Good LabVIEW Habits

Brian Powell

#OurGiantsAreFemale

Dr. Jennifer Golbeck

Professor, Information Sciences
University of Maryland

- Expert in computational social network analysis, social media, social trust, security, and privacy
- Research in how humans interact with information on social media

- Also rescues golden retrievers

@thegoldenratio4

Who Am I?

- LabVIEW R&D 20+ years, NI 26 years
- Athenahealth, Director of Engineering
- The Zebra, VP of Engineering
- CEO of Stravaro LLC
  - LabVIEW Champion
  - Certified LabVIEW Architect
  - Certified Professional Instructor

Passionate about

- team culture
- organizational structure
- people management
- software engineering processes

Special Thanks To The LabVIEW Champions

for providing input and ideas

“Habits” not “Rules”

- There’s no particular order
- Some are small. Some are big.
- Not everybody agrees with these
- Don’t adopt these unless you understand the “why”

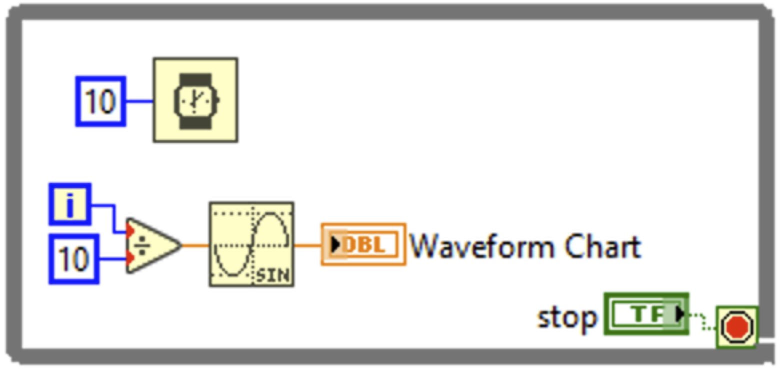
- If we have time at the end, offer your own good habits

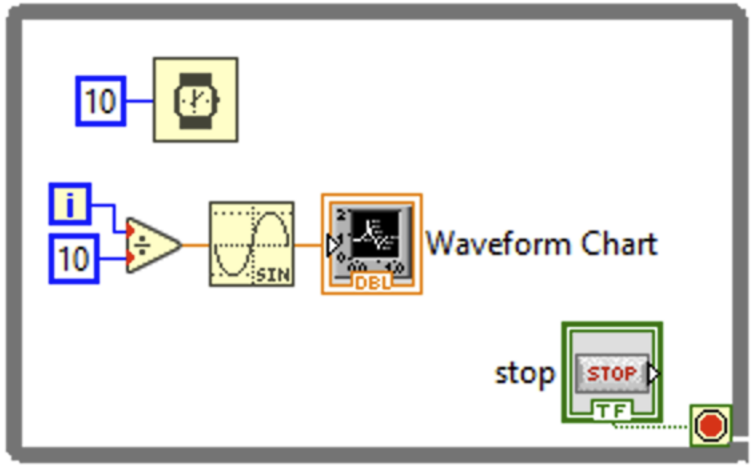
#1

Why?
Takes up extra space.
Hard to visually distinguish from subVI.

Tools >> Options settings I always change:

- Turn off "Place front panel terminals as icons"



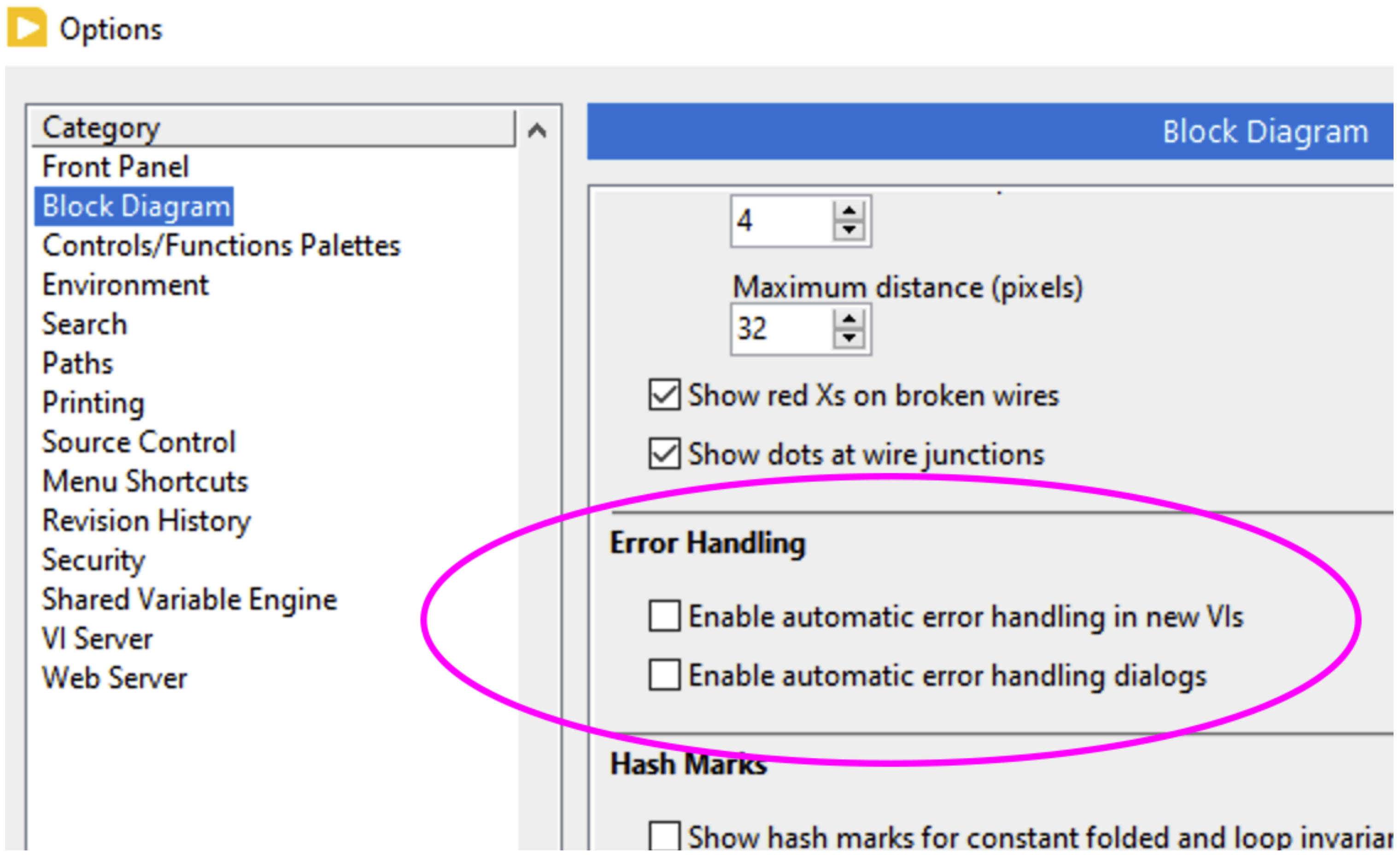


#2

Why?
Different behavior on different machines.
Substitute for good error checking.

Tools >> Options settings I always change:

- Turn off automatic error handling



Options

Category
Front Panel
Block Diagram
Controls/Functions Palettes
Environment
Search
Paths
Printing
Source Control
Menu Shortcuts
Revision History
Security
Shared Variable Engine
VI Server
Web Server

