**Automated Step Response Test Stand**

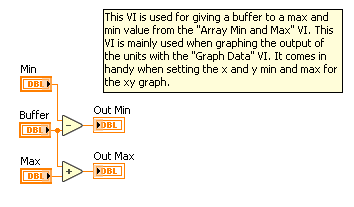
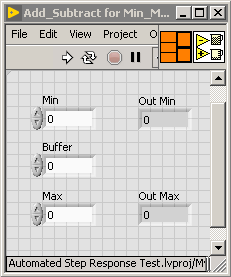
**Software Documentation**

David Boullie – 2019

MKS Instruments®

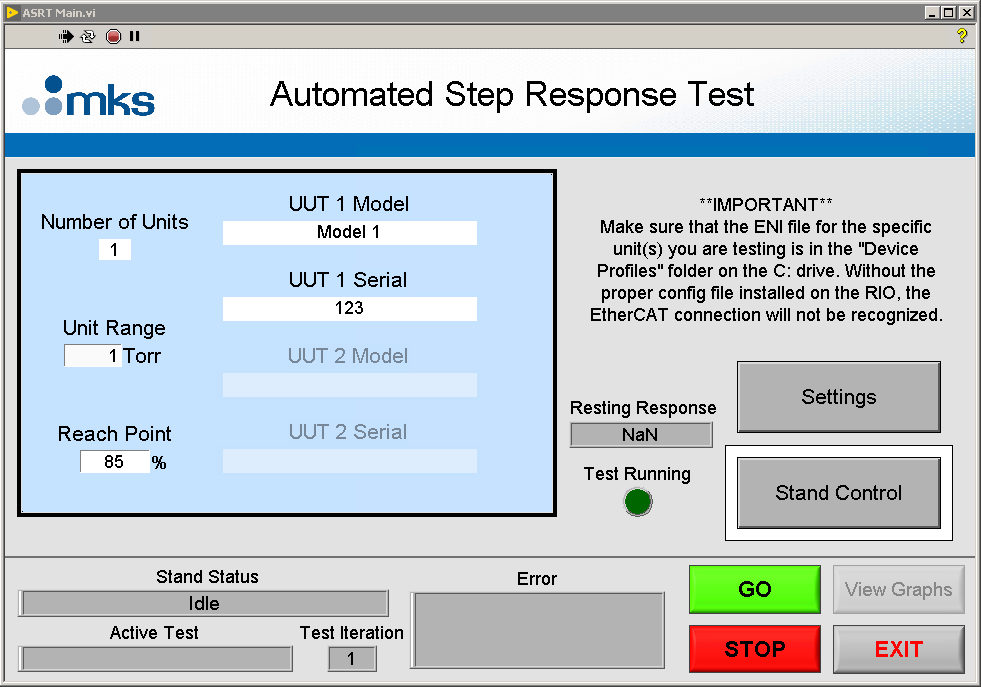
**Computer VIs**

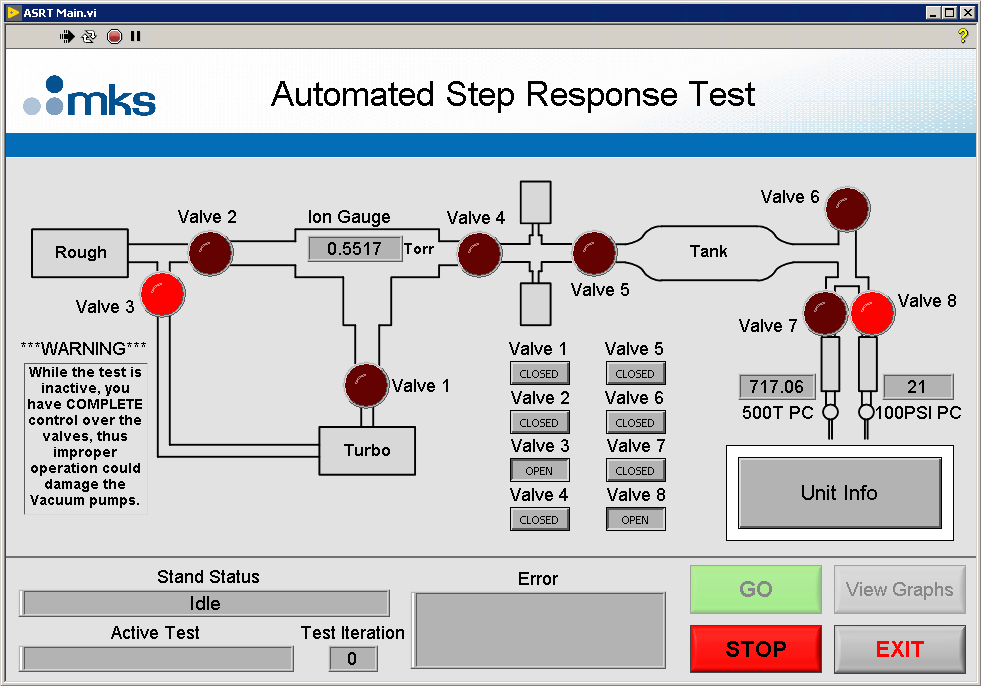
Add\_Subtract for Min\_Max

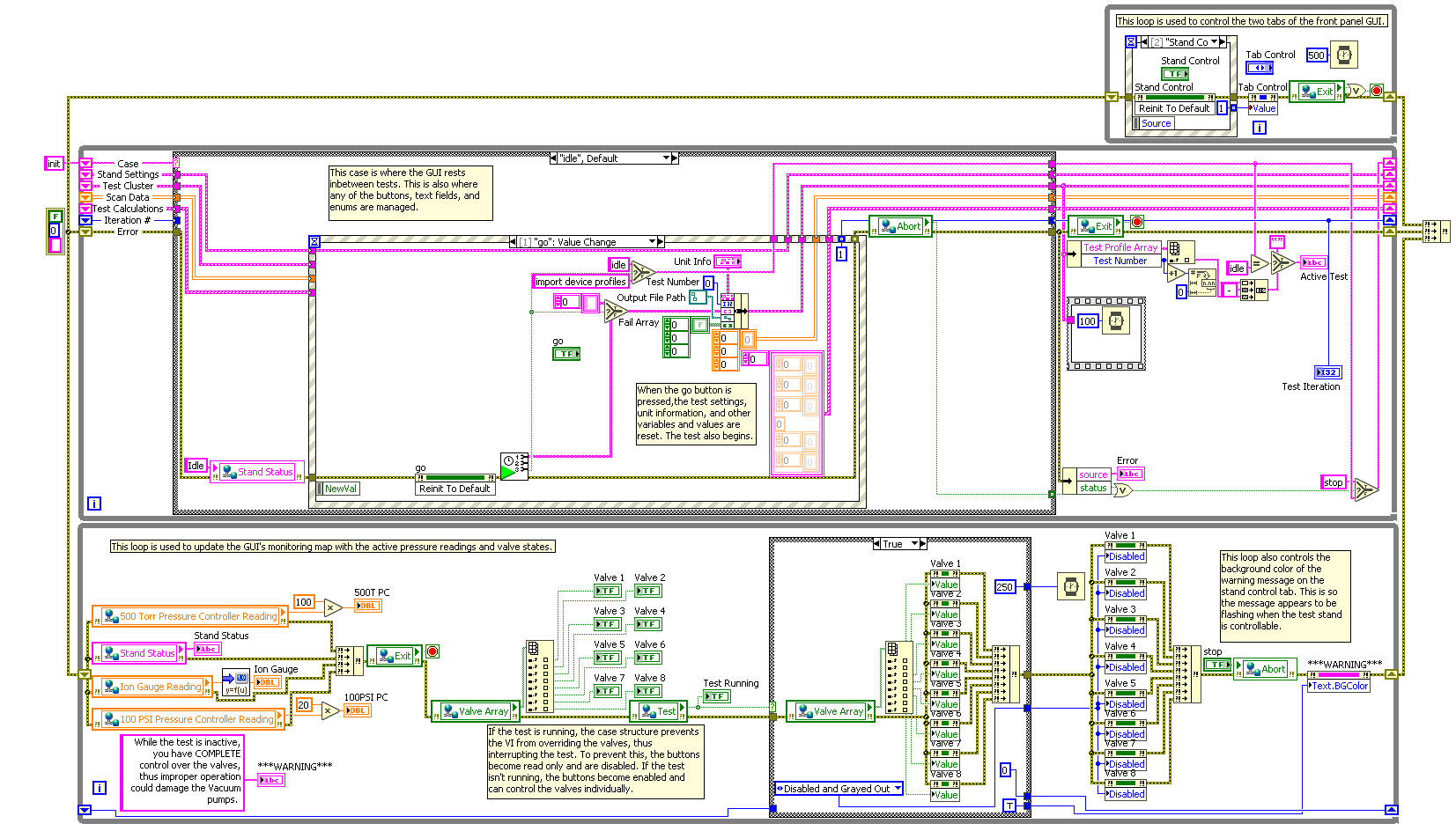


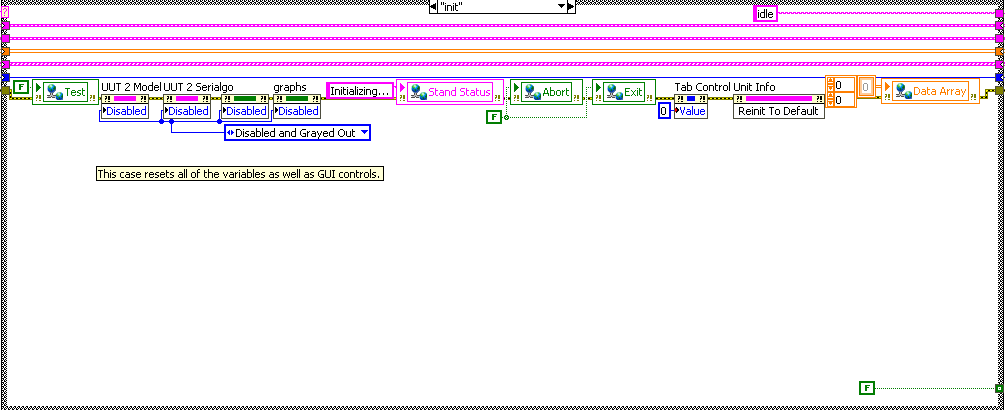
This VI is used for giving a buffer to a max and min value from the "Array Min and Max" VI. This VI is mainly used when graphing the output of the units with the "Graph Data" VI. It comes in handy when setting the x and y min and max for the xy graph.

ASRT Main

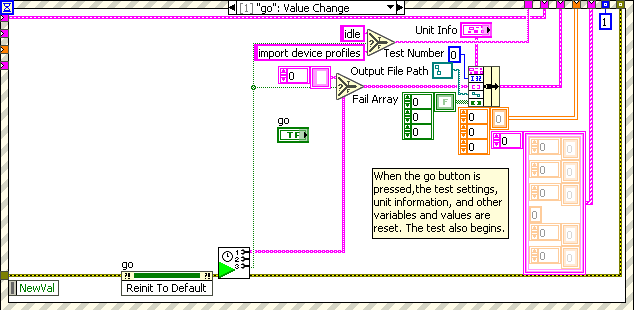






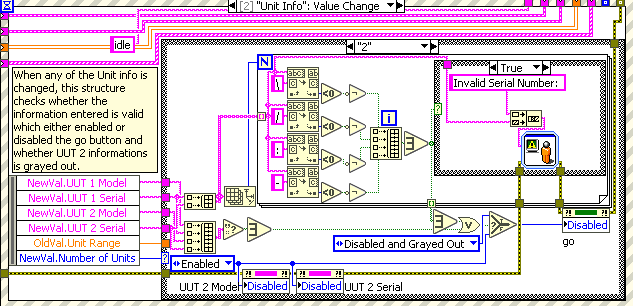


This case resets all the variables as well as GUI controls.

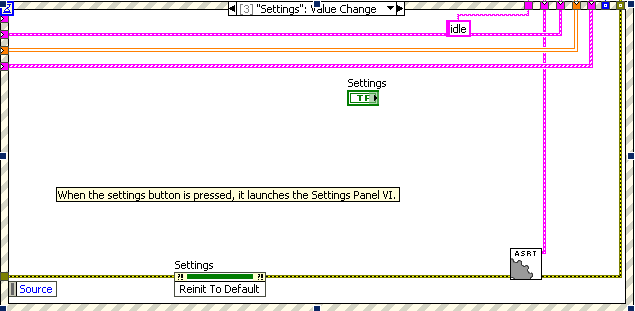


This case is where the GUI rests inbetween tests. This is also where any of the buttons, text fields, and enums are managed.

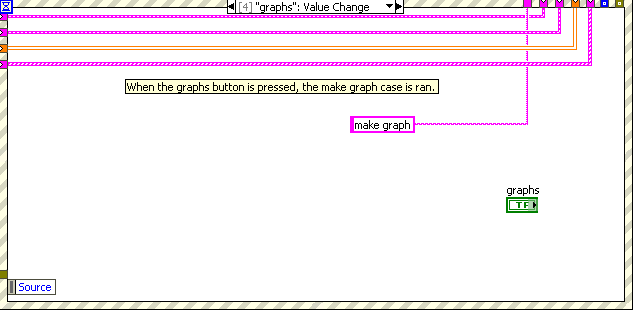
When the go button is pressed, the test settings, unit information, and other variables and values are reset. The test also begins.



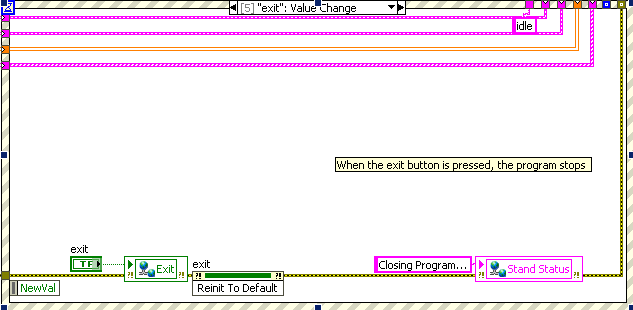
When any of the Unit info is changed, this structure checks whether the information entered is valid which either enabled or disabled the go button and whether UUT 2’s information is grayed out.



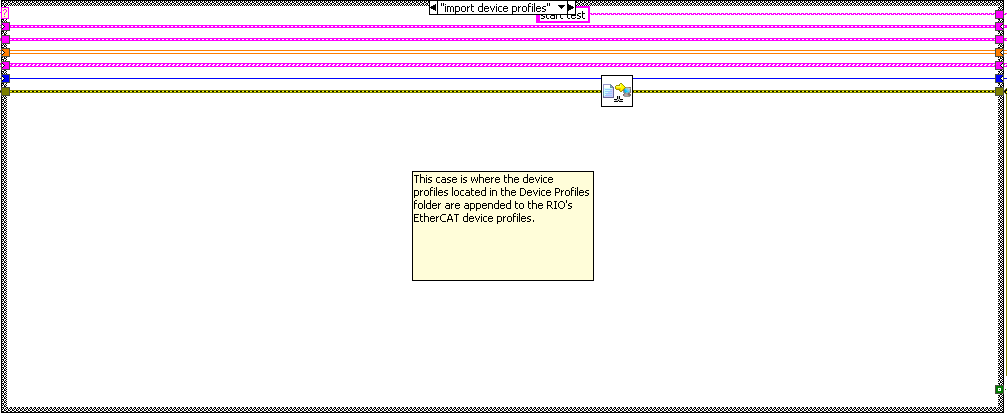
When the settings button is pressed, it launches the Settings Panel VI.



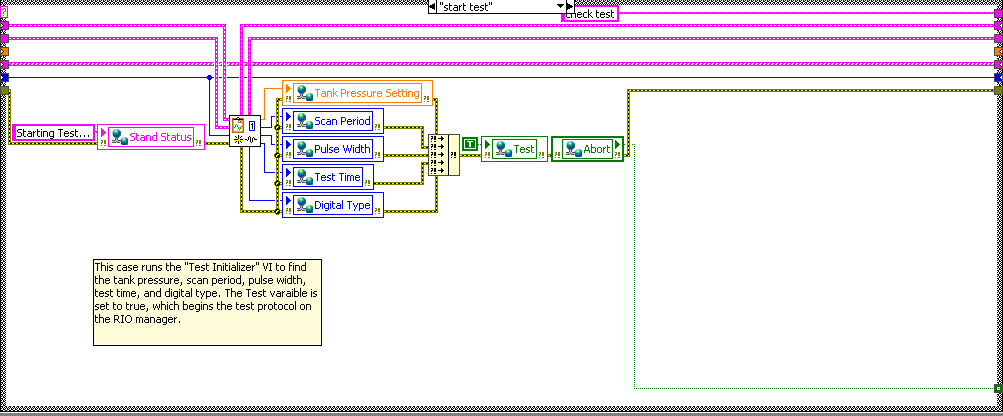
When the graphs button is pressed, the make graph case is run.



