Your LabVIEW Code Is a Work of Art…

…But I Can't Read It

Darren Nattinger

Chief TSE, NI

Certified LabVIEW Architect

Before we get started

- All of my LabVIEW presentations are available at:

dna rg

- Link to this presentation: https://bit.ly/labviewreadability

Download link for ZoomIt

Prologue

DNatt Presentation Catalysts

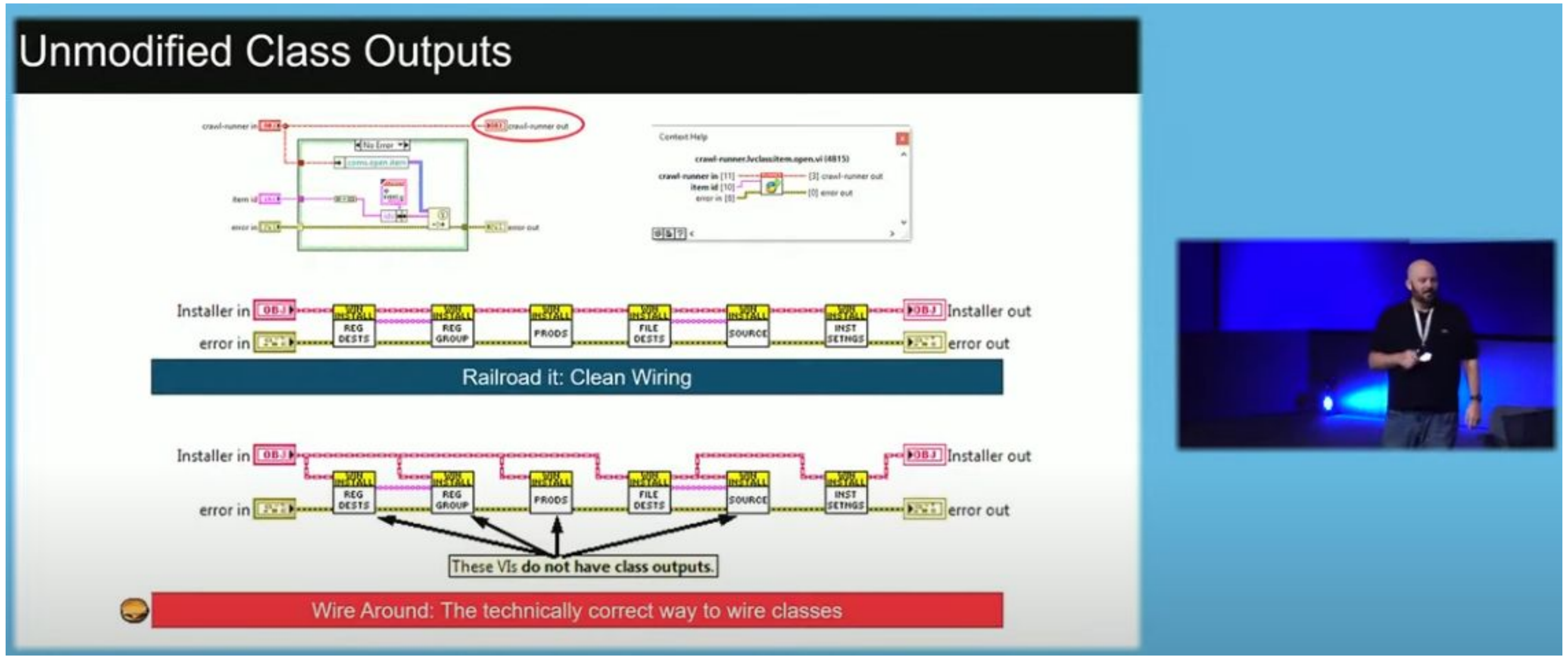
- "List Buildup"
  - *Hidden Gems in vi.lib*
  - *An End to Brainless LabVIEW Programming*
  - *Ludicrous Ways to Fix Broken LabVIEW Code*
  - *Quick! Drop Your VI Execution Time!*
- "Colleague Enrichment"
  - *Introduction to DQMH*
  - *All About Collection Data Types*
- "DNatt Stuff"
  - *I Find Your Lack of LabVIEW Programming Speed Disturbing*
  - *Don't Wait for LabVIEW R&D… Implement Your Own LabVIEW Features!*
  - *Improving Your LabVIEW Code with the VI Analyzer*

EMERSON. | ni

The Catalyst for *This* Presentation

- “A passing comment I heard during someone else’s presentation that made me mad”
- Extreme LabVIEW Style Showdown – Hunter Smith vs. Tom McQuillan
  - GDevCon #4, Glasgow UK 2023

The Catalyst for *This* Presentation


Unmodified Class Outputs

Railroad it: Clean Wiring

These VIs do not have class outputs.