Block Diagram

4

Maximum distance (pixels)
32

☑ Show red Xs on broken wires
☑ Show dots at wire junctions

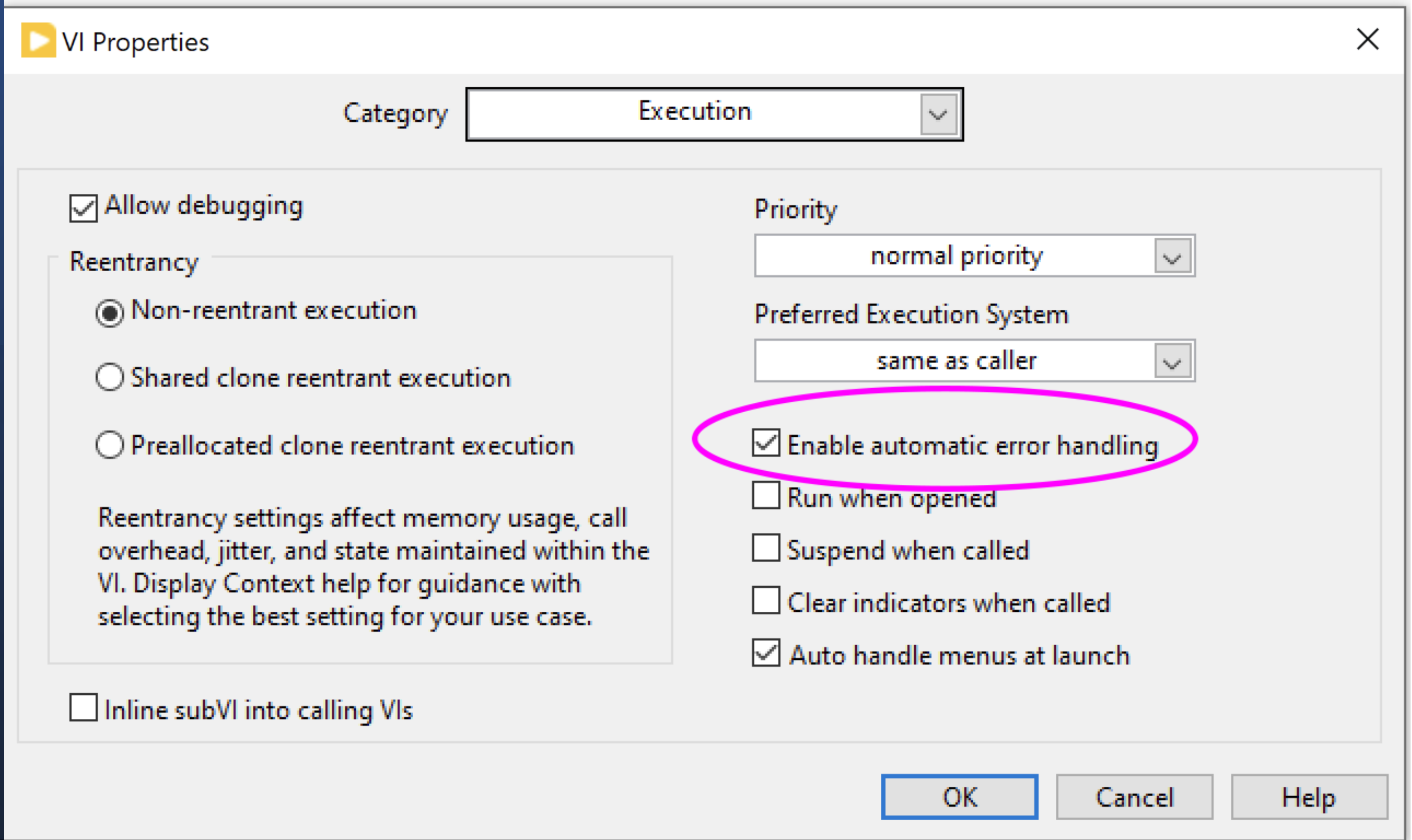
Error Handling
☐ Enable automatic error handling in new VIs
☐ Enable automatic error handling dialogs

Hash Marks
☐ Show hash marks for constant folded and loop invariar

#2b

Also saved with the VI.

Use the VI Analyzer to find these.


VI Properties

Category: Execution

[✓] Allow debugging

Reentrancy
- [✓ Non-reentrant execution]
- Shared clone reentrant execution
- Preallocated clone reentrant execution

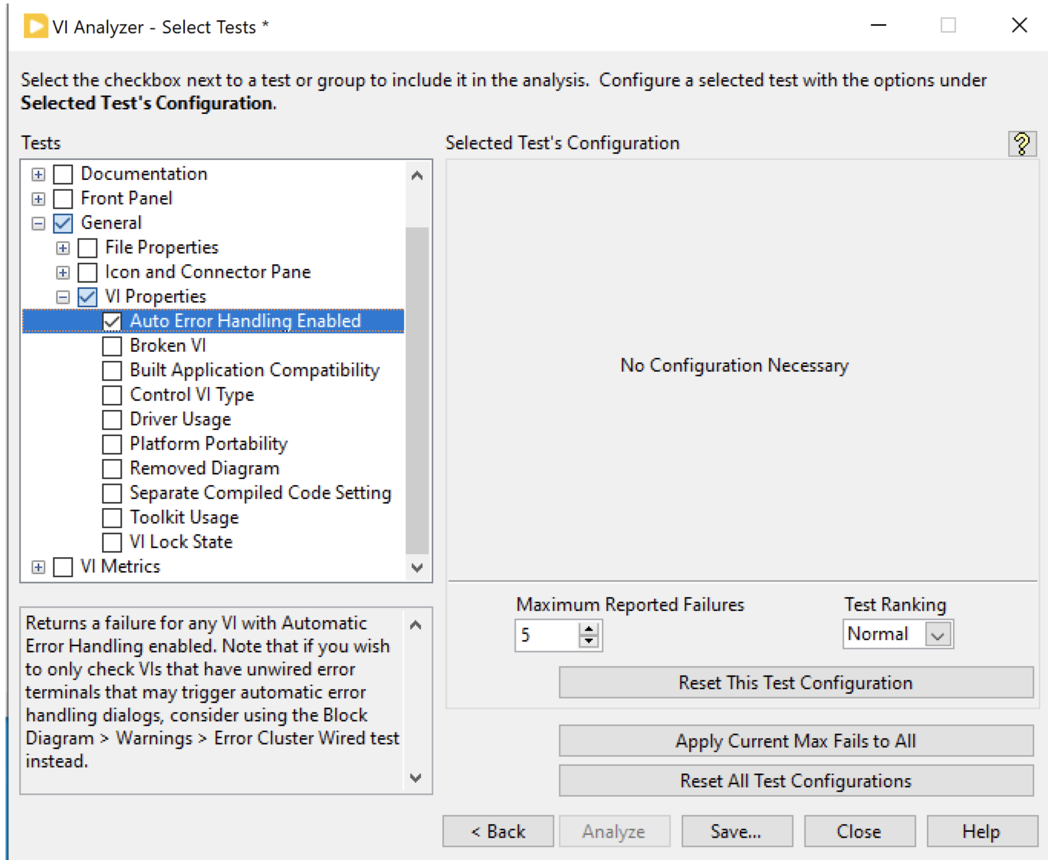
Reentrancy settings affect memory usage, call overhead, jitter, and state maintained within the VI. Display Context help for guidance with selecting the best setting for your use case.

[ ] Inline subVI into calling VIs

Priority: normal priority

Preferred Execution System: same as caller

[✓] Enable automatic error handling
[ ] Run when opened
[ ] Suspend when called
[ ] Clear indicators when called
[✓] Auto handle menus at launch


VI Analyzer - Select Tests *

Select the checkbox next to a test or group to include it in the analysis. Configure a selected test with the options under **Selected Test's Configuration.**

Tests
- [ ] Documentation
- [ ] Front Panel
- [✓] General
  - [ ] File Properties
  - [ ] Icon and Connector Pane
  - [✓] VI Properties
    - [✓] Auto Error Handling Enabled
    - [ ] Broken VI
    - [ ] Built Application Compatibility
    - [ ] Control VI Type
    - [ ] Driver Usage
    - [ ] Platform Portability
    - [ ] Removed Diagram
    - [ ] Separate Compiled Code Setting
    - [ ] Toolkit Usage
    - [ ] VI Lock State
- [ ] VI Metrics

Returns a failure for any VI with Automatic Error Handling enabled. Note that if you wish to only check VIs that have unwired error terminals that may trigger automatic error handling dialogs, consider using the Block Diagram > Warnings > Error Cluster Wired test instead.

Selected Test's Configuration

No Configuration Necessary

Maximum Reported Failures: 5

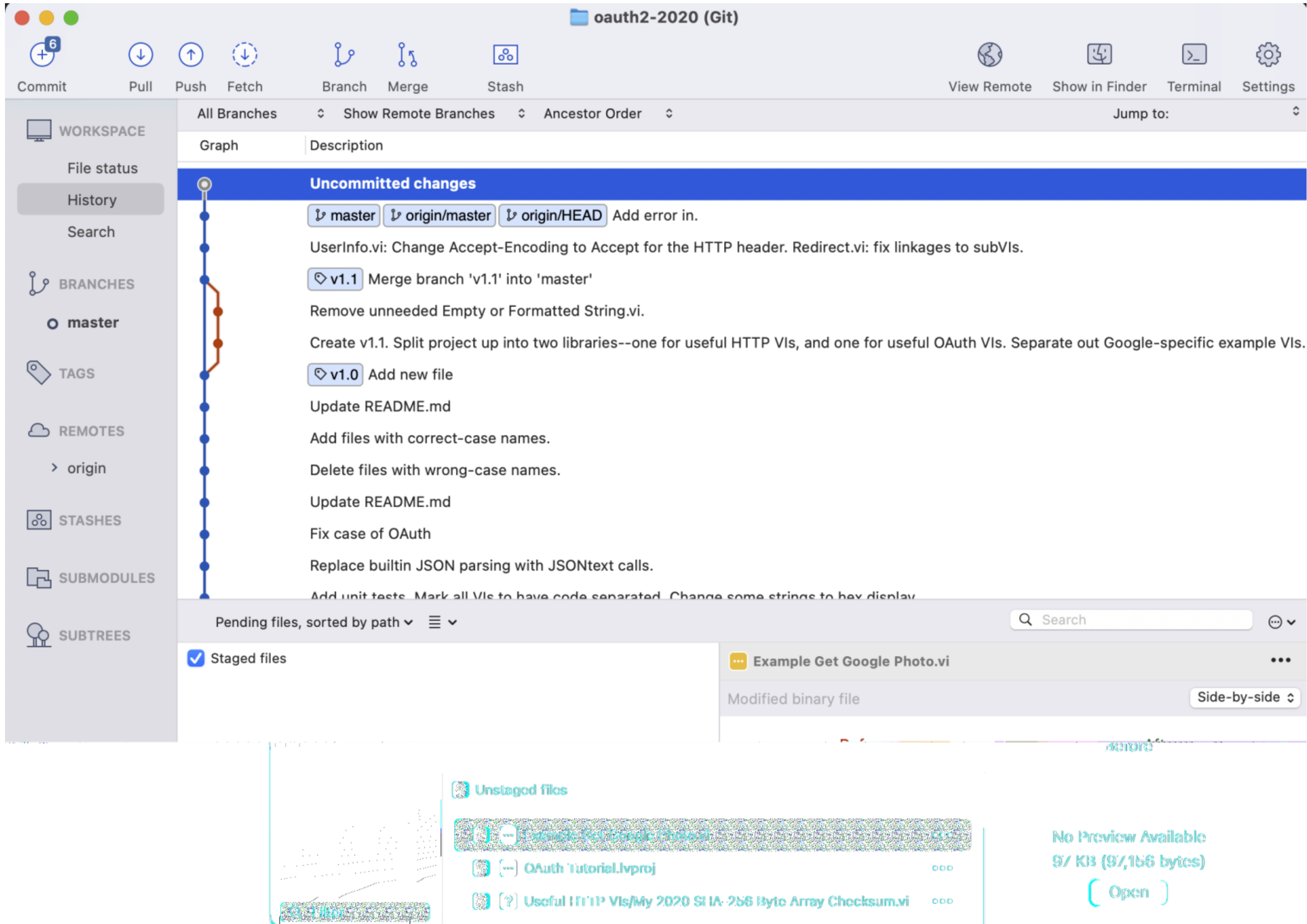
Test Ranking: Normal

Reset This Test Configuration
Apply Current Max Fails to All
Reset All Test Configurations
< Back
Analyze
Save...
Close
Help

#3

Why?
Save you from yourself (and others).
Remembers history.

