LabVIEW Developer Days

Build Code. Form Communities. Gain Confidence.

NATIONAL INSTRUMENTS™

Improving Code Quality Through Automated Code Analysis

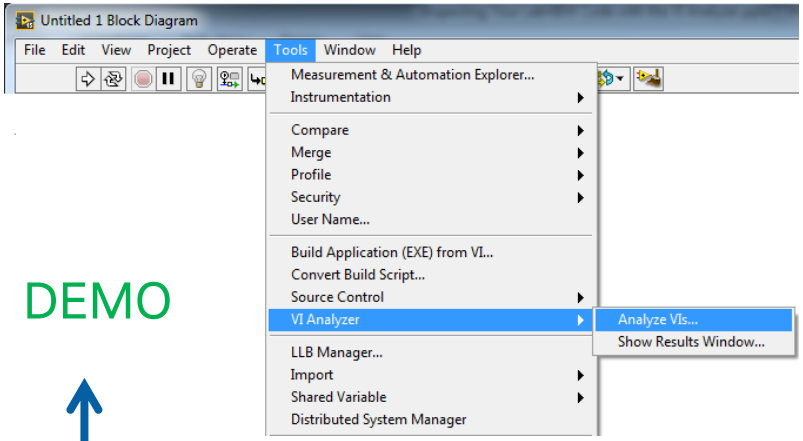
Before we get started…

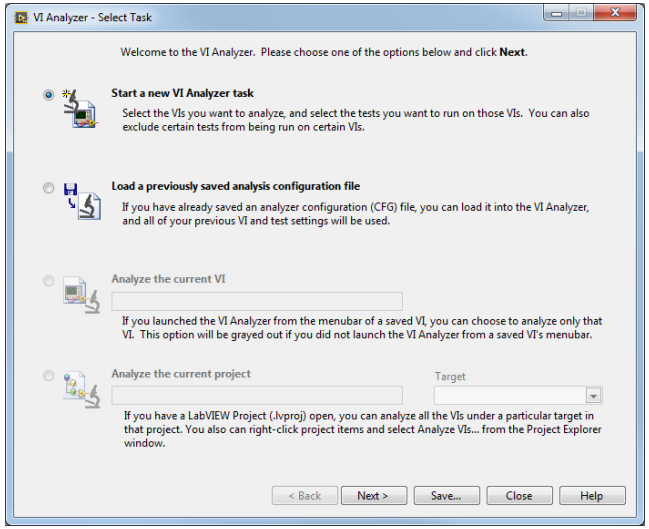
Download a copy of this presentation and all the demo files here:

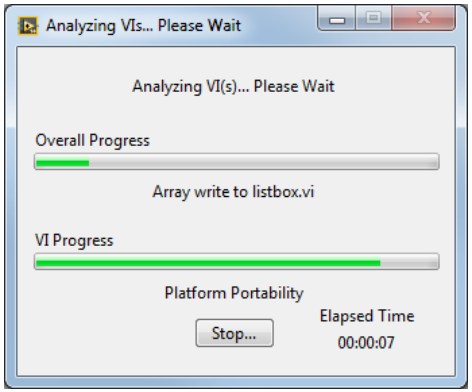
bit.ly/dnatt2015vianalyzer

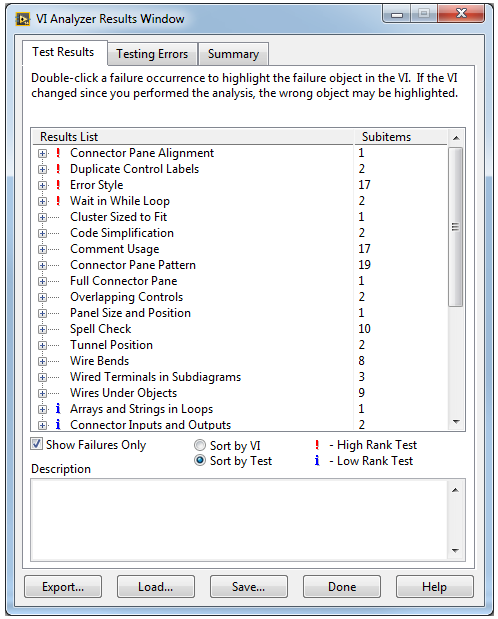
What is the VI Analyzer?

