TERADYNE TIM 2017

Di-Series Debugging Tools

Tarra Marchetti | Teradyne | North Reading, MA| tarra.marchetti@teradyne.com

Contents

Introduction

Whether performing new TPS development, TPS rehost, or routine testing, there is always a chance of running into unexpected results that require debug. When those tests involve the Teradyne Di-Series digital test instrument, numerous tools are installed as part of the standard product. This paper will explain what these tools are, when they can be used, and how to use them.

Di-Series Alarm Viewer

The Di-Series Alarm Viewer is installed with iStudio Digital Test Editor V3.2 and higher. This tool can be used to easily identify when a Di-Series instrument experiences an alarm condition from the test environment. These alarm conditions are: OverVoltage, OverCurrent, and OverTemperature.

The Di-Series Alarm Viewer is automatically started during the iStudio Digital Test Editor installation with software versions 4.3 and later. For earlier versions, the Alarm Viewer must be manually started. When an event occurs with the alarm viewer active, a notification bubble can be seen from the taskbar icon tray. The Alarm Viewer can be launched to view additional alarm information by selecting from the taskbar icon tray, or from the Start menu -> Teradyne -> Di-Series -> Alarm Viewer.

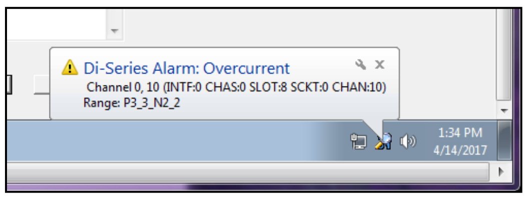


Figure 1 - Alarm Viewer Notification

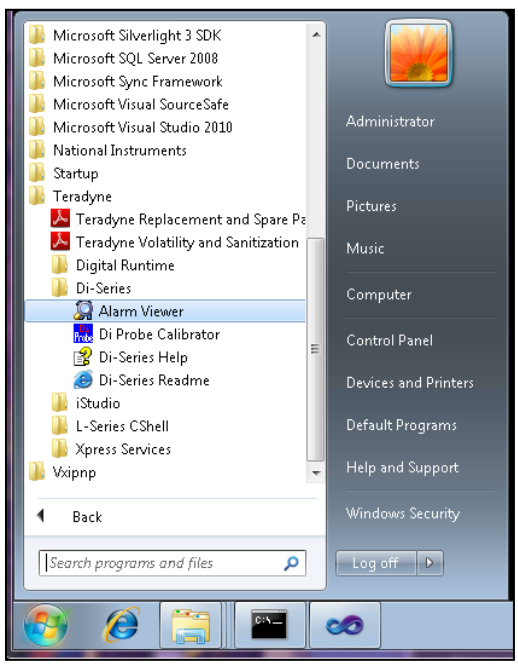
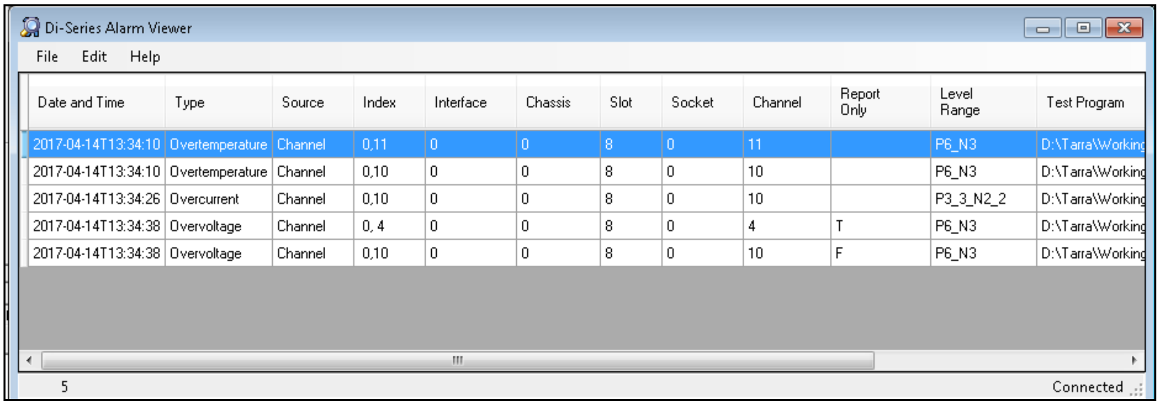


Figure 2 - Launching Di-Series Alarm Viewer from the Start menu

An OverVoltage alarm occurs when a Di-Series channel is subjected to an input voltage outside of the programmed voltage range, which could cause damage to the Di-Series instrument. When the event is detected the channel's relay is opened, which could cause unexpected TPS failures that require debugging. The Di-Series Alarm Viewer clearly identifies when an OverVoltage event is detected, which channel(s) experienced the event, and what programmed voltage range the channel(s) was set to when the event occurred. A user can make use of this information to correct their test program or test setup.