Use source code control


oauth2-2020 (Git)

Uncommitted changes
master | origin/master | origin/HEAD Add error in.
UserInfo.vi: Change Accept-Encoding to Accept for the HTTP header. Redirect.vi: fix linkages to subVIs.
v1.1 Merge branch 'v1.1' into 'master'
Remove unneeded Empty or Formatted String.vi.
Create v1.1. Split project up into two libraries--one for useful HTTP VIs, and one for useful OAuth VIs. Separate out Google-specific example VIs.
v1.0 Add new file
Update README.md
Add files with correct-case names.
Delete files with wrong-case names.
Update README.md
Fix case of OAuth
Replace builtin JSON parsing with JSONtext calls.
Add unit tests. Mark all VIs to have code separated. Change some strings to hex display.

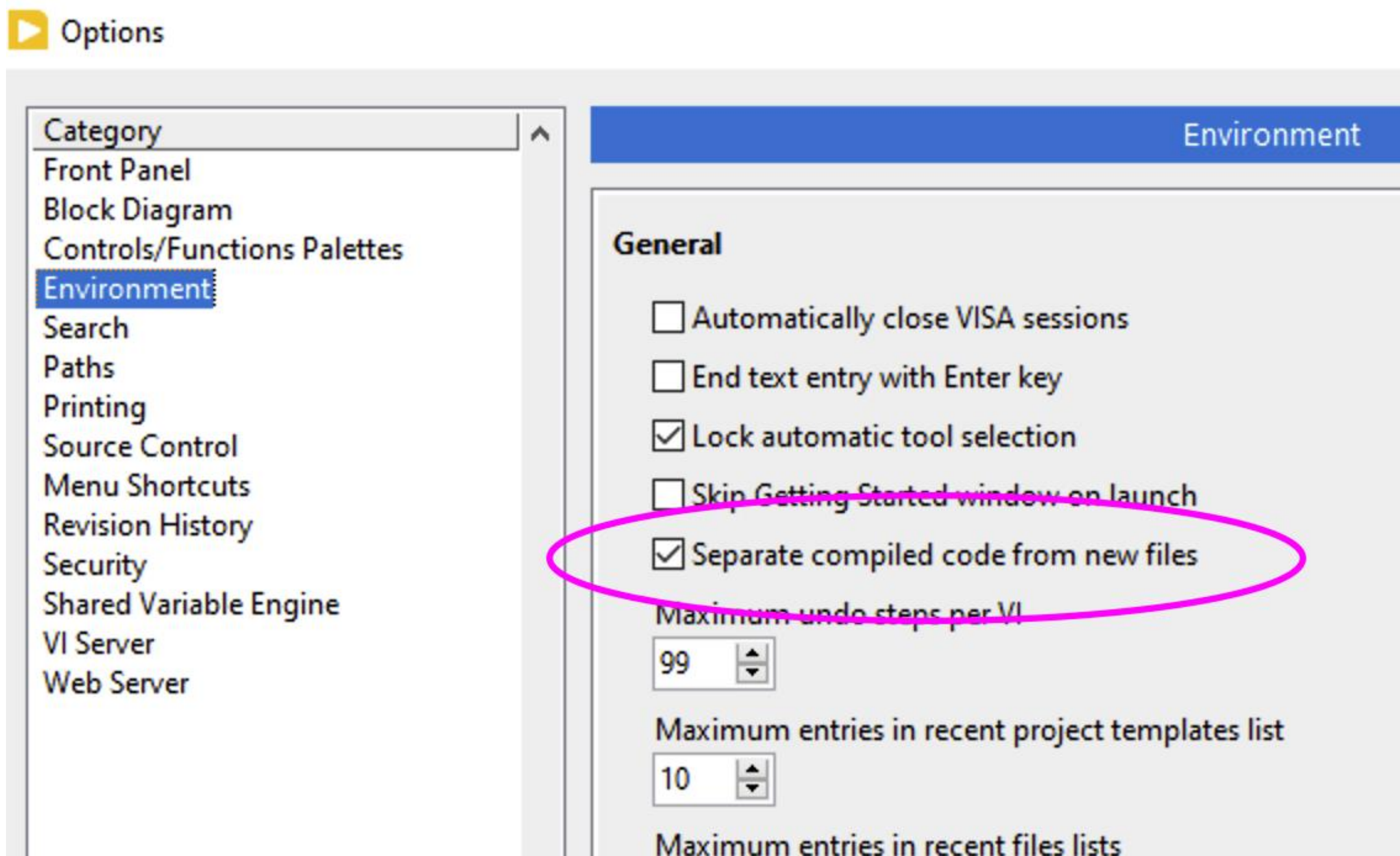
Pending files, sorted by path
Staged files
Example Get Google Photo.vi
Modified binary file
Side-by-side
Unstaged files
OAuth Tutorial.lvproj
Useful HTTP VIs/My 2020 SHA-256 Byte Array Checksum.vi
No Preview Available
97 KB (97,156 bytes)
Open

#4

Why?
Avoid source code control changes just due to recompile.
Smaller file size in SCC.

Tools >> Options settings I always change:

- Separate compiled code from VIs


Options

Category
Front Panel
Block Diagram
Controls/Functions Palettes
Environment
Search
Paths
Printing
Source Control
Menu Shortcuts
Revision History
Security
Shared Variable Engine
VI Server
Web Server

Environment

General

- [ ] Automatically close VISA sessions
- [ ] End text entry with Enter key
- [x] Lock automatic tool selection
- [ ] Skip Getting Started window on launch
- [x] Separate compiled code from new files

Maximum undo steps per VI: 99

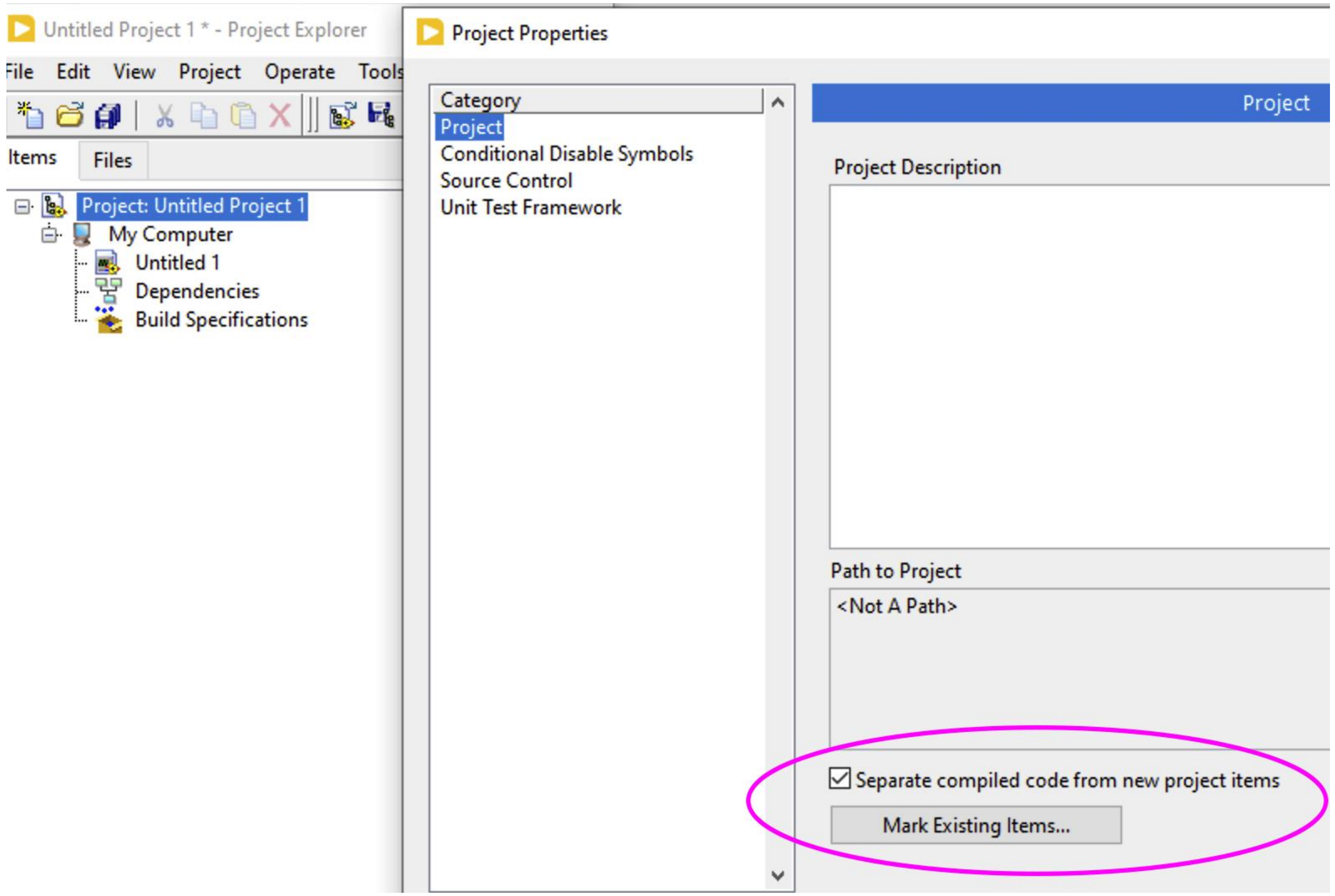
Maximum entries in recent project templates list: 10