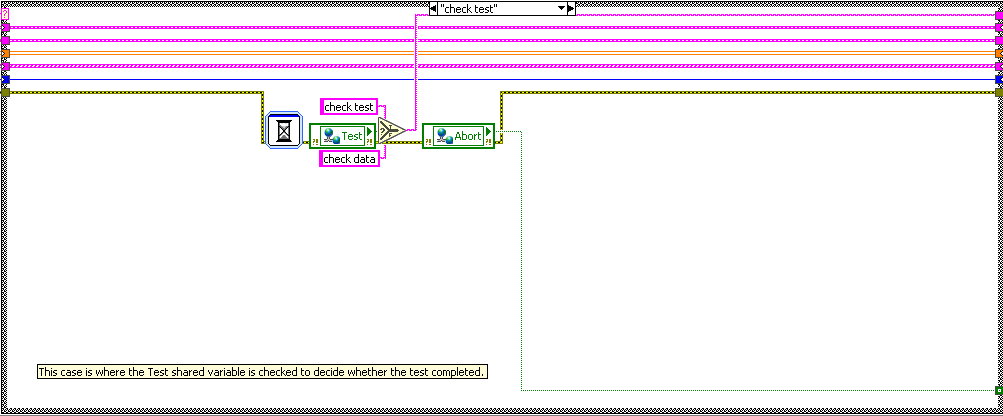
When the exit button is pressed, the program stops.



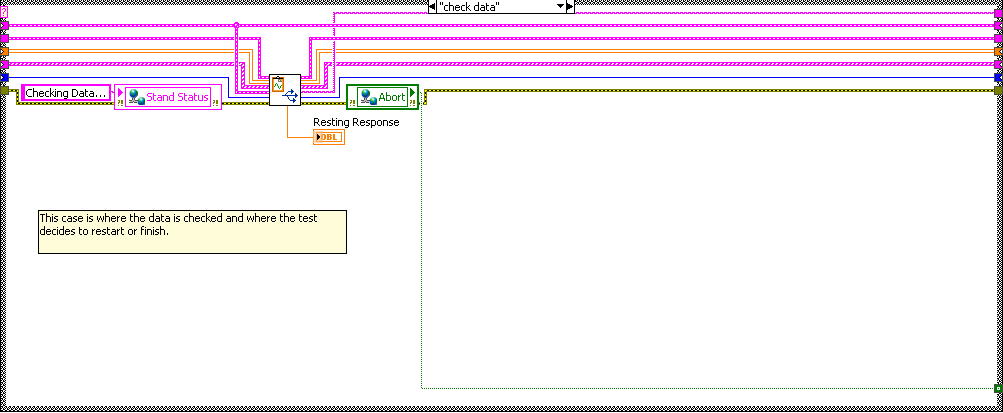
This case is where the device profiles located in the Device Profiles folder are appended to the RIO's EtherCAT device profiles.



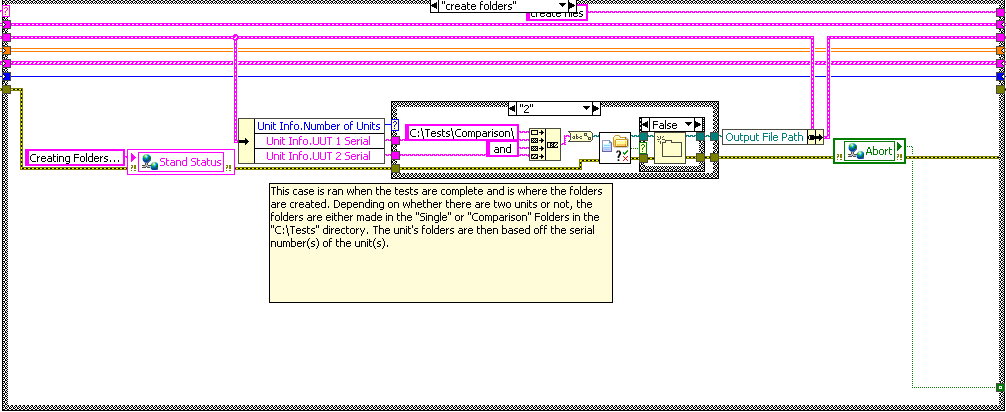
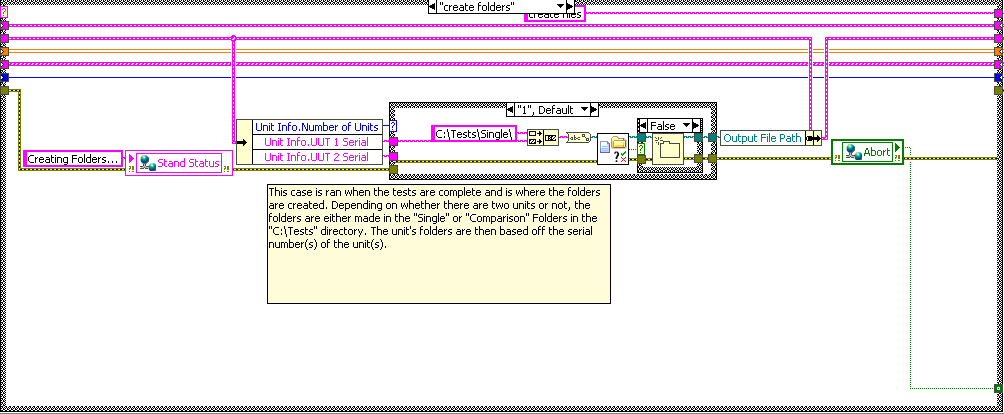
This case runs the "Test Initializer" VI to find the tank pressure, scan period, pulse width, test time, and digital type. The Test variable is set to true, which begins the test protocol on the RIO manager.



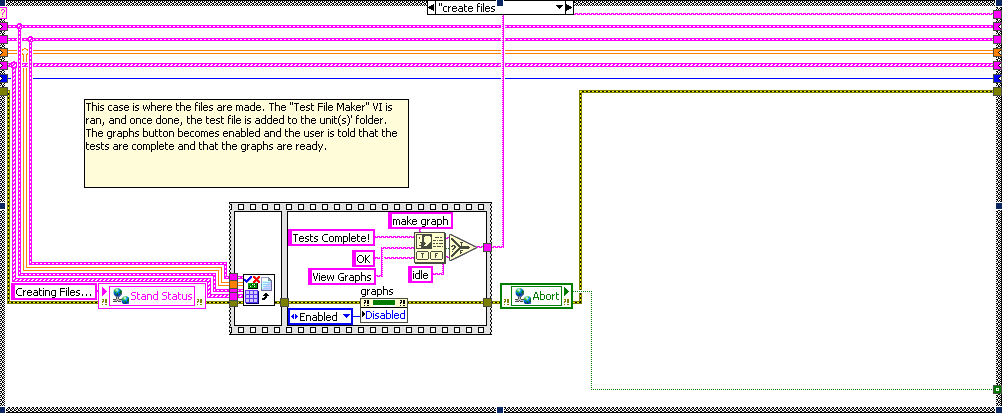
This case is where the Test shared variable is checked to decide whether the test completed.



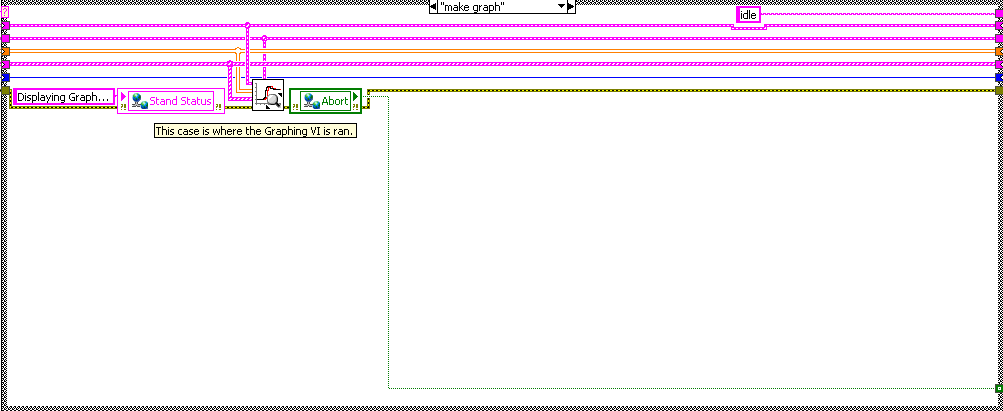
This case is where the data is checked and where the test decides to restart or finish.



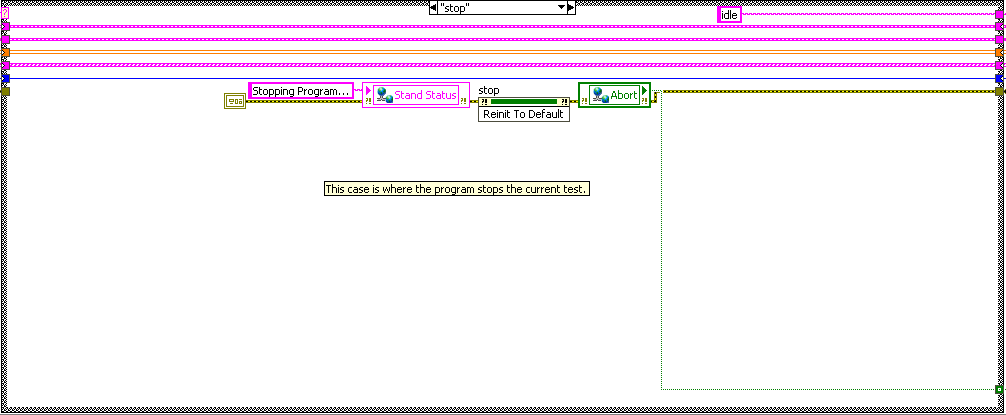
This case runs when the tests are complete and is where the folders are created. Depending on whether there are two units or not, the folders are either made in the "Single" or "Comparison" Folders in the "C:\Tests" directory. The unit's folders are then based off the serial number(s) of the unit(s).



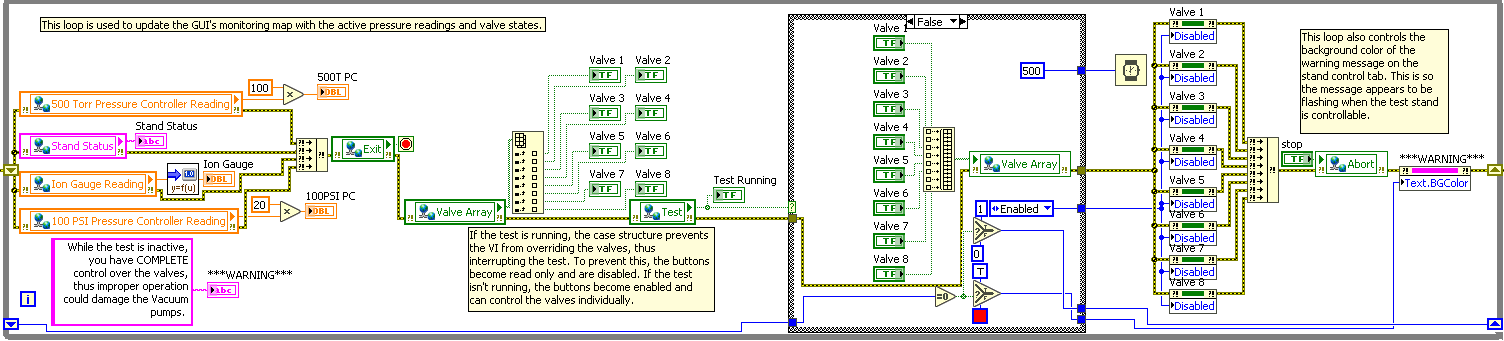
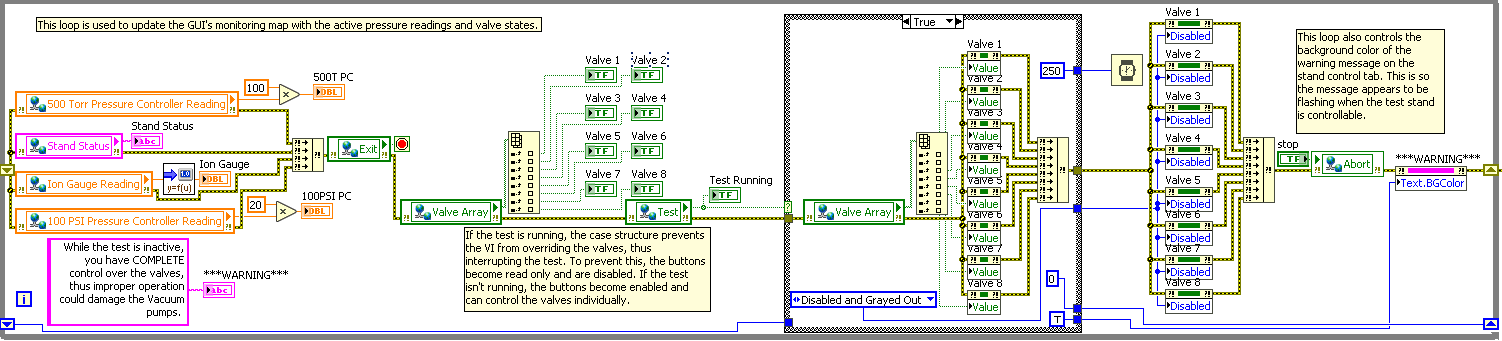
This case is where the files are made. The "Test File Maker" VI is ran, and once done, the test file is added to the unit(s)' folder. The graphs button becomes enabled and the user is told that the tests are complete and that the graphs are ready.



This case is where the Graphing VI is ran.



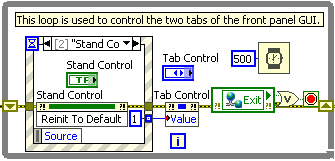
This case is where the program stops the current test.



This loop is used to update the GUI's monitoring map with the active pressure readings and valve states.

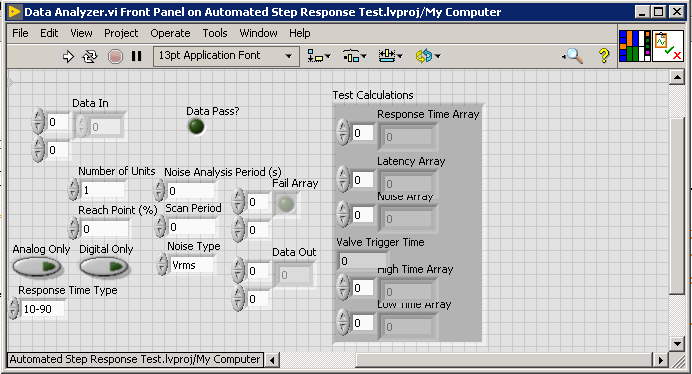
If the test is running, the case structure prevents the VI from overriding the valves, thus interrupting the test. To prevent this, the buttons become read only and are disabled. If the test isn't running, the buttons become enabled and can control the valves individually.

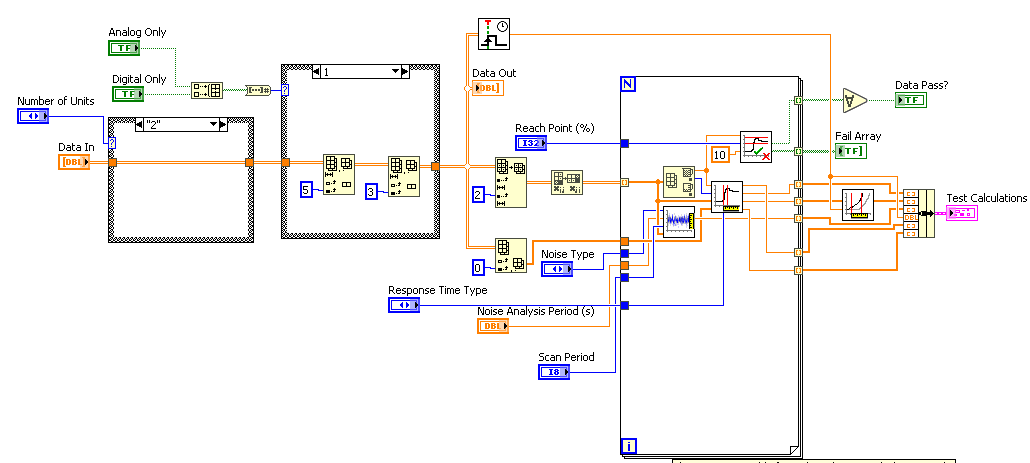
This loop also controls the background color of the warning message on the stand control tab. This is so the message appears to be flashing when the test stand is controllable.



This loop is used to control the two tabs of the front panel GUI.