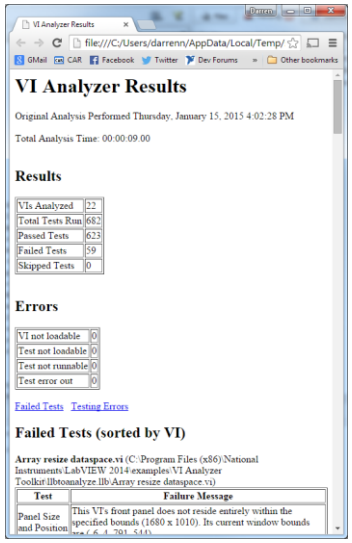
DEMO











NATIONAL INSTRUMENTS

What is the VI Analyzer?

A tool for performing **static code analysis** on one or more VIs.

- **Static code analysis**

  inspecting *non-running* LabVIEW code

  - **VI Analyzer**
  - Find
  - Show Buffer Allocations
  - VI Metrics

- **Run-time code analysis**

  analyzing performance and memory issues with *running* LabVIEW code

  - Profile Performance and Memory
  - Desktop Execution Trace Toolkit

Why should I use the VI Analyzer?

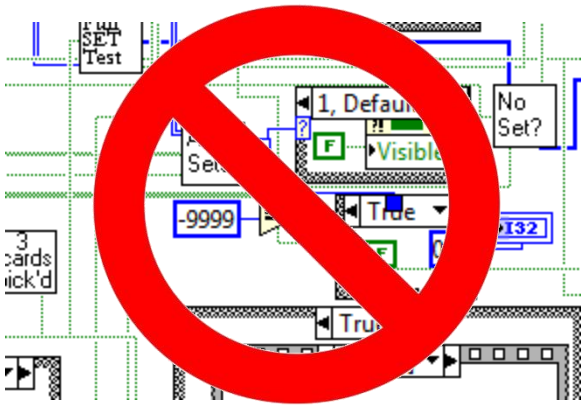
- The VI Analyzer improves the quality of your code by analyzing block diagrams and front panels to see if coding *best practices* are followed.

- Use the VI Analyzer **interactively** to detect issues during your daily programming work.

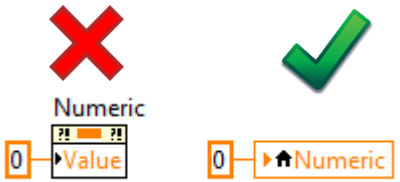
- Use the VI Analyzer **programmatically** to detect issues as part of a nightly build of your LabVIEW application.

Why should I use the VI Analyzer?