Maximum entries in recent files lists

#4b

Don't forget to "Mark Existing Items". I check this for any stragglers from time to time.

Project settings, too:


Untitled Project 1 * - Project Explorer

File Edit View Project Operate Tools

Items Files

- Project: Untitled Project 1
  - My Computer
    - Untitled 1
    - Dependencies
    - Build Specifications

Project Properties

Category
- Project
- Conditional Disable Symbols
- Source Control
- Unit Test Framework

Project

Project Description

Path to Project

<Not A Path>

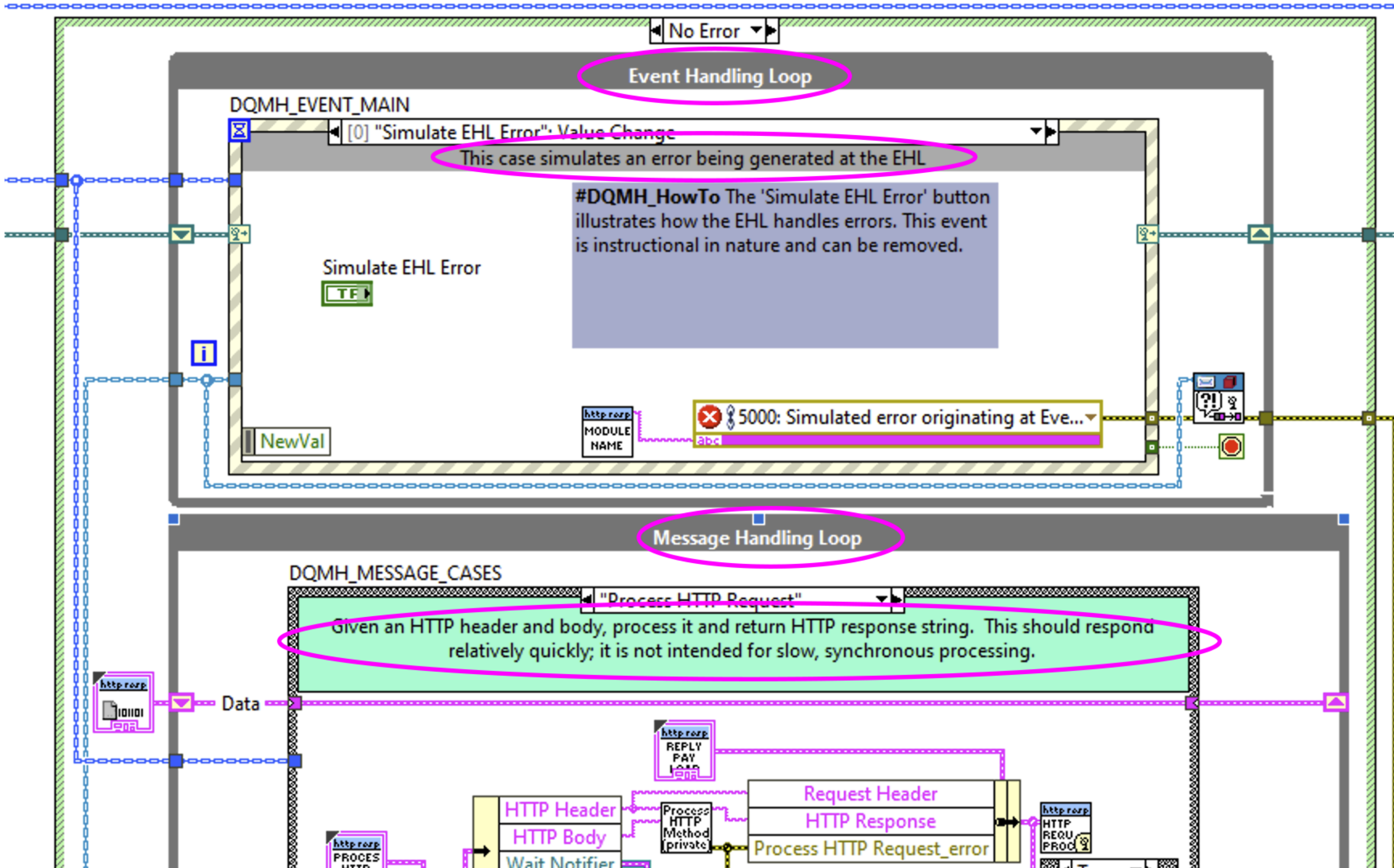
☑ Separate compiled code from new project items

Mark Existing Items...

#5

Use subdiagram labels

Why?
Good documentation practice.
Not free-floating; stays with each loop/case.


This case simulates an error being generated at the EHL

#DQMH_HowTo The 'Simulate EHL Error' button illustrates how the EHL handles errors. This event is instructional in nature and can be removed.

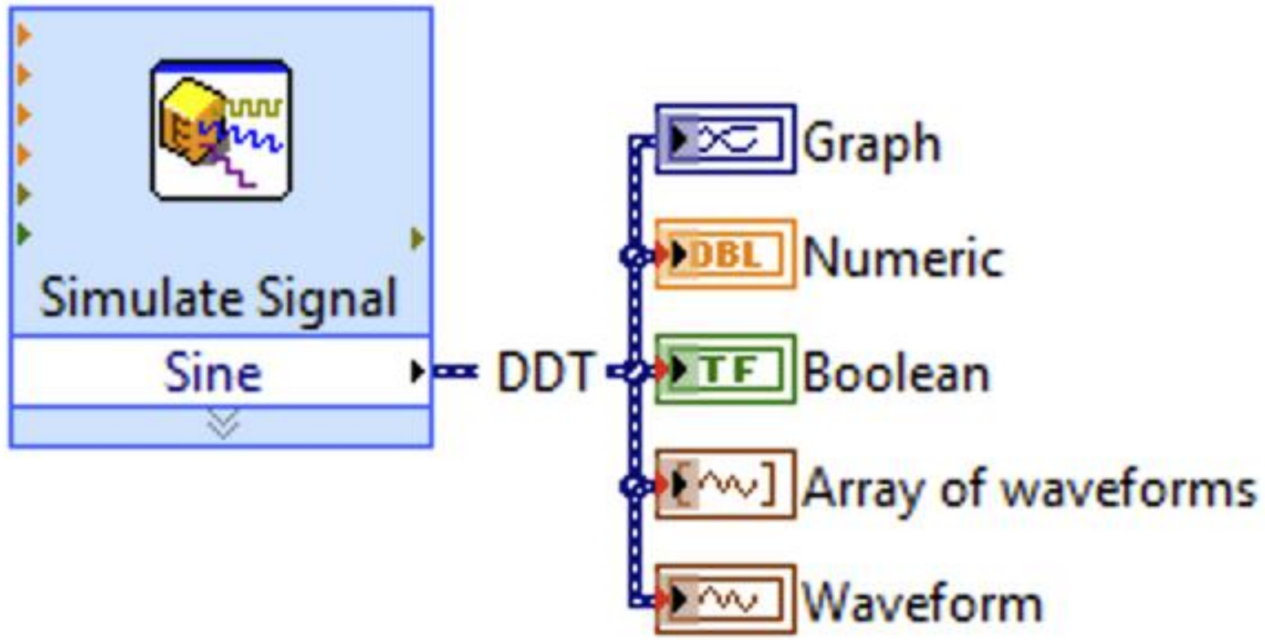
5000: Simulated error originating at Eve...

Given an HTTP header and body, process it and return HTTP response string. This should respond relatively quickly; it is not intended for slow, synchronous processing.

#6

Why?
Kitchen sink data type.
Too easy to do the wrong thing.