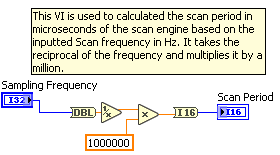
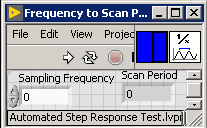
Data Analyzer





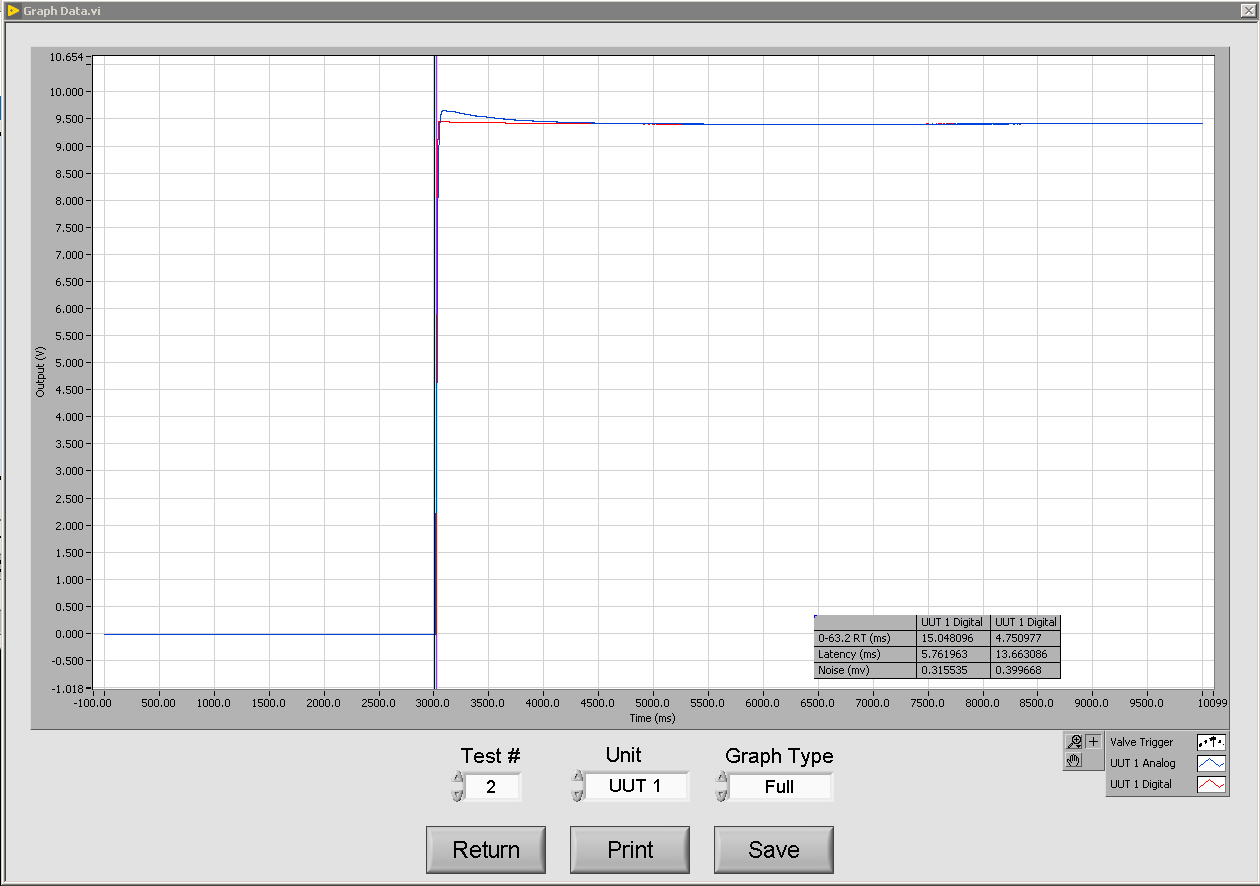
This VI is responsible for making the test calculations and determining if the test was successful in reaching the reach point. Depending on the number of units, and if the units are analog only, the scanned data from the Data Array is parsed. Then, the scanned data of each unit runs through a for loop, where noise and response time are calculated, and where the VI checks whether the test was successful. Then, the latency is calculated, and bundled with the other test calculations.

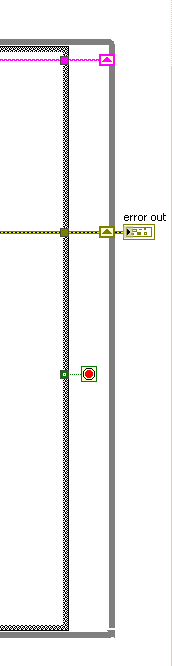
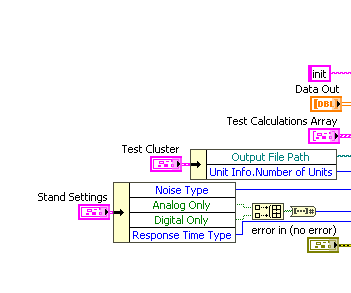
Frequency to Scan Period

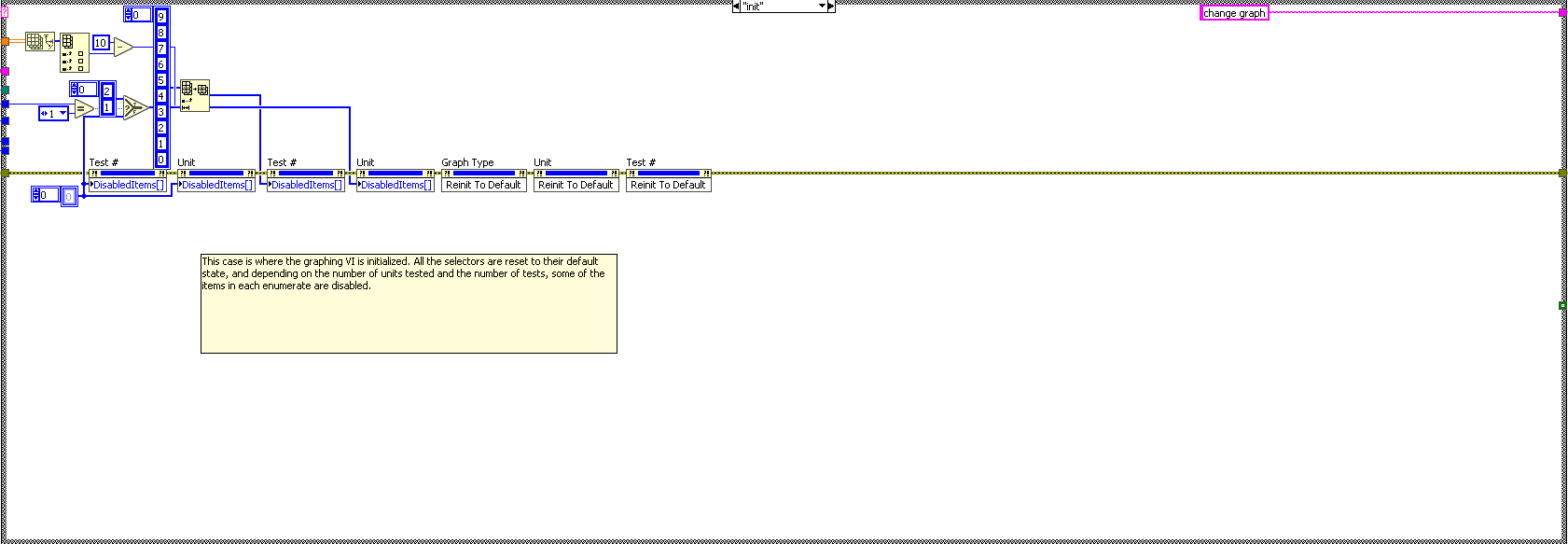


This VI is used to calculate the scan period in microseconds of the scan engine based on the inputted Scan frequency in Hz. It takes the reciprocal of the frequency and multiplies it by a million.

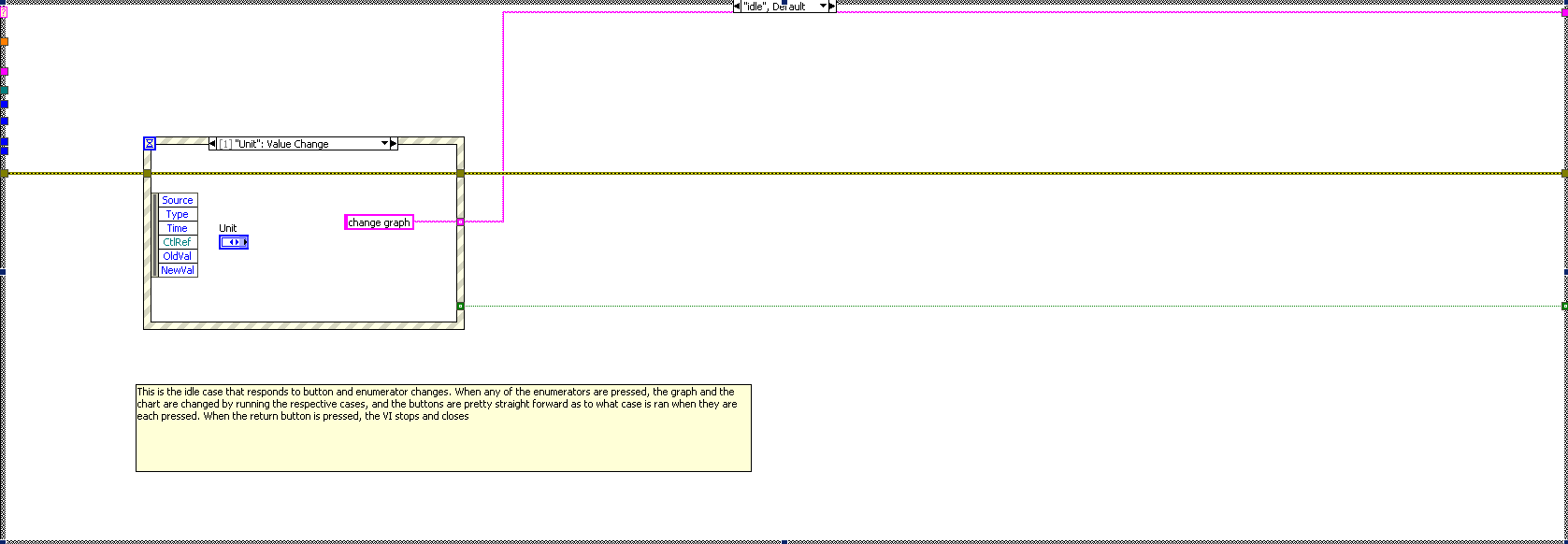
Graph Data

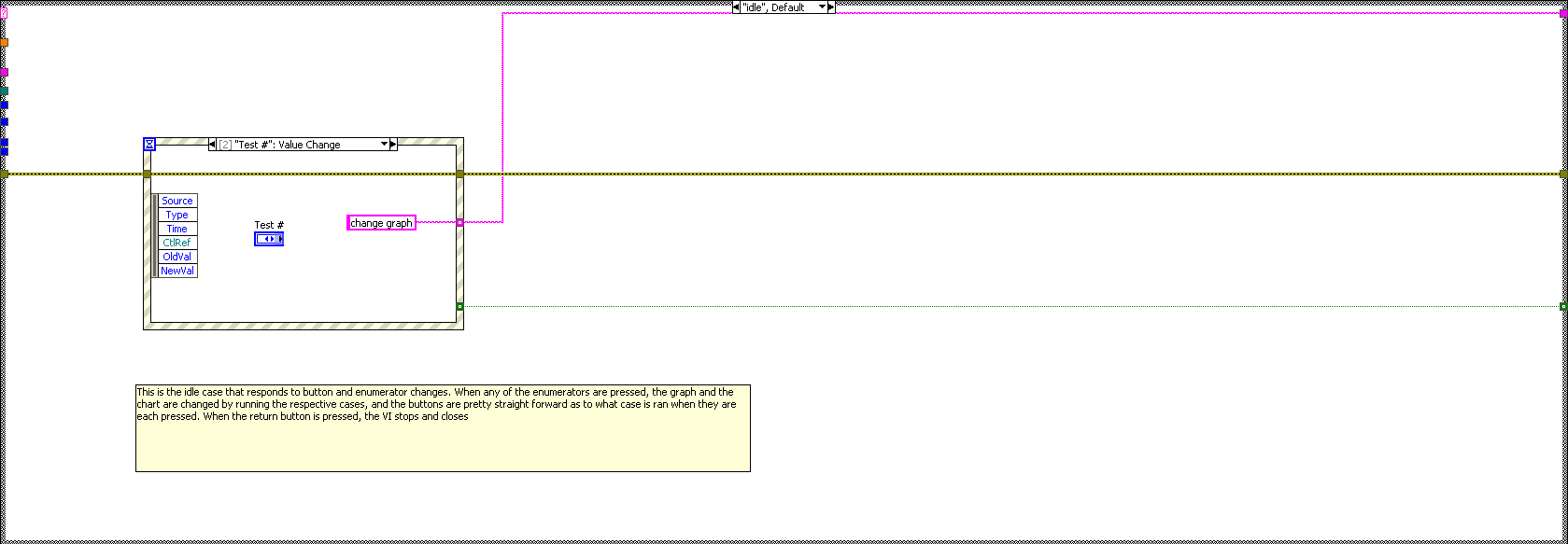


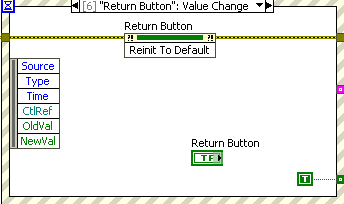
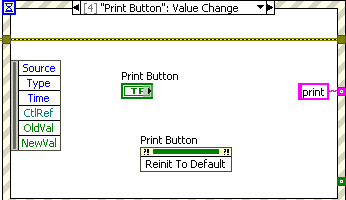


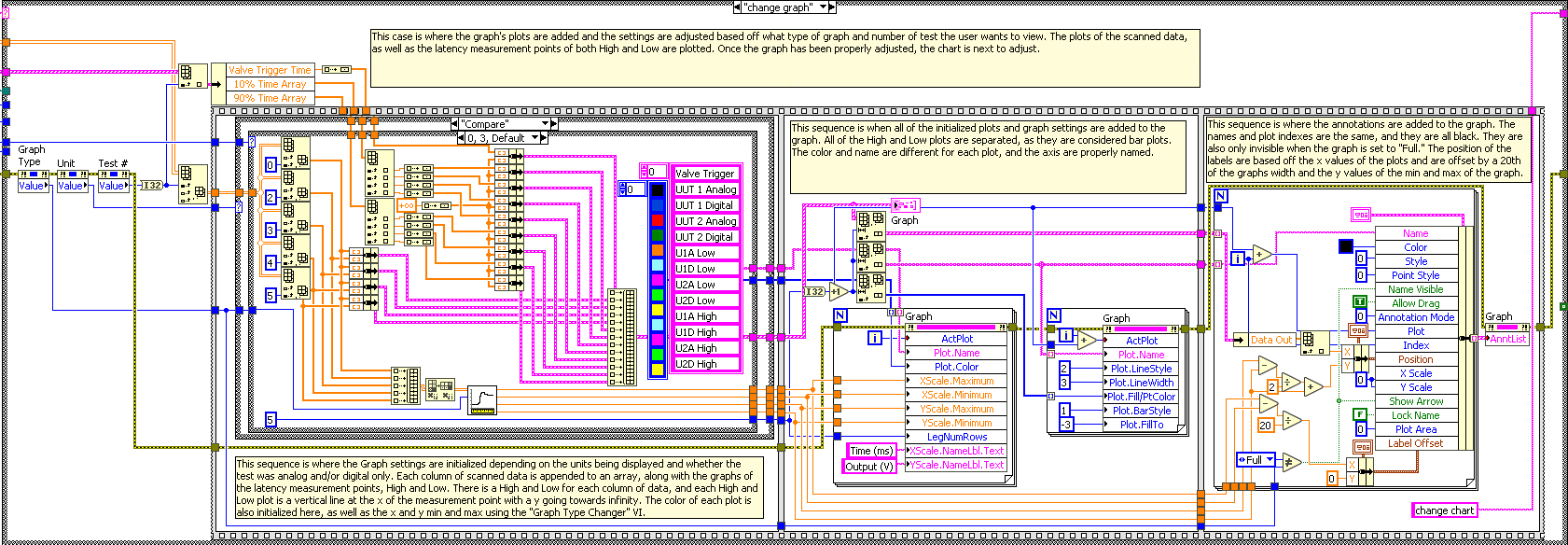


This case is where the graphing VI is initialized. All the selectors are reset to their default state, and depending on the number of units tested and the number of tests, some of the items in each enumerate are disabled.