An OverCurrent alarm occurs when current overage is detected on a Di-Series channel. When the event is detected the channel driver is tristated, which could cause unexpected TPS failures that require debugging. The Di-Series Alarm Viewer can also clearly identify when an OverCurrent event is detected and which channel(s) experienced the event so corrective actions can be taken in the test program or test setup.

An OverTemperature alarm occurs when a Di-Series instrument is subjected to temperatures that could potentially damage hardware. When this event is detected, the Di-Series pin electronics are shut down, which could cause TPS failures that require debug.



Date and Time	Type	Source	Index	Interface	Chassis	Slot	Socket	Channel	Report Only	Level Range	Test Program
2017-04-14T13:34:10	Overtemperature	Channel	0,11	0	0	8	0	11		P6_N3	D:\Tarra\Working
2017-04-14T13:34:10	Overtemperature	Channel	0,10	0	0	8	0	10		P6_N3	D:\Tarra\Working
2017-04-14T13:34:26	Overcurrent	Channel	0,10	0	0	8	0	10		P3_3_N2_2	D:\Tarra\Working
2017-04-14T13:34:38	Overvoltage	Channel	0, 4	0	0	8	0	4	T	P6_N3	D:\Tarra\Working
2017-04-14T13:34:38	Overvoltage	Channel	0,10	0	0	8	0	10	F	P6_N3	D:\Tarra\Working

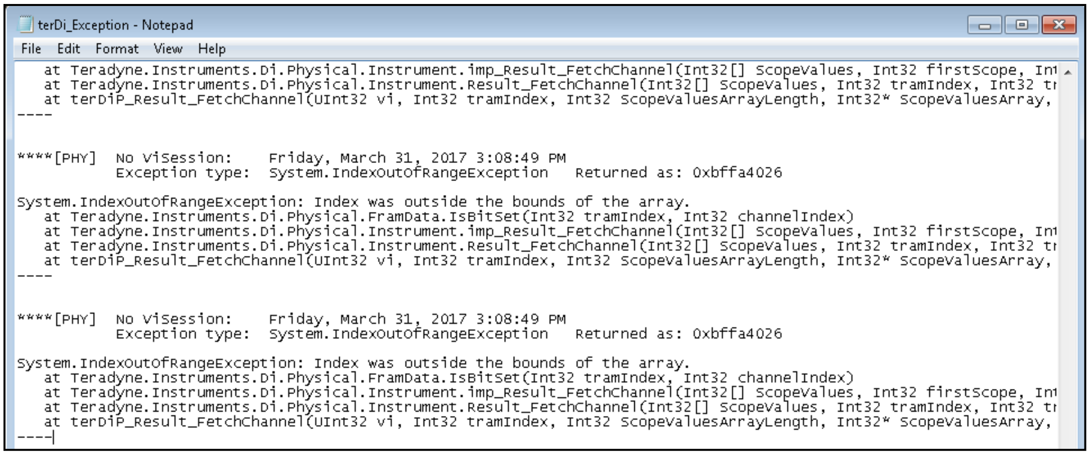
Figure 3 - Di-Series Alarm Viewer

This debug tool requires no user setup or maintenance and can be referenced whenever there is question around suspicious test results from the Di-Series instrument.

Log Files

Test program failures may occur because of errors thrown by the Di-Series driver that are not handled properly in a test program or do not directly translate to a legacy instrument error when in simulation. Log files created by the driver are automatically updated when these errors occur, and they can be referenced to debug unexplained test failures.

