SAND2017-5287C

Thank you for attending NIWeek 2017.


NATIONAL
INSTRUMENTS™

A software-centric platform that accelerates the development and increases the productivity of test, measurement, and control systems.

Tim Vargo

Exceptional service in the national interest

Instrumentation Engineer

tdvargo@sandia.gov

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Ravi Beniwal

Binary Palette

Crafting Software & Systems

Owner

ravi@binarypalette.com

Tried & True, Conventional Debugging Tools

… which still work very well, but with limitations

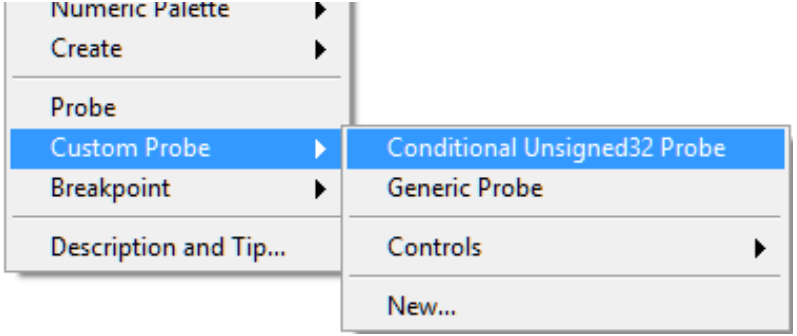
Built-in Debug Tool	Pros	Cons
Execution Highlighting 	Visualizes data flow	Slows execution speed (impedes discovery of timing problems)
Single Stepping 	One operation at a time	Slows execution speed (impedes discovery of timing problems)
Breakpoints 	Pause execution at this exact point (useful for beginning execution highlighting and single stepping)	Persistent (must remember to remove them before deployment)

NATIONAL INSTRUMENTS

Probes !!!

- Generic (default) Probes
  - Retain Wire Values

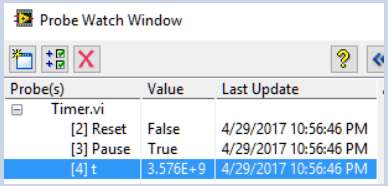
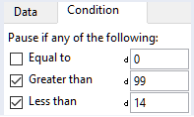
- **Custom Probes**
  - Custom Probes ▶ Conditional (native)
  - Custom Probes ▶ Controls (look Mom, no code!)
  - Custom Probes ▶ New… (roll your own)
  - An awesome suite of custom probes
    - by Saphir, 3rd party, Free, ViBox – Probes
  - Variant Probe
    - Courtesy of LAVA member Ton Plomp



NATIONAL INSTRUMENTS

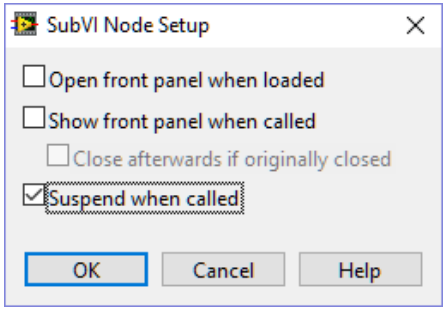
Tried & True, Conventional Debugging Tools

… which still work very well, but with limitations

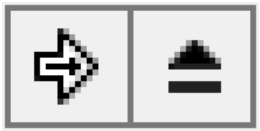
Built-in Debug Tool	Pros	Cons
Default Probes 	Peek at wire values throughout execution	**Difficult to place in clones**, Non-persistent
Conditional Probes 	+ “pause if value” logic	Not available for all datatypes
Custom Probes 	+ custom view and/or actions	Per LV version

NATIONAL INSTRUMENTS

Often forgotten, overlooked, or just plain ol' "huh, I-never-knew-about-that"



- Suspend when called
  - Access this from "SubVI Node Setup"
  - Suspends a subVI when called and waits for user interaction
    - Allows changes to input values, then re-execute
    - Allows changing the output, then return to caller



  - A poor-man's (yet extremely effective) unit testing
  - Video Demo: https://youtu.be/AgXcyf2Ta_A

There are many Free, 3rd-party Debugging Tools
These two will retain information even if LabVIEW crashes!

