Import** button.

To update the display on the **General** property page, click the **Refresh** button on the **Source** property page.

Note that any named inputs referencing system constants (e.g. CurDate and other reserved words) will not be editable, either during test plan design or during run time.

Variable values generated by execution of the Page mnemonic are not directly accessible by a Result mnemonic. Use a variable snapshot to access these variables. See **Section 8.1.8. VAR_SNAPSHOT Mnemonic**.

8.1.10. GOTO Mnemonic

This mnemonic causes the Test Run to branch to another Group Header.

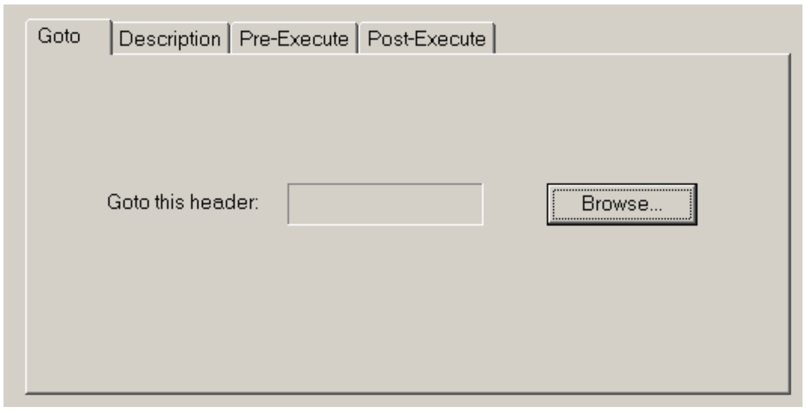


Figure 112: GOTO mnemonic

Click the **Browse** button to display a list of available Group Headers:

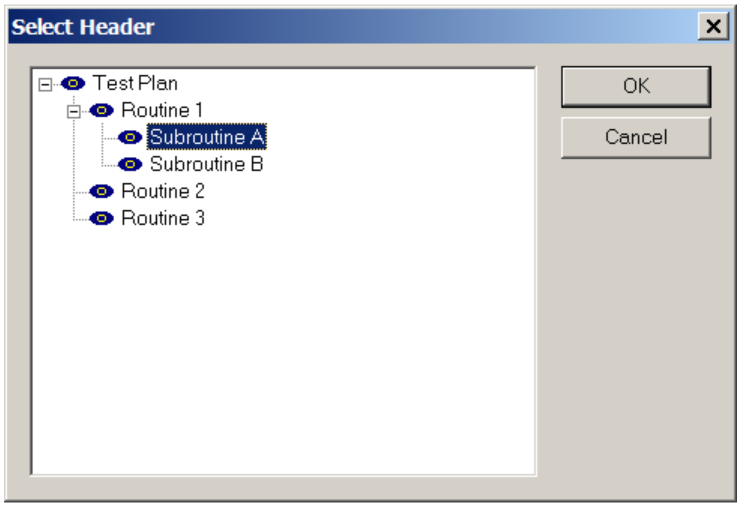
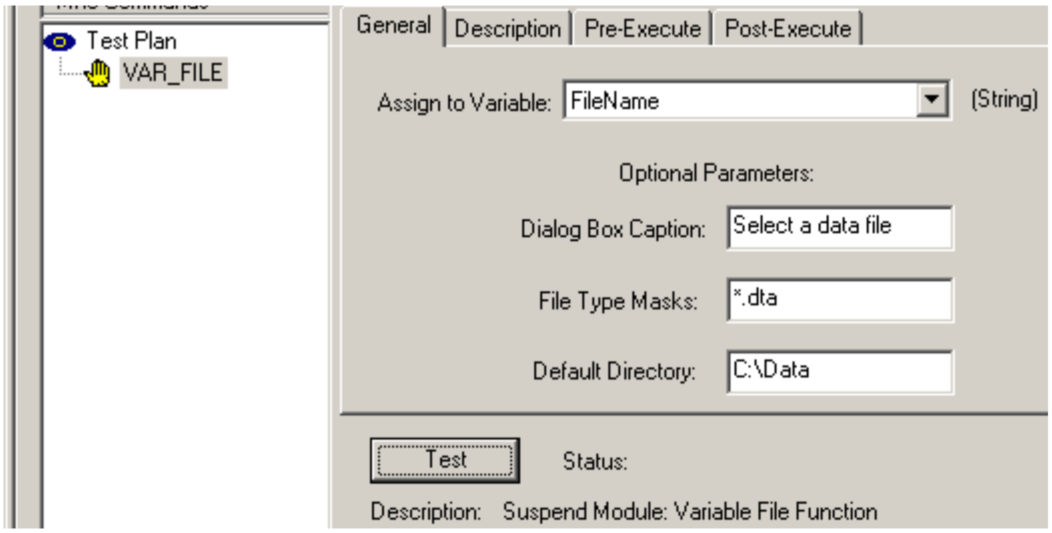


Figure 113: GOTO mnemonic, Group Header selection

This mnemonic can be used with its **Pre-Execute** property page to implement conditional branching.

8.1.11. VAR_FILE Mnemonic

The Variable File (**VAR_File**) mnemonic permits run-time input from the operator of a filename. When executed the **VAR_FILE** mnemonic causes PI-DATS to display a standard Windows **Open File** dialog box. The complete path and name of the file selected by the operator is then assigned to a pre-defined string variable, where it can then be used by subsequent mnemonics as the source or destination for data.



Assign to Variable: FileName (String)

Optional Parameters:

Dialog Box Caption: Select a data file

File Type Masks: *.dta

Default Directory: C:\Data

Status:

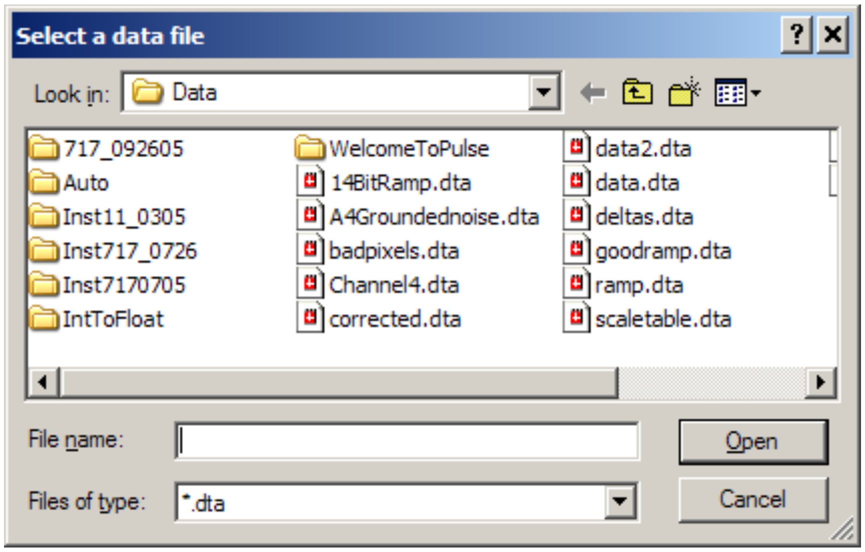
Description: Suspend Module: Variable File Function

Figure 114: VAR_FILE property page

A string variable must be defined in the PIP file before the **VAR_FILE** mnemonic can be used.

After inserting the VAR_FILE mnemonic into your test plan, use the **Assign to Variable** drop-menu to choose the string variable that will hold the path and filename after execution. The **Dialog Box Caption**, **File Type Mask**, and **Default Directory** can be used to control the appearance of the dialog box that appears at run-time. Use the Test button to test the appearance of the dialog box.

For example, the **VAR_FILE** as populated above will generate the following dialog box at run-time:



717_092605	WelcomeToPulse	data2.dta
Auto	14BitRamp.dta	data.dta
Inst11_0305	A4Groundednoise.dta	deltas.dta
Inst717_0726	badpixels.dta	goodramp.dta
Inst7170705	Channel4.dta	ramp.dta
IntToFloat	corrected.dta	scaletable.dta

File name:

Files of type: *.dta

Figure 115: Run-time dialog box generated by VAR_FILE

If the operator clicks **Cancel** at run-time, the string variable will contain an empty string.

9. RESULT Mnemonic

The **Result** mnemonic is used to process test data for analysis, plots, and/or generating reports. It can also be used to perform image post-processing, such as Non-Uniformity Correction (NUC) and Bad Pixel Replacement. Conceptually, the Result mnemonic takes one or more tables (arrays) of data as inputs and produces tables or scalars as outputs.

Most objects in a test plan make data accessible as inputs to the Result mnemonic. The following is a sample of available data types:

- Programmed settings for instruments (voltage levels, rise/fall times, channel settings, acquisition parameters, etc.)
- Measured values (from DVMs, ‘scopes, temperature monitors)
- Image data from Data Acquisition instruments or from data files
- Test plan constants (date, time, operator)
- Data returned from other Result mnemonics

Data can be processed within the Result mnemonic or exported to a variety of external analysis tools, including Matlab, Excel, or your custom application. Built-in analysis and data reduction tools include:

- General statistical functions
- Subset extraction
- Threshold detection
- Array arithmetic

Data passed into a Result mnemonic can be reduced to scalars or to arrays, and/or made available to other Result mnemonics for further processing.

9.1. Overview

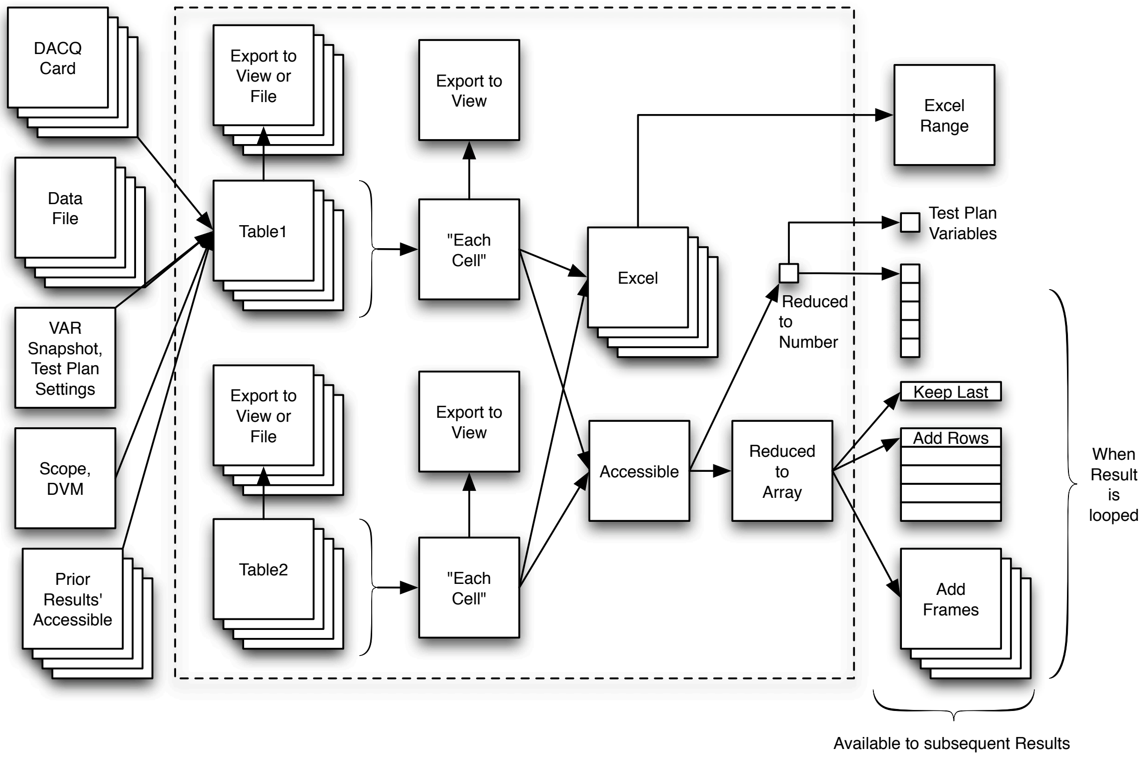


Figure 116: Data Flow To/From Result Mnemonic

Result mnemonic in your test plan must have one or more **Tables** defined. Tables can consist of data from any of the sources described above, and may have 1D (an array of a scalar variable or value), 2D (an array of multiple scalars or values, or an acquired image), or 3D (multiple frames of 2D data).

9.2. Setup

Result mnemonics operate on tables of data. The **Setup** property page is used to define the input data tables.

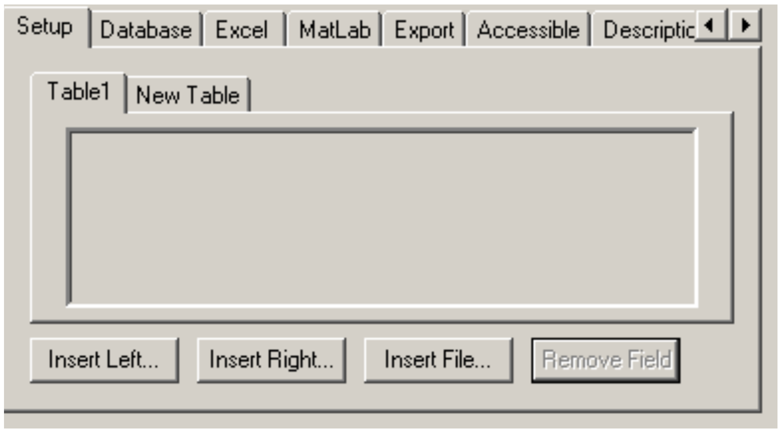


Figure 117: Result Mnemonic, Setup property page

By default, a Result mnemonic starts with one empty table, named **Table1**. Additional tables can be created by selecting the **New Table** tab.

Multiple tables can be inserted into a single Result mnemonic either for performing multi-table processing (e.g. subtracting one table from another) or for constructing reports.

9.2.1. Populating Data Tables

To begin populating a data table, click **Insert Left** or **Insert File**. If **Insert Left** is clicked a dialog box will show all available data objects in the current test plan, arranged in a tree-structured hierarchy that mirrors the test plan. With the exception of the Constants, only mnemonics preceding this **Result** mnemonic and located within the same or "deeper" loop in the test plan will be shown:

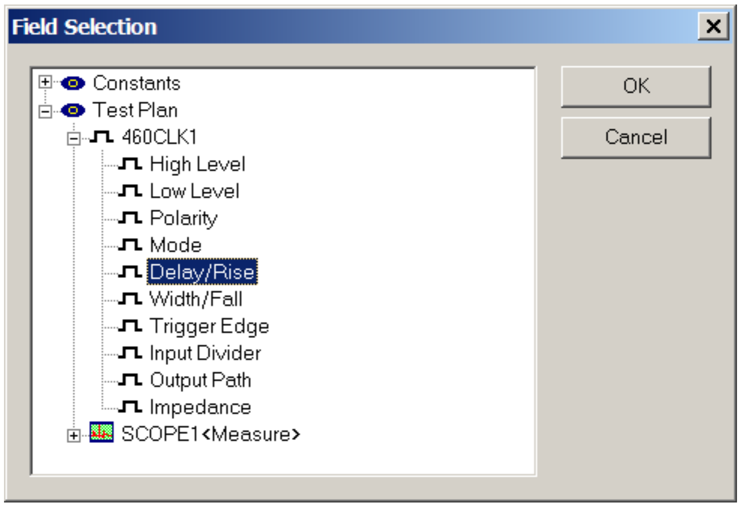


Figure 118: Result Mnemonic, Insert Field dialog box

Each available mnemonic in the test plan will appear within its group header. When expanded, each mnemonic will display all data items available to the Result mnemonic. In the example above, a Clock Driver mnemonic can return any of its programmable parameters as a data field.

Mnemonics may return data as scalars or arrays. Scalar values can be returned either by hardware control parameters (e.g. settable parameters), variable snapshots, or measurement devices (e.g. oscilloscope or DVM mnemonics).

Certain data types, such as image data from an acquisition system, are inherently array-structured. Scalar data types, such as voltage levels, will be converted into arrays if the mnemonics returning them are looped. Each field placed into a table serves as a column definition for a report or an axis definition for a plot. Arrays placed into a table take up the entire table. Therefore each table can contain either fields or an array, but not both.

9.2.2. Data From Scalars

Click one or more desired scalar items and click **OK**. To de-select a source, click it again. All data items from a test plan mnemonic can be selected by clicking the mnemonic itself. Each inserted item will become a column header in the current table of the Result mnemonic.

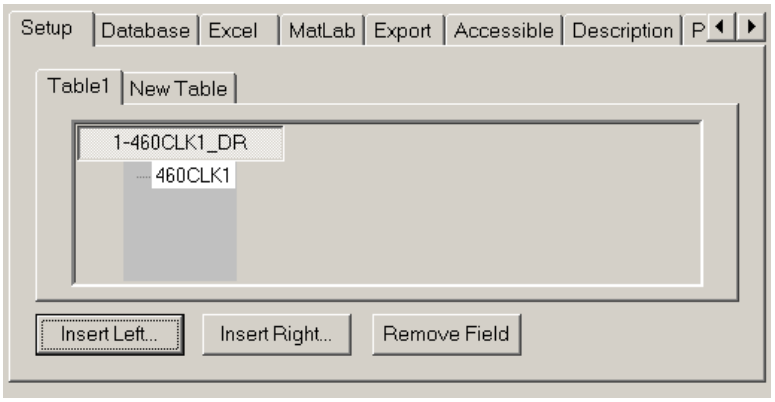


Figure 119: Result Mnemonic, Field Inserted

Once the table has been populated with data, the table can be renamed by double-clicking on its title (e.g. **Table1** in the figure above).

The default label for a column header is constructed from the column number, mnemonic name and a suffix indicating the returned parameter. In the example above, the column header refers to the Delay/Rise time (DR) programmed parameter for a PI-40460 Clock Driver channel. Columns can be renamed by double-clicking the table header label. Column labels can be reset to their default labels by clicking and deleting the user-defined name.