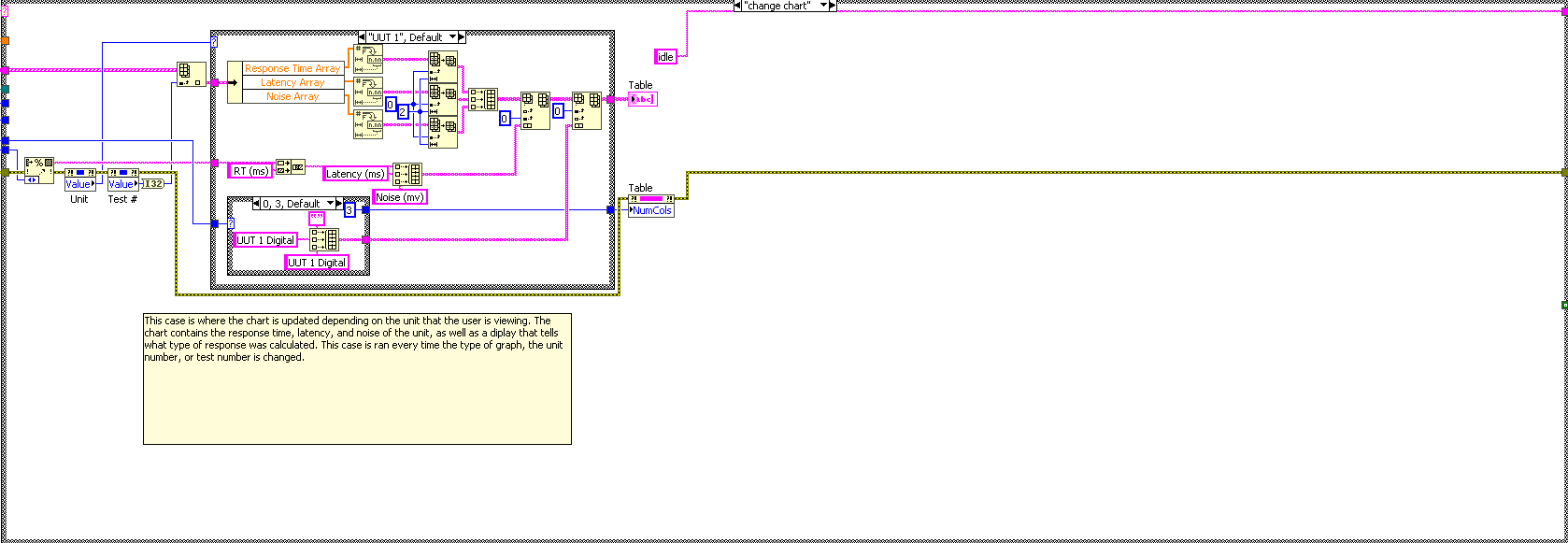


This case is where the graph's plots are added and the settings are adjusted based off what type of graph and number of test the user wants to view. The plots of the scanned data, as well as the latency measurement points of both High and Low are plotted. Once the graph has been properly adjusted, the chart is next to adjust.

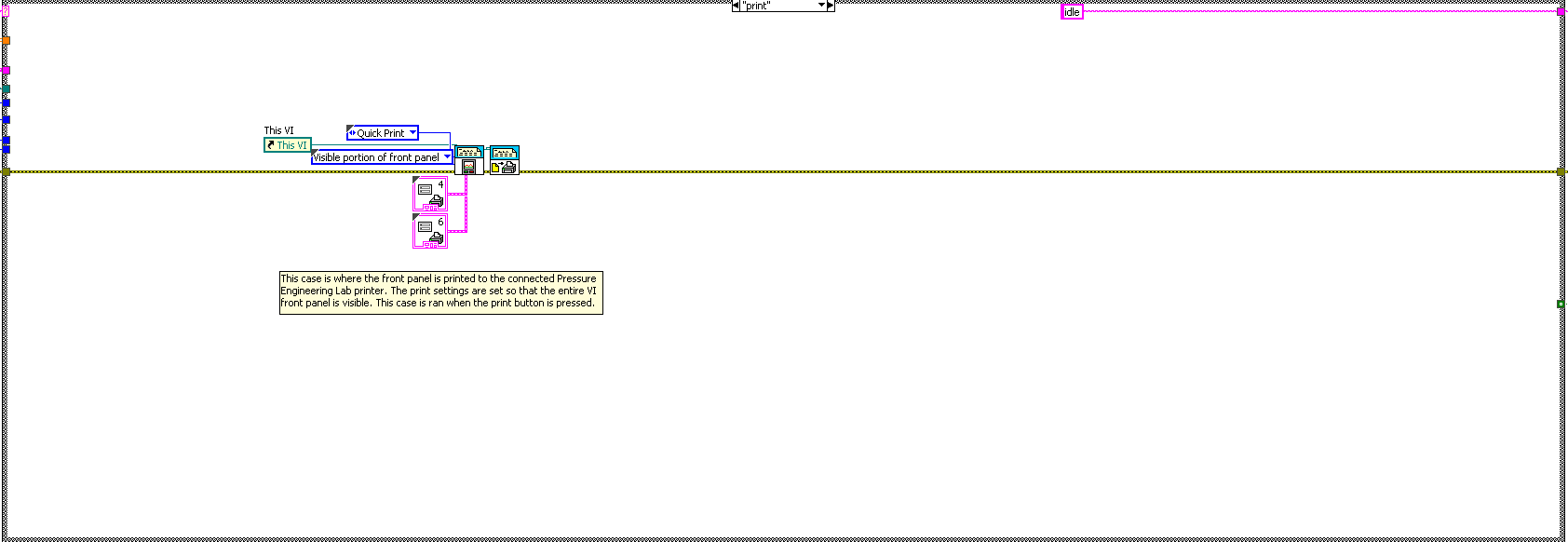
This sequence is where the Graph settings are initialized depending on the units being displayed and whether the test was analog and/or digital only. Each column of scanned data is appended to an array, along with the graphs of the latency measurement points, High and Low. There is a High and Low for each column of data, and each High and Low plot is a vertical line at the x of the measurement point with a y going towards infinity. The color of each plot is also initialized here, as well as the x and y min and max using the "Graph Type Changer" VI.

This sequence is when all the initialized plots and graph settings are added to the graph. All the High and Low plots are separated, as they are considered bar plots. The color and name are different for each plot, and the axis are properly named.

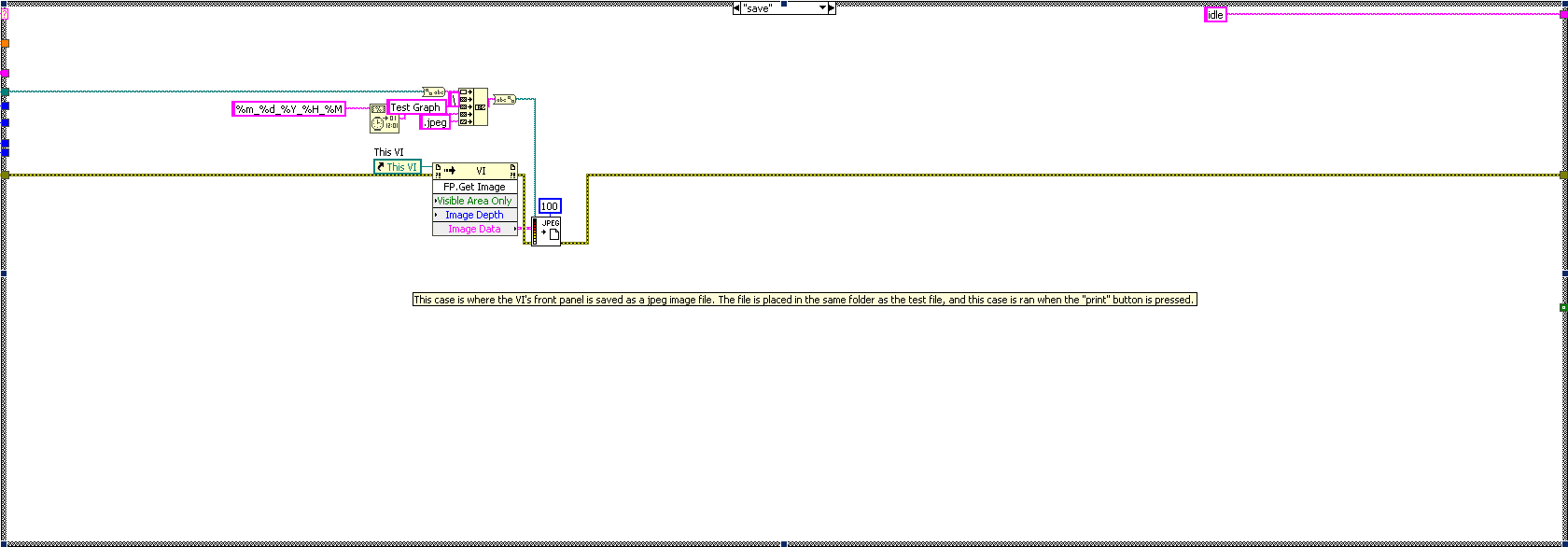
This sequence is where the annotations are added to the graph. The names and plot indexes are the same, and they are all black. They are also only invisible when the graph is set to "Full." The position of the labels is based off the x values of the plots and are offset by a 20th of the graphs width and the y values of the min and max of the graph.



This case is where the chart is updated depending on the unit that the user is viewing. The chart contains the response time, latency, and noise of the unit, as well as a display that tells what type of response was calculated. This case runs every time the type of graph, the unit number, or test number is changed.

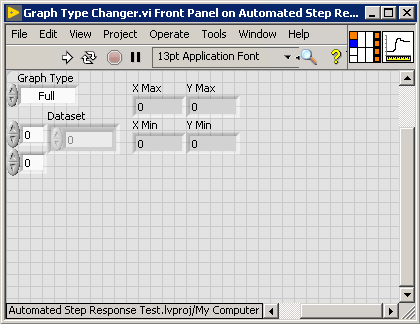


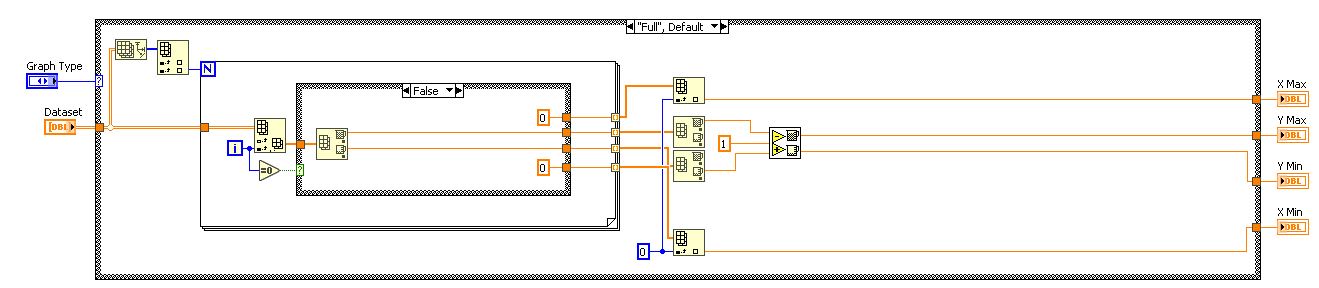
This case is where the front panel is printed to the connected Pressure Engineering Lab printer. The print settings are set so that the entire VI front panel is visible. This case runs when the print button is pressed.



This case is where the VI's front panel is saved as a jpeg image file. The file is placed in the same folder as the test file, and this case runs when the "print" button is pressed.

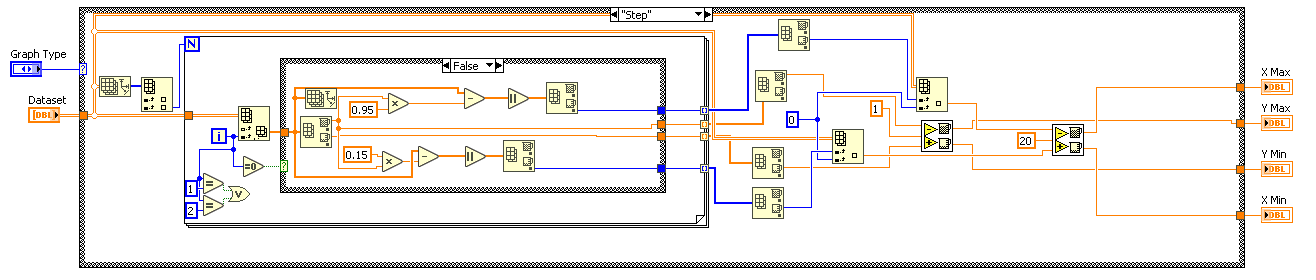
Graph Type Changer



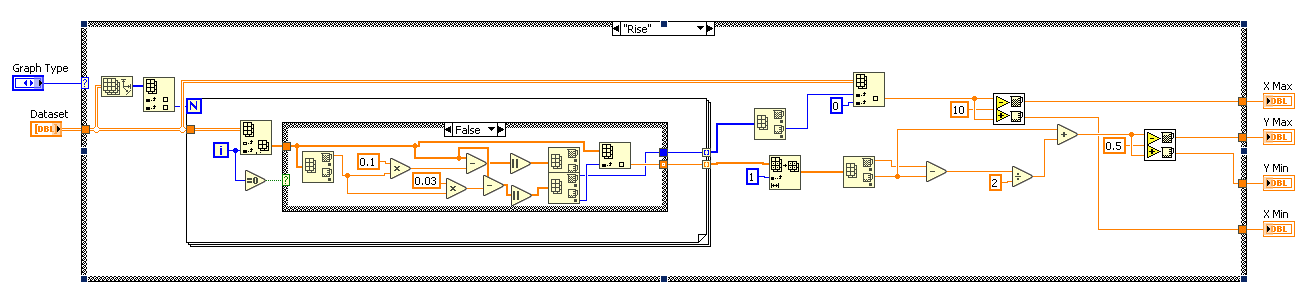


This VI is used to find the x and y min and max for the xy graph in the "Graph Data" VI. Depending on the graph type, the VI will use the inputted data to set the boundaries on the graph that best display the data the user wants.

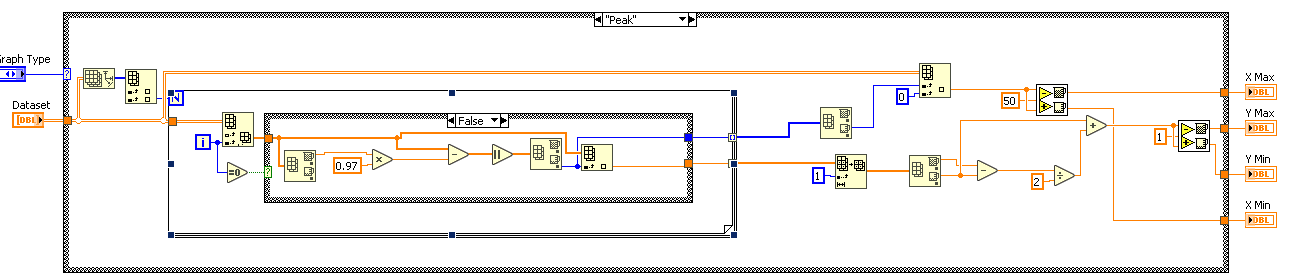
The Full graph setting is simply to show all the scanned data, so the highest and lowest points in time and output voltage are used to calculate the mins and maxs of the graph.



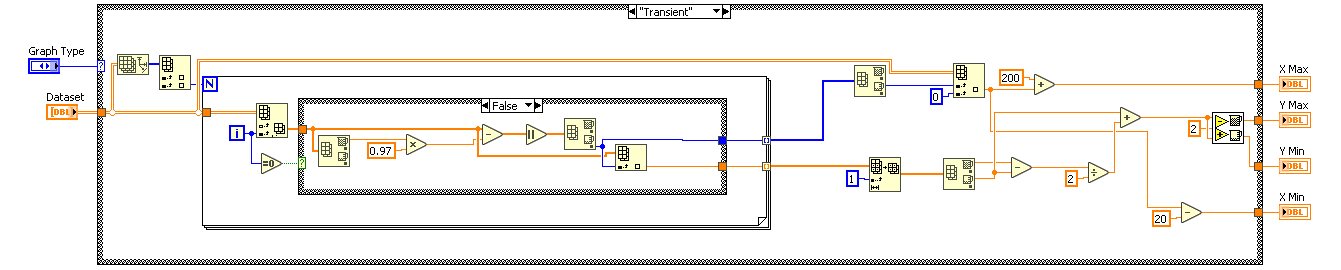
The Step graph setting is used to show the transition of the step and the climb to the response. It does so by taking the times at which the output voltage is at 15% and 95% of the maximum value reached and giving those times a 20 ms buffer on each side. The y min and max are the same and the full graph.



The Rise graph setting is used to show when the output of the unit begins to rise. The time at which the output reaches 3% of the maximum valve is used to set the x min and max, and the output at that time is used for the y min and max.