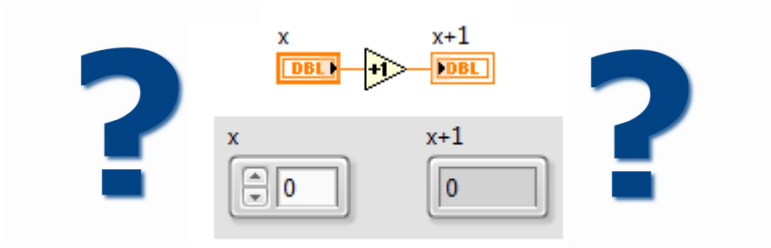
- Style compliance



- Performance analysis



- **Finding bugs!**

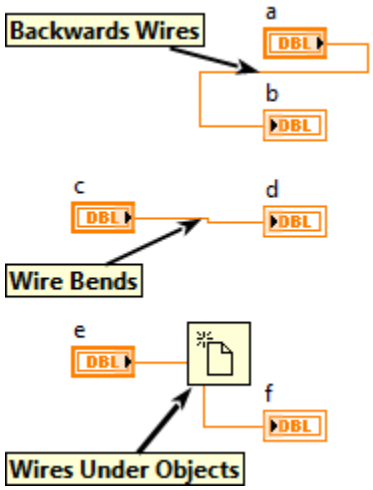


NATIONAL INSTRUMENTS

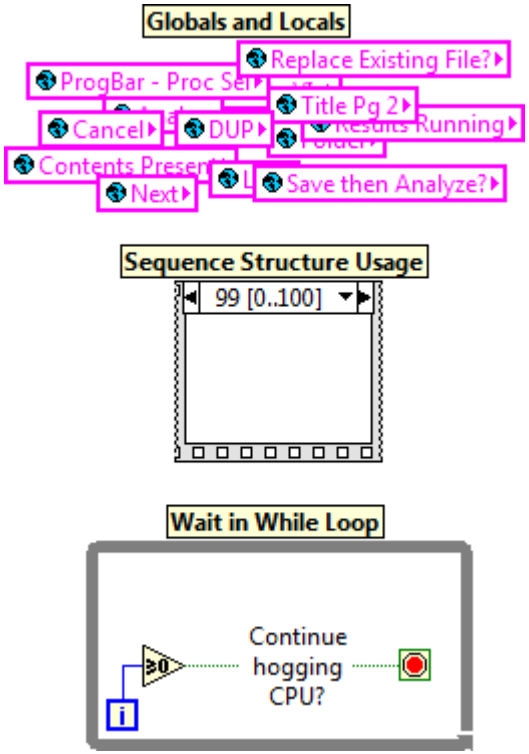
Use Case #1 – Style Compliance

Use the VI Analyzer to detect noncompliance with style guidelines

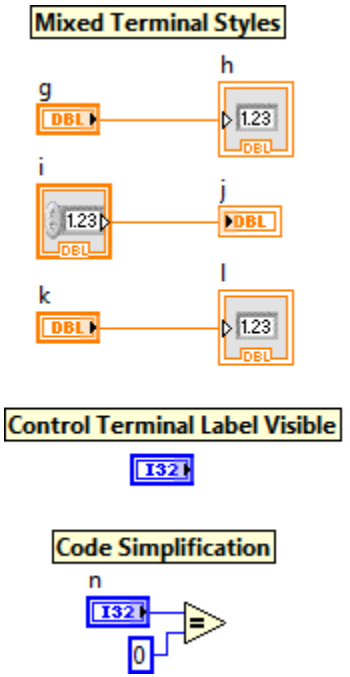
Wiring



Best Practices



General Aesthetics

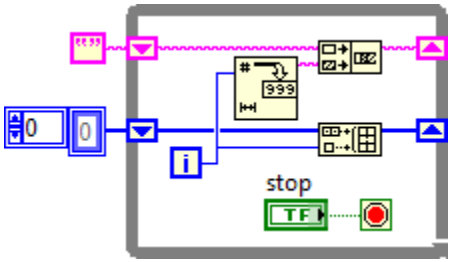


NATIONAL INSTRUMENTS

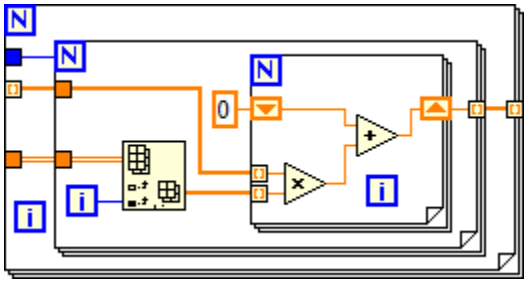
Use Case #2 – Performance Issues

Use the VI Analyzer to detect potential performance issues
(via **static code analysis**)