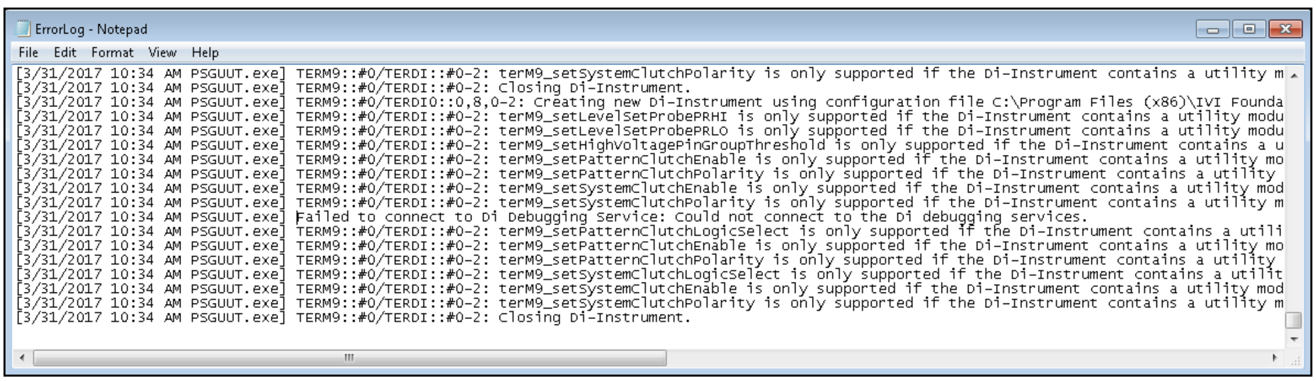
Di-Series exceptions are logged in the terDi_Exception.txt file located in the following location: "%LOCALAPPDATA%\Teradyne\DiDriver". This log file can be referenced during test program development if there is a chance exceptions are not being handled properly within a test program. Each exception is logged with the date and time of occurrence, exception type, and some additional information that can help identify at which Di-Series API the exception was thrown.



```
terDi_Exception - Notepad
File  Edit  Format  View  Help
   at Teradyne.Instruments.Di.Physical.Instrument.imp_Result_FetchChannel(Int32[] ScopeValues, Int32 firstScope, Int
   at Teradyne.Instruments.Di.Physical.Instrument.Result_FetchChannel(Int32[] ScopeValues, Int32 tramIndex, Int32 tr
   at terDiP_Result_FetchChannel(UInt32 vi, Int32 tramIndex, Int32 ScopeValuesArrayLength, Int32* ScopeValuesArray,
----


****[PHY]  No ViSession:     Friday, March 31, 2017 3:08:49 PM
           Exception type:   System.IndexOutOfRangeException   Returned as: 0xbffa4026

System.IndexOutOfRangeException: Index was outside the bounds of the array.
   at Teradyne.Instruments.Di.Physical.FramData.IsBitSet(Int32 tramIndex, Int32 channelIndex)
   at Teradyne.Instruments.Di.Physical.Instrument.imp_Result_FetchChannel(Int32[] ScopeValues, Int32 firstScope, Int
   at Teradyne.Instruments.Di.Physical.Instrument.Result_FetchChannel(Int32[] ScopeValues, Int32 tramIndex, Int32 tr
   at terDiP_Result_FetchChannel(UInt32 vi, Int32 tramIndex, Int32 ScopeValuesArrayLength, Int32* ScopeValuesArray,
----


****[PHY]  No ViSession:     Friday, March 31, 2017 3:08:49 PM
           Exception type:   System.IndexOutOfRangeException   Returned as: 0xbffa4026

System.IndexOutOfRangeException: Index was outside the bounds of the array.
   at Teradyne.Instruments.Di.Physical.FramData.IsBitSet(Int32 tramIndex, Int32 channelIndex)
   at Teradyne.Instruments.Di.Physical.Instrument.imp_Result_FetchChannel(Int32[] ScopeValues, Int32 firstScope, Int
   at Teradyne.Instruments.Di.Physical.Instrument.Result_FetchChannel(Int32[] ScopeValues, Int32 tramIndex, Int32 tr
   at terDiP_Result_FetchChannel(UInt32 vi, Int32 tramIndex, Int32 ScopeValuesArrayLength, Int32* ScopeValuesArray,
----
```