Don’t use the Dynamic Data Type (DDT)

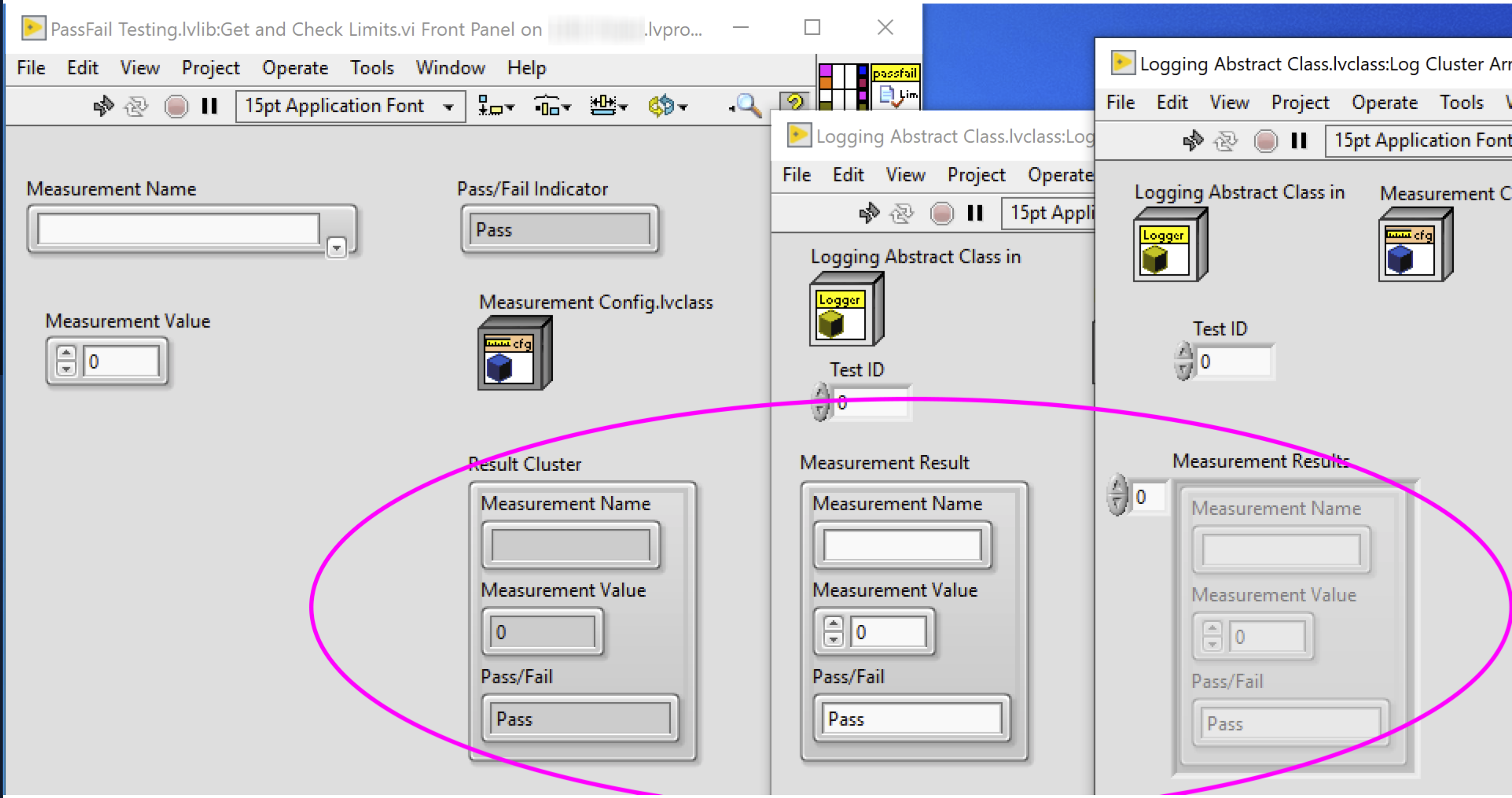


#7

Why?
Easy to propagate changes to all users of the data type.

When used in more than one place:

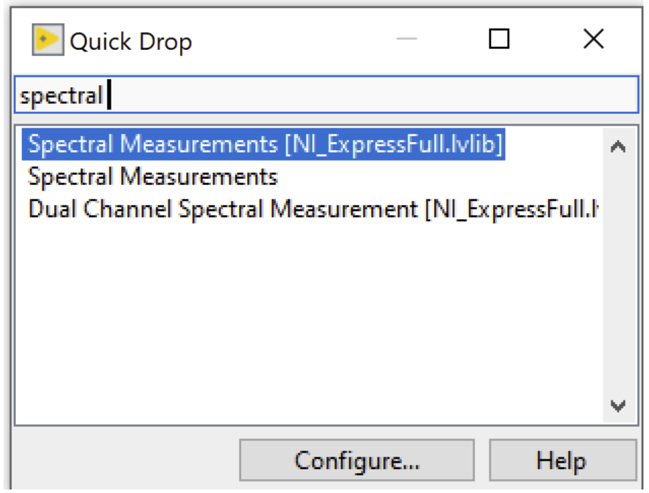
- Use typedefs (or classes) for clusters
- Use typedefs for enums.



#8

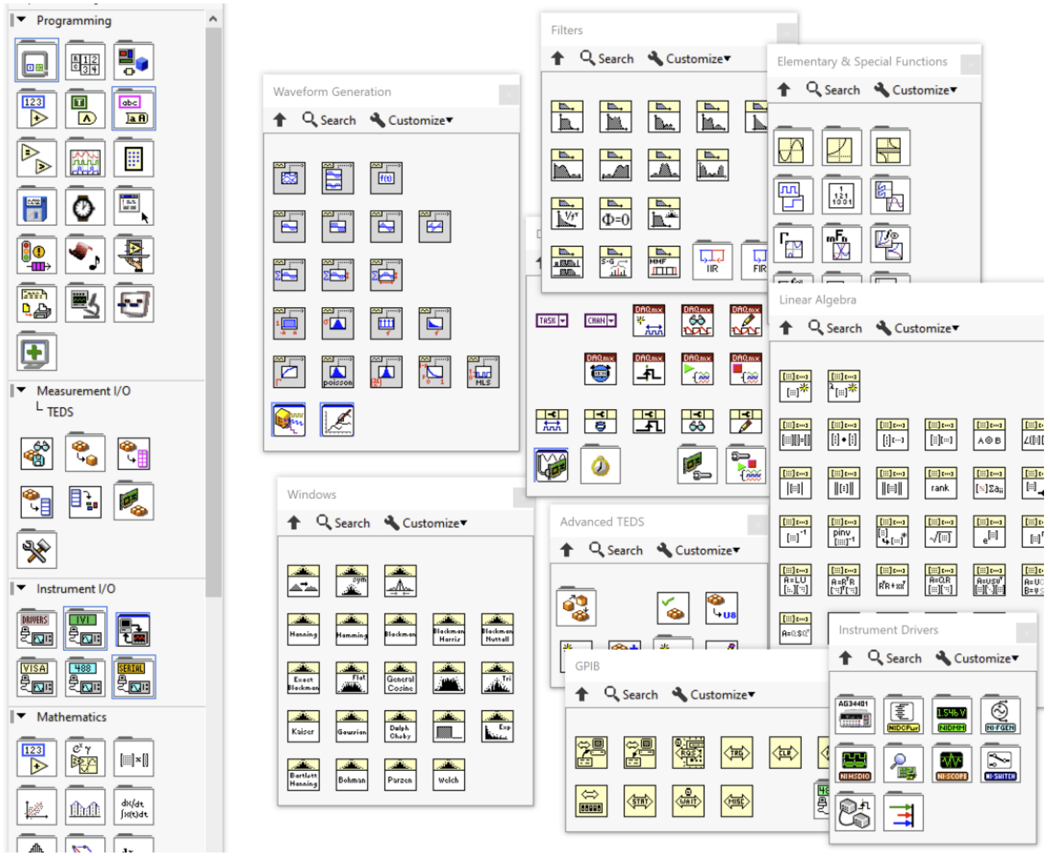
Why?
Saves time.

Use Quickdrop



spectral
Spectral Measurements [NI_ExpressFull.lvlib]
Spectral Measurements
Dual Channel Spectral Measurement [NI_ExpressFull.l

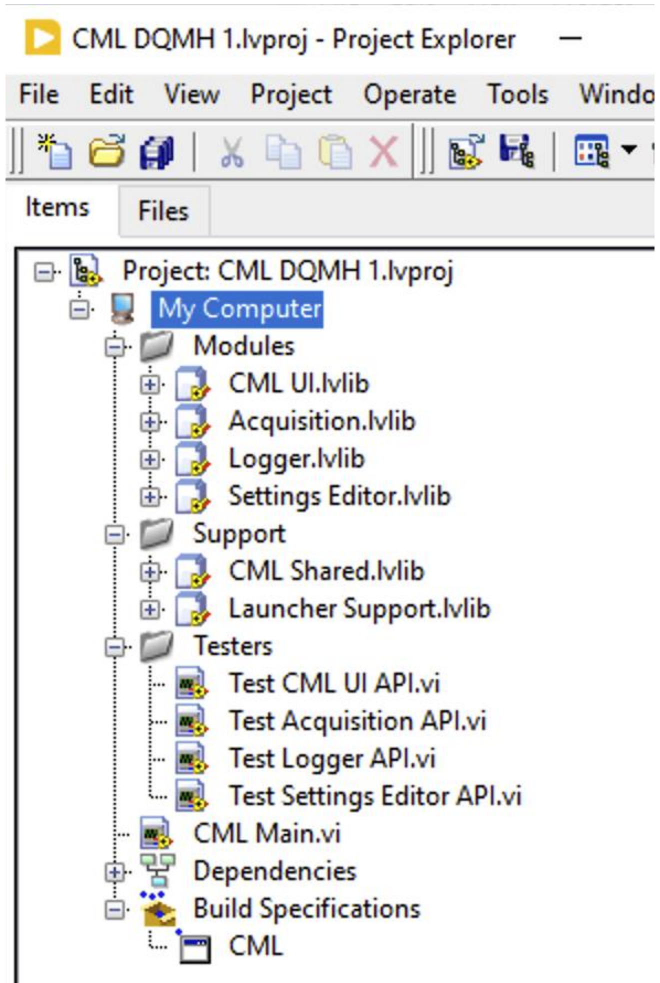
vs.