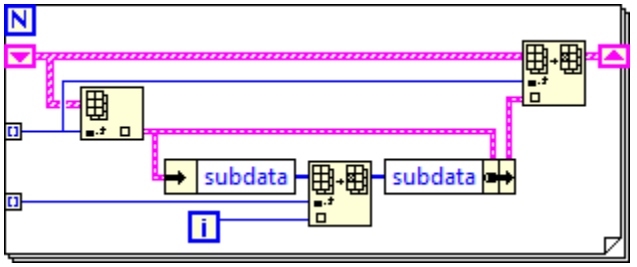
Arrays and Strings in Loops



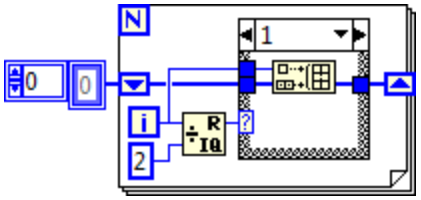
Parallelizable Loops



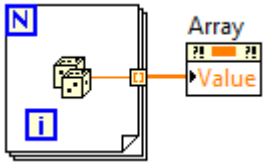
In Place Element Structure Usage



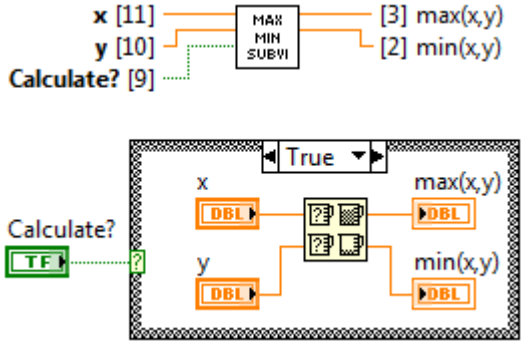
Prepend Scalar With Build Array



Value Property Usage



Wired Terminals in Subdiagrams



NATIONAL INSTRUMENTS

Use Case #3 – Finding Bugs!

Use the VI Analyzer to *find bugs in your code*.

Usually bugs are discovered and fixed via:

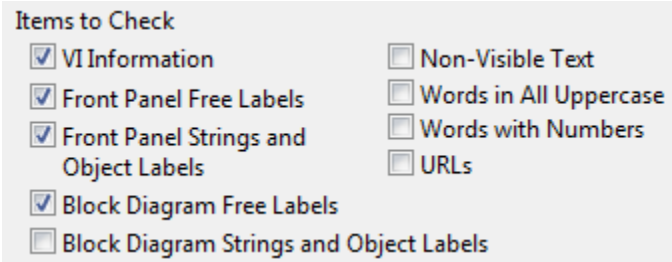
- Formal runtime unit testing (UTF, e.g.)
- Visual code reviews
- Code execution and reactionary debugging

But the VI Analyzer provides a way to discover and fix bugs *before* all of that.

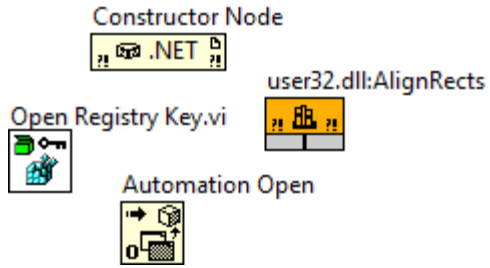
DEMO

Miscellaneous Other Tests

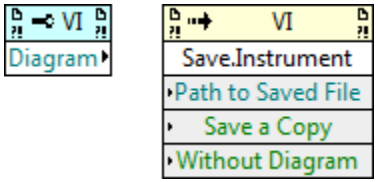
- **Spell Check** – Checks spelling of user-visible text in a VI
  - Includes standard, industry, and user dictionaries
  - Customizable options



- **Platform Portability** – Indicates issues that may arise when moving a VI from one desktop platform to another



- **Built Application Compatibility** – Checks for properties or methods that are not supported in built applications (EXEs)