Figure 4 - Example Di-Series Exception Log

When emulating an M9-Series instrument with the Di-Series, a file is updated that logs when emulated instruments are created and closed, when unsupported function calls or features are used in the test program, and when errors occur. The default path for this M9-Series Interface log file is: C:\Documents and Settings\userName\Local Settings\Application Data\Teradyne\M9-Series Interface\ErrorLog.txt. This location can be changed to a preferred location through the M9-Series Interface Configuration Editor. The M9-Series Interface log file is especially useful if errors returned in an M9-Series emulated test program are not sufficient to debug the test program. Numerous Di-Series errors that cannot be mapped to an existing M9-Series error are reported as "TERM9_ERROR_INTERNAL", and this log file can provide additional information required to resolve the failure for these instances.



```
ErrorLog - Notepad
File  Edit  Format  View  Help
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: terM9_setSystemClutchPolarity is only supported if the Di-Instrument contains a utility m
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: Closing Di-Instrument.
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI0::0,8,0-2: Creating new Di-Instrument using configuration file C:\Program Files (x86)\IVI Founda
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: terM9_setLevelSetProbePRHI is only supported if the Di-Instrument contains a utility modu
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: terM9_setLevelSetProbePRLO is only supported if the Di-Instrument contains a utility modu
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: terM9_setHighVoltagePinGroupThreshold is only supported if the Di-Instrument contains a u
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: terM9_setPatternClutchEnable is only supported if the Di-Instrument contains a utility mo
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: terM9_setPatternClutchPolarity is only supported if the Di-Instrument contains a utility
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: terM9_setSystemClutchEnable is only supported if the Di-Instrument contains a utility mod
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: terM9_setSystemClutchPolarity is only supported if the Di-Instrument contains a utility m
[3/31/2017 10:34 AM PSGUUT.exe] Failed to connect to Di Debugging Service: Could not connect to the Di debugging services.
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: terM9_setPatternClutchLogicSelect is only supported if the Di-Instrument contains a utili
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: terM9_setPatternClutchEnable is only supported if the Di-Instrument contains a utility mo
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: terM9_setPatternClutchPolarity is only supported if the Di-Instrument contains a utility
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: terM9_setSystemClutchLogicSelect is only supported if the Di-Instrument contains a utilit
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: terM9_setSystemClutchEnable is only supported if the Di-Instrument contains a utility mod
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: terM9_setSystemClutchPolarity is only supported if the Di-Instrument contains a utility m
[3/31/2017 10:34 AM PSGUUT.exe] TERM9::#0/TERDI::#0-2: Closing Di-Instrument.
```

Figure 5 - Example M9-Series Interface log file

These log files are automatically generated and updated, and like the Alarm Viewer require no user setup or maintenance. These tools are a quick way to begin debugging unexpected test failures with little effort.

Digital Test Editor Debugging Features

Various Di-Series debugging tools are available through the Digital Test Editor, which can be useful whether you are executing test programs though the M9-Series Interface or natively with Di-Series API's. When a test program fails, the Digital Test Editor may be used to clearly identify which channels are failing, if there are unexpected channel setup issues, and to easily loop a Di-Series test.

Debug Capture of a Test Program

The Digital Test Editor has a Debug feature that can capture a test program as it is executed on Di-Series hardware into the graphical interface. This is especially useful when debugging complicated test programs that do not provide sufficient Di-Series debug information or to graphically display failure patterns that are not obvious when reported in a test program. To utilize this feature of the Digital Test Editor create a new Digital Test from the iStudio program, and select "Debug -> Enable" from the top menu bar.

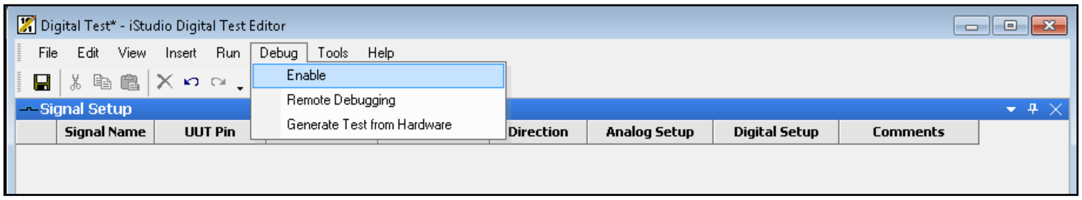


Figure 6 - Enable Debug Capture in the Digital Test Editor

Note that the Xpress Databases do not need to be manually populated before using the Debug Capture capability of the Digital Test Editor. The configuration and pinmap information for the test will be captured by the tool along with set up information and patterns.

An example Debug Capture of a test program executed through the M9-Series Interface can be seen below. This test program failed, but was difficult to debug in the original test environment due to the complexity of the test executive and limited failure reporting. After a debug capture through the Digital Test Editor, it is clear that all failures occurred on a single channel. This could point to fixture or cabling issues since all channel settings are expected.