To create additional data columns, click **Insert Left** or **Insert Right** to insert a data field to the left or right of the selected column. In the example above, adding the Scope mnemonic's measured Rise Time to the right of the current column would result in the following table definition:

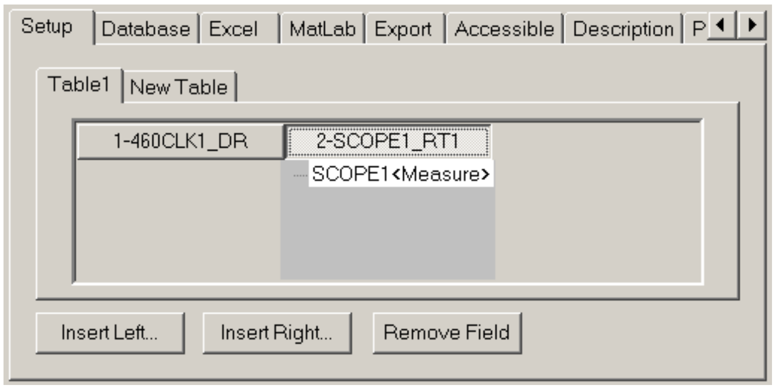


Figure 120: Result Mnemonic, Second Field Inserted

Columns can be re-ordered by dragging and dropping their headers. A column's data source can be examined or reassigned by double-clicking the lowest level of the "tree" extending below the column header. When a data item is double-clicked, PI-DATS will display the Field Selection dialog box with the current data source selected. The data source may be de-selected by clicking it, and a new source (or sources) selected by clicking.

Click **Remove Field** to remove the currently selected field from the table.

To remove a table definition entirely, remove all of its fields.

9.2.3. Data From Arrays

Arrays may be inserted as a table. Arrays are generated by **DACQ Measure** mnemonics that return acquisition data, by **RESULT** mnemonics that make data arrays accessible, or by insertion of a data file. Arrays can only be entered into empty tables.

To enter a data array from a **DACQ Measure** or **RESULT** mnemonic, use the **Insert Left** or **Insert Right** button and select the desired data source from the test plan tree.

The **DACQ Measure** mnemonic makes one data set available from each connected Master, numbered from 1 to n, ascending by slot number. If there is more than one connected Master, there will also be a "variable" Master that can be controlled by a defined integer variable. Please see the **PI-3105 Operators Manual** for the definition of an active Master.

Because whether a particular DACQ card will be an active Master, active Slave, or inactive card cannot be determined by PI-DATS at test plan time, it is the user's responsibility to ensure that an appropriate Master is chosen as the data source.

The **DACQ** mnemonic does not preserve data frames on loops, so a **Result** mnemonic must be in the same loop as a DACQ mnemonic in order to "see" more than one frame.

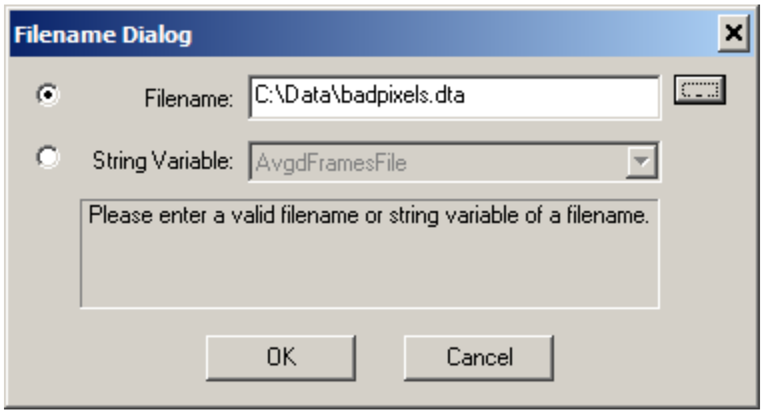
If a **Result** mnemonic attempts to read acquisition data from a **DACQ** mnemonic in an inner loop, it will only receive the first frame of the last loop, regardless of how many iterations are in the loop or how many frames are acquired on each loop.

To process frames from multiple iterations of an inner loop, use a **Result** mnemonic inside the loop to write and append the frames to a data file, then read that data file into the outer **Result** mnemonic (see below).

9.2.4. Data From File

To enter a data set from a file, use the **Insert File** button. The file must use Pulse Instruments' data file format, generated either by the DACQ mnemonic in PI-Controller mode or by an Export operation from a RESULT mnemonic.

The Insert File button will cause the following dialog box to be displayed:



[✓ Filename:] C:\Data\badpixels.dta
String Variable: AvgdFramesFile
Please enter a valid filename or string variable of a filename.

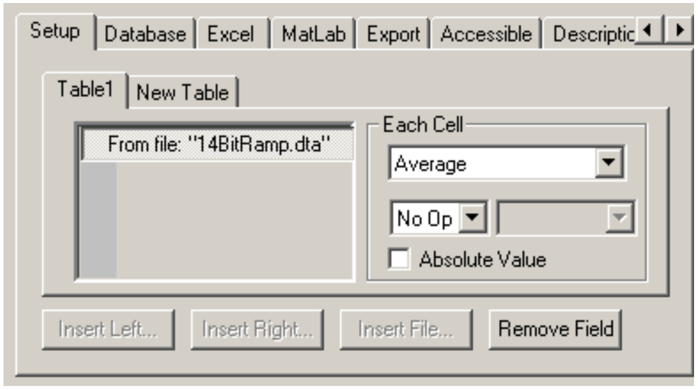
Figure 121: Result Mnemonic, Insert From File

The **Filename** option permits the user to enter or select the path and filename of a specific file. To select an existing path and filename, use the Browse (**…**) button to display a standard Windows **Open File** dialog.

To enter the name manually, type a path and filename in the box. If a filename is specified without a path, PI-DATS will attempt to load the file from the same directory that contains the PIP file.

The **String Variable** option permits the user to use an existing string variable to specify the file at run-time. For example, the string **AvgdFramesFile** might contain the path and filename of the desired data file. The **VAR_FILE** mnemonic may be used at a previous point in the test plan to allow the operator to specify this value.

Once the file or string variable has been selected, the filename will appear in the Table definition:



From file: "14BitRamp.dta"
Each Cell: Average
No Op
Absolute Value

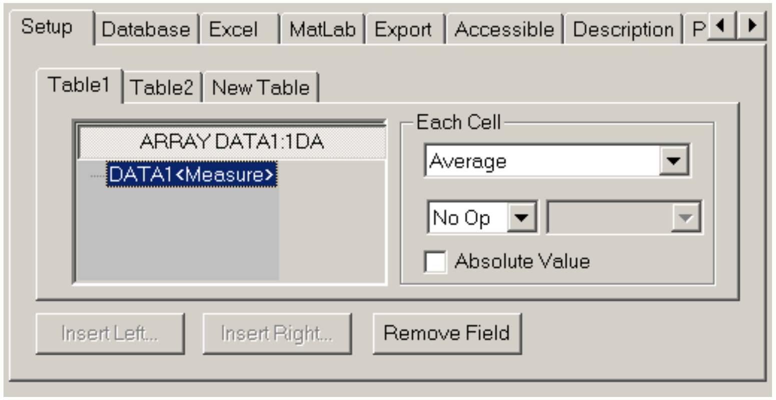
Figure 122: Result Mnemonic, File Inserted as Table

The path and filename can be edited by clicking on the filename, prompting the file selection dialog box to re-appear

Any path and filename may be entered, including filenames that do not yet exist or variables that do not yet point to a file. This can be used to make references to files that will be created during test plan execution (e.g. data processed and exported by a RESULT mnemonic). If the specified file does not exist when the dialog box is closed, the Result mnemonic will report "**From File: Open Fail**," to warn the user. If the file still does not exist when this mnemonic is executed, the mnemonic will fail execution.

9.2.5. Data Reduction

If the field entered is an array, then array reduction functions will appear:



ARRAY DATA1:1DA
DATA1<Measure>

Each Cell
Average
No Op
Absolute Value

Figure 123: Result Mnemonic, Array Operations

Choose a reduction algorithm by selecting from the drop-menu. The data will be returned as a single array. For example, the array above refers to a Data Acquisition mnemonic. If that mnemonic were programmed to acquire 100 frames of data, each 1024 x 1024, then the reduction functions will return a single 1024 x 1024 frame, with each returned pixel value being the result of the selected operation (in this case an Average) across all frames.

The array reduction functions are meaningful only for multiple sets of an array (e.g. multiple frames of image data from an acquisition system, etc). If the array does not have depth (e.g. a single image frame), choose the **Average** setting, as none of the other operations is meaningful for a single array of data.

The array set can be reduced to a single array in one of the following ways.

9.2.6. Average

Each pixel will be returned as the average value of that pixel across all acquired frames.

e.g.: $\frac{\sum x_n}{n}$

9.2.7. Std Deviation (Sample)

Each pixel will be returned as the standard deviation (sample) of that pixel across all acquired frames, e.g.:

$$\sqrt{\frac{n\sum x^2 - \left(\sum x\right)^2}{n(n-1)}}$$

9.2.8. Std Deviation (Population)

Each pixel will be returned as the standard deviation (population) of that pixel across all acquired frames, e.g.:

$$\sqrt{\frac{n\sum x^2 - \left(\sum x\right)^2}{n^2}}$$

9.2.9. Has a Column

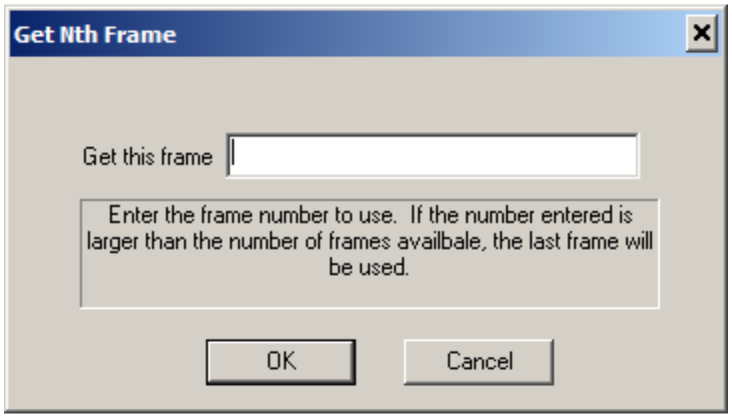
Each pixel will be returned as a column of data. The number of rows will be the number of frames. This function is intended primarily for data from linear imagers.

9.2.10. RMS

Each pixel will be returned as the RMS value of that pixel across all acquired frames.

9.2.11. Nth Frame

The nth frame will be returned. If **Nth Frame** is chosen, a dialog box will appear, prompting the user to enter a frame number to select:



Enter the frame number to use. If the number entered is larger than the number of frames availbale, the last frame will be used.

Figure 124: Result Mnemonic, Select Nth Frame

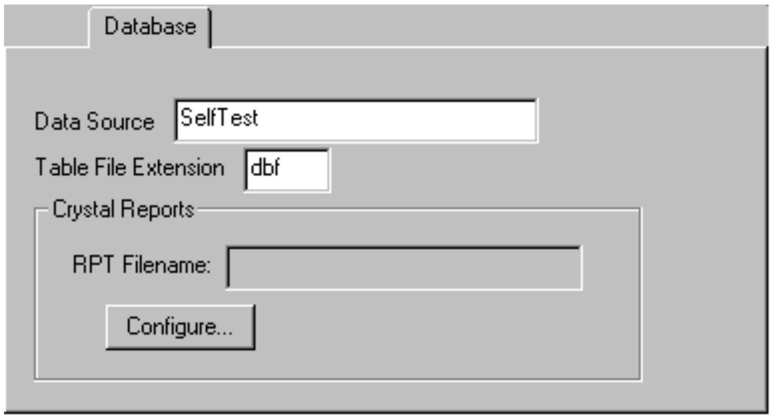
An integer or a PI-DATS integer variable can be used to select the frame number. If the selected frame number is greater than the total number of frames, the last frame will be selected. If the selected frame number is less than 1, the first frame will be selected.

9.2.12. Scalar Transform

Each pixel in the reduced array can also be transformed by application of a constant value or user-defined variable. Choose any operator other than **No Op** from the drop-menu then select a variable or type a constant value into the text box. Choose **No Op** to leave the array data unchanged. Resultant values can also be returned as absolute value by clicking the **Absolute Value** checkbox.

9.3. Database Property Page

This property page enables PI-DATS to build database tables for Crystal Reports. This will channel data to a database using ODBC Connectivity.



Data Source: SelfTest
Table File Extension: dbf

Figure 125: Result Mnemonic, Database property page

9.3.1. Data Source

Enter the name of the data source here. The name must be identical to a name that is listed in the "**Data Sources**" dialog box. The "**Data Sources**" dialog box can be shown by clicking the "**32bit ODBC**" icon within the Control Panel Window. (Access the Control Panel by selecting "**Start**", then "**Settings**", then "**Control Panel**").

It is recommended that a data source using the "**Microsoft dBase Driver (*.dbf)**" is selected.

If there are no "**Microsoft dBase Driver (*.dbf)**" data sources defined, then one may be created as follows:

Click on "**Add**".

Select "**Microsoft dBase Driver (*.dbf)**", and then click on "**OK**".

Enter a suitable "data source name" and "description". Remember the "data source name" as this will be what is to be entered for the "Data Source" for PI-DATS.

Select "**dBase IV**" for the database version.

Select a directory where the database tables will be stored. Selecting the "**Pulse20**" directory will be sufficient.

9.3.2. Table Extension

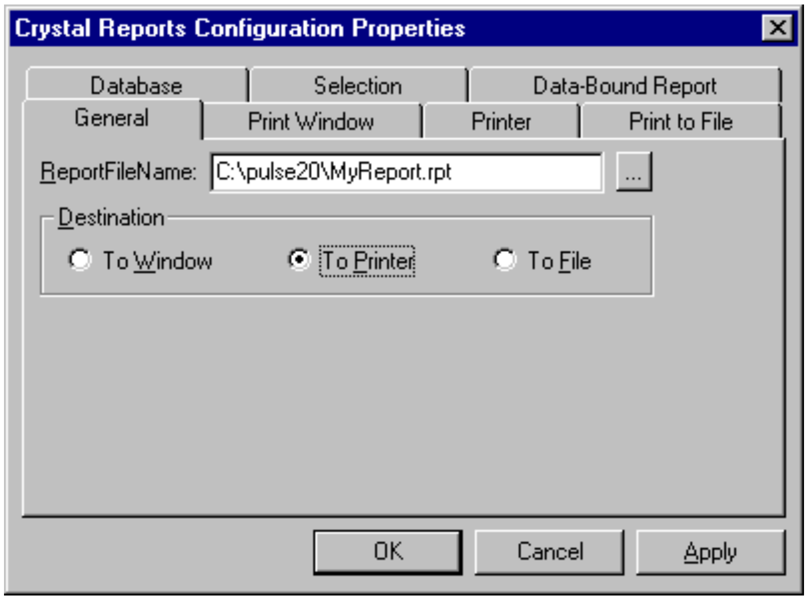
Enter the desired database table extension here. Enter “**dbf**” if the data source driver is “**Microsoft dBase Driver (*.dbf)**”.

9.3.3. Crystal Reports Filename

This displays the currently selected Crystal Report filename. A Crystal Report filename usually has an “rpt” filename extension. Select the Crystal Reports filename by clicking on “**Configure**”.

9.3.4. Crystal Reports Configuration

Click on this button to access the Crystal reports configuration window as shown below:



ReportFileName: C:\pulse20\MyReport.rpt

Destination: To Window [✓ To Printer] To File

Figure 126: Result Mnemonic, Crystal Reports property page

Enter the Crystal Reports path and filename to provide Crystal Reports with the report file. The report file must already exist. This file must be created using the “Crystal Reports” Application. (Refer to the Crystal Reports users manual).

The “**Destination**” selection will designate where the report will be sent when the RESULT mnemonic is reached during a Test Plan run.

9.3.4.1. To Window

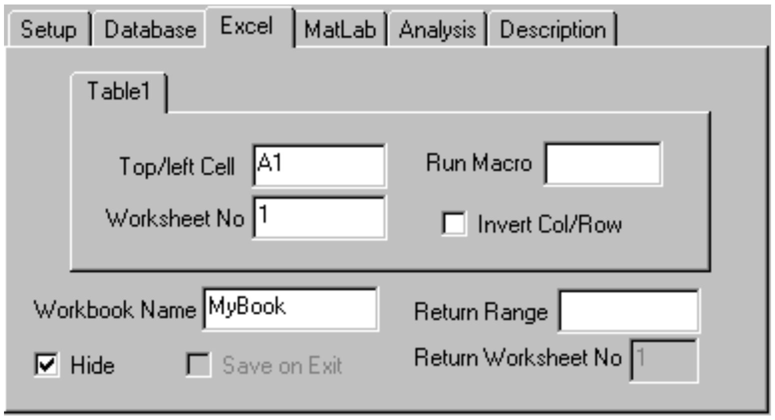
If “To Window” is selected, then a preview of the report will be displayed. The Preview window remains displayed until the user manually closes it.

9.3.4.2. To Printer/To File

The report is sent immediately to the appropriate device. There are no windows for the user to close as in the case of “To Window”.

9.4. Excel Property Page

This will allow data exchange with Excel Microsoft 2013 or higher.



Top/left Cell: A1
Worksheet No: 1
Run Macro:
Invert Col/Row: ☐
Workbook Name: MyBook
Return Range:
Hide: ☑
Return Worksheet No: 1

Figure 127: Result Mnemonic, Excel property page

Use the **Excel** property page to specify where and how each defined table should be exported to Microsoft Excel. Each defined Table can have its own export destination, including multiple locations on the same worksheet. Once exported, a user-defined Excel Macro can be executed, and the results returned back to PI-DATS.

Excel may be running or not when PI-DATS executes the Result mnemonic. If Excel is not running, it will be launched and the specified workbook opened. If the workbook does not exist, it will be created.

If Excel is already running, the workbook will be opened if it exists, or created if it does not. If the specified workbook is already open in Excel, and it has un-saved changes, Excel will display a dialog box prompting the user to discard changes before re-opening the workbook. If the user clicks Yes, Excel will discard any unsaved changes, re-open the workbook, and populate it with new data from the tables. If the user clicks No, Excel will populate the open workbook with the new data from the tables.

9.4.1. Top/left Cell (Required)

Enter the cell address where the top left datum of the table should be exported to. The default destination cell is **A1.** The destination cell should be different for each exported Table, and in general the exported data ranges should not overlap.

If the destination cell is blank for a table, that table will not be exported.