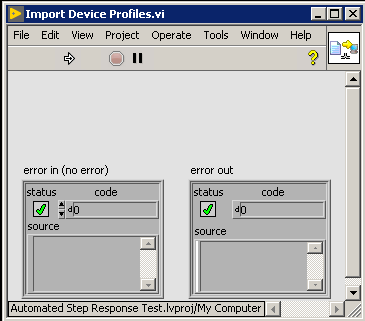


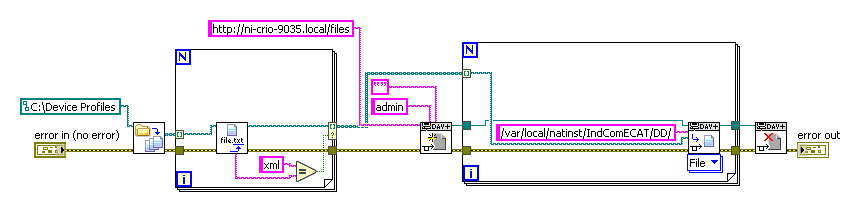
The Peak graph setting is used to show when the output of the unit slows down its response to the burst of pressure and begins to settle. The x min and max are found by taking the time at which the maximum value of the data is achieved, and the y min and max are based off that maximum value.



The Transient graph setting is used to show the Peak and a few ms beyond that. The y min and max are found the same way as the Peak setting, but this time the x max is set to a little farther.

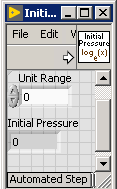
Import Device Profiles

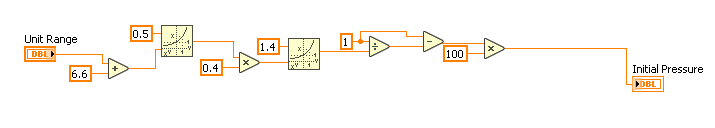




This VI is used to install the XML files from the "Device Profiles" directory on the C: Drive onto the RIO target. Using WebDAV protocols, the XML files only are added to the RIOs EtherCAT device profiles folder. If the file already exists on the RIO then it is simply overwritten.

Initial Tank Pressure Calculator

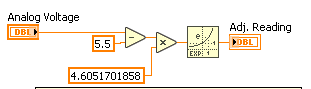
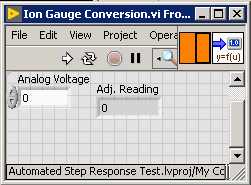




This VI is used to calculate the initial tank pressure based off the unit's range in Torr. It uses the equation:

100(1 - 1 / (0.4(6.6 + x) ^0.5)^1.4) ) - 0.5 = Tank Pressure

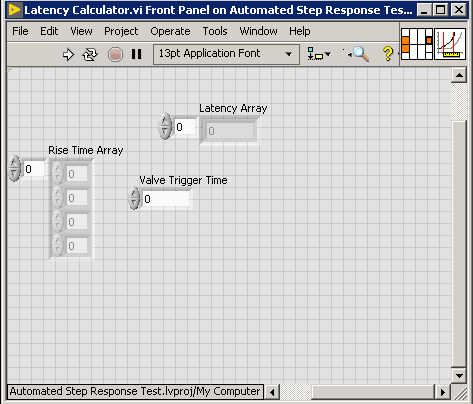
Ion Gauge Conversion

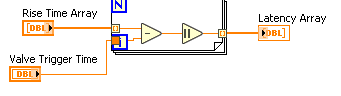


This VI is used to scale the Ion Gauge analog reading to its proper output. The equation for finding the reading based off the output voltage is:

exp(4.6051701858(x - 5.5)) = reading

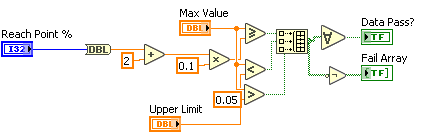
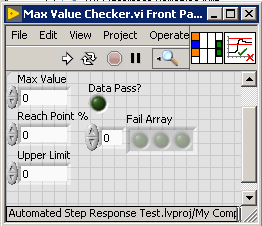
Latency Calculator





This VI is responsible for finding the latency of each scanned data array. Depending on the number of units, the Valve Trigger time is subtracted from the Rise Time of each array (the time at which the response of the unit begins to rise from its near vacuum voltage). These values give the latency of each unit, and are built into the Latency array.

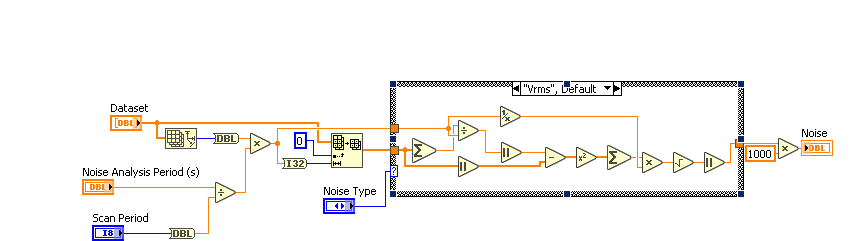
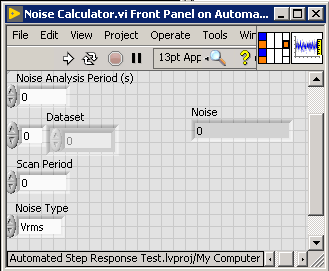
Max Value Checker



This VI is responsible for checking to see if the data from the test passes or not. The VI takes the reach point percentage as the lower limit, and 10 V as the upper limit and sees if the max value from the scanned data is in-between those numbers. This VI also appends to the Fail Array, which keeps track how the data failed, as well as if the valve did not fire, which is found by seeing if the max value is less than 0.05 V.

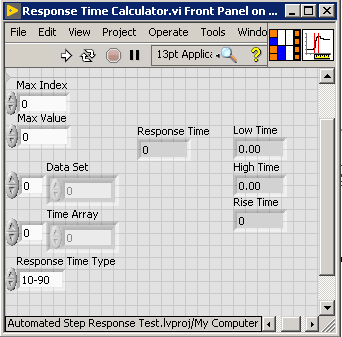
The reach point percentage is added to because the maximum value of the scanned data is not exactly equal to the resting response, so added a higher threshold for the data to pass is a way of circumventing the issue.

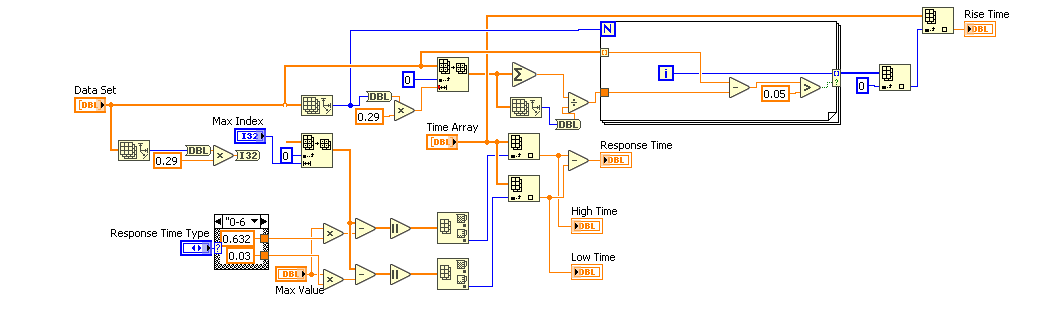
Noise Calculator



This VI is responsible for calculating the noise of the unit(s) being tested before the step is initiated. The scanned data array for each unit is parsed to include just the data before the step began. Then, depending on the settings, the noise is calculated using root mean square or peak-to-peak.

Response Time Calculator

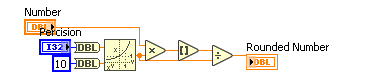
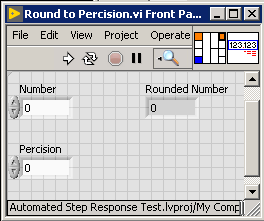




This VI is responsible for calculating the response time of the scanned data. The scanned data is parsed from the beginning to the point where the maximum value of the data is, or where the crest of the response would be. The type of response time decides whether the High and Low response measurements will be from 10-90 or from 0-63.2. From the parsed data, the High and Low of maximum value points in the data are found by subtracting those values from the array and finding the index of the lowest value (the data point that best matches the Low and High values). Those indexes are used to find the points of the time array where those values are present. The Time Array values are subtracted, giving the response time in milliseconds.

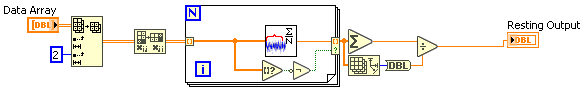
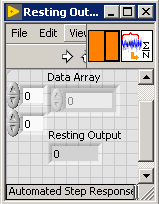
This VI also calculates the "Rise Time" of the scanned data. It does this by averaging the first 29% of the data (because the ALD is triggered at 30% of the test time) and subtracts it from the rest of the data in a for loop. Then, any value resulting from that subtraction that is greater than 0.05 has the index appended to an array of indexes. The first value in that array is then used to index the Time Array. This is done because the first value of the scanned data that is 0.05 V greater than the average of the unit at near vacuum would be the data point in which the rise of the response begins, thus any latency calculations would be made based off the value at that index in the Time Array.

Round to Precision



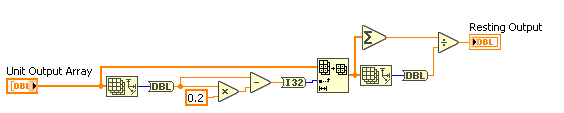
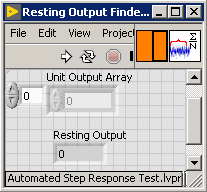
This VI can round a number to a specific precision. It does this by multiplying the number inputted by 10 to the power of the precision wanted. Then, the number is rounded and divided by that same number.

Resting Output Displayer



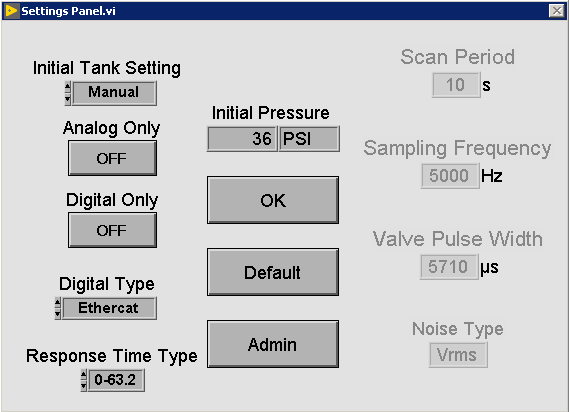
This VI is used to find the resting output of the units that have been tested and average them to display on the front panel display. Using the "Resting Output Finder" VI, each column of scanned data is calculated and then averaged to find the average resting output.

Resting Output Finder

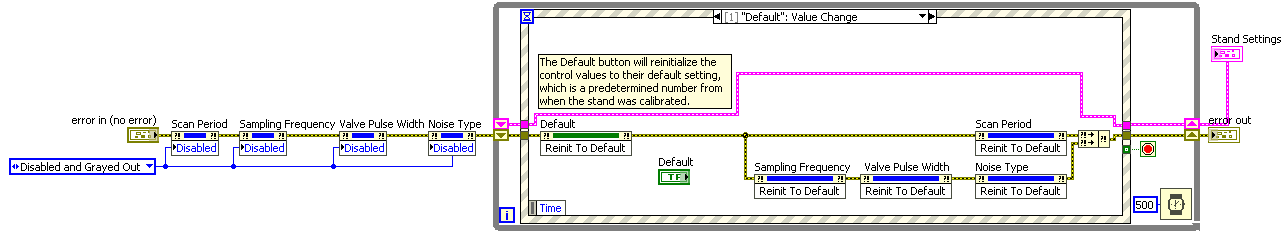


This VI is used to find the resting output of the scanned data inputted. The last 20% of the data is averaged to find this number.

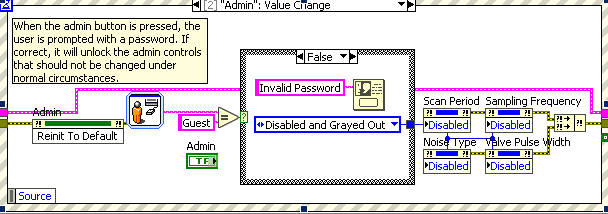
Settings Panel



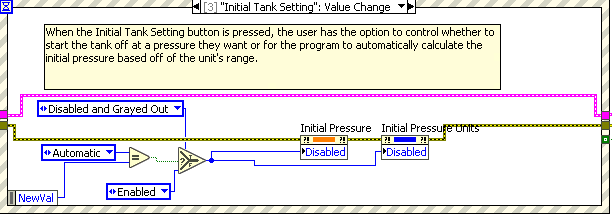
This VI is responsible for changing certain values that are used throughout the Test VI and change how the test will be carried out.



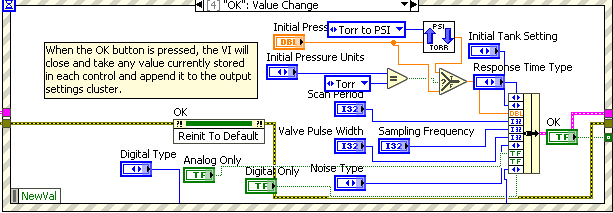
The Default button will reinitialize the control values to their default setting, which is a predetermined number from when the stand was calibrated.



When the admin button is pressed, the user is prompted with a password. If correct, it will unlock the admin controls that should not be changed under normal circumstances.

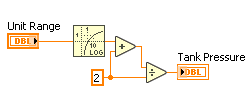
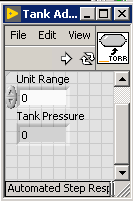


When the Initial Tank Setting button is pressed, the user has the option to control whether to start the tank off at a pressure they want or for the program to automatically calculate the initial pressure based off the unit's range.



When the OK button is pressed, the VI will close and take any value currently stored in each control and append it to the output settings cluster.

Tank Adjustment Pressure Calculator

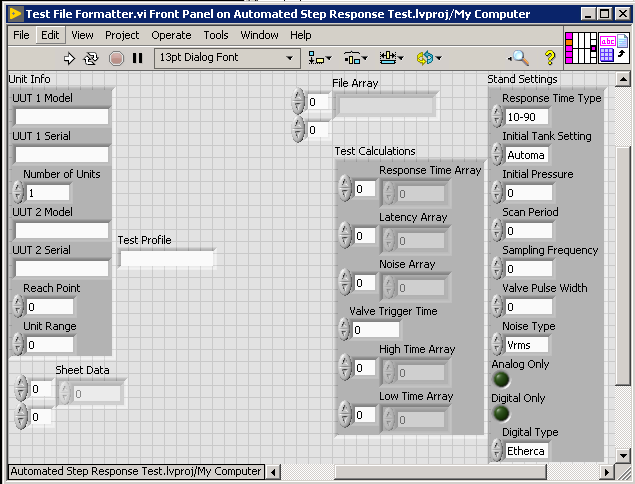


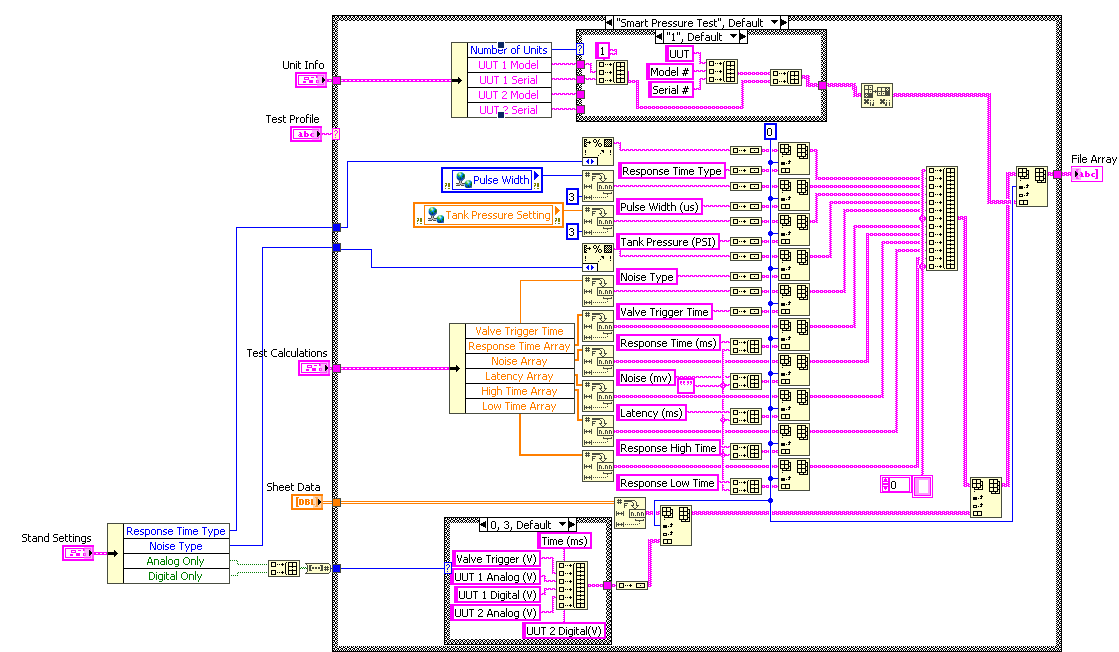
This VI is used to calculate the optimal tank pressure change based on the unit range.