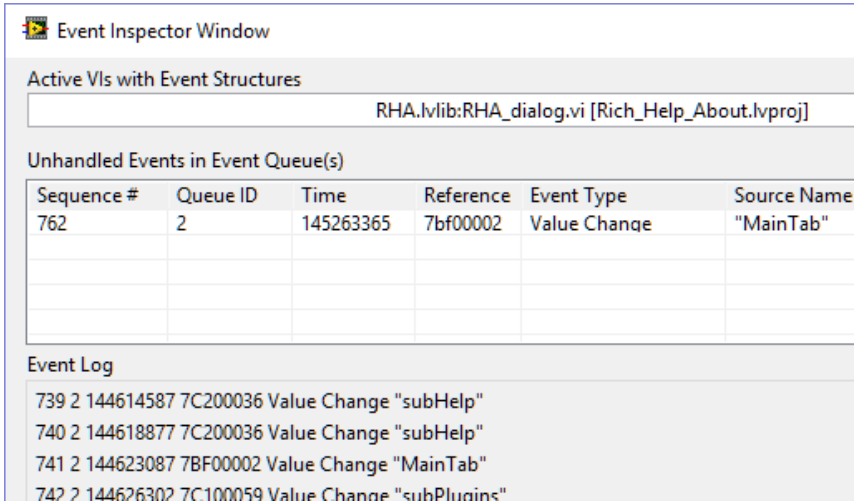
- WinDebugLogging 
  - Write debug strings to the Windows debug interface
  - Dr. Damien (DFGray), Nov-2008
  - DebugView (from Sysinternals [Microsoft])

- WinDebugLogProbe 
  - Tim Vargo

More NI debugging tools

Built-in to LabVIEW

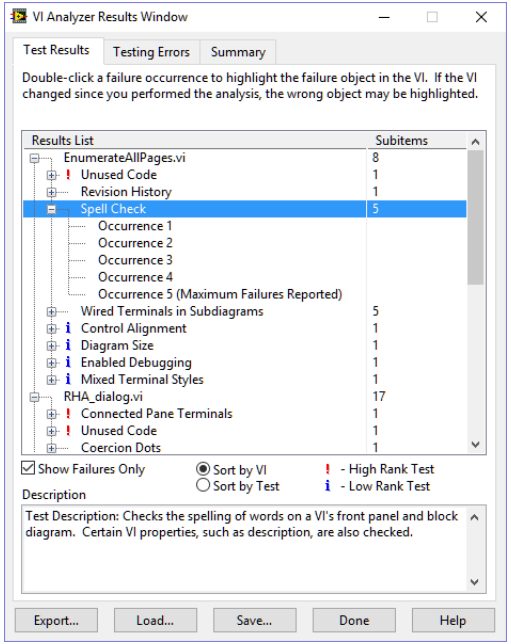
- Event Inspector
  - Introduced with LabVIEW 2013, the Event Inspector added an ability to gain new understandings into the inner workings of our event structures



More NI debugging tools

Toolkits for extra cost (but worth it, IMHO)

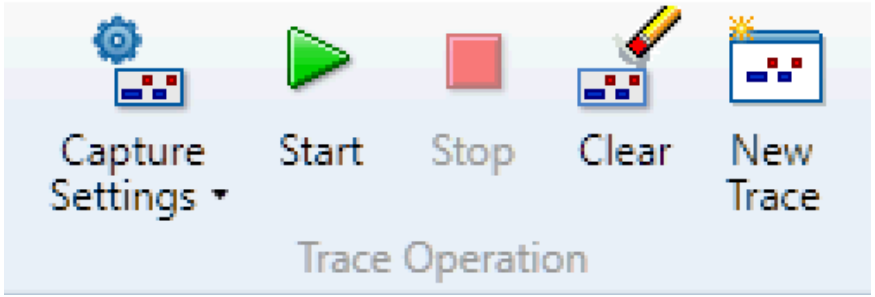
- VI Analyzer
  - VI Analyzer does its job well, but it intends to perform a static analysis of VIs, not a dynamic analysis of code while running



More NI debugging tools

Toolkits for extra cost (but worth it, IMHO)

- Desktop Execution Trace Toolkit (DETT)
  - The DETT product is very good at what it was designed to do
    - dynamic execution traces provide a fantastic wealth of evidence when event timing and/or sequence information is critical
    - … but they don't provide non-event related information



Some impediments to advanced troubleshooting …

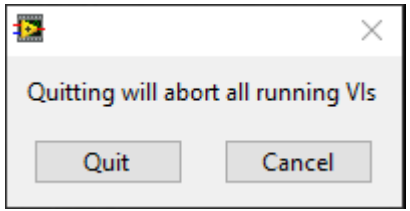
- All of these built-in tools are extremely useful during troubleshooting of individual VIs or small collections of VIs
  - particularly when we already know which VIs are the troublesome ones
  - but the LabVIEW IDE still lacks a debugging tool to provide insights with a "bigger picture" view of an entire project

Some impediments to advanced troubleshooting …

- Using only the native troubleshooting tools
  - reentrant VIs are especially difficult to debug, since each preallocated clone will have its own data space in memory
  - shared clones can be even more complex, since they share data space
- Dynamically launched VIs, even if not reentrant, present their own particular frustrations during a debug session, since the developer is unable to place probes **BEFORE** executing the code.

As LabVIEW enthusiasts, we all love visuals

… but **THESE** visuals make our faces cringe, and our sphincters tighten



WHAT?! There are still VIs running?!

What to do when debugging becomes too complex ???