9.4.2. Worksheet No. (Required)

Enter the desired worksheet number in which this table will be inserted. The default value is 1. Data from a table can be exported to multiple worksheets at once. This can be used to create common header information s for multiple test reports on multiple sheets of a workbook. To send a table to multiple worksheets, separate the destination worksheet numbers using plus signs e.g. "1+2+4".

Data can also be exported to different worksheets on different loop iterations by entering worksheet numbers separated by commas. The worksheet entry "1,2,3" will enter data into worksheets 1,2, and 3 on loops 1,2, and 3, respectively. It is also possible to combine the “+” method and the comma method. For example, "1+2+3,4+5+6,7+8+9" will enter same values into worksheets 1-3 on loop 1, worksheets 4-6 on loop 2, etc.

The destination worksheet can also be incremented on loops. The entry "+1" will enter data into worksheets 1,2,3, etc., on each loop increment. The entry "+2" will enter data into worksheets 1,3,5,7, etc. on each loop increment.

The worksheet number can also be a PI-DATS integer variable.

If the specified worksheet number is larger than the last worksheet in the workbook, no data will be exported and the Result mnemonic will fail execution.

9.4.3. Run Macro (Optional)

Enter the full path to an Excel macro to be executed after the data has been imported into the spreadsheet.

9.4.4. Invert Col/Row

Check this box to transpose columns and rows.

9.4.5. Workbook Name

Enter the workbook name here, with or without a directory path. If this field is blank, then no Excel data is exported. If the specified Workbook name does not exist, a new workbook with this name will be created created.

If the workbook name is entered without a path, PI-DATS will look for the named file in the same directory that holds the current PIP file.

9.4.6. Hide

Check this if it is not necessary to view the worksheet results. Workbooks containing only intermediate calculations may be hidden during test runs. This will prevent the user from having to manually close the spreadsheet. Workbooks containing final output reports should not be hidden.

9.4.7. Save

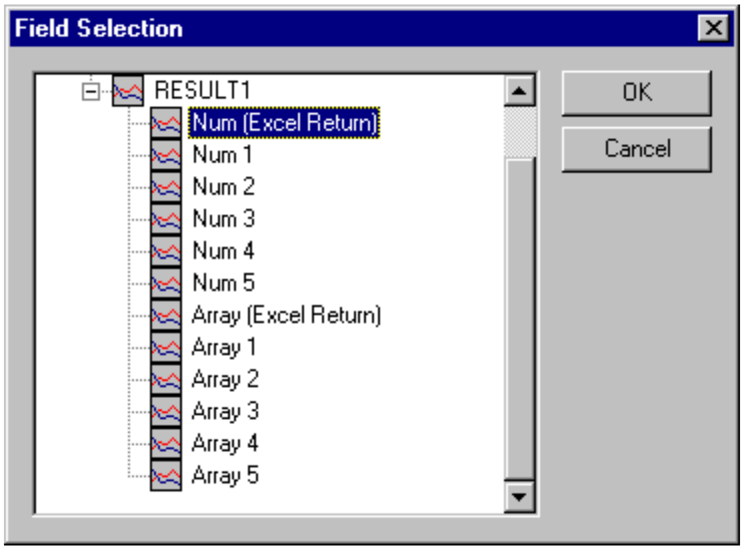
Check this to **Save** the workbook after exporting the data. Excel does not mark the workbook as “dirty” when data is exported to it from PI‑DATS, so it is possible for the user to manually close the workbook without saving, and thus discard exported data.

9.4.8. Don't Reopen File

By default the **Result** mnemonic will open (or re-open) the specified workbook from file before exporting data. If previous **Result** mnemonics have exported data to this workbook, but have not **Saved** it (see above), the workbook could be opened from file, and the latest data lost. Check this box to prevent **Result** from re-opening a workbook that is already open.

9.4.9. Return Range

It is possible to return spreadsheet results back into PI-DATS. For example, suppose that a spreadsheet has performed some math on data and has produced a range of results. These results may be imported back into PI-DATS by entering the range into this field. For example “F7:H9” would return a 3x3 range. The return range can be accessed by creating another subsequent RESULT mnemonic and inserting the appropriate selection into a Table. The field selection will look as follows:



RESULT1
- Num (Excel Return)
- Num 1
- Num 2
- Num 3
- Num 4
- Num 5
- Array (Excel Return)
- Array 1
- Array 2
- Array 3
- Array 4
- Array 5

Figure 128: Result Mnemonic, Excel Return Range

Select the Num (Excel Return) field if the Return Range is a single cell. Select the Array (Excel Return) field if the Return Range is a range of cells.

9.4.10. Return Worksheet No

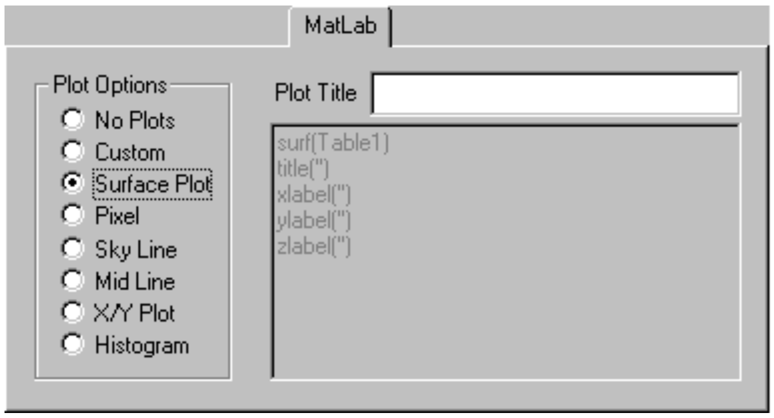
Enter the worksheet number containing the range of data to be imported into PI-DATS. (Refer to “**Return Range**” described above).

9.4.11. Save

If the **Save** box is checked the Excel worksheet will be saved upon completion of the export operation. This is useful when multiple passes of a loop must export data to the same spreadsheet. If **Save** is not checked then each pass of the loop will reopen the same original file, and only the data from the last loop iteration will be exported.

9.5. MatLab Property Page

This is the interface to make MatLab plots.


MatLab

Plot Options
- No Plots
- Custom
- [✓ Surface Plot]
- Pixel
- Sky Line
- Mid Line
- X/Y Plot
- Histogram

Plot Title

```
surf(Table1)
title('')
xlabel('')
ylabel('')
zlabel('')
```

Figure 129: Result Mnemonic, MatLab property page

9.5.1. Custom

Allows operator to enter custom MatLab commands.

9.5.2. Surface

Makes a 3-D color-shaded surface plot of Table1 (must be Array).

9.5.3. Pixel

Makes a 2-D color-shaded surface plot of Table1 (must be Array).

9.5.4. Sky Line

Makes a single Line plot of entire array of Table 1 (must be Array).

9.5.5. Mid Line

Makes a single Line plot of the middle row of array of Table 1 (must be Array).

9.5.6. X/Y Plot

Plots Field1 (X) versus Field2 (Y).

9.5.7. Histogram

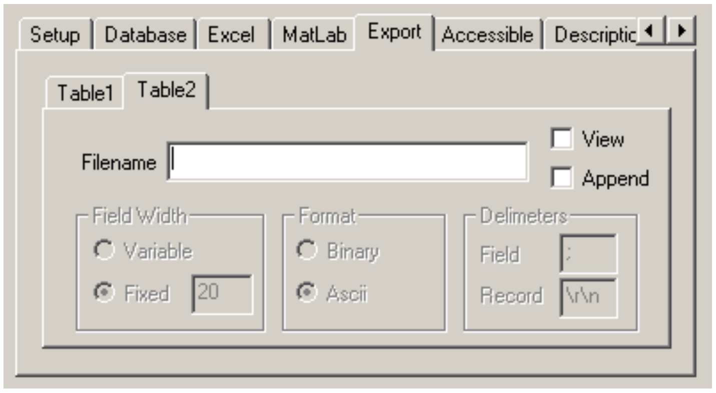
Makes a 10 bin histogram plot of items in Table1.

9.6. Export property page

The **Export** property page enables exporting of each data table to a file for use by PI-DATS, PI-PLOT, or by another application. The **Export** property page appears in two formats, depending on the type of table.

9.6.1. Exporting Data Tables

If the table is composed of scalar column entries, the data may be exported as ASCII text with several different options:



Filename: (empty)
View: ☐
Append: ☐
Field Width: Fixed [✓], 20
Format: Ascii [✓]
Delimeters: Field ; Record \r\n

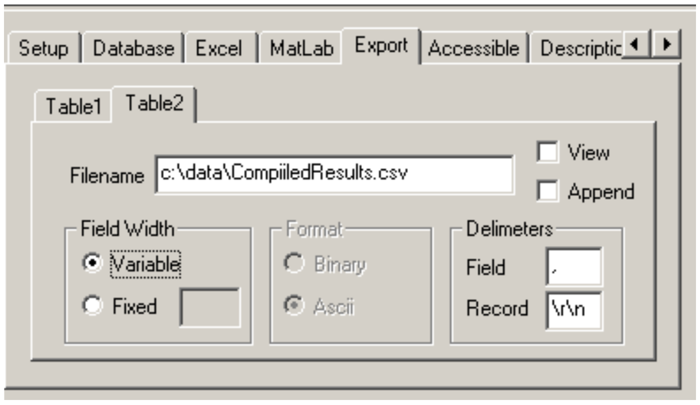
Figure 130: Result Mnemonic, Export property page

Type the destination path and filename in the **Filename** box. If you wish to append the new data to the file instead of replacing the existing file, click the **Append** checkbox.

To view the data on the screen as a table of data, check the **View** checkbox. Exporting data to the screen can be useful when diagnosing or debugging a test plan, or it may be used as a "program status" indicator to display the current state of variables or test results. It may be desirable to insert a **Halt** mnemonic after a Result mnemonic exports to the screen.

When **View** is checked, the **Filename** text entry box will be disabled.

After the **View** box is un-checked and a filename has been entered, the export controls will be enabled:



Filename: c:\data\CompiiledResults.csv
View: ☐
Append: ☐
Field Width: Variable [✓]
Format: Ascii [✓]
Delimeters: Field , Record \r\n

Figure 131: Result Mnemonic, Export property page

The **Field Width** radio button controls whether the text file will have variable or fixed widths per entry. If a **Fixed** width is chosen, any data longer than the fixed width will be truncated.

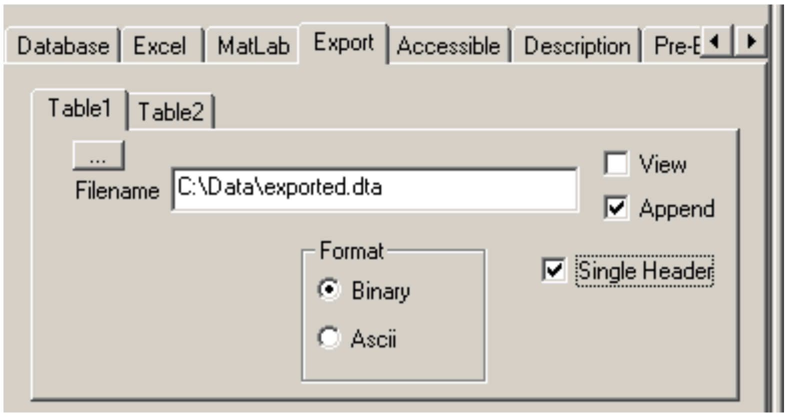
Use the **Delimiters** entries to specify the character(s) to append to the end of each **Field** and to the end of each **Record**. Escaped character notation may be used to specify non-typed characters, such as “\t” for a tab character, and “\r\n” for a carriage-return and line-feed combination.

9.6.2. Exporting Image Data

If the table is composed of an array of image data, there are two options, image export and NUC export.

9.6.2.1. Image Export

For image export, the default choice, the only settable format parameter is the **Binary** or **ASCII** format:



Filename: C:\Data\exported.dta

View [ ]
Append [✓]
Single Header [✓]

Format: [✓ Binary] Ascii

Figure 132: Result Mnemonic, Export property page

Data saved from an array table in binary format with a “.dta” extension can be viewed in PI-PLOT or inserted into another RESULT mnemonic as a file.

To output multiple frames of data to a single file, use the **Append** and **Single Header** checkboxes in conjunction with conditional execution in a loop. For example, create a Boolean flag variable (e.g. “FileCreated”) with an initial value of False, and two Result mnemonics that both take the same set of input frames.

In the second Result mnemonic, export the frame to a filename, leaving the **Append** and **Single Header** boxes unchecked. On the **Pre-Execute** property page, execute this mnemonic only if the FileCreated variable is False, and on the Post-Execute property page, set the FileCreated variable to True.

In the first Result mnemonic, export the frame to the same filename, but check both the **Append** and **Single Header** checkboxes. Execute this mnemonic only if FileCreated is True.

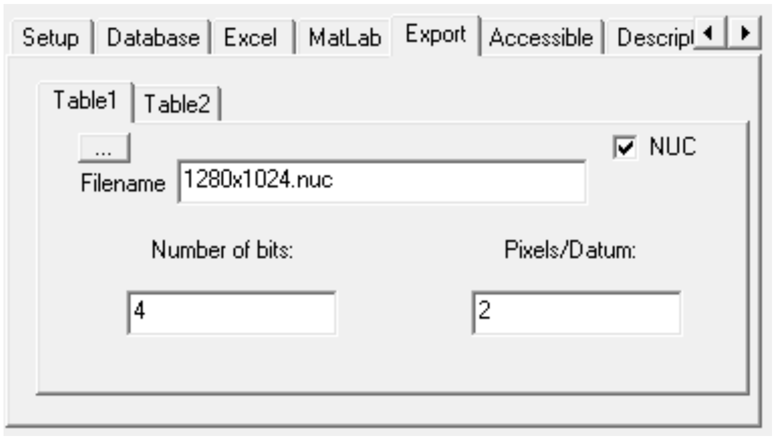
On the first iteration of the loop, the output file will be created with the first frame of data, replacing any existing file of the same name. On every subsequent loop iteration, the new frames will be appended to the file, and the file header information will automatically be incremented by one frame.

Note that, when using the flag method of execution control, the "create" mnemonic should be placed below the "append" mnemonic in the test plan, otherwise both mnemonics will execute on the first iteration of the loop, and the first frame of data will appear twice in the output file.

9.6.2.2. NUC Export

Floating point image data can be exported as a scaled, N-bit digital value, to allow injection of non-uniformity correction (NUC) vectors into ROICs that support NUC in hardware. See the application note at **Section** 12. NUC Vector Injection for a more complete description of the procedure.

To enable NUC export, check the NUC checkbox and enter a filename (typically with a .nuc extension):



☑ NUC

Filename: 1280x1024.nuc

Number of bits: 4

Pixels/Datum: 2

Figure 133: Result Mnemonic, NUC Export property page

Each pixel will be converted from floating point to its unsigned binary integer representation, right-aligned, according to the specified **Number of bits**. Fractional values will be truncated, e.g. 11.213 converted to 4 bits will result in 1011., and 3.11 will convert to 0011.

Converted data will be packed for storage according to the **Pixels/Datum** setting. The **Number of bits** x **Pixels/Datum** will be rounded up to the nearest logical storage size (8-bit Byte, 16-bit WORD, or 32-bit DWORD) and right-aligned, with any remaining leading bits packed with zeroes.

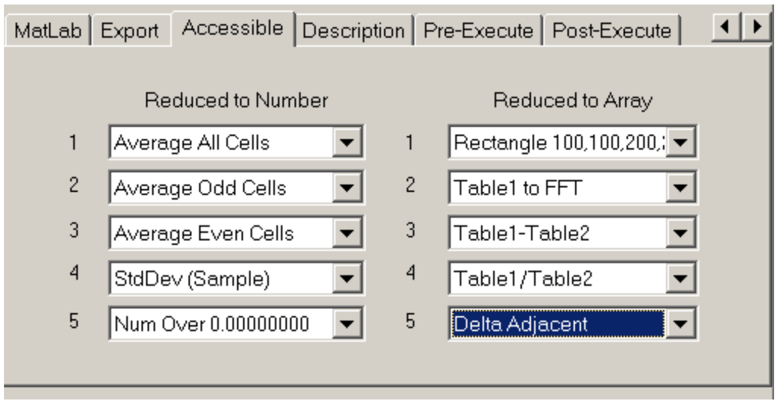
Data saved from an array table in NUC format with a ".nuc" extension can then be injected to a PI-PAT timing file using the **NUC Pattern** mnemonic, described in **Section** 6.5. NUC Pattern Injection**.**

9.7. Accessible property page

The **Accessible** property page reduces data into other meaningful forms. Data can be converted into a single number (scalar) or arrays can be operated on to produce other arrays. Choose one or more data reduction options from the drop menus.

Some reduction options require additional arguments to be entered. Selecting these menu items will display a dialog box allowing the user to enter the required arguments.

Some reduction options treat the data set as a two-dimensional array, while others treat it as a one-dimensional array of sequential pixels. For one-dimensional operations, pixels in two-dimensional data sets are numbered in ascending order by pixel and then by row, e.g. in an ***m x n*** array, the pixel at address ***(x, y)*** will be numbered as ***xm + y***.



	Reduced to Number		Reduced to Array
1	Average All Cells	1	Rectangle 100,100,200,:
2	Average Odd Cells	2	Table1 to FFT
3	Average Even Cells	3	Table1-Table2
4	StdDev (Sample)	4	Table1/Table2
5	Num Over 0.00000000	5	Delta Adjacent

Figure 134: Result Mnemonic, Accessible property page

9.7.1. Reduced to Number

The **Reduced to Number** functions each operates on an array and returns a scalar value. These scalar values can then be assigned to defined variables and/or exported to reports. If the Result mnemonic is placed in a loop the resulting array scalar values can be inserted as a column in subsequent Result mnemonic.

9.7.1.1. Average All Cells

Returns the average all the cells (pixels) of the array.

9.7.1.2. Average Odd Cells

Returns the average of only the odd-numbered pixels.

9.7.1.3. Average Even Cells

Returns the average of only the even-numbered pixels.

9.7.1.4. Get Elements 0 +- 0

Returns the average of a subset of cells (pixels). Selecting this item will display a dialog box allowing you to enter the center pixel number and the number of pixels above and below to average. For example: “Get Elements 7 +- 2” would average pixels 5, 6, 7, 8 and 9.