#9

Use a Project (.lvproj) for your application.

Why?
Organize your files.
Only way to build an executable.

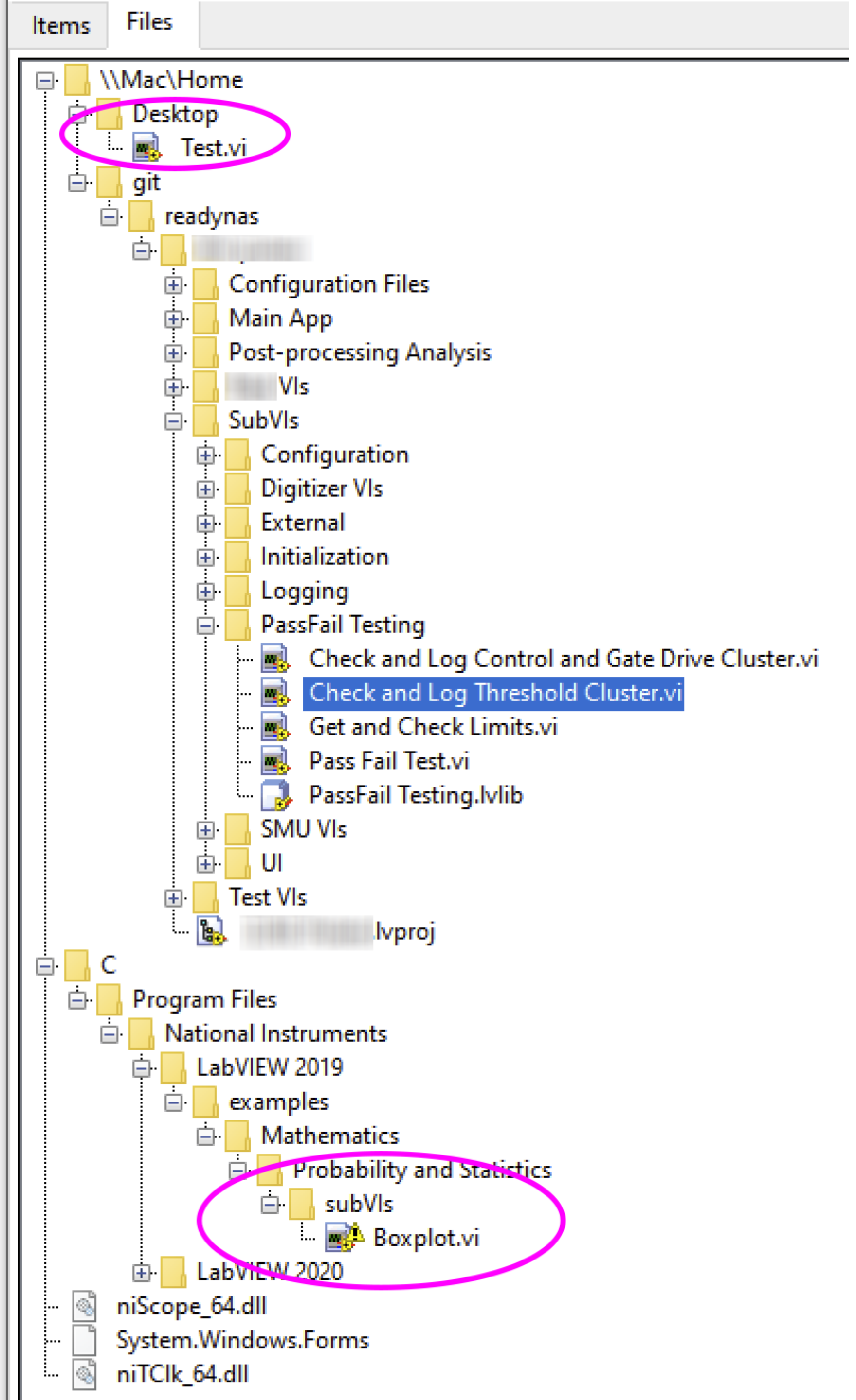

CML DQMH 1.lvproj - Project Explorer
File Edit View Project Operate Tools Windo
Items Files

```
Project: CML DQMH 1.lvproj
  My Computer
    Modules
      CML UI.lvlib
      Acquisition.lvlib
      Logger.lvlib
      Settings Editor.lvlib
    Support
      CML Shared.lvlib
      Launcher Support.lvlib
    Testers
      Test CML UI API.vi
      Test Acquisition API.vi
      Test Logger API.vi
      Test Settings Editor API.vi
    CML Main.vi
    Dependencies
    Build Specifications
      CML
```

#10

Why?
Make sure you're using the files you think you are.

Check "Files" view in the project for stray files that are out of place.



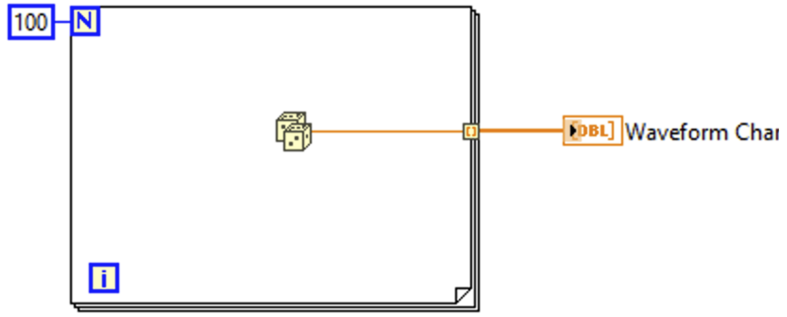
```
\\Mac\Home
    Desktop
        Test.vi
    git
        readynas
            [redacted]
                Configuration Files
                Main App
                Post-processing Analysis
                [redacted] VIs
                SubVIs
                    Configuration
                    Digitizer VIs
                    External
                    Initialization
                    Logging
                    PassFail Testing
                        Check and Log Control and Gate Drive Cluster.vi
                        Check and Log Threshold Cluster.vi
                        Get and Check Limits.vi
                        Pass Fail Test.vi
                        PassFail Testing.lvlib
                    SMU VIs
                    UI
                Test VIs
                [redacted].lvproj
C
    Program Files
        National Instruments
            LabVIEW 2019
                examples
                    Mathematics
                        Probability and Statistics
                            subVIs
                                Boxplot.vi
            LabVIEW 2020
niScope_64.dll
System.Windows.Forms
niTClk_64.dll
```

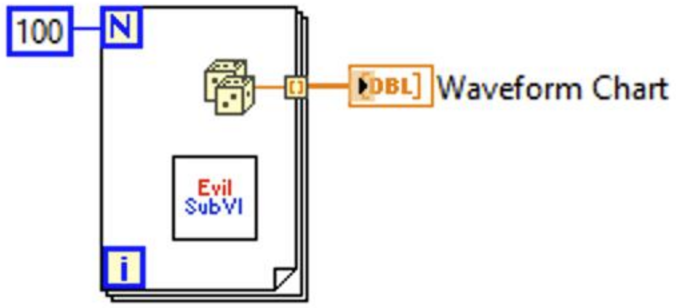
#11

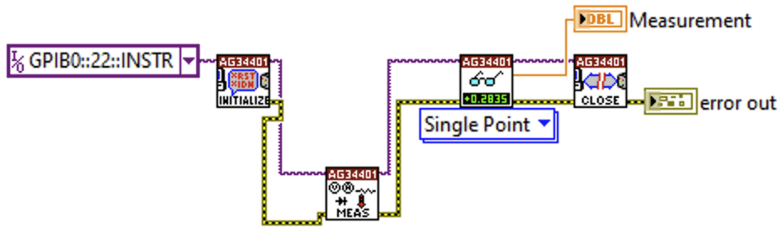
Use block diagram cleanup.

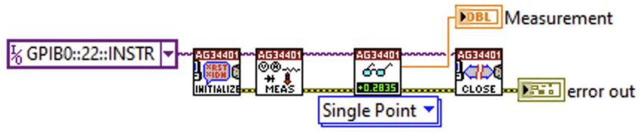
(But be prepared to undo.)

Why?
Let's you build VIs quickly, clean up as you go.
Finds miswired terminals.
Finds hidden objects.