- How to defeat these shortcomings, and many other debugging related challenges …
  - have long been the topics of many user community discussions
  - several ad-hoc solutions and processes, although usually quite limited in scope, have been developed over the years by the user community
  - but they lack the coherence of a unified tool

- **Introducing the**

LabVIEW
Task Manager

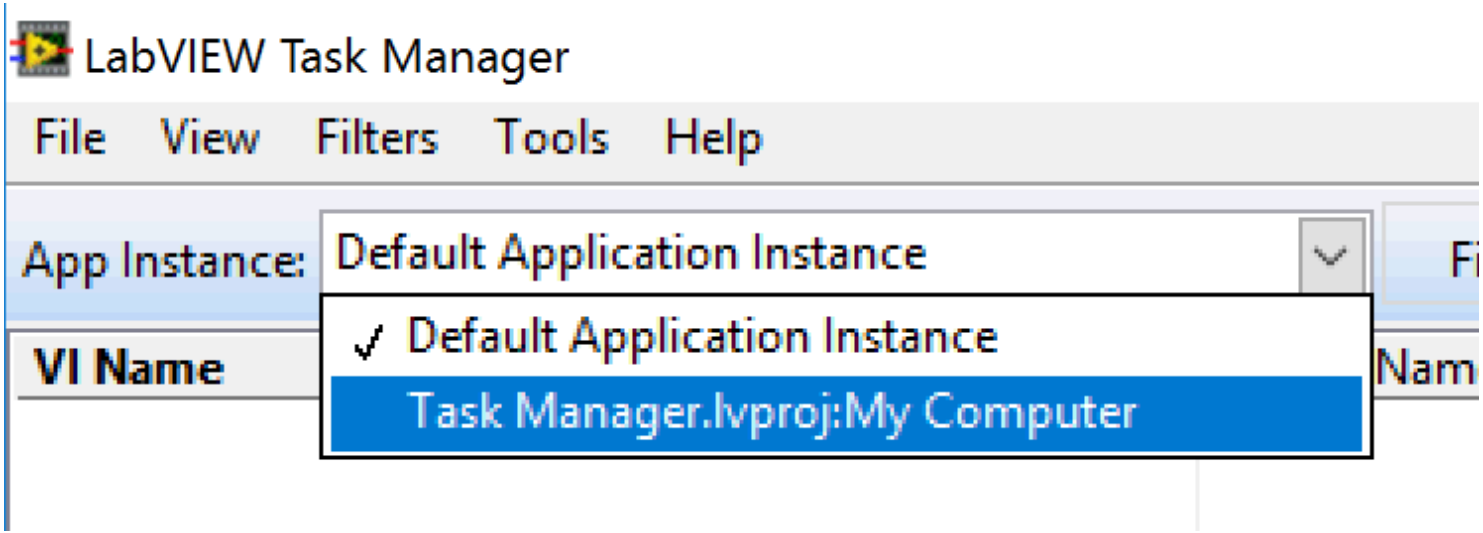
NATIONAL INSTRUMENTS

LabVIEW Task Manager

The *LabVIEW Task Manager* seeks to be that missing unified debugging tool

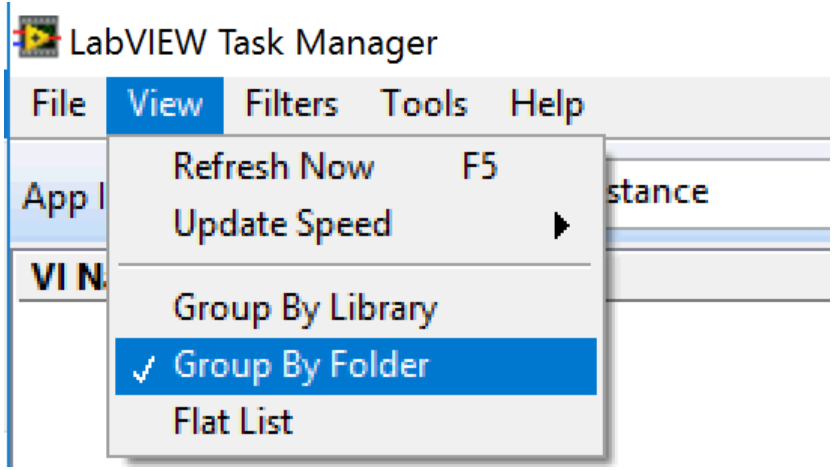
- Providing a dynamic & big-picture view of all VIs currently in memory
  - by delivering new comprehensions into your running code
- Conquering those difficulties concerning reentrancy, clones, dynamic launching, finding & aborting hung VIs, and other sticky complications
  - by enabling interaction with individual or groups of VIs in many various ways, providing significant benefits while troubleshooting