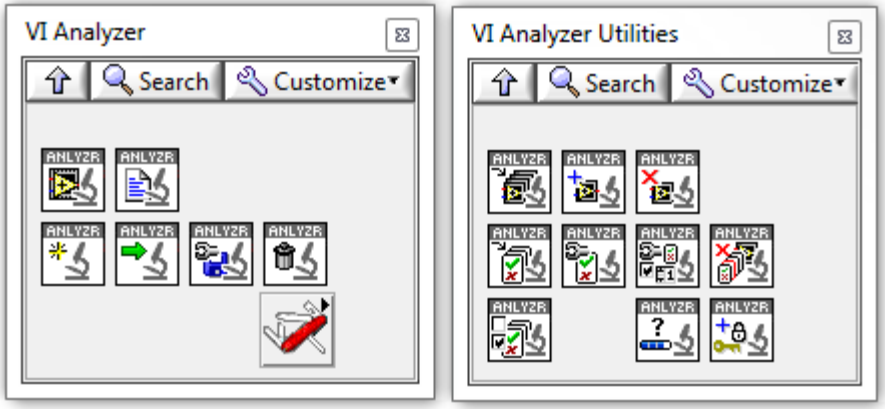


For a complete list of all 92 VI Analyzer Toolkit tests (with descriptions), google '*list of vi analyzer toolkit tests*'.

NATIONAL INSTRUMENTS

Programmatic Analysis

Use the VI Analyzer VIs to perform an analysis programmatically.



DEMO

The most common use case for this API is an automated build process that generates analysis reports on code repositories.

Find examples for the VI Analyzer API here:

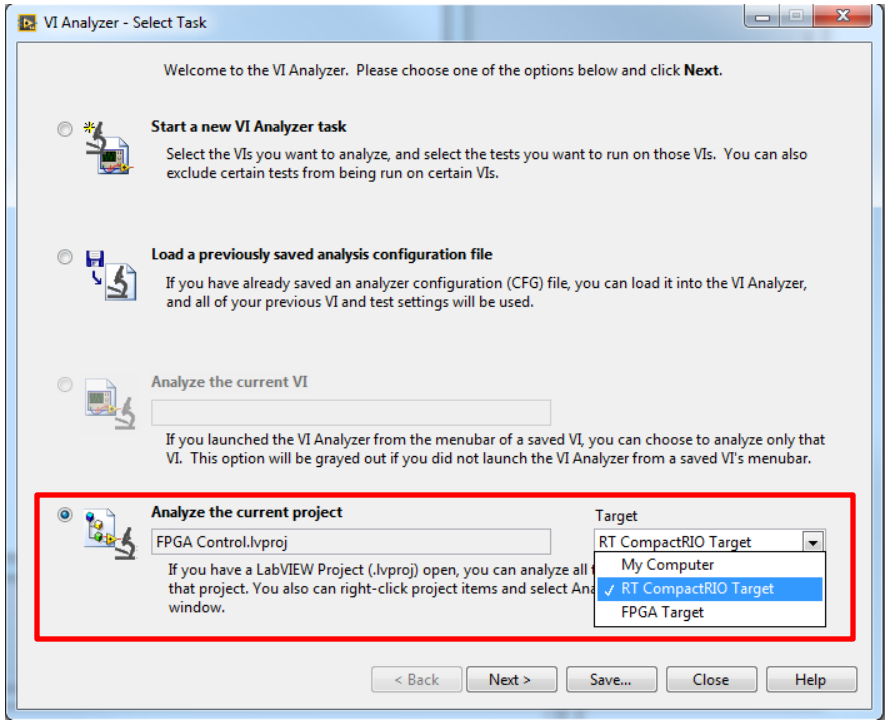
[LabVIEW 2013 and earlier]\examples\VIAnalyzer