Pixels are numbered starting at zero, and integer variables or fixed values may be used.

9.7.1.5. Maximum

Returns the maximum value in the array.

9.7.1.6. Minimum

Returns the minimum value in the array.

9.7.1.7. Num Over 0.00

Returns the number of cells (pixels) that are greater than a given value. Selecting this item will display a dialog box allowing you to enter a constant or a user-defined variable.

9.7.1.8. Num under 0.00

Returns the number of cells (pixels) that are less than a given value. Selecting this item will display a dialog box allowing you to enter a constant or a user-defined variable.

9.7.1.9. StdDev (Pop)

Returns the Standard Deviation (population) of the entire array. (Fastest if array size is a power of 2)

9.7.1.10. StdDev (Sample)

Returns the Standard Deviation (sample) of the entire array. (Fastest if array size is a power of 2)

9.7.1.11. 50% Width

Returns the width of the cross section at the 50% contour (see **Section 9.7.2.15. 50% Array**).

9.7.1.12. 50% Height

Returns the height of the cross section at the 50% contour (see **Section 9.7.2.15. 50% Array**).

9.7.1.13. Array Info (Num Columns, Num Rows, Num Frames, Data Type, Num Bits)

Each of these choices returns selected the indicated data from the header for Table1. For example the **Num Frames** choice will return the number of frames in the original data set. This result can then be assigned to a variable which can then be used as a looping control, for example.

If the requested data type does not exist because the input array is not an image array, the result will be zero.

9.7.2. Reduced to Array

The **Reduced to Array** functions each operate on an array and each returns an array. These arrays can be inserted into subsequent Result mnemonics where they can be exported to Excel, exported to files, or processed again. In the following examples the size of the returned array assumes the input array is **m x n**.

9.7.2.1. Get Cells 0-0

Return a selected portion of the array from the ***xth*** to the ***yth*** pixel. The Pixels are numbered from 1 to the end of the array. For example: "Get Cells 3-7 would return pixels 3,4,5,6,7.

Pixels are numbered starting at zero, and integer variables or fixed values may be used.

9.7.2.2. Rectangle 0,0,1,1

Return a rectangular section of the array. Parameters are Left column, Top row, Width, Height. For example, "Rectangle 3,2,2,2 would return a 2 x 2 array of pixels starting at row 2 column 3.

The starting pixels are numbered starting at zero, and integer variables or fixed values may be used, including negative numbers (see below). The width and height are 1-based, and must be greater than 1.

If the specified rectangle extends beyond the original data table, any "out of bounds" pixels will be returned with a value of 0. For example, if a 200 x 200 array is passed into a Result mnemonic, but returned as a Rectangle (-100, -100, 400, 400), the resulting rectangle would be 400 x 400 and have the original data in the "middle" 200 x 200, padded on all sides by 100 rows and columns of zeroes.

When variables are used, the legend displayed in the **Reduce to Array** drop-menu will be "Rectangle X, Y, Z, W" to indicate that the arguments are variables.

9.7.2.3. Table1 to Array

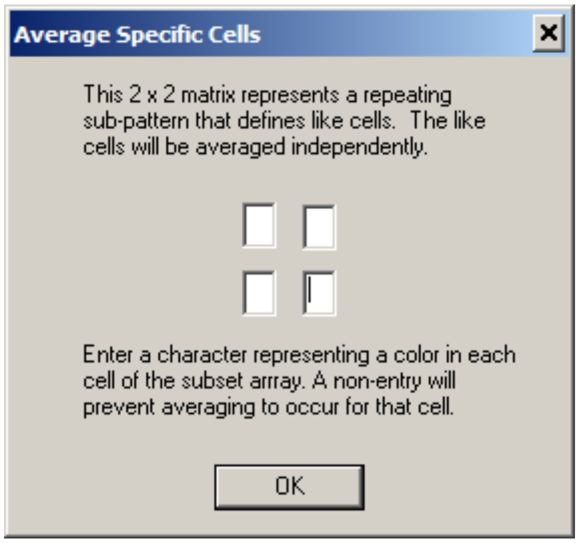
Returns the data in Table1 as an array. This array can then be used in another Result mnemonic.

9.7.2.4. Table1 to Transposed Array

Returns the data in Table1 as an array, but with each row of Table1 returned as a column. This array can then be used in another Result mnemonic.

9.7.2.5. AvgRepPat2x2

Returns different averages for cells across a repeating pattern. Selecting this option displays a dialog box:

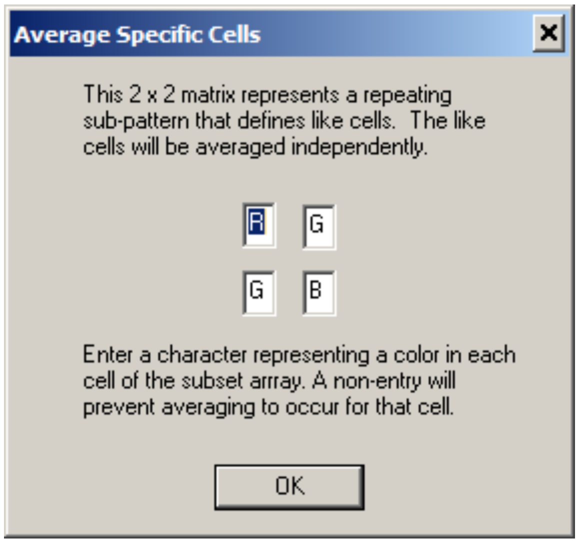


This 2 x 2 matrix represents a repeating sub-pattern that defines like cells. The like cells will be averaged independently.

Enter a character representing a color in each cell of the subset array. A non-entry will prevent averaging to occur for that cell.

Figure 135: Result Mnemonic, Accessible property page

Each cell of the matrix may be populated with a “key” that determines which cells will be averaged.



This 2 x 2 matrix represents a repeating sub-pattern that defines like cells. The like cells will be averaged independently.

R	G
G	B

Enter a character representing a color in each cell of the subset array. A non-entry will prevent averaging to occur for that cell.

Figure 136: Result Mnemonic, Accessible property page

For example the above entry will return an array of 3 values—one average for the “R” cells (even column, even row), one average for the “B” cells (odd column, odd row), and one for the average of the “G” cells (E/O and O/E).

9.7.2.6. Table1 to FFT

Returns an array that is an FFT of Table1. (Fastest if the array size is a power of 2)

9.7.2.7. Table1+Table2

Returns an array where each element is the sum of the corresponding elements in Table1 and Table2. Table1 and Table 2 must be the same size; otherwise PI-DATS will return an array of zero values.

PI-DATS can also perform “saturation” math which is often useful for image data. When using saturation math, integer operations that would ordinarily overflow a standard integer will instead “saturate.” For example, 0xFFF0 + 0x0200 = 0xFFFF when using saturation math.

9.7.2.8. Table1-Table2

Returns an array where each element is the difference of the corresponding elements in Table1 and Table2. Table1 and Table 2 must be the same size; otherwise PI-DATS will return an array of zero values.

PI-DATS can also perform “saturation” math which is often useful for image data. When using saturation math, integer operations that would ordinarily underflow a standard integer will instead “saturate.” For example, 0x0200 - 0x0800 = 0x0000 when using saturation math.

9.7.2.9. Table1*Table2

Returns an array where each element is the product of the corresponding elements in Table1 and Table2. Table1 and Table 2 must be the same size; otherwise PI-DATS will return an array of zero values.

PI-DATS can also perform “saturation” math which is often useful for image data. When using saturation math, integer operations that would ordinarily overflow a standard integer will instead “saturate.” For example, 0xFFF0 * 0x002 = 0xFFFF when using saturation math.

9.7.2.10. Table1/Table2

Returns an array where each element is the ratio of the corresponding elements in Table1 and Table2. Table1 and Table 2 must be the same size; otherwise PI-DATS will return an array of zero values. If any element of Table2 is zero, the returned array will contain have an undetermined value at that location.

9.7.2.11. Table1?Table2

Returns an array where each element is 1 only if the corresponding elements in Table1 and Table2 are equal, otherwise the element contains 0. Table1 and Table 2 must be the same size; otherwise PI-DATS will return an array of zero values.

9.7.2.12. Delta Adjacent

Returns a one-dimensional array containing the differences $\boldsymbol{pixel_{n+1} - pixel_n}$. The returned array has a size $\boldsymbol{m \times n - 1}$.

9.7.2.13. Find Over

Returns an array that is filled with zeroes except for those cells that are greater than the specified value.

9.7.2.14. Find Under

Returns an array that is filled with zeroes except for those cells that are less than the specified value.

9.7.2.15. 50% Array

Returns an array with pixel values capped at the average value of the original array. For example, assume a 16 x 16 array with the following values were passed into the Result mnemonic:

0.1	0.6	1.0	1.3	1.4	1.5	1.5	1.5	1.5	1.5	1.5	1.4	1.3	1.0	0.6	0.1
0.6	1.3	1.8	2.2	2.4	2.5	2.5	2.5	2.5	2.5	2.5	2.4	2.2	1.8	1.3	0.6
1.0	1.8	2.5	3.0	3.3	3.4	3.5	3.5	3.5	3.5	3.4	3.3	3.0	2.5	1.8	1.0
1.3	2.2	3.0	3.6	4.1	4.4	4.5	4.5	4.5	4.5	4.4	4.1	3.6	3.0	2.2	1.3
1.4	2.4	3.3	4.1	4.8	5.3	5.5	5.5	5.5	5.5	5.3	4.8	4.1	3.3	2.4	1.4
1.5	2.5	3.4	4.4	5.3	6.0	6.4	6.5	6.5	6.4	6.0	5.3	4.4	3.4	2.5	1.5
1.5	2.5	3.5	4.5	5.5	6.4	7.2	7.5	7.5	7.2	6.4	5.5	4.5	3.5	2.5	1.5
1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.4	8.4	7.5	6.5	5.5	4.5	3.5	2.5	1.5
1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.4	8.4	7.5	6.5	5.5	4.5	3.5	2.5	1.5
1.5	2.5	3.5	4.5	5.5	6.4	7.2	7.5	7.5	7.2	6.4	5.5	4.5	3.5	2.5	1.5
1.5	2.5	3.4	4.4	5.3	6.0	6.4	6.5	6.5	6.4	6.0	5.3	4.4	3.4	2.5	1.5
1.4	2.4	3.3	4.1	4.8	5.3	5.5	5.5	5.5	5.5	5.3	4.8	4.1	3.3	2.4	1.4
1.3	2.2	3.0	3.6	4.1	4.4	4.5	4.5	4.5	4.5	4.4	4.1	3.6	3.0	2.2	1.3
1.0	1.8	2.5	3.0	3.3	3.4	3.5	3.5	3.5	3.5	3.4	3.3	3.0	2.5	1.8	1.0
0.6	1.3	1.8	2.2	2.4	2.5	2.5	2.5	2.5	2.5	2.5	2.4	2.2	1.8	1.3	0.6
0.1	0.6	1.0	1.3	1.4	1.5	1.5	1.5	1.5	1.5	1.5	1.4	1.3	1.0	0.6	0.1

Figure 137: Sample Array, pixels over mean value shaded for clarity

PI-DATS would compute the average value to be 3.45 and return the following as the **50% Array**:

0.1	0.6	1.0	1.3	1.4	1.5	1.5	1.5	1.5	1.5	1.5	1.4	1.3	1.0	0.6	0.1
0.6	1.3	1.8	2.2	2.4	2.5	2.5	2.5	2.5	2.5	2.5	2.4	2.2	1.8	1.3	0.6
1.0	1.8	2.5	3.0	3.3	3.4	3.5	3.5	3.5	3.5	3.4	3.3	3.0	2.5	1.8	1.0
1.3	2.2	3.0	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.0	2.2	1.3
1.4	2.4	3.3	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.3	2.4	1.4
1.5	2.5	3.4	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.4	2.5	1.5
1.5	2.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	2.5	1.5
1.5	2.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	2.5	1.5
1.5	2.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	2.5	1.5
1.5	2.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	2.5	1.5
1.5	2.5	3.4	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.4	2.5	1.5
1.4	2.4	3.3	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.3	2.4	1.4
1.3	2.2	3.0	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.0	2.2	1.3
1.0	1.8	2.5	3.0	3.3	3.4	3.5	3.5	3.5	3.5	3.4	3.3	3.0	2.5	1.8	1.0
0.6	1.3	1.8	2.2	2.4	2.5	2.5	2.5	2.5	2.5	2.5	2.4	2.2	1.8	1.3	0.6
0.1	0.6	1.0	1.3	1.4	1.5	1.5	1.5	1.5	1.5	1.5	1.4	1.3	1.0	0.6	0.1

Figure 138: 50% Array

PI-DATS can also return the width and height of the contour at the average value (please see **Section 9.7.1.11. 50% Width** and **Section 9.7.1.12. 50% Height** functions) by computing the difference between the maximum and minimum row or column number with pixels exceeding the average value. In this example, PI-DATS would return 12 for the width and 12 for the height.

9.7.2.16. Table1 to Replacement Table

Expects an input table of bad pixels and returns a table of replacement vectors. PI-DATS evaluates the input table as Booleans; therefore any pixel with a value of 0 is good, and any non-zero value is bad. For each bad pixel location, PI-DATS calculates a vector to the “nearest” good pixel, based on a “spiral-search” algorithm.

9.7.2.17. Table1 Repl(Tbl2)

Expects an input image (Table1) and an input Replacement Table (Table2). Returns a corrected image array, replacing all bad pixels with good pixels based on the replacement table.

9.7.2.18. Table1 to Clamp

Returns Table1 with values clamped at upper- and lower-bounds. When Table1 to Clamp is selected, PI-DATS will display a dialog box enabling the user to enable and enter values for the upper and/or lower bounds. Clamping values can be fixed values or variables.

9.7.2.19. Sum All Tables

Returns a table that is the sum of all defined tables in the Result mnemonic. The dimensions and type (Int or Float) of the output table is the defined at run-time by the first non-null input table that PI-DATS encounters when summing the tables. An input table may be null if it references a mnemonic that did not execute. Once the table size and type has been established, PI-DATS will sum all subsequent tables of the same size and type, and ignore any subsequent table of different size or type.

9.7.2.20. Table1 to Float

Returns a table converted to the Floating Point data type.

9.7.2.21. Interleave Tbls (Interleave Tables)

Interleaves all defined tables into a single table, by Rows or by Columns. Selecting **Interleave Tbls** will display a dialog box, allowing the user to choose **Rows** or **Cols** for the interleave.

Interleave by Columns example:

Input Table1:

1	2	3
4	5	6
7	8	9

Input Table2:

10	11	12
13	14	15
16	17	18

Input Table3:

19	20	21
22	23	24
25	26	27

Output Table:

1	10	19	2	11	20	3	12	21
4	13	22	5	14	23	6	15	24
7	16	25	8	17	26	9	18	27

Interleave by Rows example:

Input Table1:

1	2	3
4	5	6
7	8	9

Input Table2:

10	11	12
13	14	15
16	17	18

Input Table3:

19	20	21
22	23	24
25	26	27

Output Table:

1	2	3
10	11	12
19	20	21
4	5	6
13	14	15
22	23	24
7	8	9
16	17	18
25	26	27

9.7.2.22. Tbl1 to Bool (Table1 to Boolean)

Returns a table where each cell is converted to either 1 or 0. Cells containing 0 retain their value of 0. Cells containing any other value are converted to 1.

9.7.3. Notes on Table Math

Many of the data reduction features in the Result mnemonic perform mathematical transforms on multiple data tables (e.g. Table1 + Table2 or Table1*Table2). All PI-DATS math requires that the operand tables be of the same data type. For example a table of Float data cannot be added to a table of Integer data.

If math is attempted on data of different types, PI-DATS will return an array of zeroes.

Use Table1 to Float to convert integer data to the Floating Point type before performing operations with other Floating Point tables.

9.7.4. Passing Reduced Data To Other Mnemonics

The reduced data can be accessed by creating another Result mnemonic below the current mnemonic. The data reduction items chosen in the first mnemonic will appear as **Field Selection** items in any subsequent **Result** mnemonic. For example, if **Figure 134** were the **Accessible** property page for Result2, then any Result mnemonic inserted below Result2 would have the following Fields available to it:

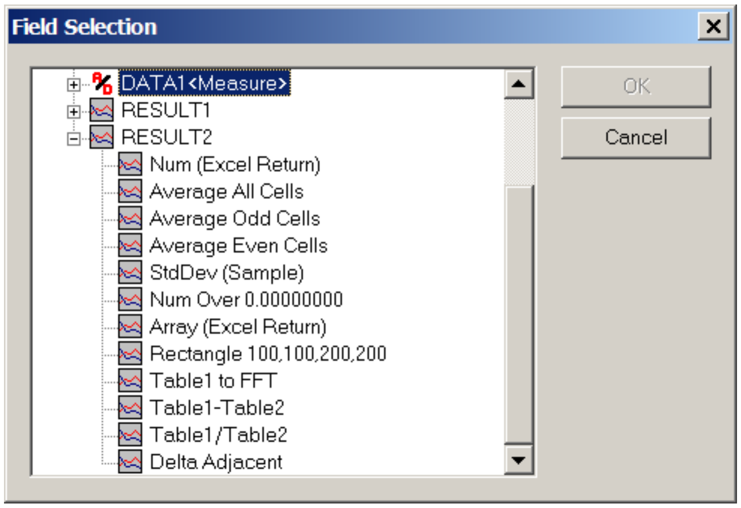


Figure 139: Result Mnemonic, Passing Reduced Data

In this way, mnemonics can be chained together to perform complex image analysis such as Non-Uniformity-Correction (NUC), etc. Using the Excel, Matlab and general Export features, data can also be passed to third-party application for specialized processing.

If a Result mnemonic reads scalar outputs from a Result mnemonic in an inner loop, it will be read in as a column of data. If a Result mnemonic reads array outputs from a Result mnemonic in an inner loop, it will only read in the output from the last loop iteration.

To pass multiple frames from Result mnemonics in inner loops, export them to a file (using the create/append methodology described above) and read them into the outer Result mnemonic.

9.7.5. Passing Reduced Data To Variables

Data reduced to scalar values can also be passed into PI-DATS variables by using the Post-Execute property page.