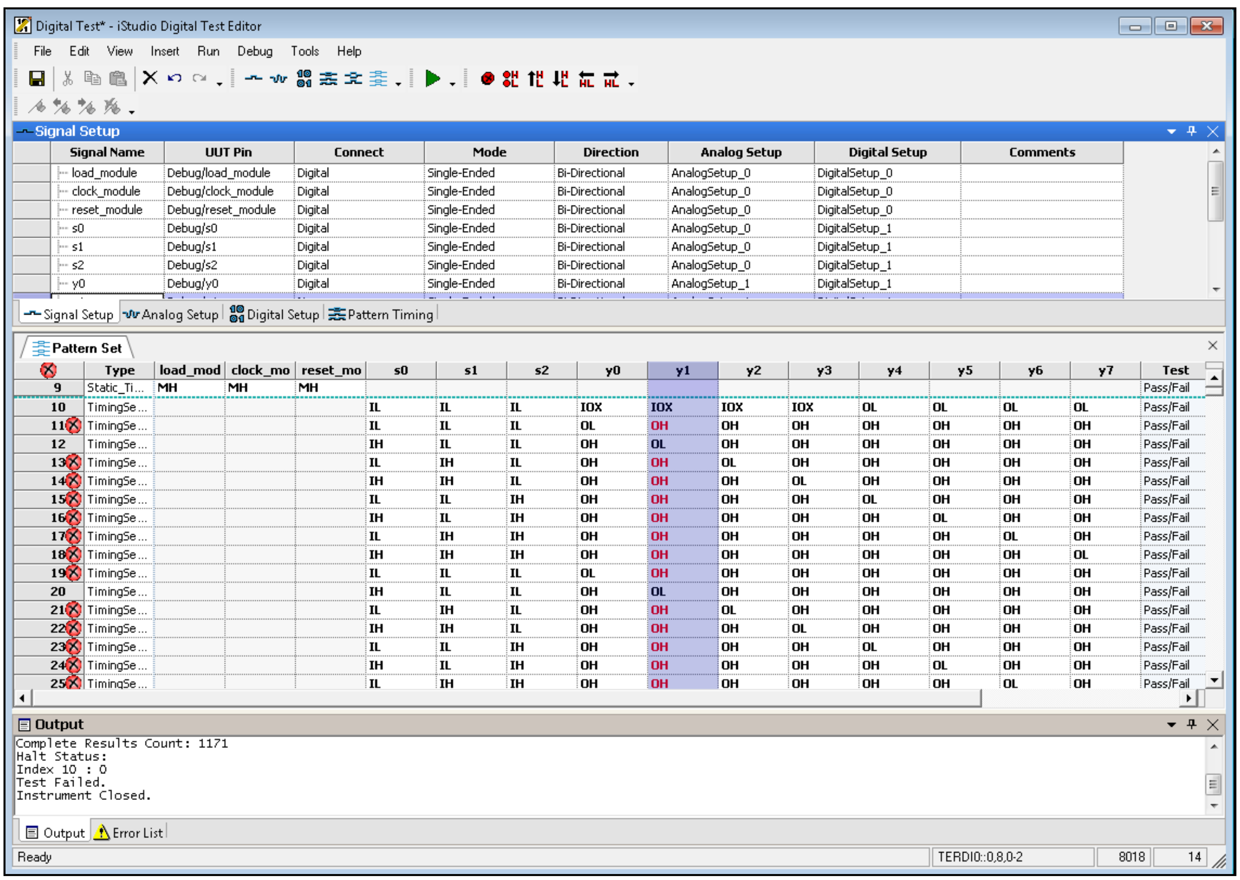


Figure 7 - Debug Captured Digital Test

Debug captured tests and Xpress Databases can be saved and used for debug of the captured test program at a later time.

Generating Test from Hardware

The Digital Test Editor also has the ability to generate a test from the current state of the Di-Series hardware. This is useful to capture instrument and channel settings at a specific point in a test program, particularly if your test program does not easily allow insight to this information or to view settings that were not available on legacy instruments that are being emulated. This should be done immediately after the static pattern or digital pattern set in question, before any instrument settings are reset. To generate a test from Di-Series hardware from within the Digital Test Editor, create a new Digital Test from an iStudio project, and select "Debug -> Generate Test from Hardware" from the top menu bar.

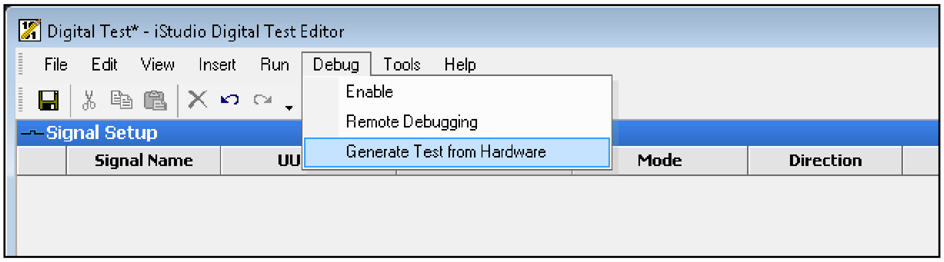


Figure 8 - Generate Test from Hardware from within the Digital Test Editor

The Generate Test from Hardware dialog box requires some inputs to get the desired information from the Di-Series hardware. First, the resource string of the target virtual instrument must be provided for the empty project. If an invalid string is provided, an error will be reported in the Output window. The user can select whether to only generate a Static Pattern or Dynamic Pattern Set, or both. If Dynamic Patterns Set is selected, there are options of gathering all patterns up to the PATC Halt, or to only generate a specific pattern range to isolate a failure. There is also the option to capture the test results from the hardware.