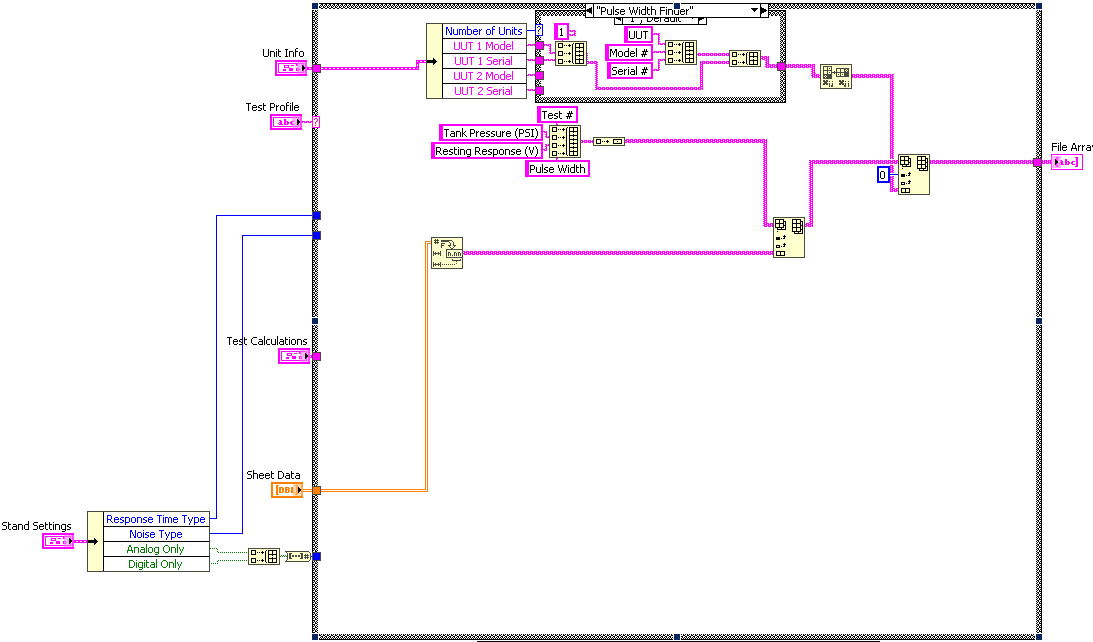
The formula used is:

(log(x) + 2) / 2 = Adjustment pressure

Test File Formatter

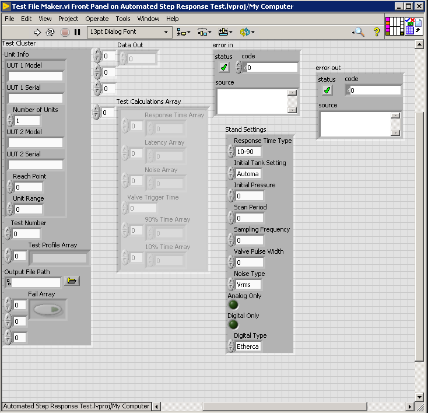


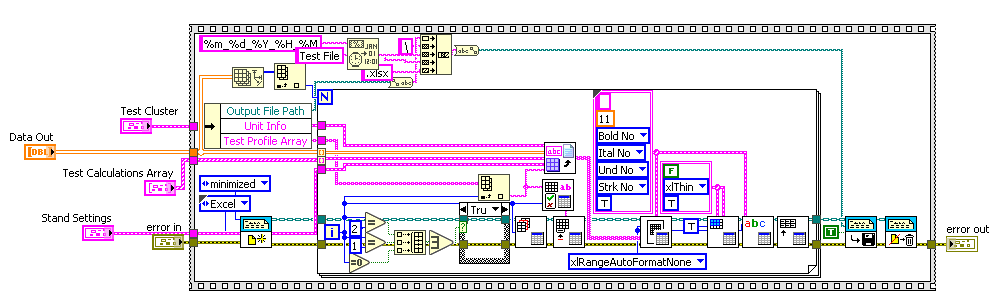




This VI is responsible for formatting the test data so that it can be properly appended to the test file. The way it is formatted depends on the type of test; Set Pressure, and both Variable Pressure tests are the same, and the Finder type tests are the same as well. This was made because the data that is being appended to the file varies based on the type of test. For the Pressure Tests, the file data includes the calculations plus the readings from the units and trigger and feedback. For the Finder Tests, the data just includes the test settings along with the resting response of the unit.

Test File Maker

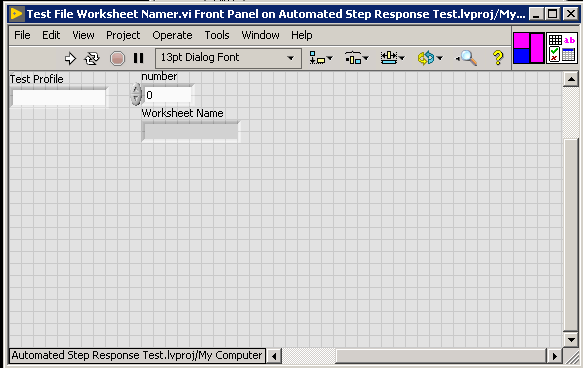


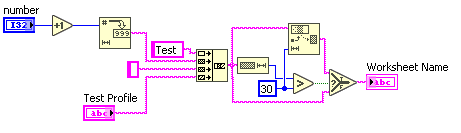


This VI is responsible for creating the test file. An excel report is created, along with the file name of which is the time when the file is being made. A for loop is ran once for each test, creating a new worksheet and appending the data. The data is formatted in the Test File Formatter VI, which runs differently depending on the type of test. Once the report is complete, it is saved to the file and then deleted.

It is important to note that this version of Excel autogenerates 3 worksheets when opened. If the number of tests is less than 3, there will be one or two empty sheets titled "Sheet 2" and/or "Sheet 3." If the number of tests is 3 or greater, the sheets will be renamed and the proper file data will be appended to them. There is no current way to programmatically delete the extra worksheets without making the LabVIEW code specific to this machine only.

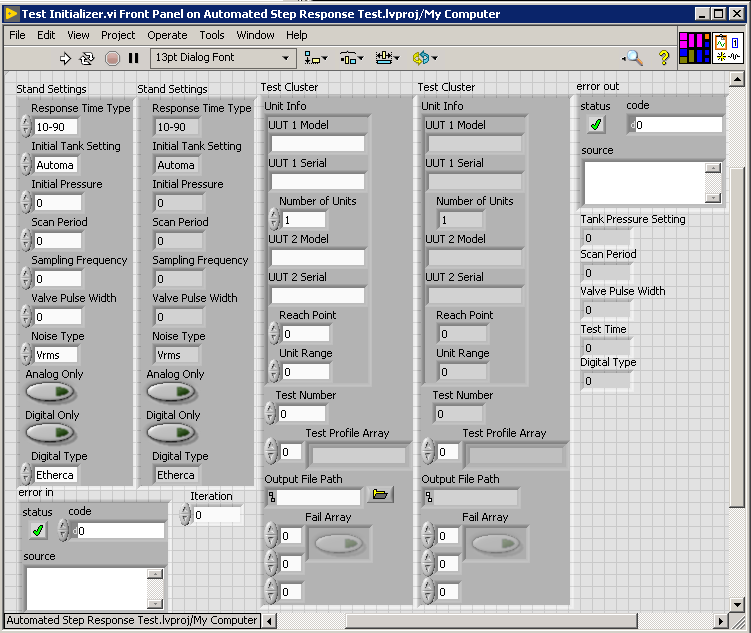
Test File Worksheet Namer

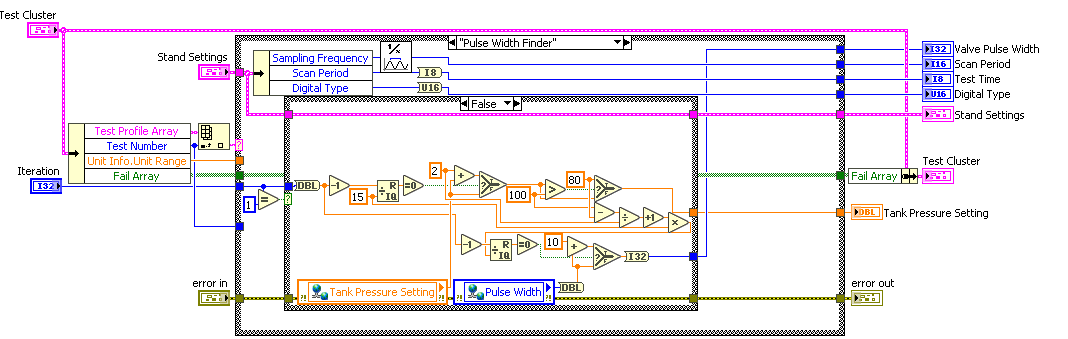


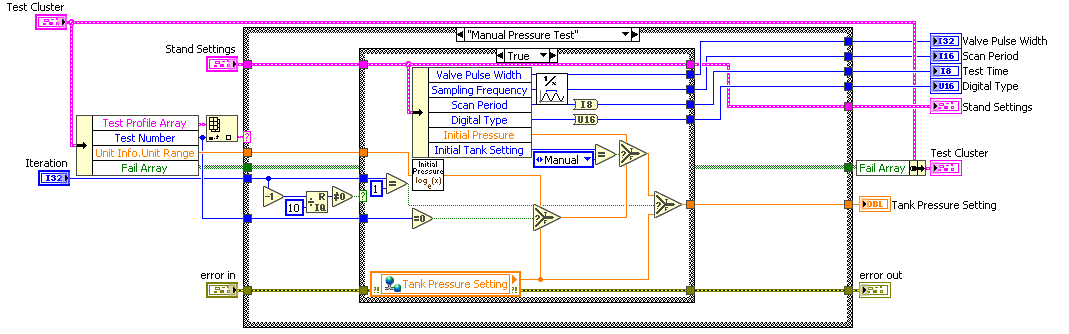


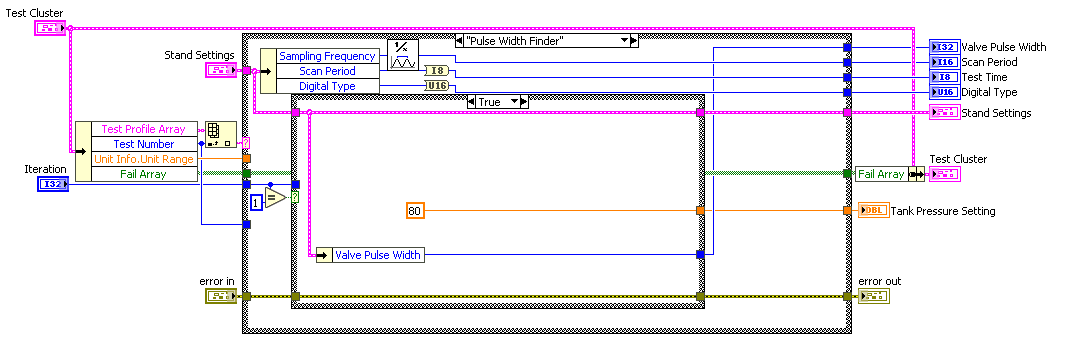
This VI is responsible for naming each worksheet in the Test File. It takes the iteration of the for loop, and creates a unique name for the worksheet including the name of the test. This VI was created because you cannot create two worksheets with the same name, and keeping track of how many different types of individual tests were run would be too complicated considering this much easier solution.

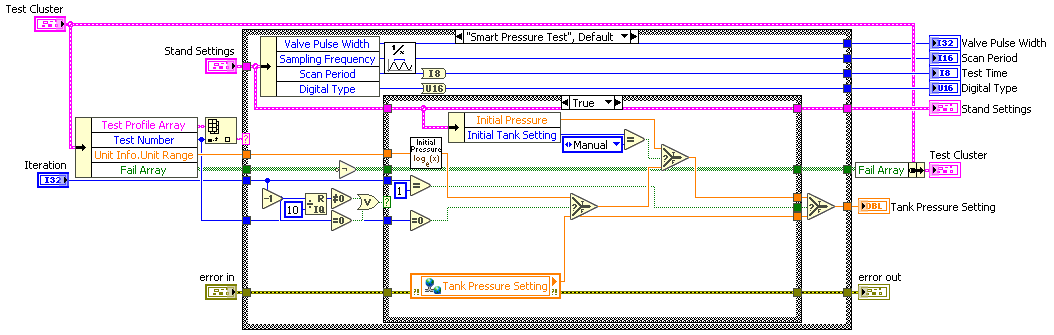
Test Initializer

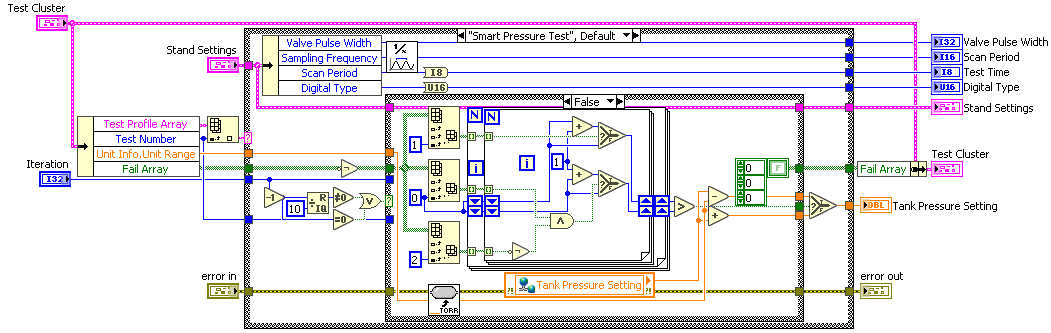












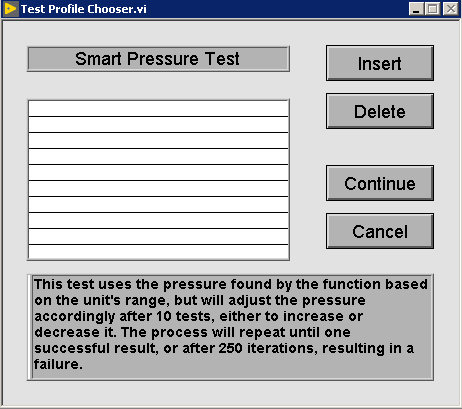
This VI is used to change the test settings based on the current test profile. The Stand Settings, Test Cluster, Iteration, and error are wired in and the pulse width, scan period, test time, digital type, stand settings, Test cluster, tank pressure setting, and error are wired out.

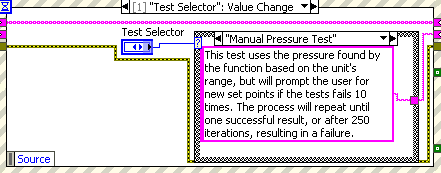
For the Smart Pressure Test, there are a lot of conditions that control the conditions of the test. If it's the first iteration of the test, the tank pressure is either automatically calculated by the initial pressure calculator VI, or by the manually-set initial pressure set by the user in the settings panel. However, if it is not the very first test in the Test Profile Array, then the initial tank pressure is set to be wherever the last tank pressure setting was. Every 10 tests the Fail Array is checked to see how the previous tests failed. If the tests failed by being too high, the tank pressure is lowered by a pressure found by the tank adjustment pressure calculator VI, and it the tests were too low, then the tank pressure is lowered. If the valve did not fire (third column of the Fail Array), then the "too low" result of the test is not compared to the "too high" results. This is too preventing the times where the pressure is too much, but because the valve did not fire, the test was considered "too low." The Fail Array is also reset after every 10 tests.

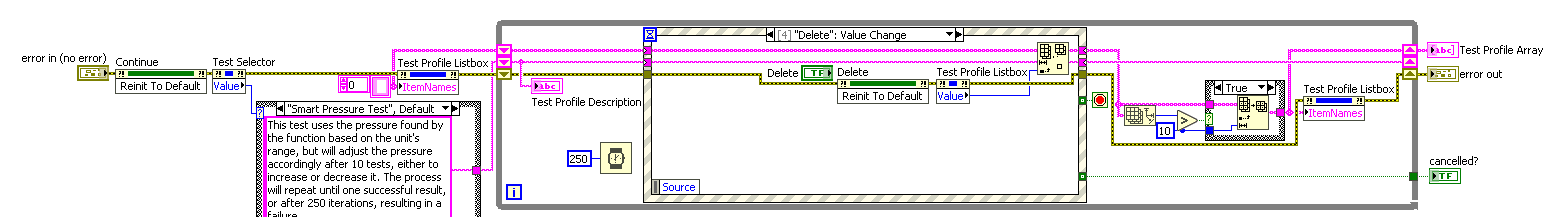
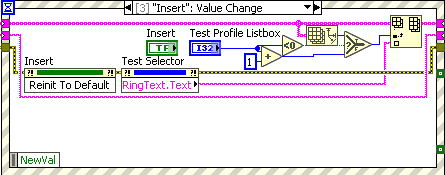
For the Manual Pressure Test, the settings panel is brought up every 10 tests, allowing the user to customize the settings of the test.