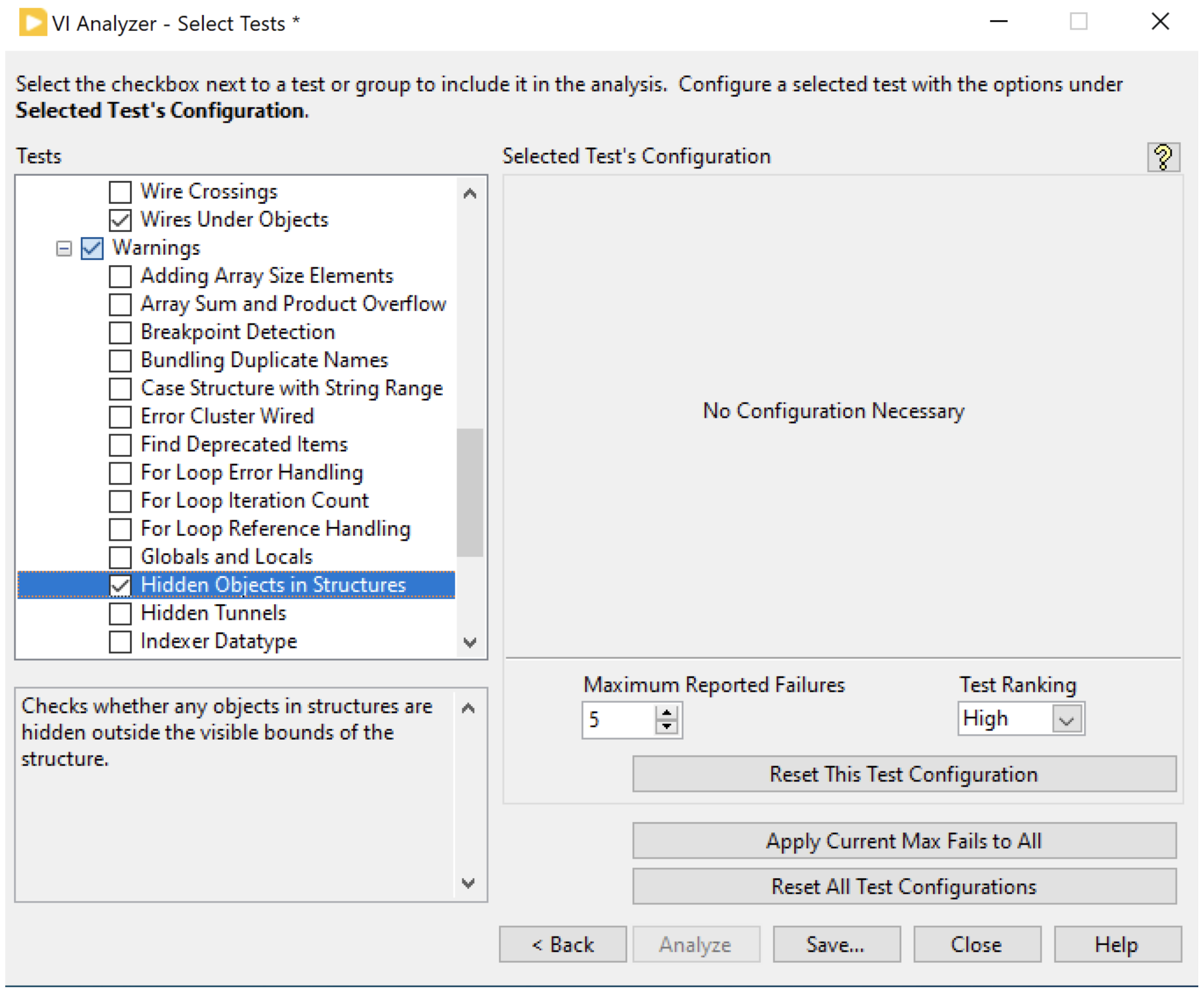




#11b

There are specific tests for hidden objects, miswired terminals, etc.

If you don't want to use Block Diagram Cleanup, use the VI Analyzer. It can help find similar problems.



VI Analyzer - Select Tests *

Select the checkbox next to a test or group to include it in the analysis. Configure a selected test with the options under **Selected Test's Configuration.**

Tests

- ☐ Wire Crossings
- ☑ Wires Under Objects
- ☑ Warnings
  - ☐ Adding Array Size Elements
  - ☐ Array Sum and Product Overflow
  - ☐ Breakpoint Detection
  - ☐ Bundling Duplicate Names
  - ☐ Case Structure with String Range
  - ☐ Error Cluster Wired
  - ☐ Find Deprecated Items
  - ☐ For Loop Error Handling
  - ☐ For Loop Iteration Count
  - ☐ For Loop Reference Handling
  - ☐ Globals and Locals
  - ☑ Hidden Objects in Structures
  - ☐ Hidden Tunnels
  - ☐ Indexer Datatype

Checks whether any objects in structures are hidden outside the visible bounds of the structure.

Selected Test's Configuration

No Configuration Necessary

Maximum Reported Failures: 5

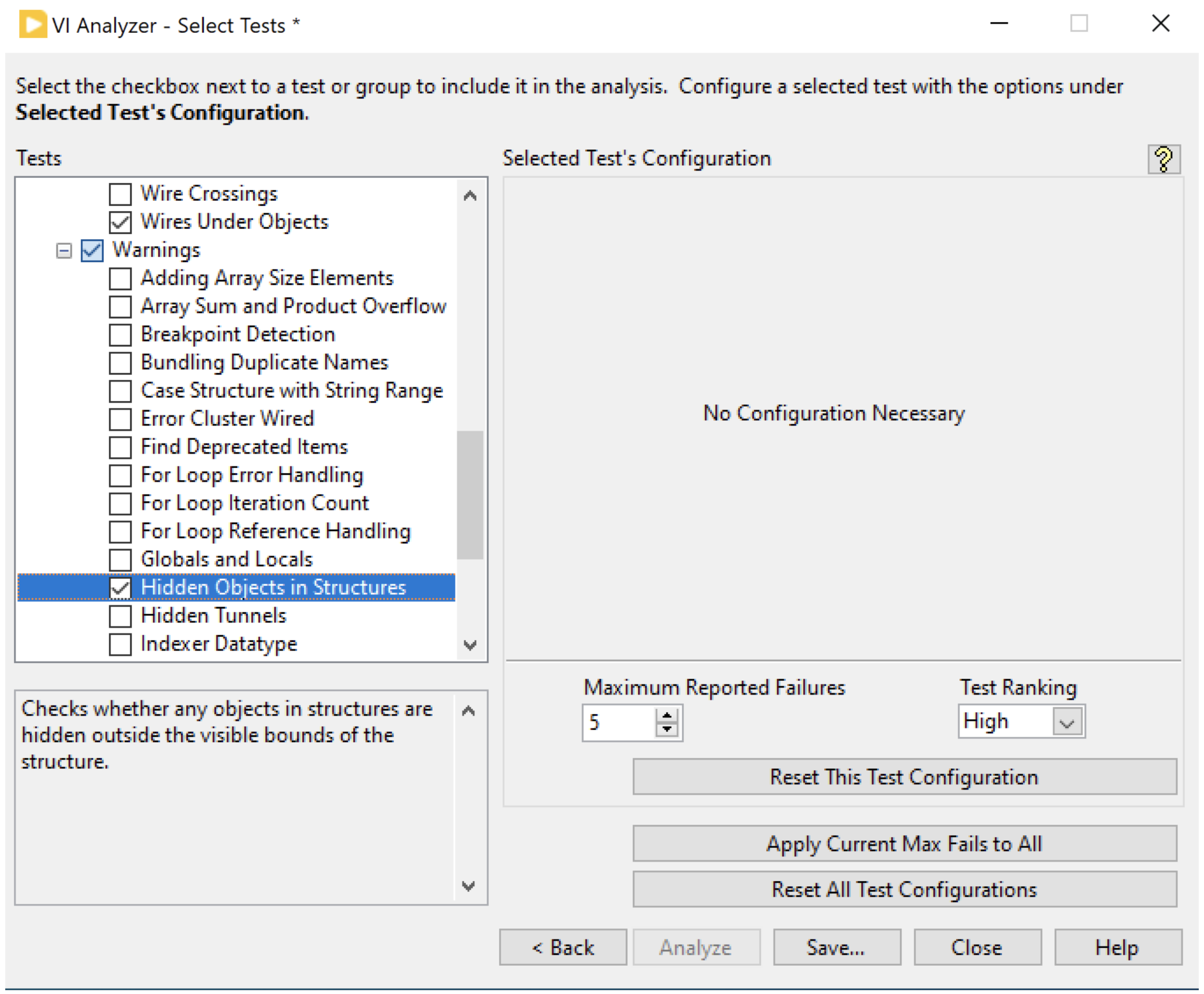
Test Ranking: High

Reset This Test Configuration
Apply Current Max Fails to All
Reset All Test Configurations
< Back
Analyze
Save...
Close
Help

#12

Use the VI Analyzer even if you do use diagram cleanup.

Why?
Enforce coding style.
Find problems.
Improve consistency.


VI Analyzer - Select Tests *

Select the checkbox next to a test or group to include it in the analysis. Configure a selected test with the options under **Selected Test's Configuration.**

Tests

- [ ] Wire Crossings
- [x] Wires Under Objects
- [x] Warnings
  - [ ] Adding Array Size Elements
  - [ ] Array Sum and Product Overflow
  - [ ] Breakpoint Detection
  - [ ] Bundling Duplicate Names
  - [ ] Case Structure with String Range
  - [ ] Error Cluster Wired
  - [ ] Find Deprecated Items
  - [ ] For Loop Error Handling
  - [ ] For Loop Iteration Count
  - [ ] For Loop Reference Handling
  - [ ] Globals and Locals
  - [x] Hidden Objects in Structures
  - [ ] Hidden Tunnels
  - [ ] Indexer Datatype

Checks whether any objects in structures are hidden outside the visible bounds of the structure.