Application Instance



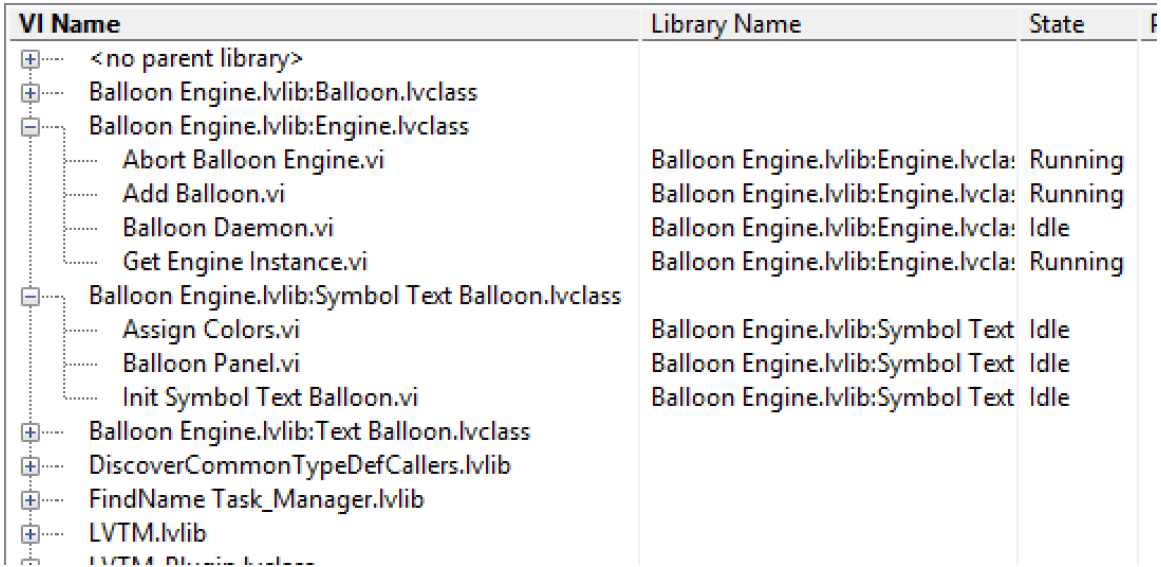
The *LabVIEW Task Manager (LVTM)* detects all open LabVIEW application instances, but discovers and displays VIs for only the selected instance

Menu: View



The View menu provides options to set an update speed for refreshing the view and a few view options

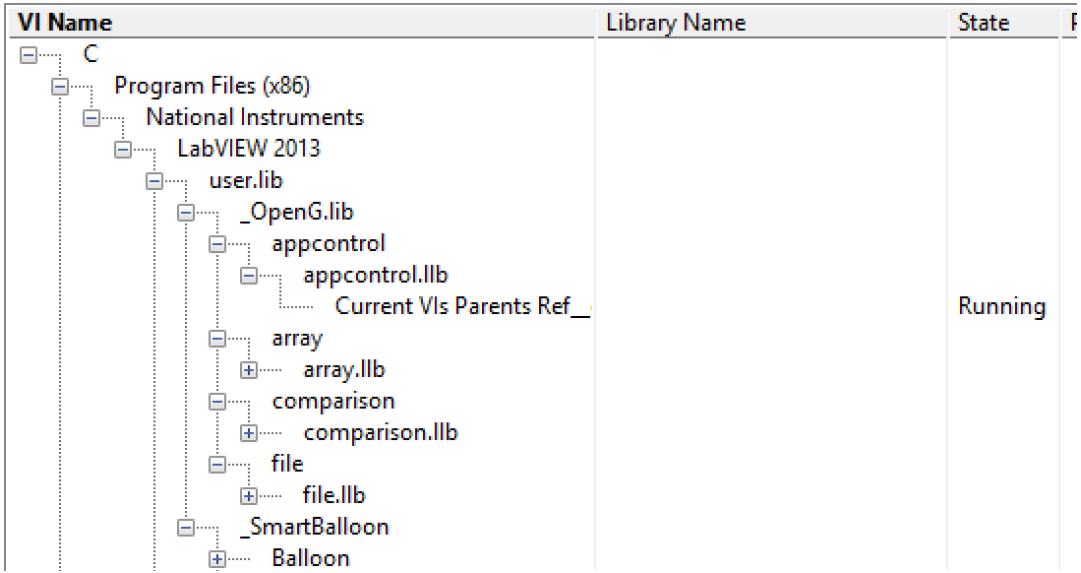
Menu: View>>Group By Library



Group By Library displays all VIs in the selected application instance grouped by the library they belong to. All VIs that are not part of any library are shown under <no parent library>.