For the Pulse Width Finder, the tank pressure is incremented from 80-100 PSI by 2, with 20 tests at each pressure setting. Once the 20 tests are done at 100 PSI, the pressure is set back to 80, and the pulse width is increased by 10 us.

Test Profile Chooser

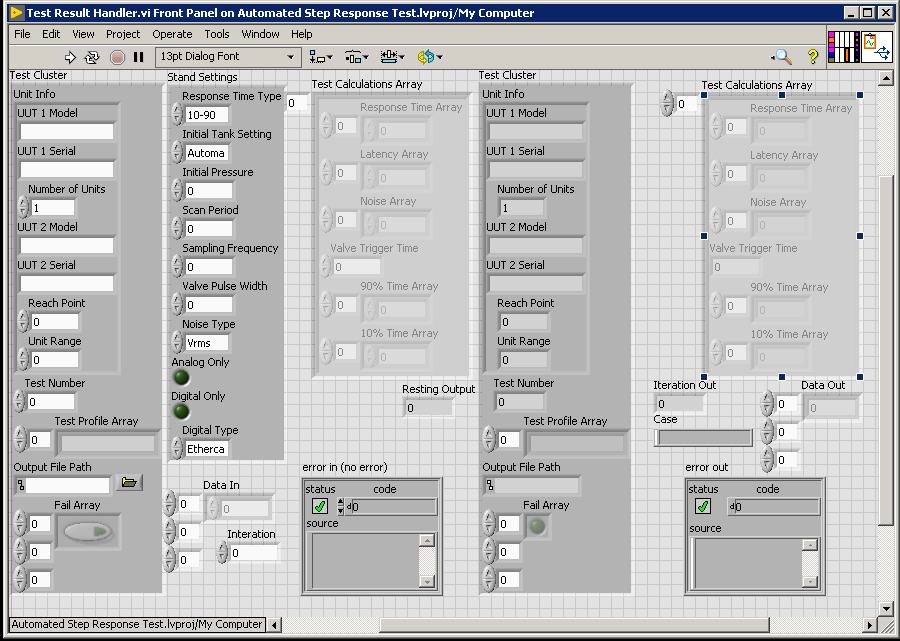


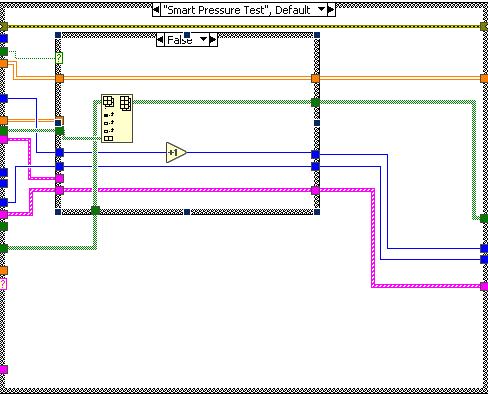
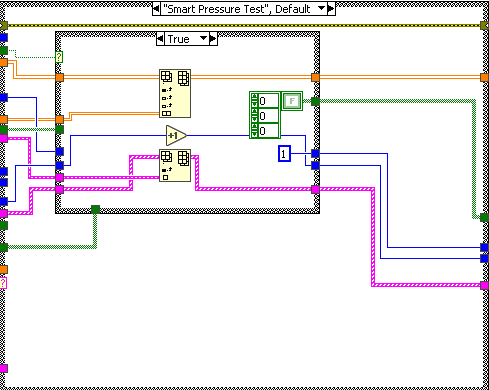


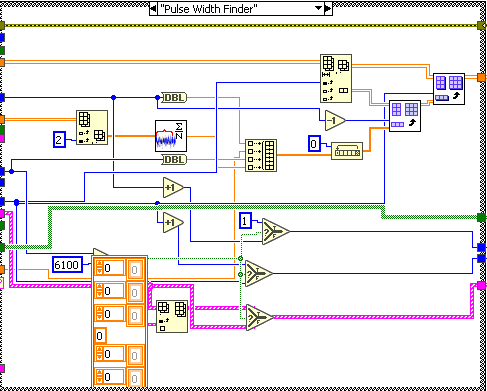


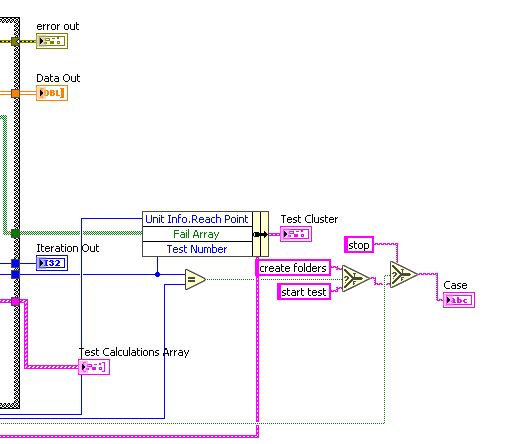
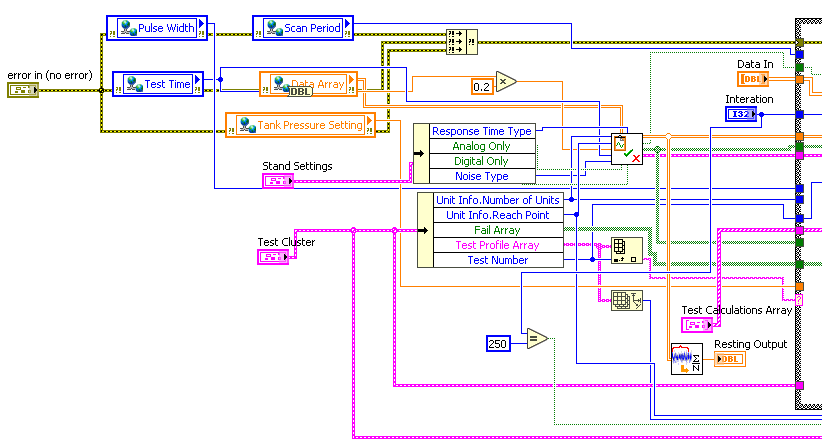
This VI is responsible for creating the test profile array that will control what tests are run. It is like the MAT Stand test selector, where the items can be added, rearranged, and deleted. Each profile has its own description that tells the user what the test does. Once the test profiles have been selected, the array of profiles is outputted. This popup window can also be cancelled.

Test Result Handler









This VI is responsible for collecting the test and stand data, and deciding what to do next based on the type and the result of the test. After calculating the response time, latency, and noise, and after deciding if the test was successful, this VI handles the appending of the test data and the next steps in the test case.

Pressure Test:

If the test passed, the scanned data and calculations are appended, the test number increases, and the iteration is set back to 1. If the test failed, then the iteration is incremented. For each Smart Variable Pressure Test, the Fail Array is also appended for every failure and reset for every success.

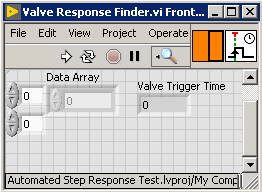
Finder:

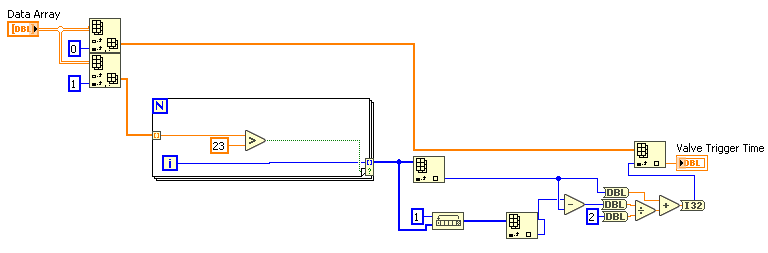
Regardless of the result of the test, the iteration, tank pressure, pulse width, and resting result of UUT 1 Analog are recorded every time. The test calculations are also disregarded. When the pulse width is greater than 6.1 ms, the test is then complete and the iteration is reset, the test number increases, and a placeholder empty calculations array is added to the test calculations.

Depending on if the number of tests completed is equal to the size of the test profile array (the number of tests), the next case is decided; either beginning the next test or creating the test folder.

If the number of iterations reaches 250, then the test is stopped regardless of the test success.

Valve Response Finder



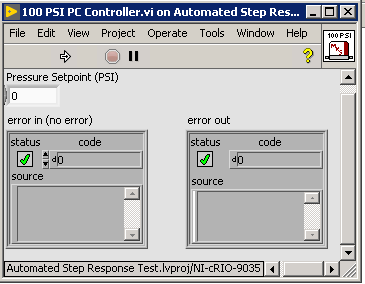


This VI is responsible for finding the time at which the high-speed valve triggered. The trigger time is used to calculate the latency of the scanned data. The trigger time is found by taking parsed Trigger and Feedback arrays, subtracting half of the maximum value in either of them, and finding the index of the smallest value (the index of the value closest to 50% of the max). Then, the index is used to find the time in the Time Array at which to measure the latency from.

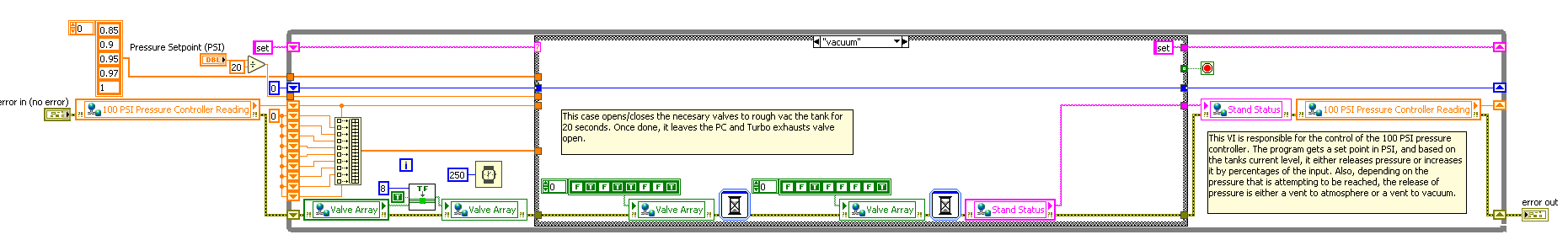
The Trigger and Feedback arrays are parsed to get the rise of voltage of the signal sent to/received from, rather than the fall. Because both signals are not clean square waves, the time at which the latency should be measured from should be before the index of the maximum value of each array, rather than afterward. Thus, each array is parsed accordingly.

**RIO VIs**

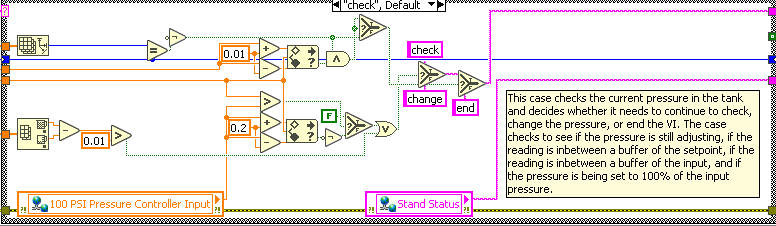
100 PSI/500 Torr PC Controller



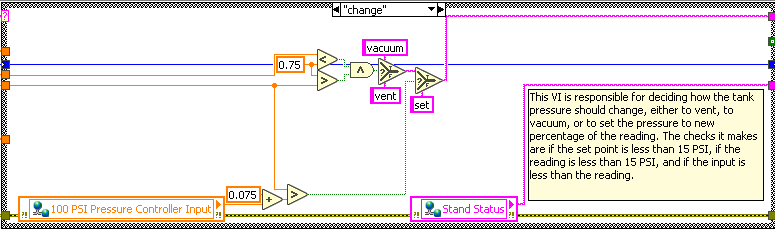
This VI is responsible for the control of the 100 PSI pressure controller. The program gets a set point in PSI, and based on the tanks current level, it either releases pressure or increases it by percentages of the input. Also, depending on the pressure that is attempting to be reached, the release of pressure is either a vent to atmosphere or a vent to vacuum.



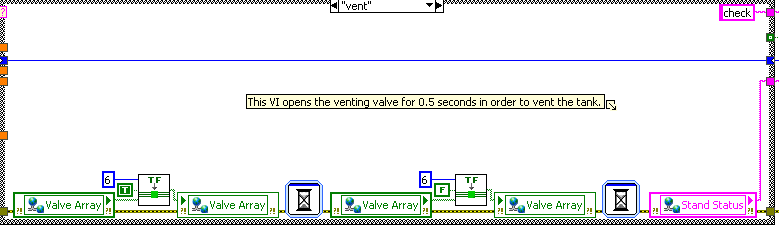
This case opens/closes the necessary valves to rough vacuum the tank for 20 seconds. Once done, it leaves the PC and Turbo exhausts valve open.



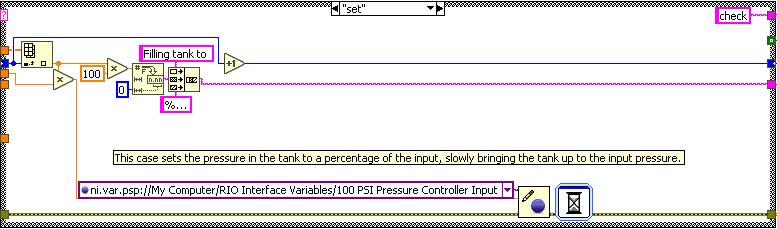
This case checks the current pressure in the tank and decides whether it needs to continue to check, change the pressure, or end the VI. The case checks to see if the pressure is still adjusting, if the reading is inbetween a buffer of the setpoint, if the reading is inbetween a buffer of the input, and if the pressure is being set to 100% of the input pressure.



This VI is responsible for deciding how the tank pressure should change, either to vent, to vacuum, or to set the pressure to new percentage of the reading. The checks it makes are if the set point is less than 15 PSI, if the reading is less than 15 PSI, and if the input is less than the reading.

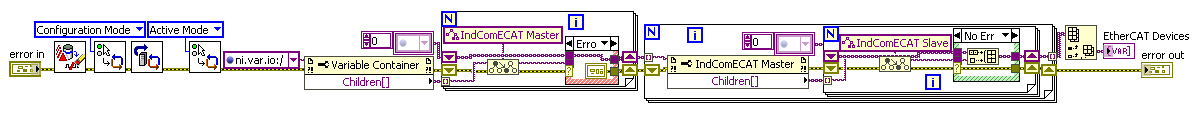
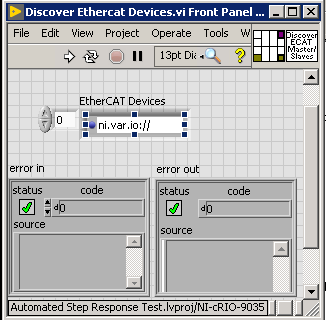


This VI opens the venting valve for 0.5 seconds in order to vent the tank.



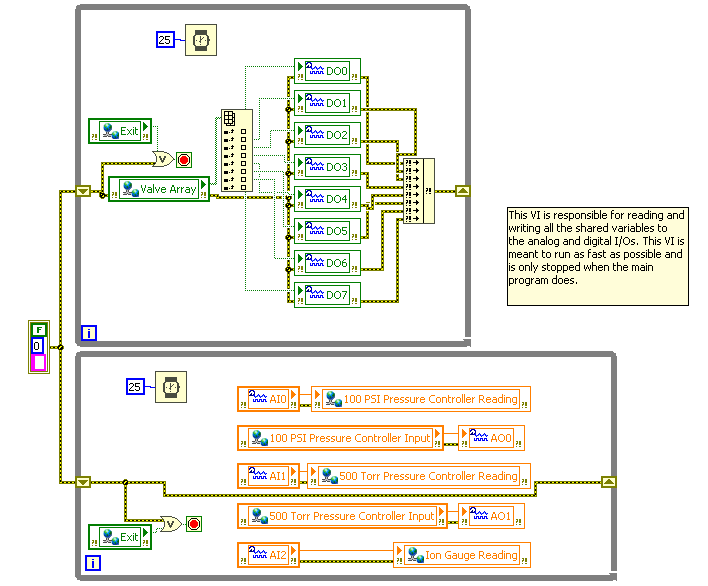
This case sets the pressure in the tank to a percentage of the input, slowly bringing the tank up to the input pressure.