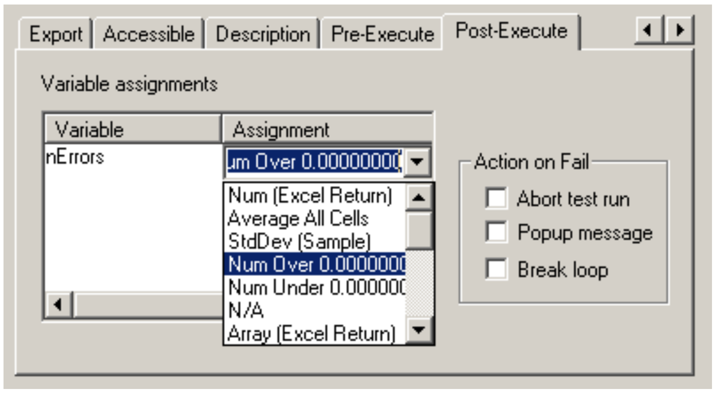


Figure 140: Result Mnemonic, Passing Reduced Data

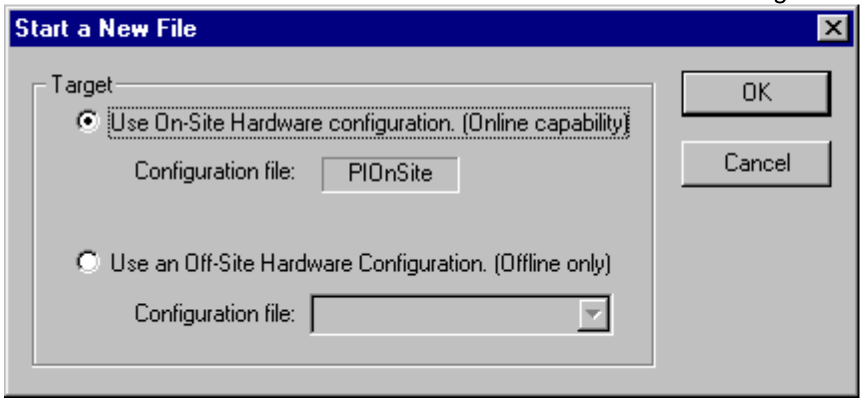
If any **Reduce To Number** selections have been entered, they will appear as choices in the **Variable Assignments** portion of the **Post-Execute** property page. To assign a **Reduce To Number** result to a variable, click in the empty space under the **Variable** column heading and choose an existing defined variable, then select the desired value in the **Assignment** column. To remove a variable assignment, click the variable name and type **Delete**.

10. Quick Start Tutorial

10.1. Building a Basic Test Plan

The following is a brief tutorial on how to build a basic Test Plan. The intent of this example Test Plan will be to command a BIAS card to 10 Volts and Pause 5 seconds before completing the test run.

Start with a new file. Select **New** from the File menu. The following will be displayed:



Start a New File

Target

[✓ Use On-Site Hardware configuration. (Online capability)]

Configuration file: PIOnSite

Use an Off-Site Hardware Configuration. (Offline only)

Configuration file:

Figure 141: Sample Test Plan, Start New File

Select **OK**.

Add a BIAS mnemonic. Select **Append Mnemonic** from the **Build** menu or press . Select a **BIAS** mnemonic from the **Mnemonic Selection**. Then select **OK**.

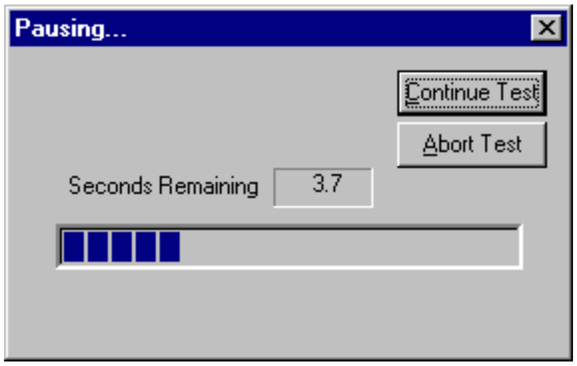
Enter **10** in the **Voltage** entry box.

Add a PAUSE mnemonic. Select **Append Mnemonic** from the **Build** menu or press . Select **Suspend Execution Commands**, then **PAUSE** mnemonic from the Mnemonic Selection display. Then select **OK**.

Enter **5** seconds in the **Seconds of pause** entry box.

Run the test. Select **Run Test Plan** from the **Tests** menu or press .

The BIAS instrument will be set to a level of 10 Volts and the following display will appear for 5 seconds.



Seconds Remaining 3.7

Figure 142: Sample Test Plan, Pause countdown

10.1.1. Building an Initialization Group

The following is a brief tutorial on how to build a section of your test plan for initialization. The intent of this example Test Plan will be to initialize the DVM, reset all clock driver and DC Bias cards, set BIAS1 card to 10 Volts, and set all 40460 clock drivers with a high level of 10 Volts.

Start with a new file. Select **New** from the File menu. Select **OK** when presented with the **Start a New File** display.

Insert a Group Header. Select **Insert Group within Group** from the **Build** Menu. This will be our initialization group header. Click the text **Group Header** and rename it to **Initialization**.

Insert a **DVM** mnemonic. Select **Insert Mnemonic within Group** from the **Build** menu. (Do not press Mn as this will not insert the mnemonic within the group). Open the **Digital Volt Meters** group then select a **DVM <Setup>** mnemonic from the Mnemonic Selection display. Then select **OK**.

Append an **XMFRM** mnemonic. Select **Append Mnemonic** from the **Build** menu or click Mn. Select an **XMFRM** mnemonic from the Mnemonic Selection. Then select **OK**. Select the **Reset All Cards** check box. Make sure the **State** is set to **OffLine**.

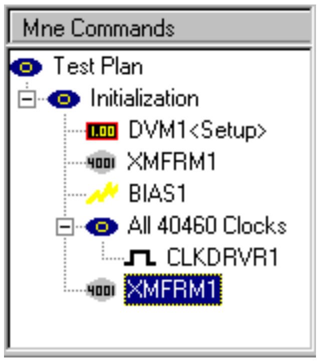
Append a **BIAS** mnemonic. Select **Append Mnemonic** from the **Build** menu or click Mn. Select a **BIAS** mnemonic from the Mnemonic Selection. Then select **OK**. Enter **10.0** in the **Voltage** field.

Add a Group Header to this group for clock driver enumeration. Select **Append Group** from the **Build** Menu or press ●. This will allow us to loop through all the 40460 clock drivers. Click the text “Group Header” and rename it to **All 40460 Clocks**. Set the **Num Executions** to the number of 40460 Clock channels installed on the system.

Insert the first **CLKDRV** mnemonic. Select **Insert Mnemonic within Group** from the **Build** menu. (Do not press Mn as this will not insert the mnemonic within the group). Select the first **CLKDRV** mnemonic from the Mnemonic Selection. Then select **OK**. Set the **High Level** voltage to 10 Volts. Also select the **Looping** tab and check the **Increment to next clock** check box. This will turn on the enumeration mechanism.

Finally, append an **XMFRM** mnemonic. First click on the "**All 40460 Clocks**" and select **Append Mnemonic** from the **Build** menu or click . Select an **XMFRM** mnemonic from the Mnemonic Selection. Then select **OK**. Set the **State** to **OnLine**.

The display should look like this:



```
Mne Commands
Test Plan
  Initialization
    DVM1<Setup>
    XMFRM1
    BIAS1
    All 40460 Clocks
      CLKDRVR1
    XMFRM1
```

Figure 143: Sample Test Plan, Mainframe mnemonic

You now have an initialization group that will be executed in your Test Plan.

10.1.2. Building a Test Plan with Loops

The following is a brief tutorial on how to build a Test Plan with nested loops. The intent of the example Test Plan will be to program a BIAS card to step the voltage from 5 volts to 9 volts at 1 volt increments. Within each BIAS stepped voltage we will step a Clock Driver's High Voltage from 2.5 Volts to 4.5 Volts. We will also display a message before each Bias step takes effect.

Start with a new file. Select **New** from the File menu. Select **OK** when presented with the **Start a New File** display.

Insert a Group Header. Select **Insert Group within Group** from the **Build** Menu. This will be our outside loop for the BIAS instrument. Click the text **Group Header** and rename it to **Outside Loop**.

Enter **5** for the **Num executions**.

Check the **Message** box, and then click the **Edit** button.

Enter the text "**About to step the BIAS voltage,**" then press **OK**.

Insert a **BIAS** mnemonic. Select **Insert Mnemonic within Group** from the **Build** menu. (Do not press as this will not insert the mnemonic within the group). Select a **BIAS** mnemonic from the **Mnemonic Selection**. Then click **OK**.

Enter **5** in the **Voltage** entry box.

Select the **Looping** tab.

Enter **1** in the **Voltage step per loop** entry box.

Add a **Group Header** to this group. Select **Append Group** from the **Build** Menu or press . This will be our inside loop for the Clock Driver instrument. Click the text “Group Header” and rename it to it **Inside Loop**.

Enter **5** for the **Num executions**.

Insert a Clock Driver mnemonic. Select **Insert Mnemonic within Group** from the **Build** menu. Select a **CLKDRVR** mnemonic from the **Mnemonic Selection**. Then select **OK**.

The display should look like this:

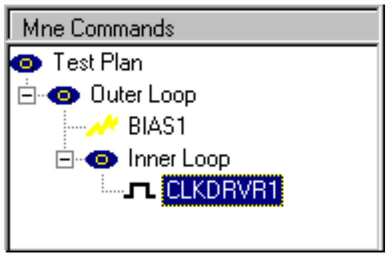


Figure 144: Sample Test Plan, Clock Driver mnemonic

Enter **2.5** for the **High Level**.

Select the **Looping** tab.

Enter **.5** in the **High Level Step** entry box.

Run the test. Select **Run Test Plan** from the **Tests** menu or press . The Message window displaying the text "**About to step the BIAS voltage"** should display 5 times. The Test Run should then terminate.

10.1.3. Preparing a Test Plan for Reporting Results

Results can only be extracted using the Results Function module. Raw data is brought into this module in the form of fields or arrays into a table. Each field placed into a table serves as a column definition for a report or an axis definition for a plot. Arrays placed into a table take up the entire table. Therefore each table can contain either fields or an array, but not both.

Create a test plan with looping, such as the Bias loop example shown above. Place a RESULT mnemonic outside the loop of interest in order to retrieve looped data. It is important to note that the Result mnemonic must be placed ***outside*** the loop.

The following is an example of how to prepare looped data for reporting.

Given a Test Plan with a looped BIAS and DVM command/measure pair as shown below.

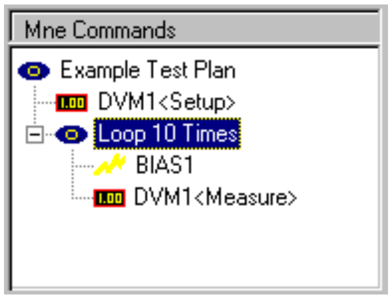


Figure 145: Sample test Plan, Bias Loop

Make sure that the Group Header **Loop 10 Times** is selected as shown. Select a RESULT mnemonic by clicking the Mn button and select RESULT1. The display will look as follows:

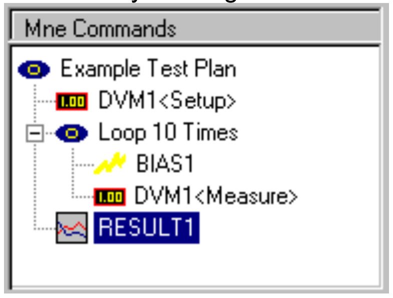


Figure 146: Sample Test Plan, Result Mnemonic

Click on **Insert Right** and Select **Loop 10 Times** then **BIAS1** then **Command Voltage**. This inserts the first field into Table1.

Click on **Insert Right** and Select **Loop 10 Times** then **DVM1<Measure>** then **Reading**. This inserts the second field into Table1 (to the right of the first field).

This Result module should look as follows:

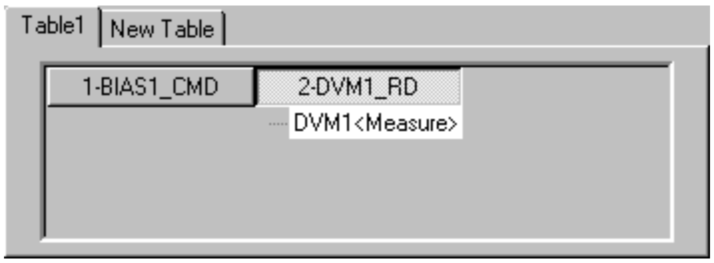


Figure 147: Sample Test Plan, Table Definition

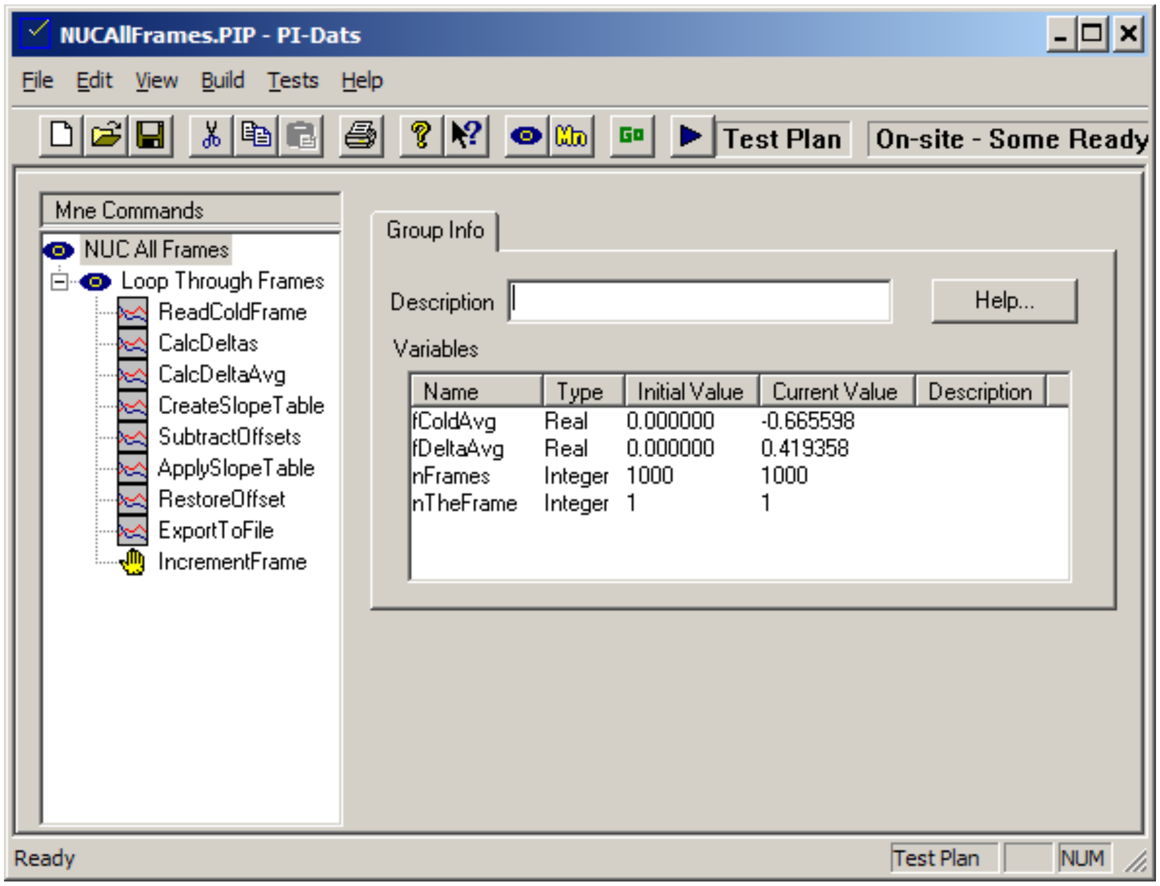
The newly-formed Table can now be used for Database, Excel, MatLab, or Analysis activities.

11. Image Processing in PI-Result

PI-Result can also be used to perform post-processing of image data after it is acquired. Post-processing operations such as Non-Uniformity Correction (NUC), histogram equalization, etc., can be performed in PI-DATS.

The following example shows how a basic, two-point non-uniformity correction can be applied to 1000 frames of previously acquired data. For this example, three data files must already exist, having been saved from the DACQ mnemonic after being acquired:

- cold.dta, a “reference” frame captured with a cold object covering the DUT’s field of view.
- hot.dta, a “reference” frame captured with a hot object covering the DUT’s field of view.
- acquired.dta, 1000 frames of raw image data



Name	Type	Initial Value	Current Value	Description
fColdAvg	Real	0.000000	-0.665598	
fDeltaAvg	Real	0.000000	0.419358	
nFrames	Integer	1000	1000	
nTheFrame	Integer	1	1	

Figure 148: Test Plan for Non-Uniformity Correction (NUC)

11.0.1. Test Plan Overview

In the test plan above, PI-DATS will read in a “cold” frame and a “hot” frame and use those two data sets to calculate the NUC table.

1. First, the “cold” frame is read from a file. This will serve as the offset-correction table.

2. The average value is calculated and assigned to the variable “fColdAvg.”

3. Next, the cold frame is subtracted from the hot frame to produce a frame of deltas. The average delta (fDeltaAvg) is calculated.