NATIONAL INSTRUMENTS

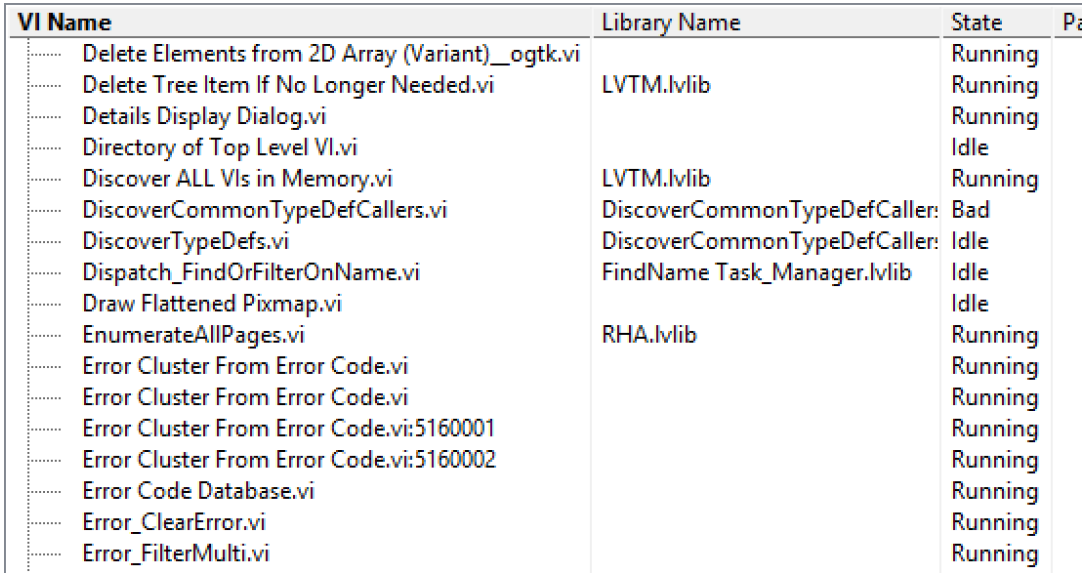
Menu: View>>Group By Folder



Group By Folder displays all VIs in the selected application instance with their folder hierarchy on disk. This view makes it easy to see where all the project dependencies live on disk.

NATIONAL INSTRUMENTS

Menu: View>>Flat List



VI Name	Library Name	State	Pa
Delete Elements from 2D Array (Variant)__ogtk.vi		Running	
Delete Tree Item If No Longer Needed.vi	LVTM.lvlib	Running	
Details Display Dialog.vi		Running	
Directory of Top Level VI.vi		Idle	
Discover ALL VIs in Memory.vi	LVTM.lvlib	Running	
DiscoverCommonTypeDefCallers.vi	DiscoverCommonTypeDefCaller:	Bad	
DiscoverTypeDefs.vi	DiscoverCommonTypeDefCaller:	Idle	
Dispatch_FindOrFilterOnName.vi	FindName Task_Manager.lvlib	Idle	
Draw Flattened Pixmap.vi		Idle	
EnumerateAllPages.vi	RHA.lvlib	Running	
Error Cluster From Error Code.vi		Running	
Error Cluster From Error Code.vi		Running	
Error Cluster From Error Code.vi:5160001		Running	
Error Cluster From Error Code.vi:5160002		Running	
Error Code Database.vi		Running	
Error_ClearError.vi		Running	
Error_FilterMulti.vi		Running	

Flat List displays all VIs in the selected application instance as a list. This view can help sorting the view by different properties of the VIs, by clicking on the column headers.

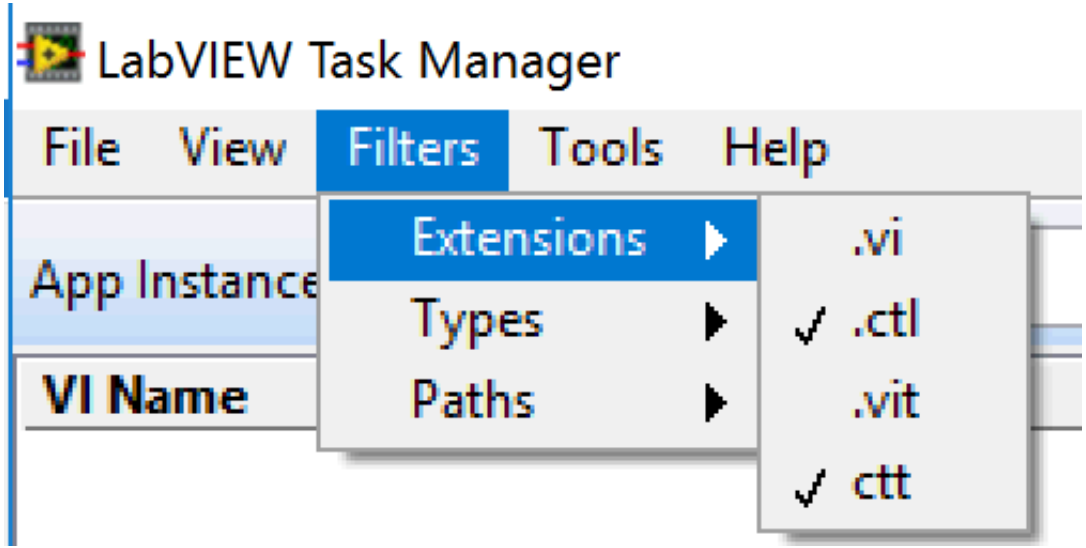
NATIONAL INSTRUMENTS

Menu: Filters

The Filters menu items provide several settings for filtering out items from view