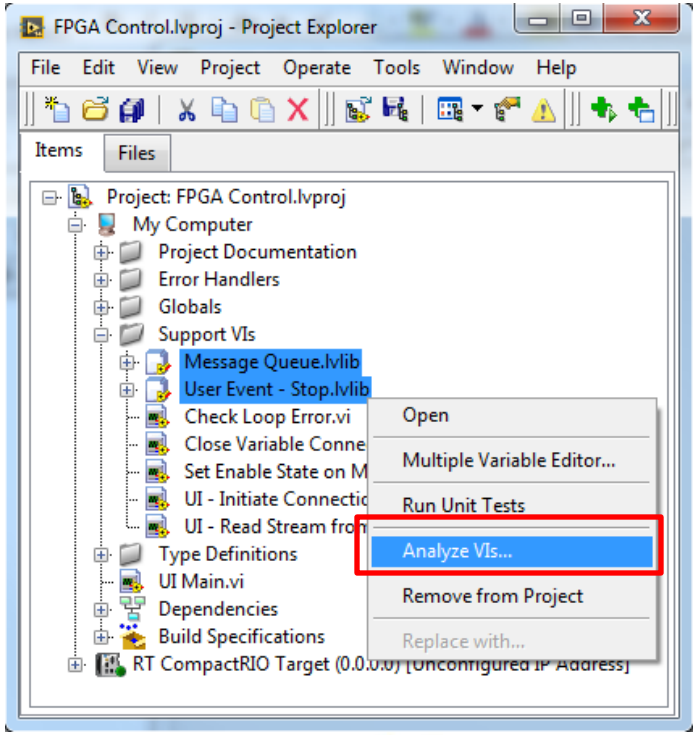
[LabVIEW 2014 and later]\examples\VI Analyzer Toolkit

Project Window Integration

The VI Analyzer Toolkit allows you to perform an analysis of all the VIs under a given target in the Project Window.

You can also right-click on project items and analyze all VIs contained within the selected project items.





NATIONAL INSTRUMENTS

VI Analyzer Customization

- Configuration Options
  - e.g. Spell Check, Globals and Locals, Arrays and Strings in Loops
- Test Exclusions
  - e.g. Exclude UI tests on internal subVIs
- Preserving Custom Settings
  - You can save all your custom settings in a VI Analyzer Configuration file (**.cfg**)

*…but what about adding **new** tests?*

VI Analyzer Test Suites

- VI Analyzer Toolkit
  - 92 tests that check for style issues, performance issues, and bugs
- Real-Time Best Practices
  - 4 tests that check for RT-specific issues
  - Installed with LabVIEW Real-Time Module 2014 and later
- Instrument Driver Guidelines
  - 23 tests that check for compliance with NI's Instrument Driver development guidelines
  - google '*instrument driver vi analyzer plug-in*'
- LabVIEW Upgrade Tests
  - Custom tests that detect potential issues when upgrading from one version of LabVIEW to another
  - Google '*vi analyzer upgrade tests*'

Creating Your Own Tests

- The **VI Analyzer Test Creator** allows you to create your own tests that plug in to the VI Analyzer
- The Test Creator is a feature of the VI Analyzer Toolkit
- Anything you can do in VI Server is fair game in a VI Analyzer test
  - Including *modifying* the VI under test!
  - DEMO

VI Analyzer Enthusiasts

The **VI Analyzer Enthusiasts** group on ni.com provides a huge repository of VI Analyzer resources

- Dozens of custom tests written by community members
- Resources for writing and using custom tests
- Discussions on accomplishing specific use cases
- Brainstorming ideas for new VI Analyzer features

Google '*vi analyzer enthusiasts*' and join the group today!

VI Analyzer – Core vs. Toolkit

VI Analyzer

- Part of LabVIEW core
- VI Analyzer UI
- VI Analyzer Engine
- Results Window
- Generating Reports

VI Analyzer Toolkit

- Separate product
  - Included in Dev Suite
  - Included with LV Pro License (2014 and later)
- 92 tests (as of 2014)
- LabVIEW API for programmatic analysis
- Project Window support
- Test Creator utility

Thanks for attending!

Remember, you can download this presentation and demo files here:

bit.ly/dnatt2015vianalyzer