Selected Test's Configuration

No Configuration Necessary

Maximum Reported Failures: 5

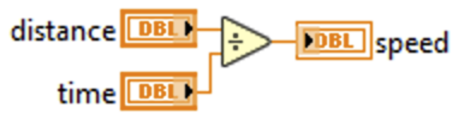
Test Ranking: High

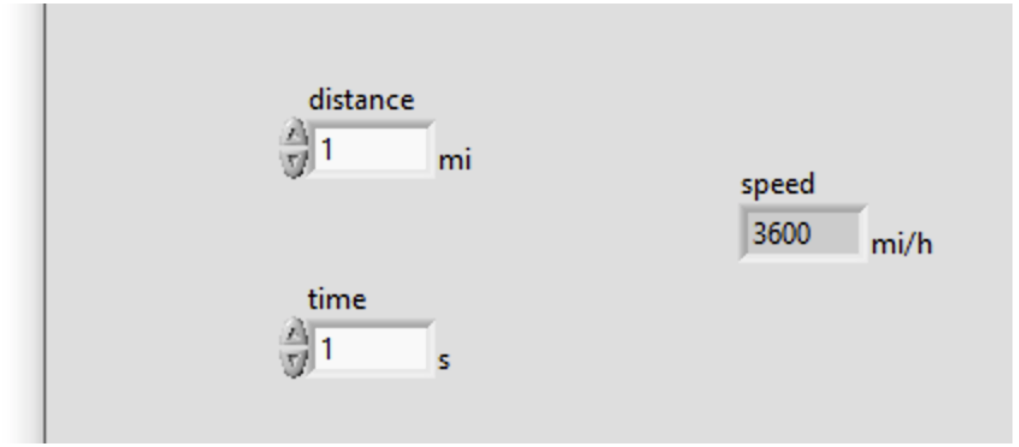
Reset This Test Configuration
Apply Current Max Fails to All
Reset All Test Configurations
< Back
Analyze
Save...
Close
Help

#13

Why?
Helps get your unit math right.
Easy to change for locale (km vs. miles, deg C vs. F)

Take advantage of the Unit Label



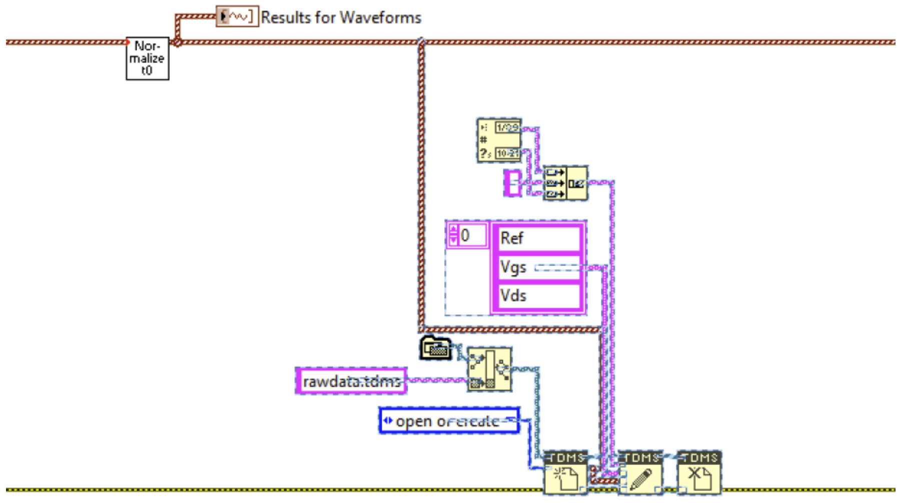


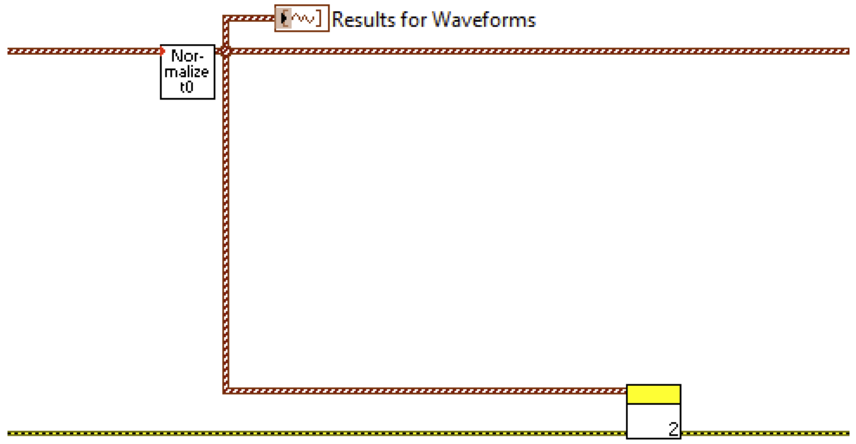
https://stravaro.com/2021/07/labview-units-part-1/

#14

Why?
Organize and simplify code.
Aids reuse.

Use Edit >> Create SubVI



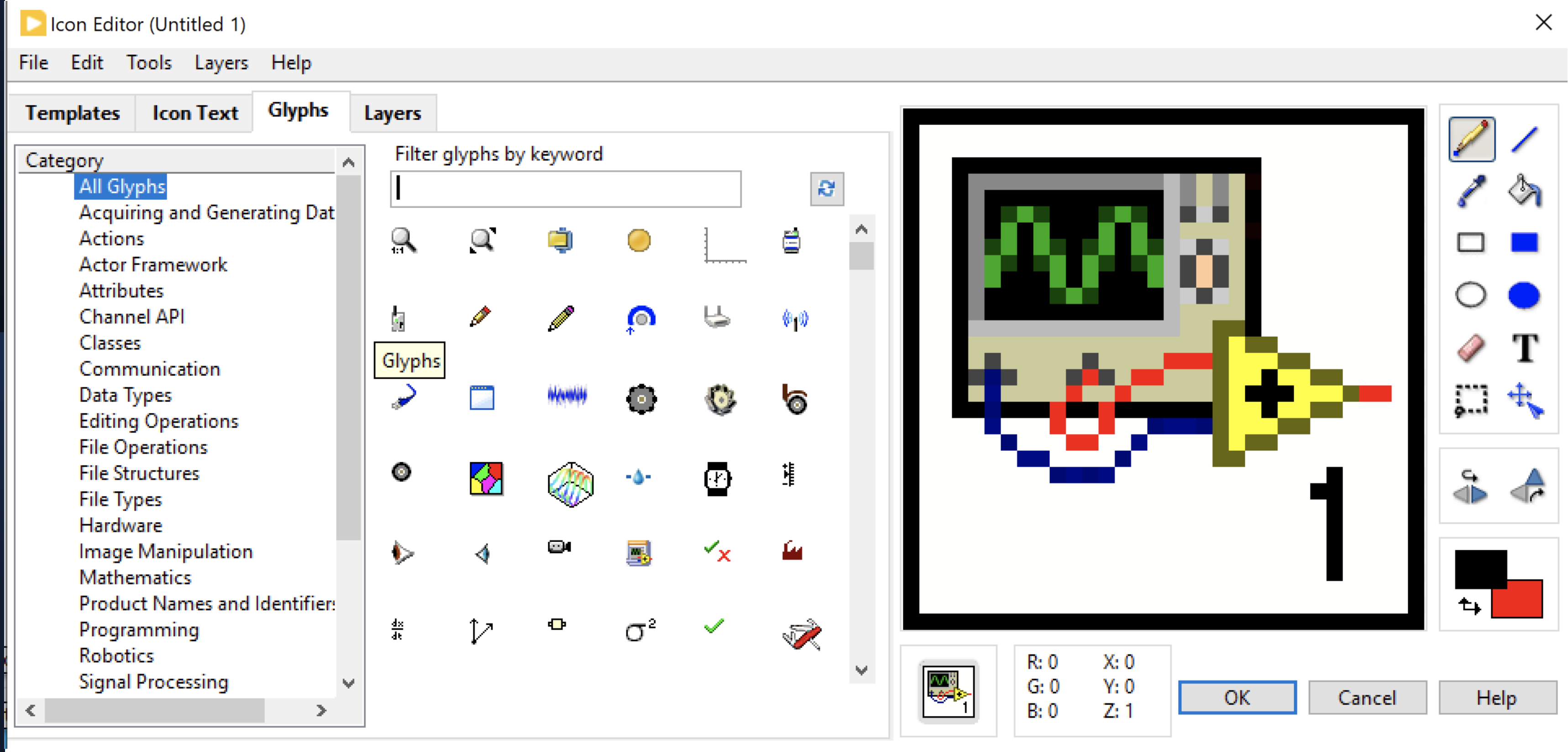


#14b

Why?
Modularity is a good thing.
Can run faster.

Also, use subVIs. They're a good thing.

And please edit the icon.



#15

Why?
No sense optimizing code you may end up rewriting/reorganizing.

Optimize for clarity. Don't optimize for speed (nor memory) until needed. (From Wiebe Walstra.)

Make it work first. Then make it work faster/better.

#16

Why?
Lets you install different toolkits/versions per VM.
Helps avoid cross-linking.

If you work on multiple projects, use virtual machines, with one VM per project.

#17

Why?
“Clever” code is hard for anyone but the author to understand and maintain.
Hard for the author, too.

Don’t be clever.

Additional Resources

- An End to Brainless LabVIEW Programming
  - http://bit.ly/brainlesslabview
- LabVIEW Journal Blog
  - https://labviewjournal.com/
- Stravaro Blog
  - https://stravaro.com/stravaro-blog