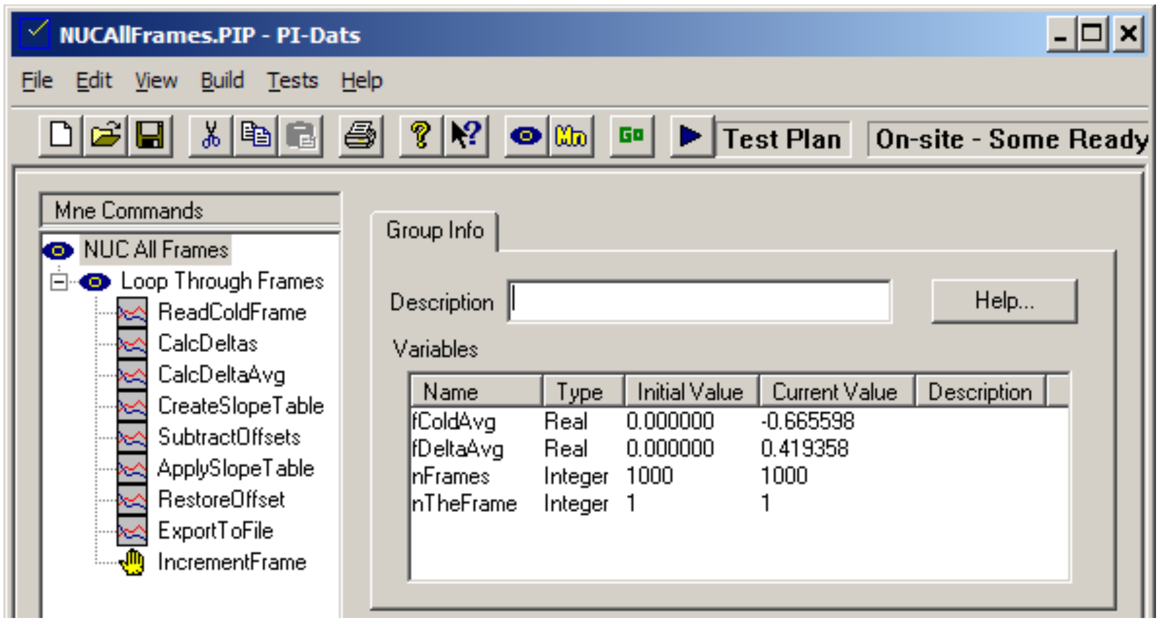
4. The delta frame is then divided by the average delta to produce a slope-correction table

5. Once the correction factors have been calculated, they can then be applied to each frame of the acquired data set. Each frame is read in from the data file, and the following operations are performed:

   a. The “cold” frame is subtracted
   b. The result is divided by the slope-correction table
   c. The average “cold” offset is added back
   d. The frame is appended to the “corrected” data file
   e. The frame number is incremented for the next loop

The following sections will examine each test plan mnemonic in detail.

11.0.2. Variables Declaration



Name	Type	Initial Value	Current Value	Description
fColdAvg	Real	0.000000	-0.665598	
fDeltaAvg	Real	0.000000	0.419358	
nFrames	Integer	1000	1000	
nTheFrame	Integer	1	1	

Figure 149: Variables Declaration

The top-level Group Header must contain declarations for each variable that will be used in the test plan. By convention the variable names in this test plan use the prefixes “f” and “n” to denote floating point and integer variables, but this is not strictly necessary. PI-DATS will assign the variable type by the selection in the drop-down menu, regardless of the user-defined name, and PI-DATS does any necessary typecasting when operations are performed using different variable types.

11.0.3. Loop Setup

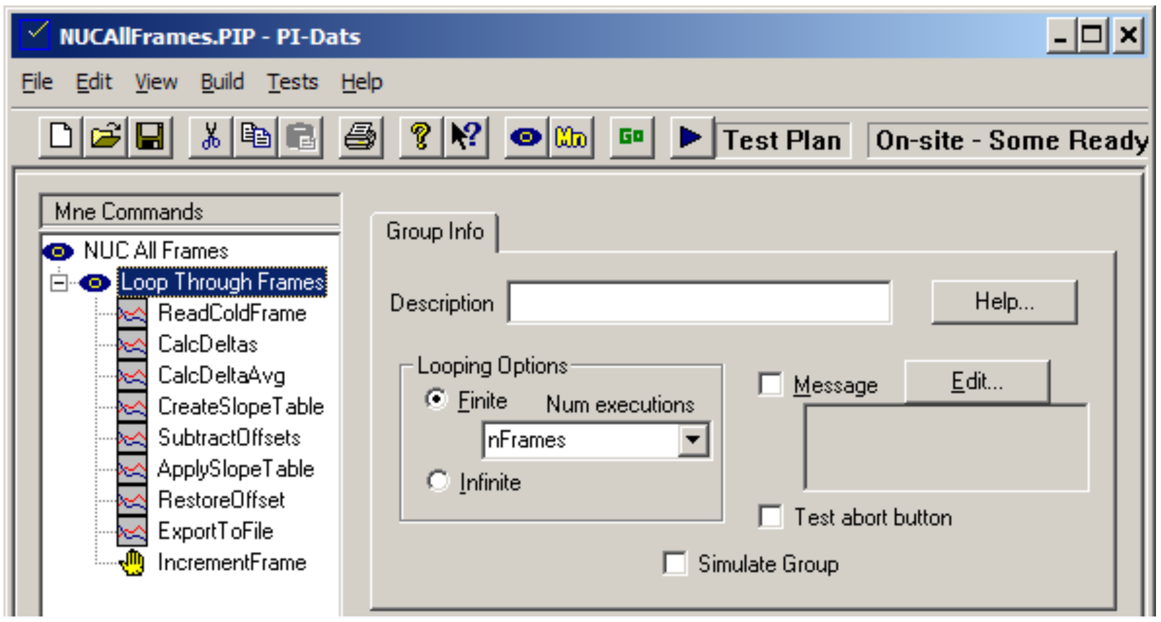


Figure 150: Loop Setup

Beneath the Test Plan header is the first Group Header. This will be used to set up a loop to process each frame in the data file. Since the variable nFrames has been assigned the initial value of 1000, this loop will execute 1000 times.

11.0.4. Read Cold Frame From File

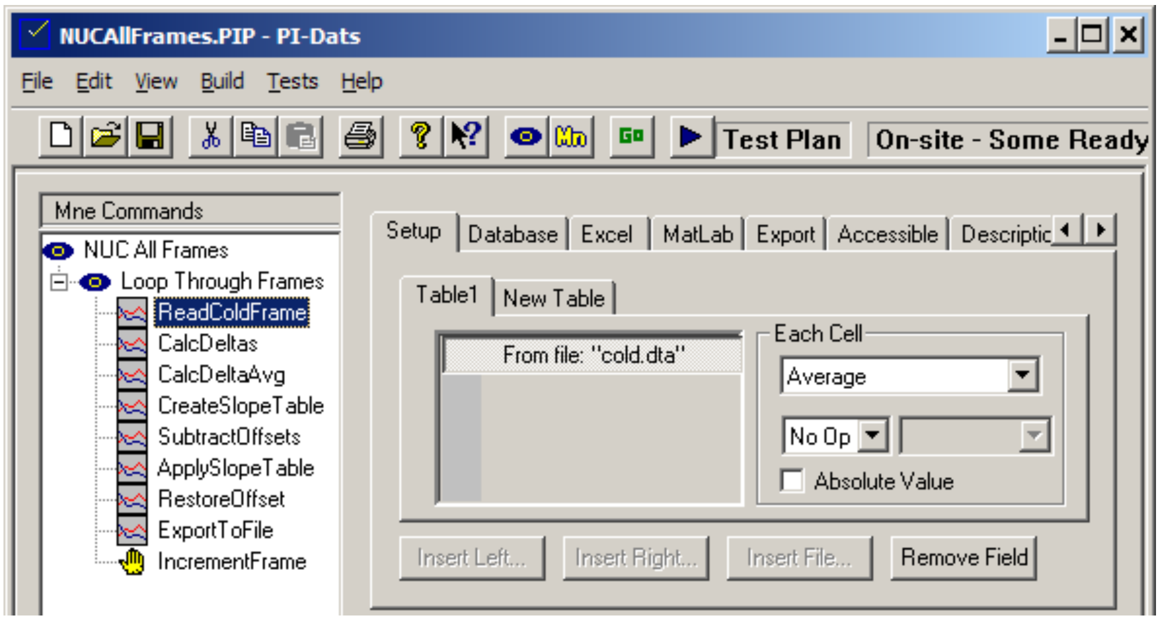
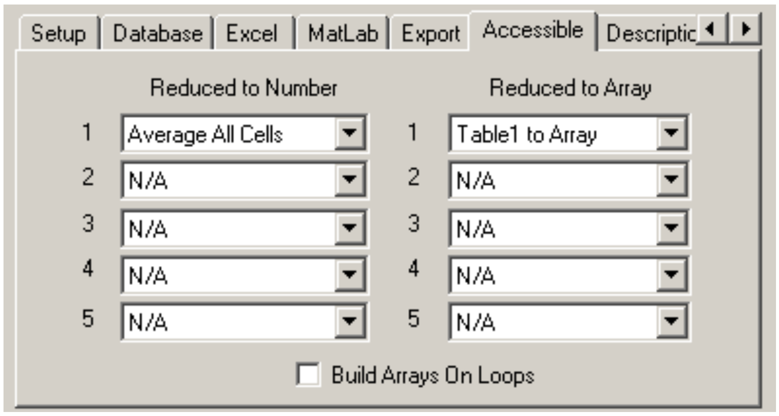


Figure 151: Read Cold Frame From File

Inside the loop, the first Result mnemonic reads the file “cold.dta” into Table 1.



Reduced to Number

1	Average All Cells
2	N/A
3	N/A
4	N/A
5	N/A

Reduced to Array

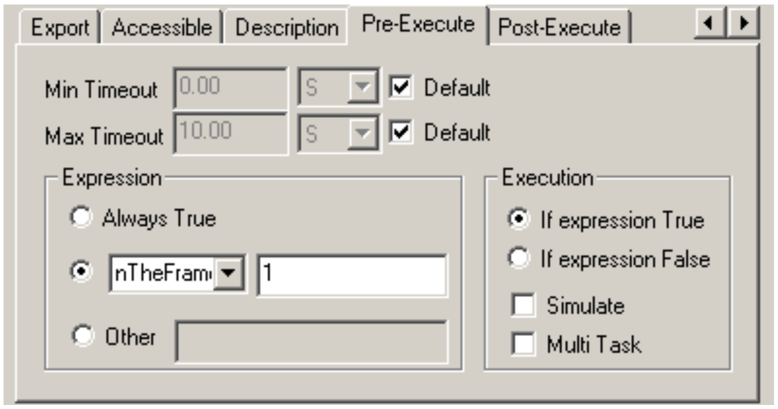
1	Table1 to Array
2	N/A
3	N/A
4	N/A
5	N/A

☐ Build Arrays On Loops

Figure 152: Processing the Cold Frame 1

The **Accessible** tab is used to reduce the Cold frame to an average value, and the "**Table1 to Array**" selection is used to pass the entire frame onto the next Result mnemonic. We un-check the **Build Arrays on Loops** box because we don't want to duplicate this array in memory on each loop iteration.

Because the correction tables need to be calculated only once, regardless of how many frames need to be processed, the Pre-Execute tab can be used to restrict execution of this mnemonic to only the first iteration of the loop (e.g. when **nTheFrame** is equal to 1). We use this same method to control execution of all the mnemonics in the loop up until the NUC is actually applied to the acquired data set.



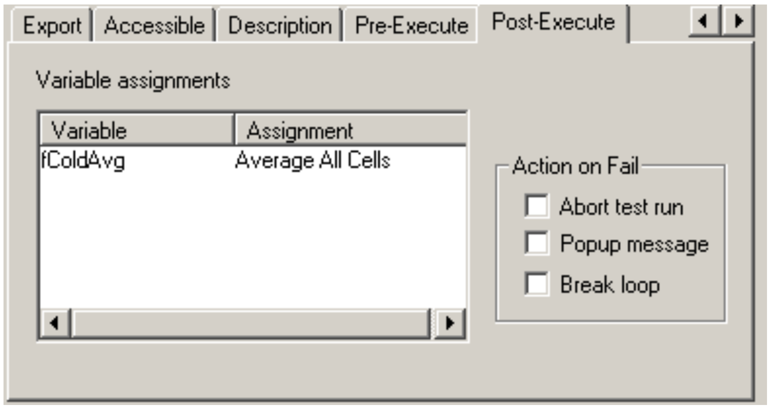
Min Timeout 0.00 S ☑ Default
Max Timeout 10.00 S ☑ Default

Expression
- ○ Always True
- ● nTheFram 1
- ○ Other

Execution
- ● If expression True
- ○ If expression False
- ☐ Simulate
- ☐ Multi Task

Figure 153: Processing the Cold Frame 2

Finally, we use the Post-Execute **control to assign the computed** Average All Cells **value to the variable named** fColdAvg**.**



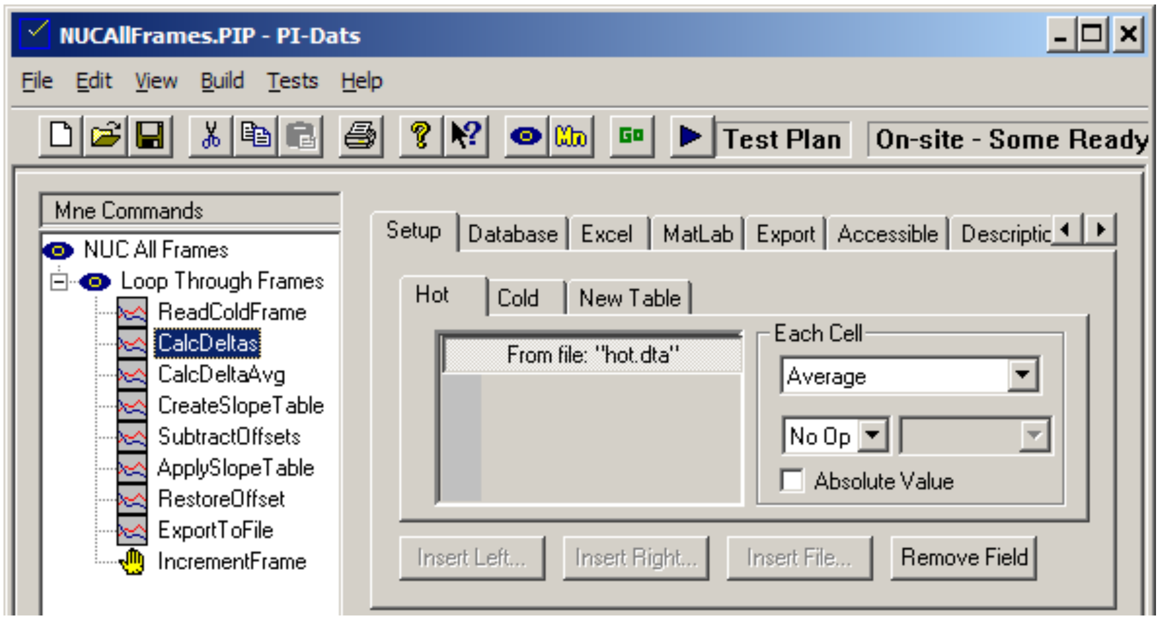
Variable assignments

Variable	Assignment
fColdAvg	Average All Cells

Action on Fail
- [ ] Abort test run
- [ ] Popup message
- [ ] Break loop

Figure 154: Processing the Cold Frame 3

11.0.5. Calculate the Delta Frame



From file: "hot.dta"

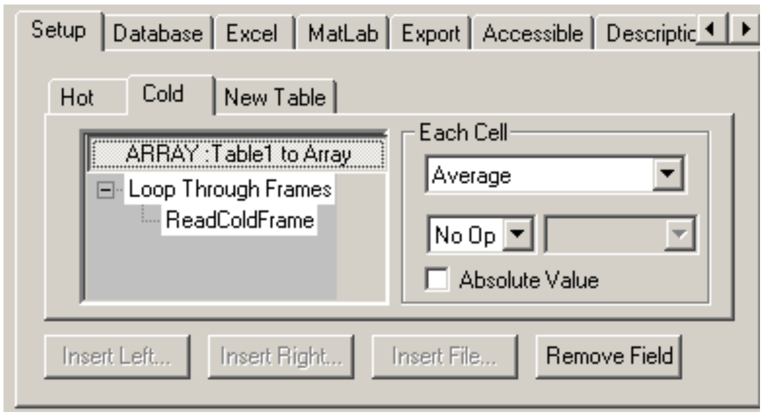
Each Cell: Average

No Op

- [ ] Absolute Value

Figure 155: Calculate the Delta Frame

Next, we want to calculate a frame of deltas between the Hot and Cold frame. We read in the file “hot.dta” into the first table.



ARRAY :Table1 to Array
Loop Through Frames
ReadColdFrame

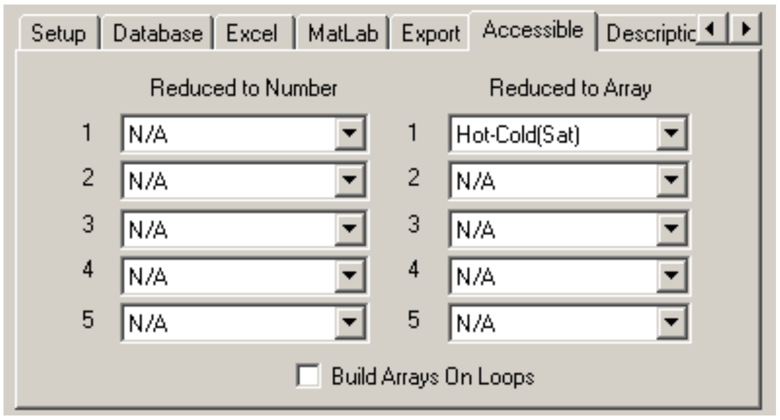
Each Cell: Average

No Op

Absolute Value: ☐

Figure 156: Calculate the Delta Frame

Into the second table we insert the output from the first Result mnemonic. This then allows us to perform operations using the two tables.