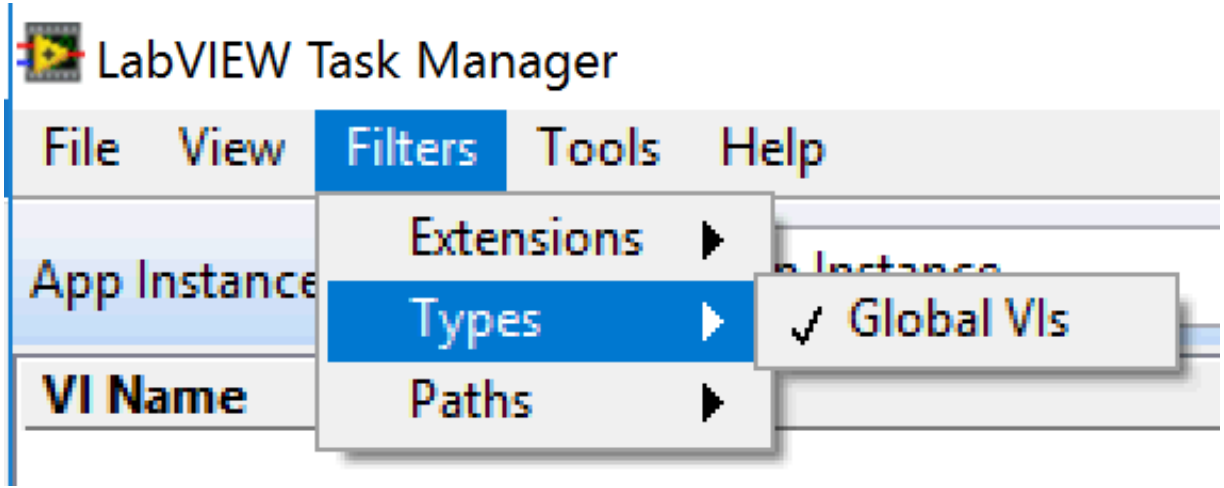
- Filtering out VIs that you don't currently care about can tremendously improve performance of the LVTM
  - Certain dynamic properties of all VIs are occasionally checked and updated. The more VIs that need to be checked, the more resources are tied up, and the longer each update cycle takes.
- Also helps the user to de-clutter the view

Menu: Filters>>Extensions



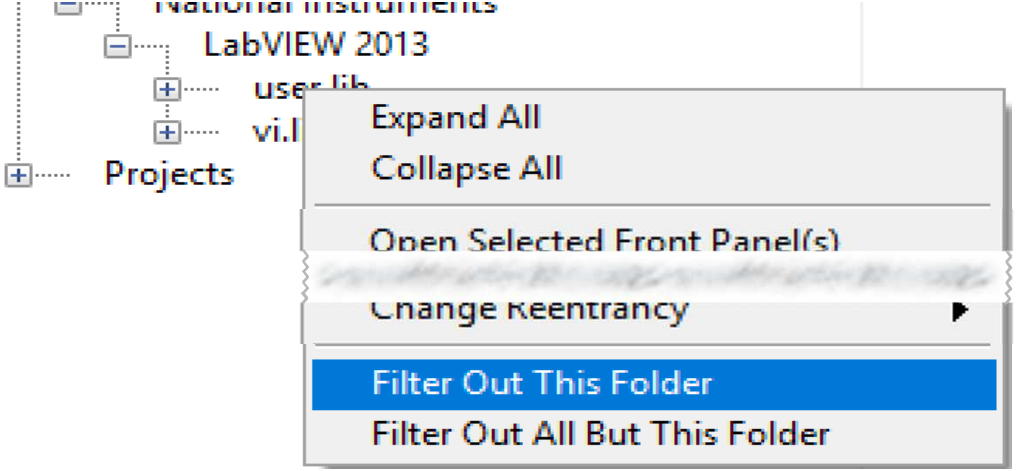
Select (√) the file extensions that you want to filter out of the view. All files with the selected extensions will be ignored.

Menu: Filters>>Types



Select (√) the VI types that you want to filter out of the view. All VIs of the selected type will be ignored.