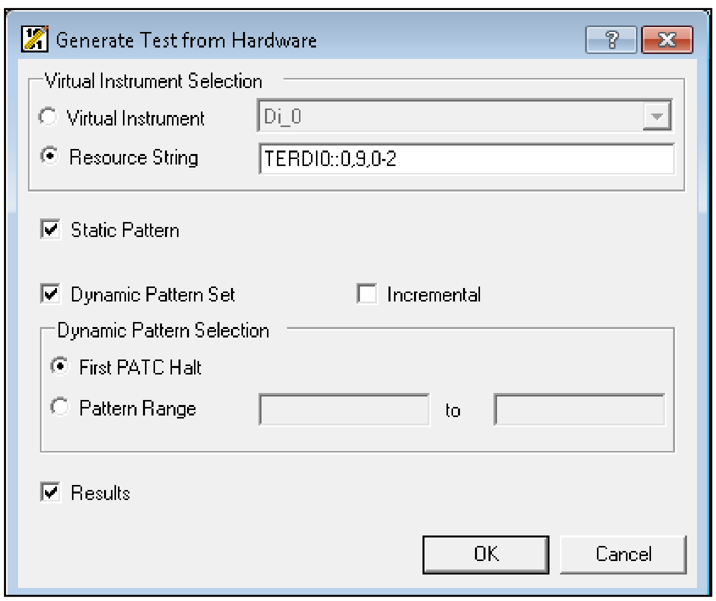


Figure 9 - Generate Test from Hardware Dialog Box

An example of a test generated from Di-Series hardware is shown below. The test program failed on multiple channels, and with the use of the Digital Test Editor it is clear the relays of the failing channels are open (Connect = None) when we expected the digital relays to be closed. This type of failure is not as easy to debug programmatically.