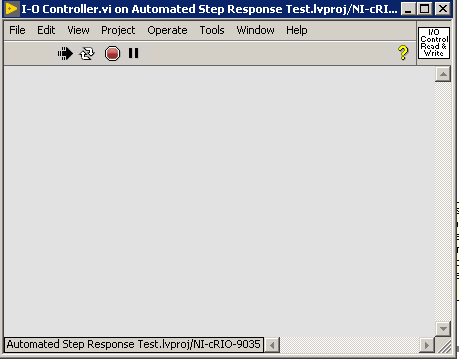
Discover EtherCAT Devices



This VI is used to detect EtherCAT devices that are connected to the CompactRIO. It does so by first resetting the scan engine and refreshing the modules. Then, it searches the RIO for connected devices. If any of those connected devices are EtherCAT Masters, it will search the EtherCAT Masters for EtherCAT Slaves. If Slaves are found, then the VI will export the array of connected Slaves.

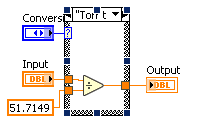
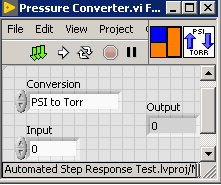
I-O Controller





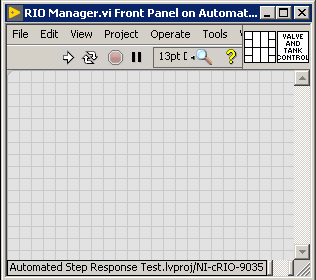
This VI is responsible for reading and writing all the shared variables to the analog and digital I/Os. This VI is meant to run as fast as possible and is only stopped when the main program does.

Pressure Converter

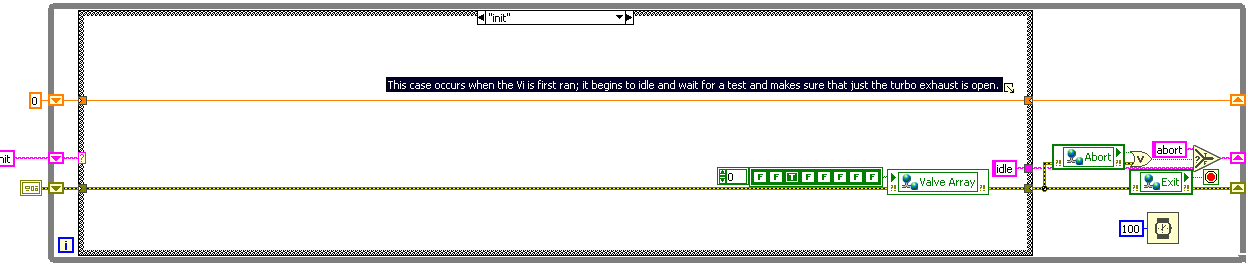


Converts pressure from Torr to PSI, or vice versa.

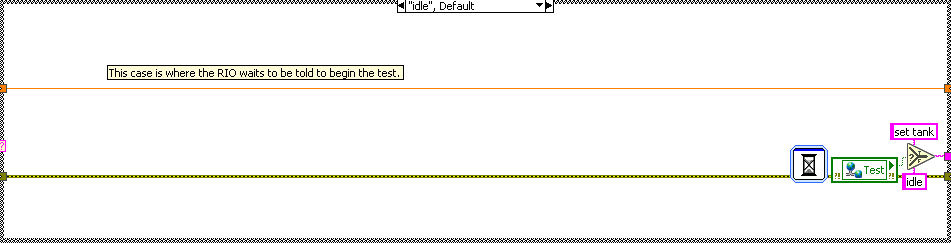
RIO Manager



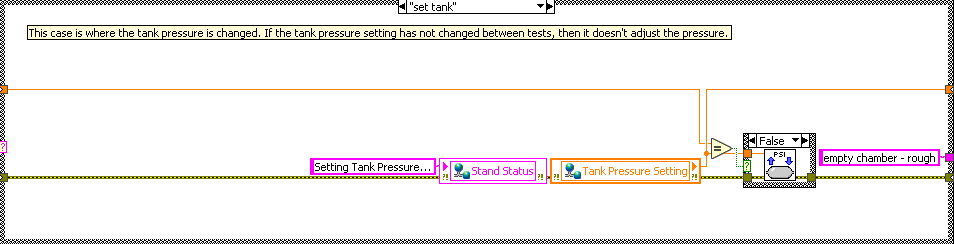
This VI is responsible for managing and controlling all the elements that are connected to the RIO. This is where the test procedure runs and where the scan occurs.



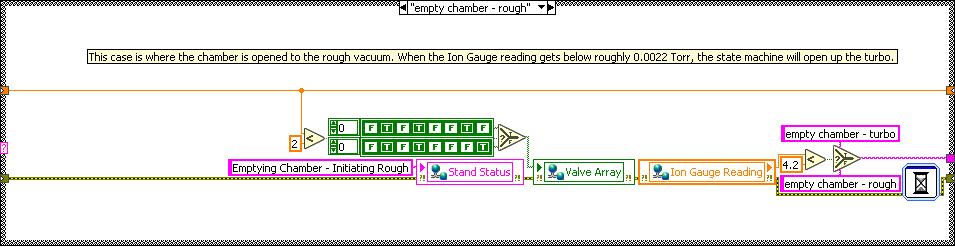
This case occurs when the Vi is first ran; it begins to idle and wait for a test and makes sure that just the turbo exhaust is open.



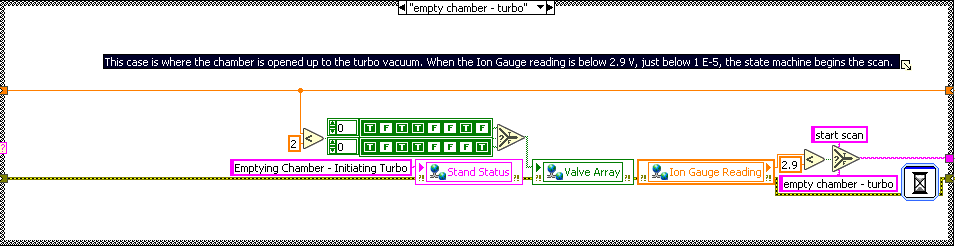
This case is where the RIO waits to be told to begin the test.



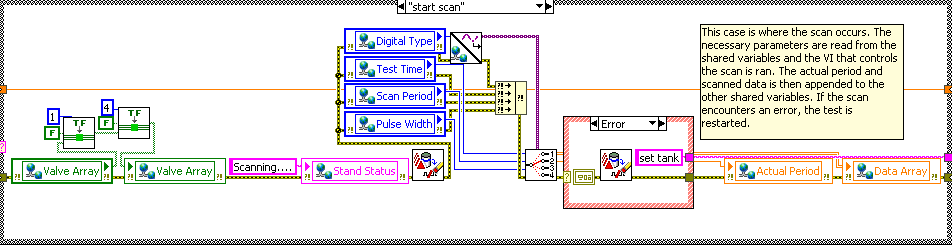
This case is where the tank pressure is changed. If the tank pressure setting has not changed between tests, then it doesn't adjust the pressure.



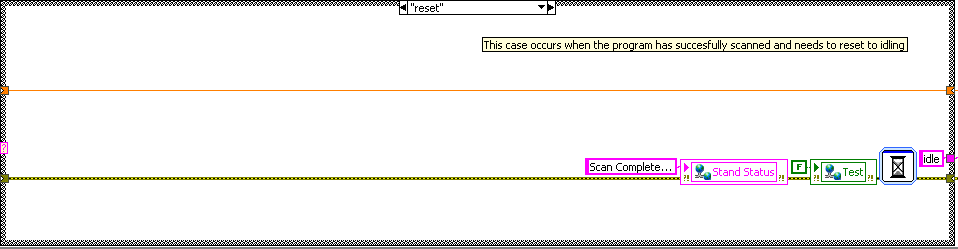
This case is where the chamber is opened to the rough vacuum. When the Ion Gauge reading gets below roughly 0.0022 Torr, the state machine will open the turbo.



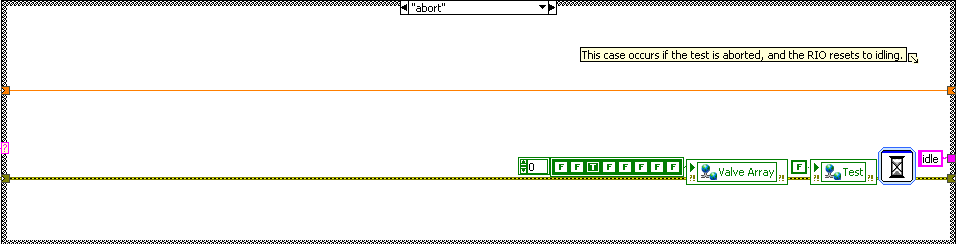
This case is where the chamber is opened to the turbo vacuum. When the Ion Gauge reading is below 2.9 V, just below 1 E-5, the state machine begins the scan.



This case is where the scan occurs. The necessary parameters are read from the shared variables and the VI that controls the scan is ran. The actual period and scanned data is then appended to the other shared variables. If the scan encounters an error, the test is restarted.

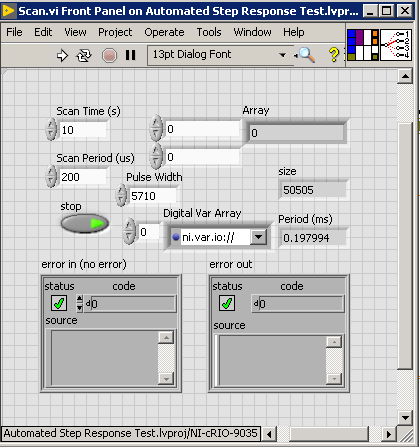


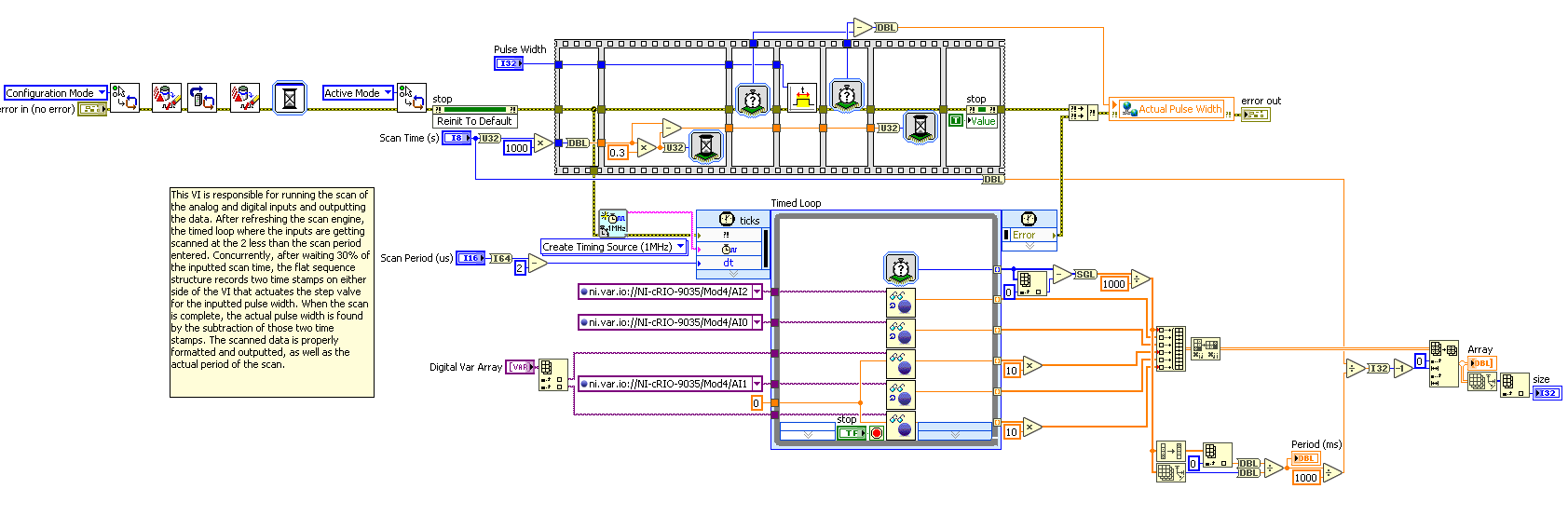
This case occurs when the program has successfully scanned and needs to reset to idling.



This case occurs if the test is aborted, and the RIO resets to idling.

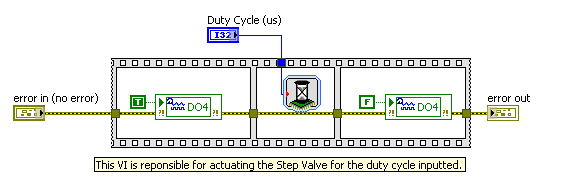
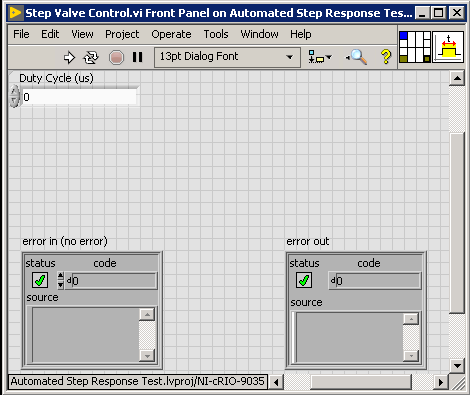
Scan





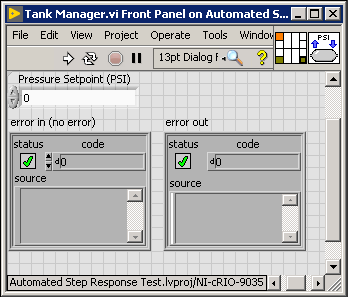
This VI is responsible for running the scan of the analog and digital inputs and outputting the data. After refreshing the scan engine, the timed loop where the inputs are getting scanned at the 2 less than the scan period entered. Concurrently, after waiting 30% of the inputted scan time, the flat sequence structure records two time-stamps on either side of the VI that actuates the step valve for the inputted pulse width. When the scan is complete, the actual pulse width is found by the subtraction of those two time-stamps. The scanned data is properly formatted and outputted, as well as the actual period of the scan.

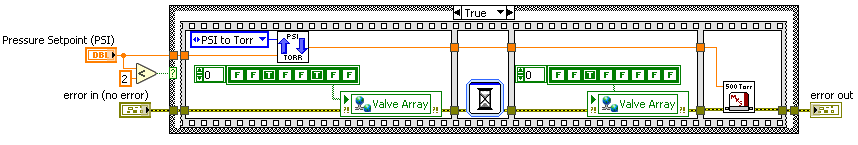
Step Valve Control



This VI is responsible for actuating the Step Valve for the duty cycle inputted.

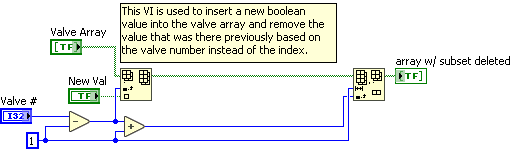
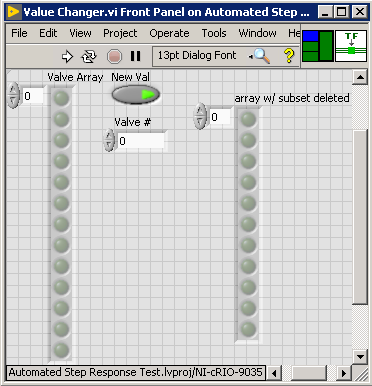
Tank Manager





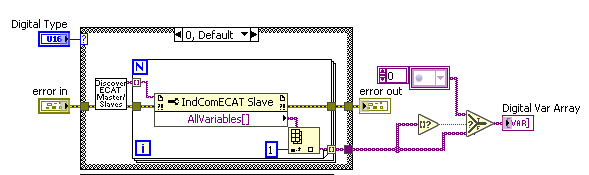
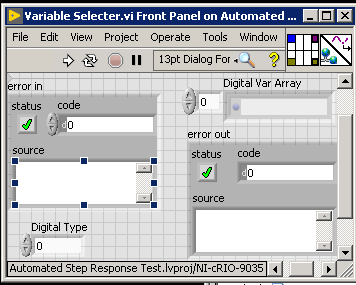
This VI is responsible for deciding the PC to use depending on the inputted tank pressure. If the input is less than 2 PSI, the VI will first vent the tank to atmosphere, to make sure the 500 Torr PC does not experience a pressure above its limitations (roughly 45 PSIA). Then, it will run the 500T PC controller VI. If the input is above 2 PSI, then the 100PSI PC controller VI will run.

Valve Changer



This VI is used to insert a new Boolean value into the valve array and remove the value that was there previously based on the valve number instead of the index.

Variable Selector



This VI is used to get the digital variables needed to complete the digital scan of the connected units. As of now, other EtherCAT units can be connected to the RIO. After discovering the EtherCAT devices using the VI, the variables of the outputted Slaves are found. The pressure reading PDO variable for all EtherCAT units is the second variable in the found variable array, so it is indexed and added to an array. If the variable array is empty, the VI makes sure it outputs a proper empty array.

**Variables**