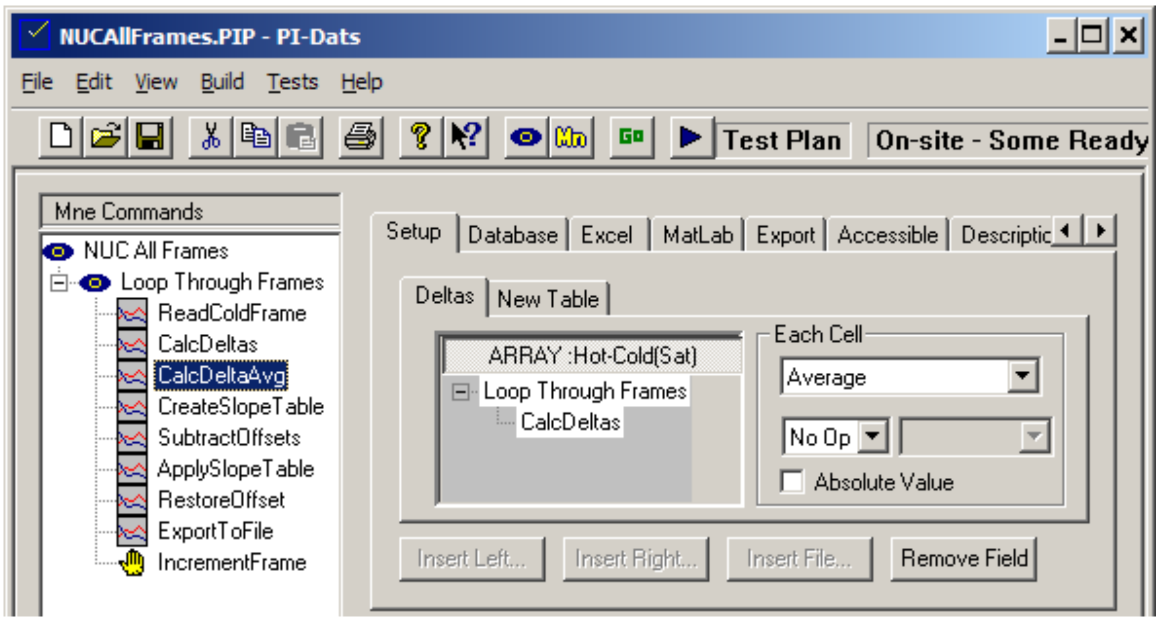
	Reduced to Number		Reduced to Array
1	N/A	1	Hot-Cold(Sat)
2	N/A	2	N/A
3	N/A	3	N/A
4	N/A	4	N/A
5	N/A	5	N/A

Build Arrays On Loops: ☐

Figure 157: Calculate the Delta Frame

In the Accessible tab, we use the Reduce to Array functions to compute the difference between the two frames. This frame of deltas will be used in the next Result mnemonic.

11.0.6. Calculate the Delta Average



NUCAllFrames.PIP - PI-Dats

File Edit View Build Tests Help

Test Plan | On-site - Some Ready

Mne Commands
- NUC All Frames
  - Loop Through Frames
    - ReadColdFrame
    - CalcDeltas
    - CalcDeltaAvg
    - CreateSlopeTable
    - SubtractOffsets
    - ApplySlopeTable
    - RestoreOffset
    - ExportToFile
    - IncrementFrame

Setup | Database | Excel | MatLab | Export | Accessible | Descriptic

Deltas | New Table

ARRAY :Hot-Cold(Sat)
- Loop Through Frames
  - CalcDeltas

Each Cell: Average

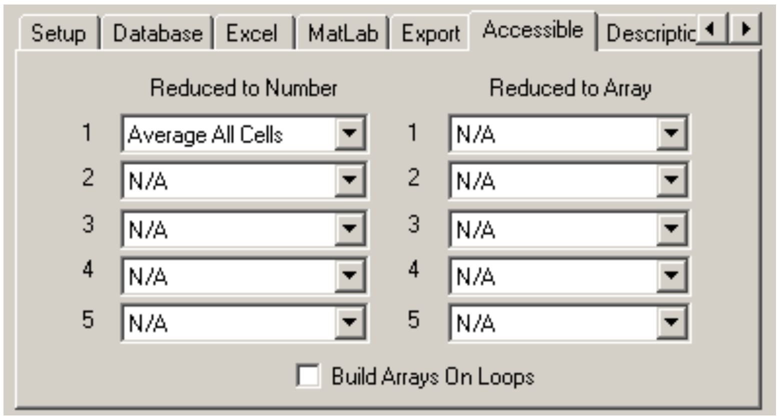
No Op

Absolute Value: ☐

Insert Left... | Insert Right... | Insert File... | Remove Field

Figure 158: Calculate the Delta Average

Once we have the frame of delta values, we can insert that into the next Result mnemonic to computer the average delta:



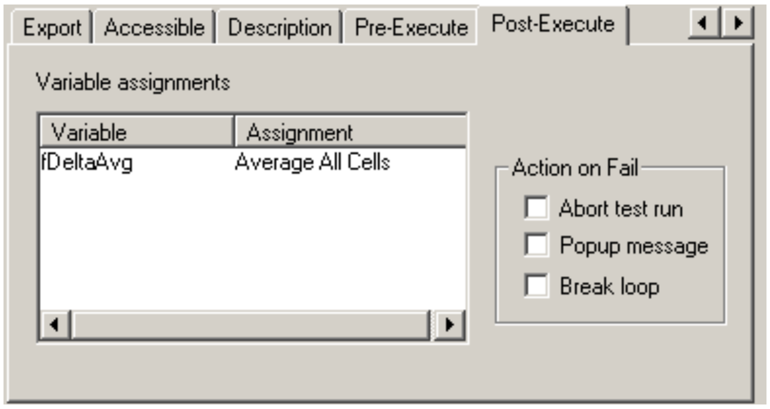
Setup | Database | Excel | MatLab | Export | Accessible | Descriptic

	Reduced to Number		Reduced to Array
1	Average All Cells	1	N/A
2	N/A	2	N/A
3	N/A	3	N/A
4	N/A	4	N/A
5	N/A	5	N/A

☐ Build Arrays On Loops

Figure 159: Calculate the Delta Average

Once the average value is computed we assign it to the variable **fDeltaAvg:**



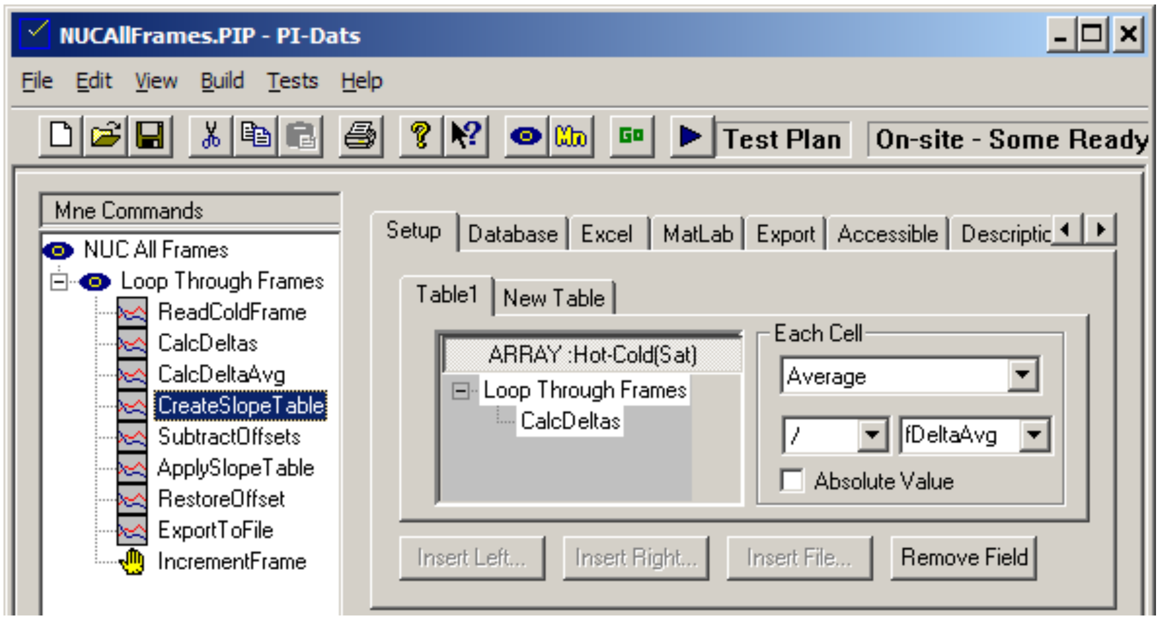
Variable assignments

Variable	Assignment
fDeltaAvg	Average All Cells

Action on Fail
- [ ] Abort test run
- [ ] Popup message
- [ ] Break loop

Figure 160: Assigning the Delta Average

11.0.7. Creating the Slope Correction Table



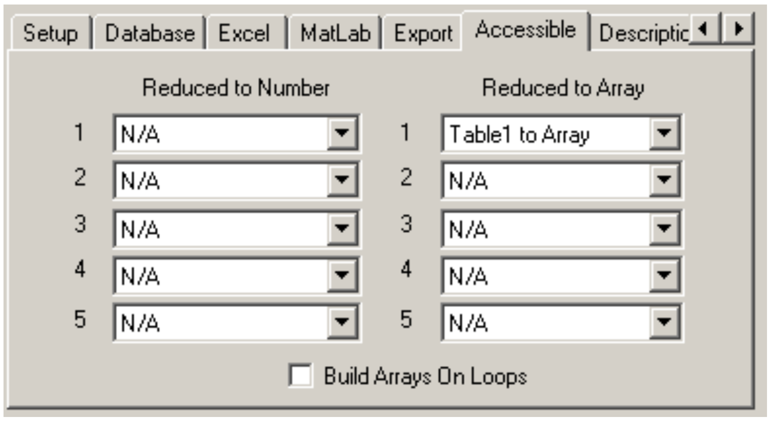
Each Cell: Average

/ fDeltaAvg

- [ ] Absolute Value

Figure 161: Creating the Slope Correction Table

Next, we divide each pixel in the Delta Frame by the average delta value. This will give us a table of slope-correction factors to apply to each frame. Again, we make the output accessible to subsequent Result mnemonics:

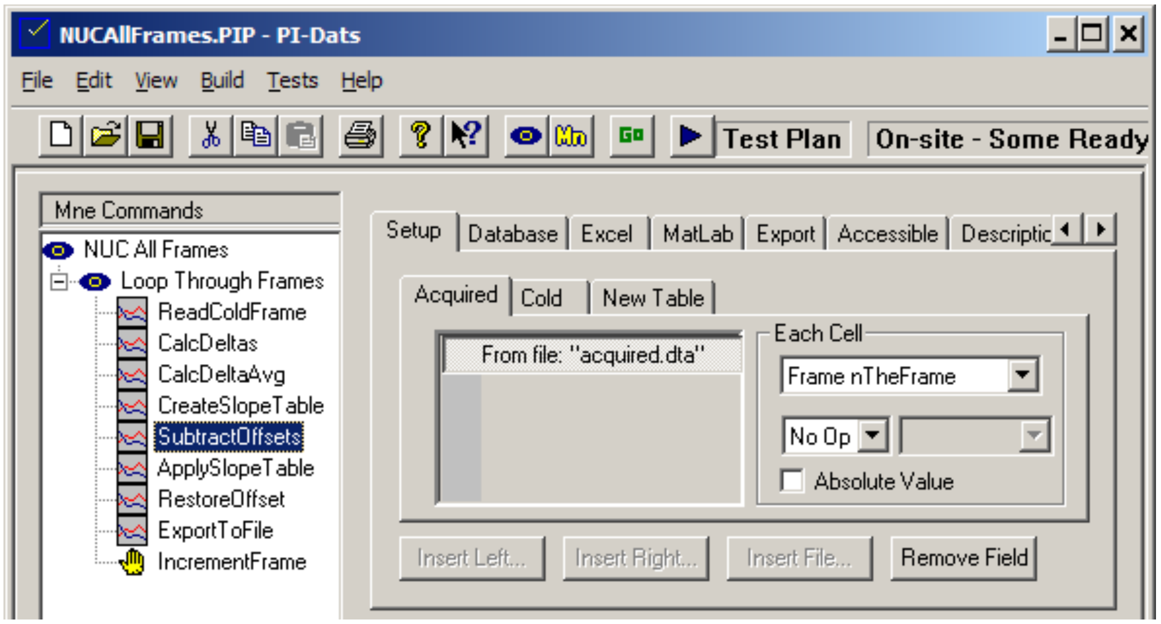

Setup | Database | Excel | MatLab | Export | Accessible | Descriptic

	Reduced to Number		Reduced to Array
1	N/A	1	Table1 to Array
2	N/A	2	N/A
3	N/A	3	N/A
4	N/A	4	N/A
5	N/A	5	N/A

☐ Build Arrays On Loops

Figure 162: Passing the Slope Correction Table

11.0.8. Applying the NUC: Offset Correction


NUCAllFrames.PIP - PI-Dats

File Edit View Build Tests Help

Test Plan | On-site - Some Ready

Mne Commands
- NUC All Frames
  - Loop Through Frames
    - ReadColdFrame
    - CalcDeltas
    - CalcDeltaAvg
    - CreateSlopeTable
    - SubtractOffsets
    - ApplySlopeTable
    - RestoreOffset
    - ExportToFile
    - IncrementFrame

Setup | Database | Excel | MatLab | Export | Accessible | Descriptic

Acquired | Cold | New Table

From file: "acquired.dta"

Each Cell: Frame nTheFrame

No Op

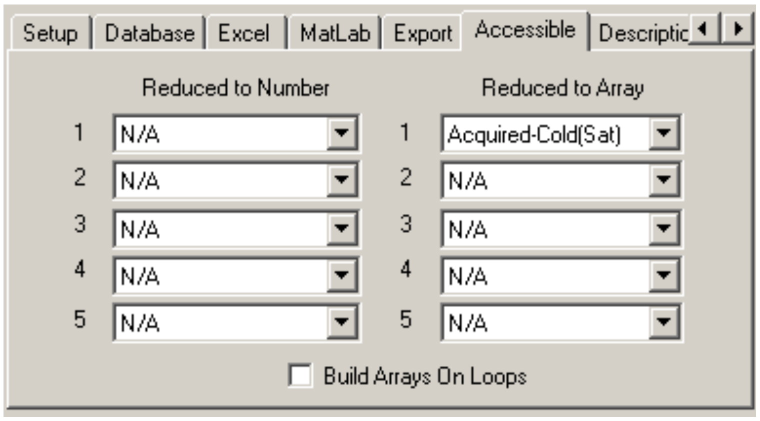
☐ Absolute Value

Insert Left... | Insert Right... | Insert File... | Remove Field

Figure 163: Applying the NUC

Once we have created the correction factors, we can now apply them to our raw video data. First, we read in the file containing the acquired data into Table1 (which we renamed “Acquired”). We want to work on each frame individually, so we choose the **Nth Frame** option with the variable **nTheFrame**.

We want to correct for pixel-to-pixel offset, so we insert the Cold frame into Table2.

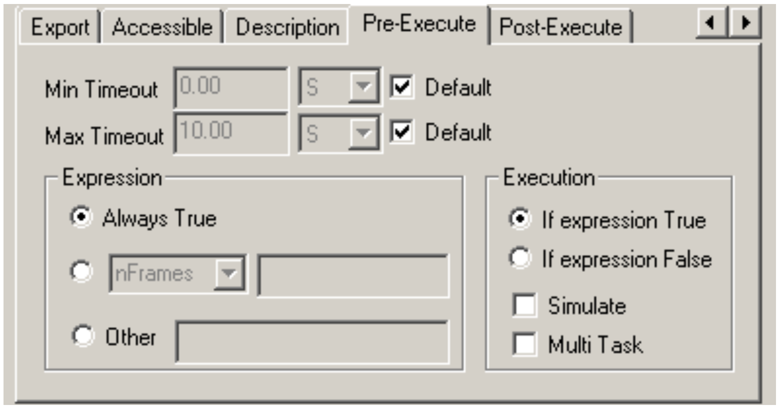


	Reduced to Number		Reduced to Array
1	N/A	1	Acquired-Cold(Sat)
2	N/A	2	N/A
3	N/A	3	N/A
4	N/A	4	N/A
5	N/A	5	N/A

☐ Build Arrays On Loops

Figure 164: Subtracting Offsets

The difference between the acquired frame and the cold frame are passed on to the next mnemonic.



Min Timeout 0.00 S ✓ Default
Max Timeout 10.00 S ✓ Default

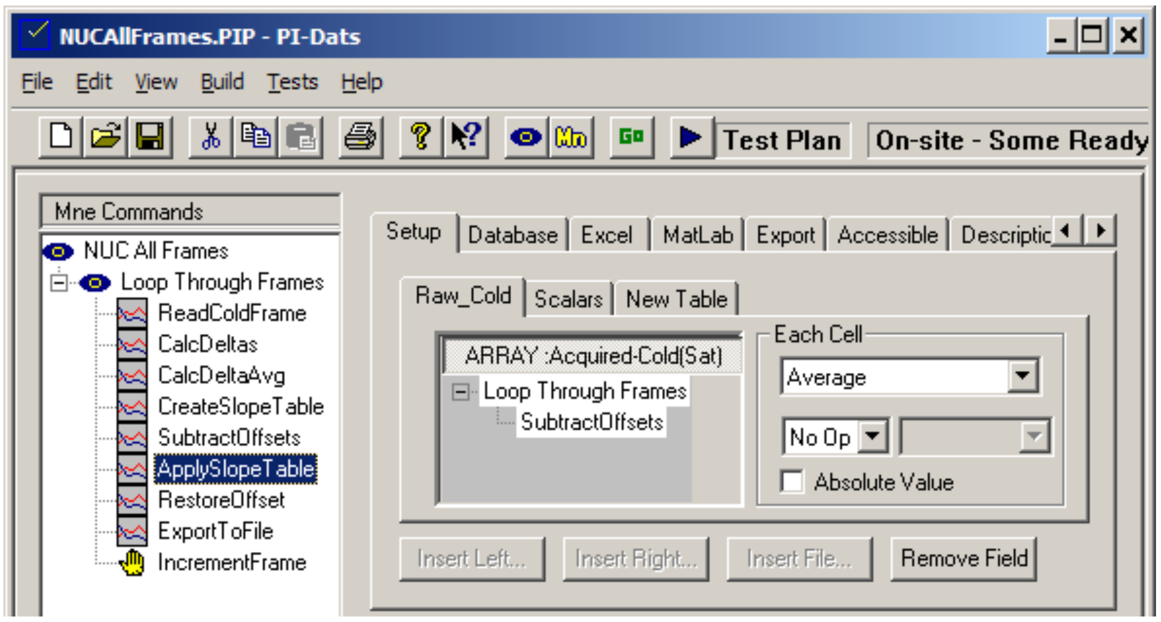
Expression
● Always True
○ nFrames
○ Other

Execution
● If expression True
○ If expression False
☐ Simulate
☐ Multi Task

Figure 165: Every Frame Gets Processed

Note that, since the one-time calculations are completed, all subsequent mnemonics in this loop should execute on every loop iteration, in order to apply the correction factors to every frame of the data set.