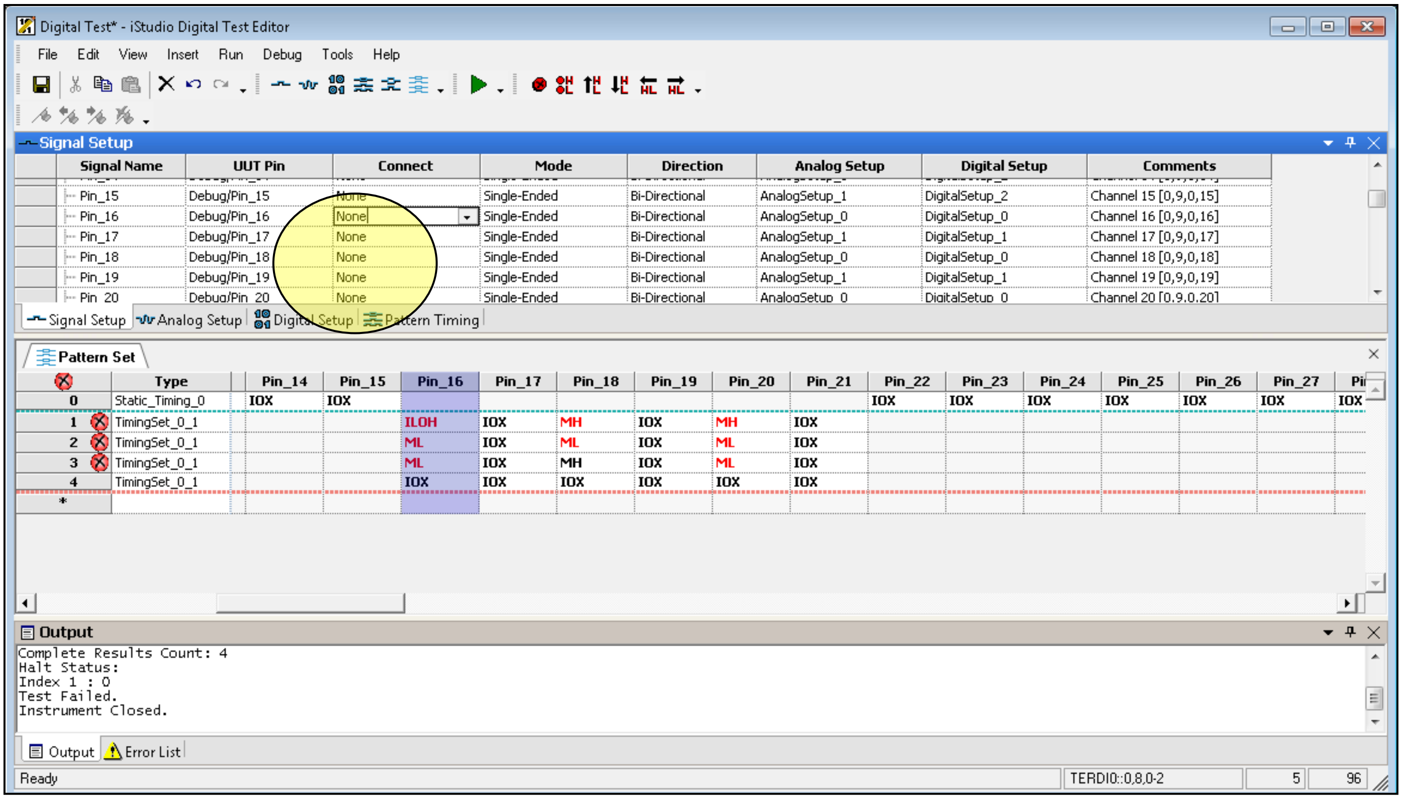


Figure 10 - Example test generated from hardware

With this new information, it can be investigated whether the relay state was programmed incorrectly, another program is opening the relay during the test execution, or an Alarm event caused the relay to open.

Loop and Sweep

The Loop and Sweep features of the Digital Test Editor can be utilized for debugging tests originally created using the Digital Test Editor, or those that were Debug Captured or generated from hardware. Looping and Sweeping can be used to pinpoint causes for unexpected failures without the potential overhead of the original test executive.

The Digital Test Editor provides the ability to indefinitely loop a test program and to stop on a failure, without the use of a separate test executive such as TestStudio. Looping tests can help to reproduce an intermittent failure, which may be more difficult or time consuming if the original test program has significant overhead. To use this feature from within a digital test, select "Tools -> Loop" from the top menu bar.

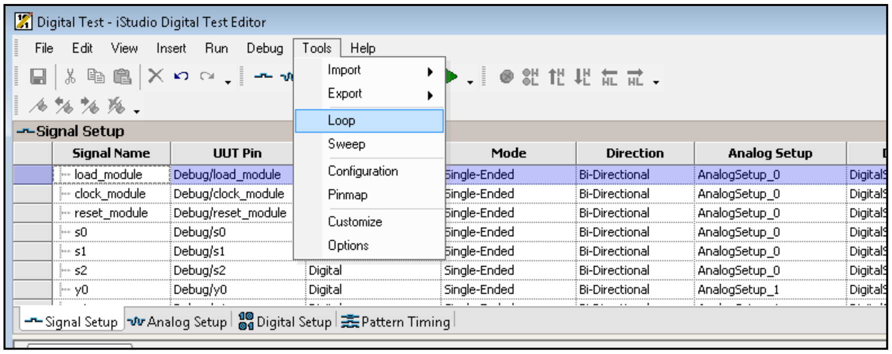


Figure 11 - Launching the Loop Dialog box from the Digital Test Editor

The Loop Dialog box allows the user to begin looping the test by selecting the Loop button and to stop at any time by selecting the Stop button. A tally of how many times the test passed and failed is displayed, as well as the number of total loops (including the one in progress), Not Tested results, and Error results. The option also exists to stop on a failure, allowing the ability to take any physical measurements on the test setup at the time of a failure. Select Cancel to exit the Loop dialog box.

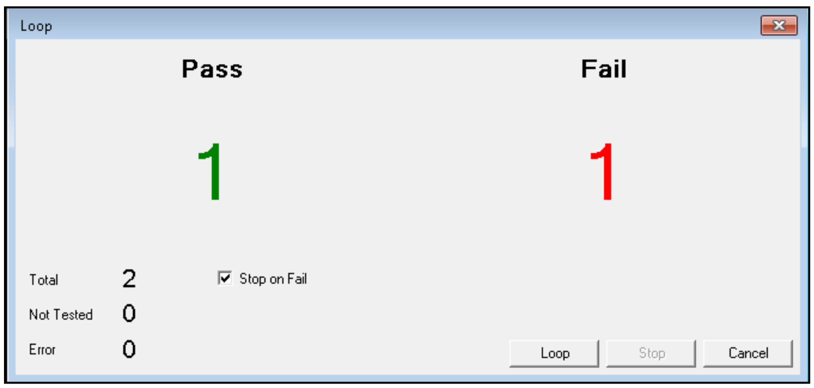


Figure 12 - Loop Dialog box stopped on failure

The Digital Test Editor also provides the ability to execute the digital test by "sweeping" over a range of either threshold voltage values or timing set values for one specific channel. This allows the user to debug at what particular voltage threshold or time applied to a channel a test will begin to pass or fail, without repeatedly modifying and rebuilding a test program. Note that in order for the "Sweep" test results to provide valuable information, the test must first be passing or have only the one channel of concern reporting a failure. This is to ensure that the modified threshold voltage or timing values are causing the new pass or fail result. This debug tool may be required if physical access to pins to measure voltage or timing of a signal is not possible. To use this feature from within the Digital Test Editor, select "Tools -> Sweep" from the top menu bar.