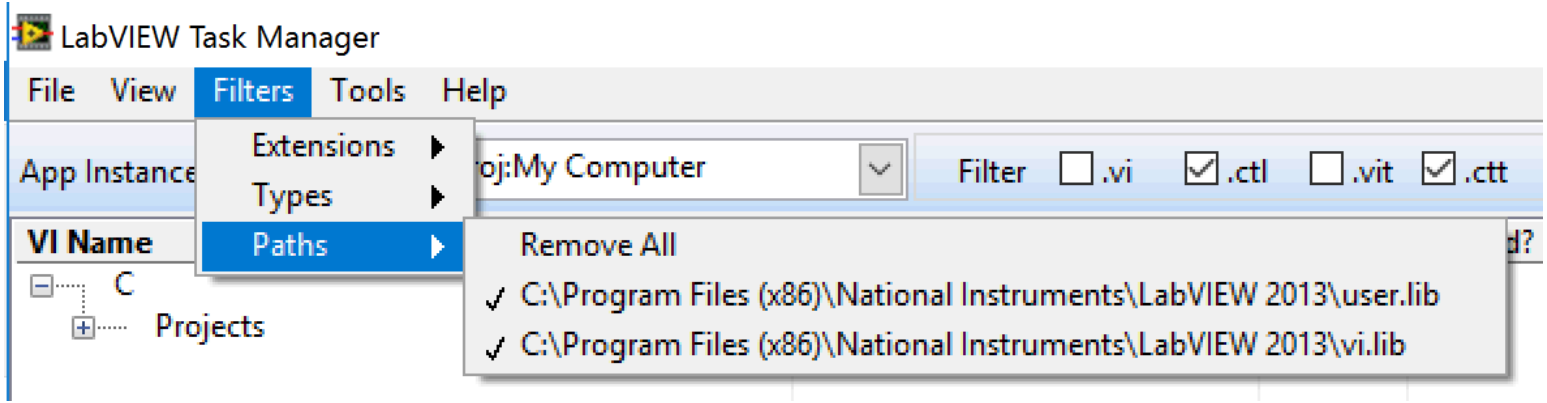
Shortcut Menu: Filter Out …



Although filtered paths can only be added or removed while in the *Group By Folder* view, the hidden paths will also persist in other views. This helps the user to de-clutter the view, and can also greatly improve performance.

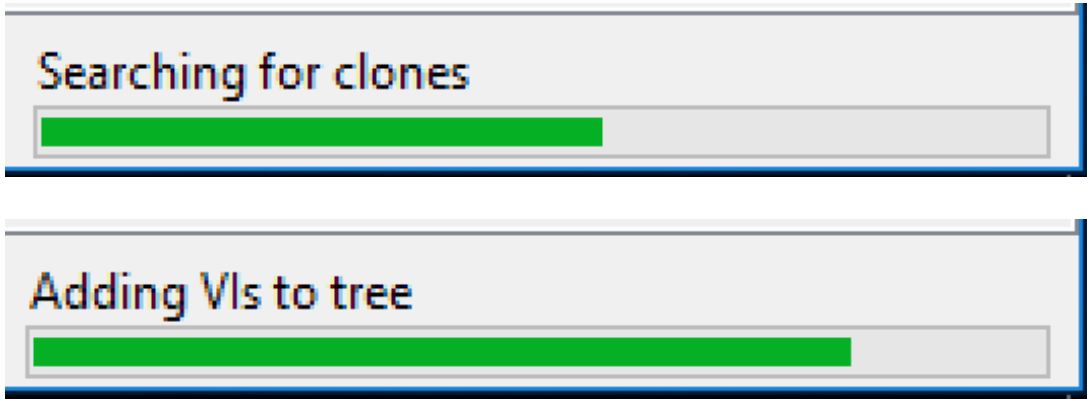
- While in the *Group By Folder* view, right click any folder and select *Filter Out This Folder*, adding that folder to an exclusion list
- Or you can select *Filter Out All But This Folder*, which will add all remaining folders to the exclusion list
- Any VIs in these filtered out folders will no longer show up in the tree

Menu: Filters>>Paths



- If one or more paths have been filtered out, they will show up under the Filters>>Paths menu. Clicking any of these paths will remove it from the exclusion list.
- Selecting *Remove All* will remove all filtered out paths, effectively disabling path based filtering.