11.0.9. Applying the NUC: Slope Correction


NUCAllFrames.PIP - PI-Dats

File Edit View Build Tests Help

Test Plan | On-site - Some Ready

Mne Commands
- NUC All Frames
  - Loop Through Frames
    - ReadColdFrame
    - CalcDeltas
    - CalcDeltaAvg
    - CreateSlopeTable
    - SubtractOffsets
    - ApplySlopeTable
    - RestoreOffset
    - ExportToFile
    - IncrementFrame

Setup | Database | Excel | MatLab | Export | Accessible | Descriptic

Raw_Cold | Scalars | New Table

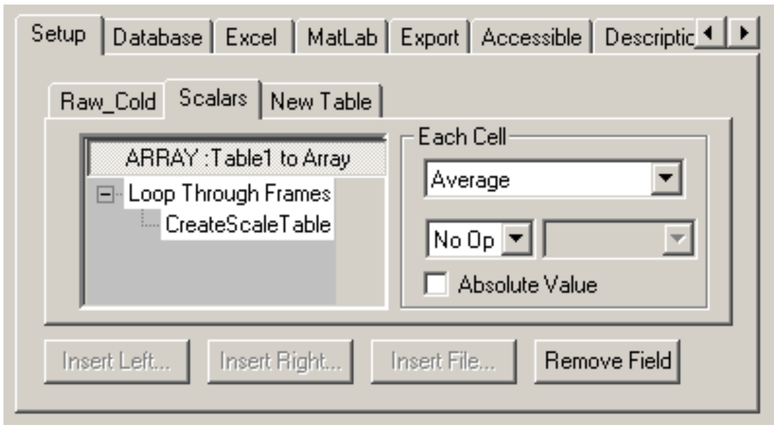
ARRAY :Acquired-Cold(Sat)
- Loop Through Frames
  - SubtractOffsets

Each Cell: Average; No Op; Absolute Value [ ]

Insert Left... | Insert Right... | Insert File... | Remove Field

Figure 166: Applying the Slope Correction Table 1

The offset-corrected frame is now inserted into a new mnemonic along with the slope-correction table generated previously.


Setup | Database | Excel | MatLab | Export | Accessible | Descriptic

Raw_Cold | Scalars | New Table

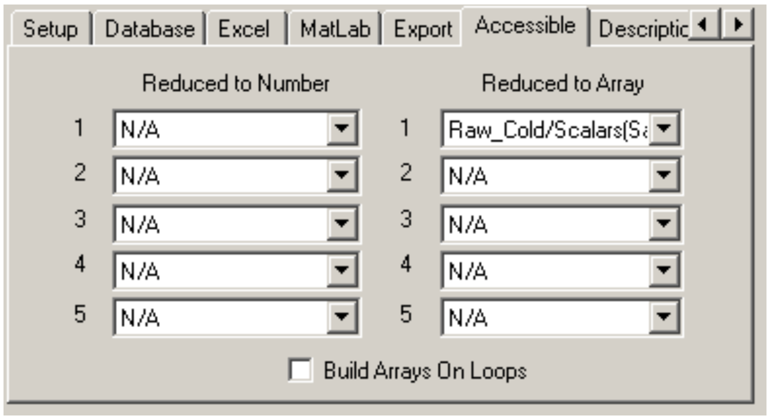
ARRAY :Table1 to Array
- Loop Through Frames
  - CreateScaleTable

Each Cell: Average; No Op; Absolute Value [ ]

Insert Left... | Insert Right... | Insert File... | Remove Field

Figure 167: Applying the Slope Correction Table 2

Each offset-corrected frame is normalized by dividing by the slope-correction table previously generated.

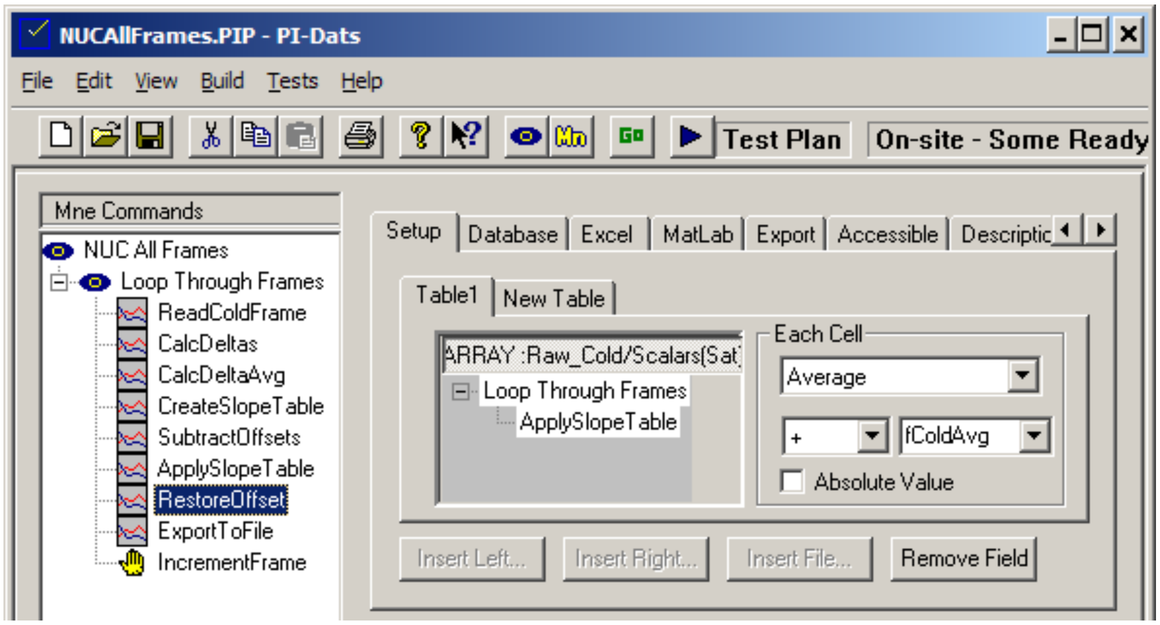

Setup | Database | Excel | MatLab | Export | Accessible | Descriptic

Reduced to Number		Reduced to Array	
1	N/A	1	Raw_Cold/Scalars(Sa
2	N/A	2	N/A
3	N/A	3	N/A
4	N/A	4	N/A
5	N/A	5	N/A

☐ Build Arrays On Loops

Figure 168: Applying the Slope Correction Table 3

11.0.10. Applying the NUC: Offset Restoration


NUCAllFrames.PIP - PI-Dats

File Edit View Build Tests Help

Test Plan | On-site - Some Ready

Mne Commands
- NUC All Frames
  - Loop Through Frames
    - ReadColdFrame
    - CalcDeltas
    - CalcDeltaAvg
    - CreateSlopeTable
    - SubtractOffsets
    - ApplySlopeTable
    - RestoreOffset
    - ExportToFile
    - IncrementFrame

Setup | Database | Excel | MatLab | Export | Accessible | Descriptic

Table1 | New Table

ARRAY :Raw_Cold/Scalars(Sat)
- Loop Through Frames
  - ApplySlopeTable

Each Cell: Average

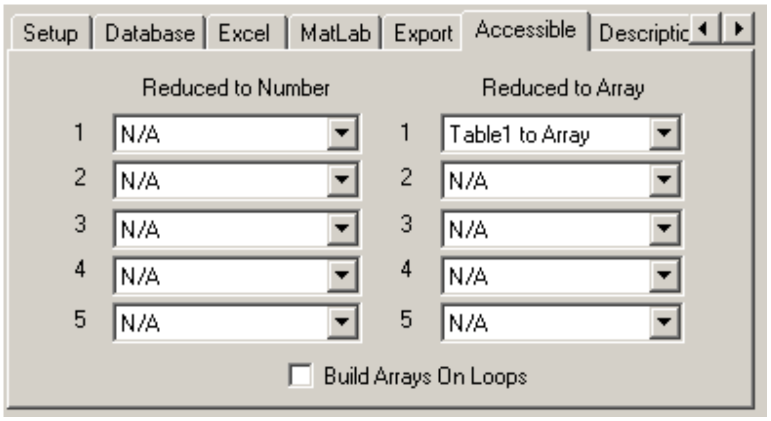
+ fColdAvg

☐ Absolute Value

Insert Left... | Insert Right... | Insert File... | Remove Field

Figure 169: Offset Restoration

Finally, we add back the average offset value (computed from the Cold frame) to restore the average DC value to the frame.



Reduced to Number

1	N/A
2	N/A
3	N/A
4	N/A
5	N/A

Reduced to Array

1	Table1 to Array
2	N/A
3	N/A
4	N/A
5	N/A

☐ Build Arrays On Loops

Figure 170: Passing the Final Results

This completes the correction, and the corrected frame is passed onto the next mnemonic for export to a file.

11.0.11. Saving The Corrected Data

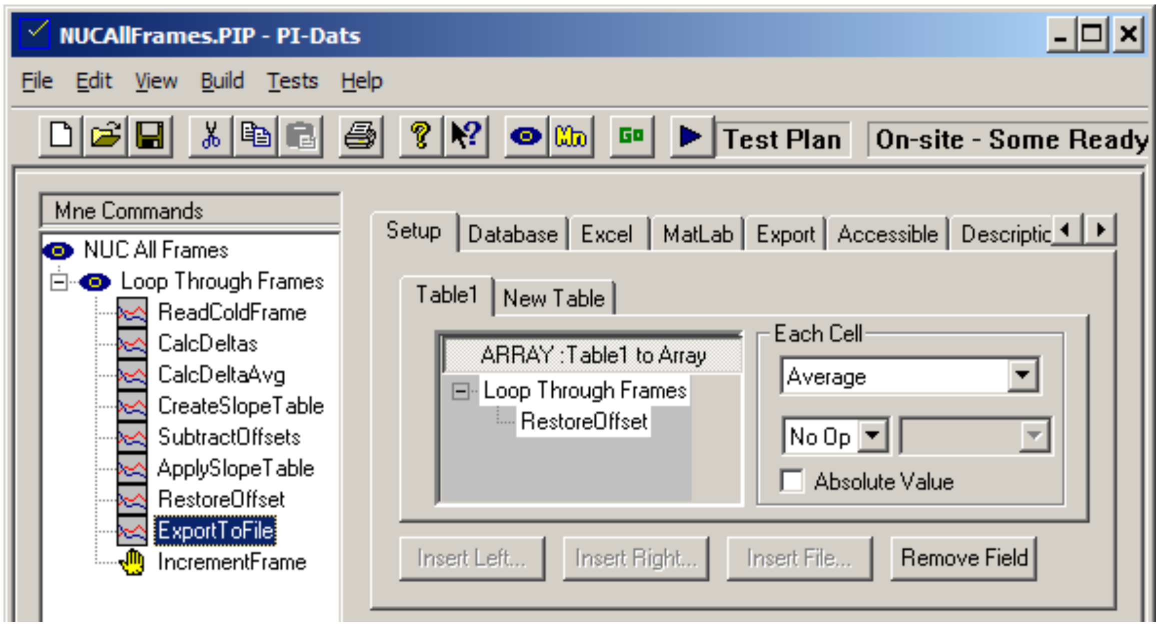
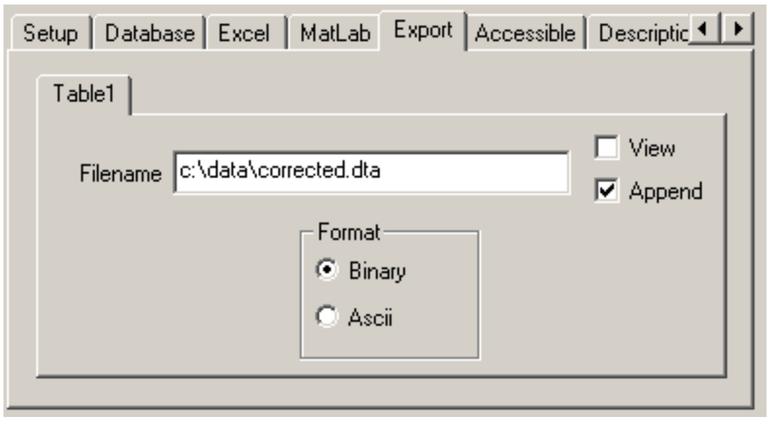


Figure 171: Saving the Corrected Data

As each frame of data is corrected, we can use a Result mnemonic to append it to a file using the Export tab:



Filename: c:\data\corrected.dta

☐ View

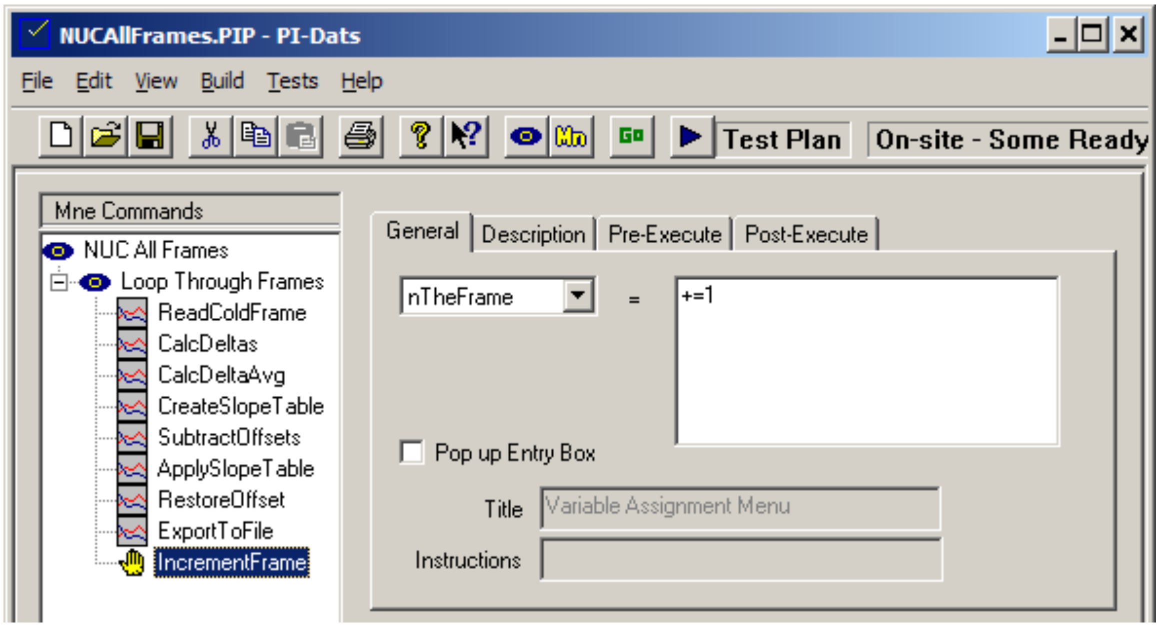
☑ Append

Format: (●) Binary ( ) Ascii

Figure 172: Appending Each Frame

The Append checkbox causes each frame to be appended to an existing file, and the header of that file updated with the correct number of frames.

11.0.12. Incrementing the Frame Number



- NUC All Frames
  - Loop Through Frames
    - ReadColdFrame
    - CalcDeltas
    - CalcDeltaAvg
    - CreateSlopeTable
    - SubtractOffsets
    - ApplySlopeTable
    - RestoreOffset
    - ExportToFile
    - IncrementFrame

nTheFrame = +=1

☐ Pop up Entry Box

Title: Variable Assignment Menu

Instructions:

Figure 173: Incrementing the Frame Number

The final step in the loop is to increment the frame counter.

Upon completion of the loop, the output file “corrected.dta” will be a valid PI data file with 1000 frames of NUCed data. It can be opened and viewed in PI-PLOT or passed on for analysis by another PI-DATS test plan, MATLAB, SBIR’s IR-Windows, or any other compatible analysis package.

11.0.13. Other Analysis and Post-Processing Options

The algorithms used in this example are for illustration of PI-DATS programming concepts only. You may choose to implement a different correction methodology using a similar test plan structure, or you may choose to modify this approach with additional options.

12. NUC Vector Injection

PI-Result can calculate, align, and inject per-pixel Non-Uniformity Correction (NUC) vectors into the timing patterns for ROICs that support this feature. This typically requires using several steps and several different PI‑DATS features, as follows:

12.0.1. Pattern Setup

Use the standard PI-PAT editing tools to write readout timing for the FPA/ROIC and acquisition timing for the PI system.

If the FPA timing is set up using repetitive loops, use one of the following features to create equivalent timing in a single, unrolled subpattern:

- Nested algorithms in the **Enter Pattern** dialog box. See **PI-2005/PI-PAT Operators Manual, Section 3.4.2.4**.
- **Compile/Unroll** to convert the repetitive timing into a single subpattern. See **PI-2005/PI-PAT Operators Manual, Section 3.4.1.8.**

12.0.2. Clock Driver, Bias and Acquisition Setup

Use the standard PI‑DATS editing tools to set up clock drivers, DC biases, and the data acquisition system to collect image data from the FPA/ROIC.

12.0.3. NUC Vector Calculation

Use the **Enter Value** method of the **NUC Pattern** feature to inject calibration values into the ROIC and acquire the resulting data. These should be selected according to your ROIC's design, but are often selected at some fraction of full scale, one LSB, full scale, etc., in order to characterize the response of the on-chip NUC circuitry. See **Section** 6.5. NUC Pattern Injection**.**

Use a series of **Result** mnemonics to calculate the desired array of correction vectors.

If the data for multiple output channels was calculated separately, use the **Interleave Tables** function to reassemble them into a single array, if appropriate. See **Section** 9.7.2.21. Interleave Tbls (Interleave Tables)**.**

Use the NUC Export feature to convert the calculated data into N-bit correction vectors, using the appropriate number of bits. See **Section** 9.6.2.2. NUC Export**.**

12.0.4. NUC Vector Injection

Use the **NUC File** method of the **NUC Pattern** feature to inject the correction vectors into unrolled timing pattern, using the appropriate bit multiplier, offset, and endianness.

13. Index

Document Change History:

4/13/07
Added index (finally!). Added preliminary section on Generic mnemonic feature.

12/3/07

Updated section on “new” Generic mnemonic

12/09

Updated Results section with block diagram

11/11

Added preliminary blackbody and motion controller sections

March 2016

Added NUC Export, NUC Inject, Interleave Tables, Table to Bool features, and NUC injection application note.