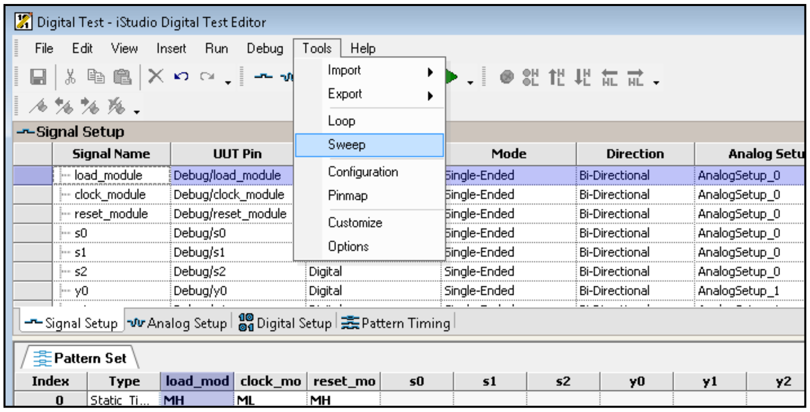


Figure 13 - Launching the Sweep Dialog box from the Digital Test Editor

From the Sweep Dialog box, the user must select from a drop-down menu which signal from the Digital Test pinmap to perform the sweep function, and whether to perform a sweep of a threshold voltage or timing values for one of the timing sets available for the selected signal. When Voltage is selected, a varying binary threshold is set for the chosen signal. When Time is selected, the capture mode is set to Strobe, and the strobe is swept over the specified values. The example below has chosen to sweep across threshold voltages of 200mV up to 4V in 100mV increments for a specified signal.

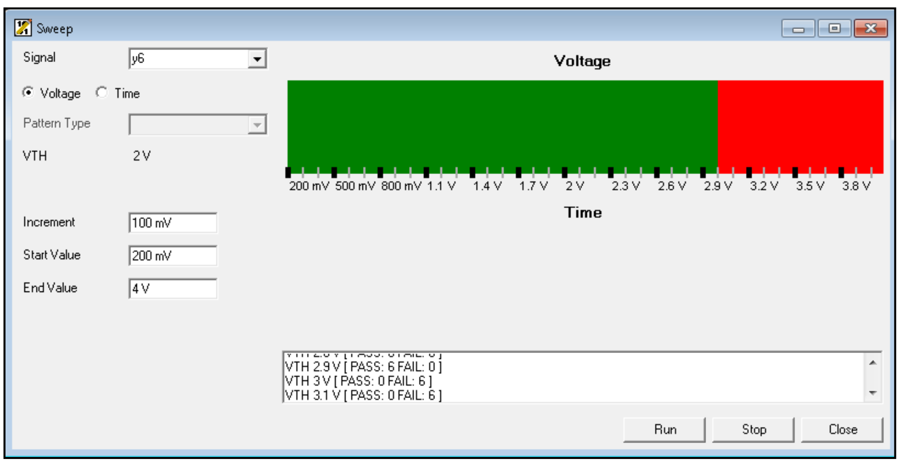


Figure 14 - Example Sweep Dialog box and results

The results show that when the threshold is set to less than 3.0V, the test passes. Therefore, we can conclude that the input to the signal is around 3.0V when it is testing for a signal high. Now, the user must determine if that is the expected input to the channel, and, if it is not, what could be causing the incorrect input. If the channel of concern were not accessible due to hardware cabling or fixture design, this tool is an effective way to verify what the Di-Series channel is sensing without modifying and recompiling a test program.

Conclusions

The standard Di-Series software package includes numerous tools to aid in debugging unexpected test program results that involve the Di-Series instrument, whether executing native Di-Series tests or using the instrument in emulation for legacy programs. Some of these tools, including the Di-Series Alarm Viewer and the automatically generated log files, require no user setup or maintenance and are a good place to quickly start the debugging process. Others require some user setup, but they can save significant time when compared to debugging solely from an existing test executive.