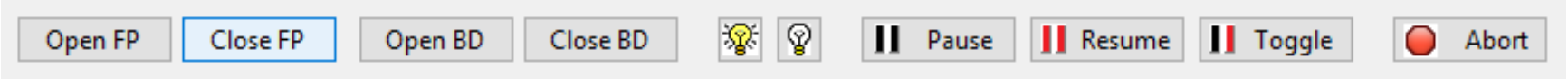
Progress Update



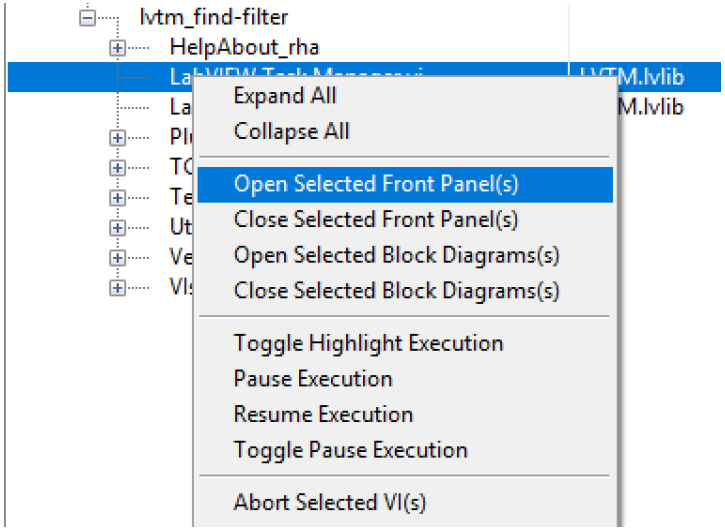
- Any time changes are made to the filters or to the view, the view is refreshed
- A progress bar, in the status bar at bottom right, shows the discovery and update progress

Toolbar

LVTM provides several operations that can be performed on one or more selected VIs in the tree. These operations may be invoked from the toolbar buttons at the bottom of the screen.



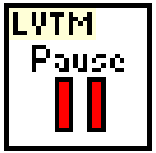
These same operations may also be invoked from the right click shortcut menu.



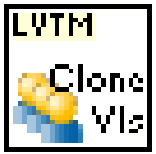
- Open Front Panel(s)
- Close Front Panel(s)
- Open Block Diagram(s)
- Close Block Diagram(s)
- Highlight Execution(s)
- Do Not Highlight Execution(s)
- Pause Execution(s)
- Resume Execution(s)
- Toggle Execution(s)
- Abort VI(s)

Helper Functions

There are helper functions available to assist with your debugging efforts, and these are available from the Functions Palette



LVTM Pause – Drop this function into any VI you wish to pause, on the condition that you are currently troubleshooting; as it will only pause if the *LabVIEW Task Manager* is also running (so this pause will NOT occur if you are not troubleshooting).



Clone Beacon – Drop this VI into any asynchronously called reentrant VI, to force it to be seen by *LabVIEW Task Manager*.

And now for the

LIVE DEMO

!!!!!!!!!!!!!!!!!!!!!!!

Demo Agenda

- …\examples\Industry Applications\Temperature Monitoring\Temperature Monitoring.lvproj
  - Demonstrate what the LVTM shows when a couple of (async?) processes didn't properly shut down after the main application shut down. Unfortunately, these process refuse to abort, so continue looking for a different example that would.
- ???
  - Demonstrate remotely closing a modal front panel, which has caused all LabVIEW application instances (including LVTM) to hang.
- ???
  - Demonstrate
- ???
  - Demonstrate
- ???
  - Demonstrate
- ???
  - Demonstrate

How to GET the tool …

https://lavag.org/files/file/245-labview-task-manager

How to CONTRIBUTE to the tool …

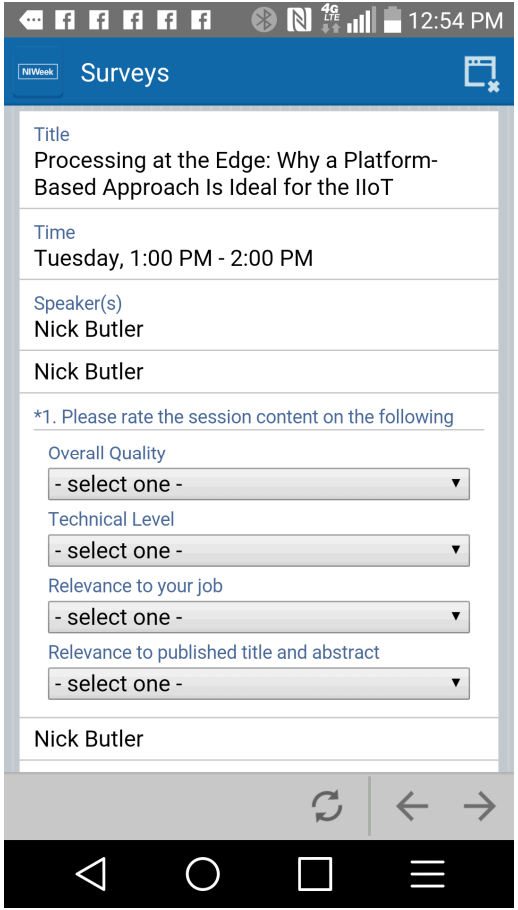
https://bitbucket.org/lavag/labview-task-manager

QUESTIONS?

NATIONAL INSTRUMENTS

We must avoid “drawing vast conclusions from half-vast data”

— Jerry R. Ehman, SETI Astronomer



Before you go, take the survey.

NATIONAL INSTRUMENTS

LabVIEW Task Manager

Stay Connected During and After NIWeek

ni.com/niweekcommunity

facebook.com/NationalInstruments

twitter.com/niglobal

youtube.com/nationalinstruments

Please provide feedback on this session via the NIWeek Mobile App