Wire Around: The technically correct way to wire classes

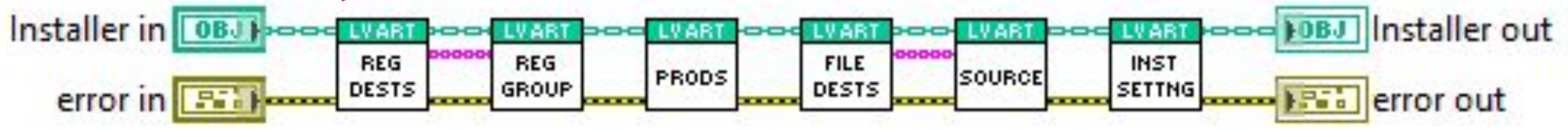
(at the 15:00 mark)

“These images are from Darren Nattinjer’s [sic] ‘End to Brainless Programming’, where he’s, of course, selected the brainless wire color… (ha ha ha)

I will concede that the bottom approach of not passing [unmodified class values] out is the more technically correct, ***but at the expense of your code readability****.” -- Hunter Smith*

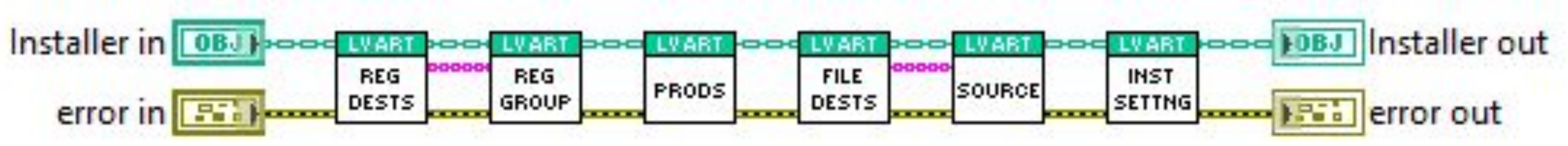
The Catalyst for *This* Presentation

- This code is **~~readable~~?**

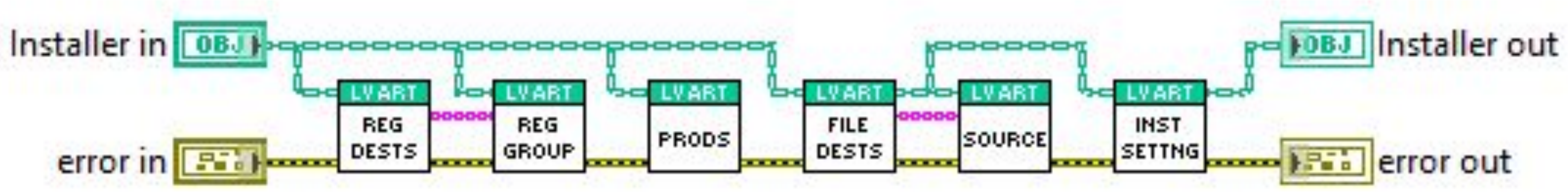


The Catalyst for *This* Presentation

- This code is **pretty**.



- This code is **readable**.



tl;dr of This Whole Presentation

- The most important thing your code can be is **functional.**
- The 2nd most important thing your code can be is **readable.**
- …
- The *n*th most important thing your code can be is **pretty**.

- **Readable code** is more important than **pretty code**.

- We need to talk about **readability** more than we talk about **style**.

What is "Readability"?

Definition of Code Readability

Readability is the attribute of your code by which a developer can **understand functionality and behavior**.

The more **readable** your code, the more efficient a developer will be when she *augments*, *troubleshoots*, and/or *refactors* your code.

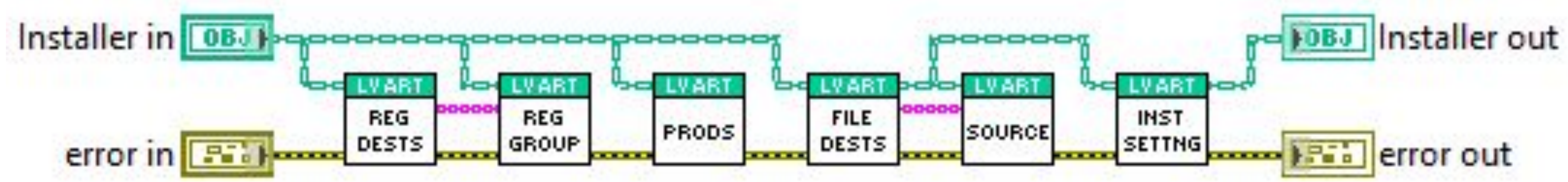
The “developer” could be someone else, or it could be you.

DevX: **Developer Experience**

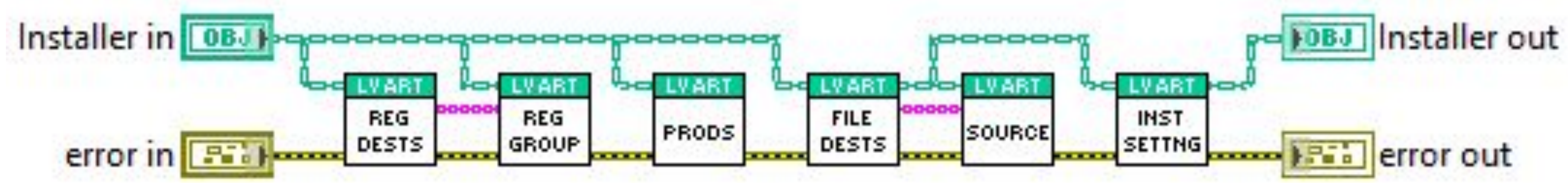
Isn’t “Readability” subjective?

DNatt vs. Norm!

- DNatt: “This code is **readable**”.



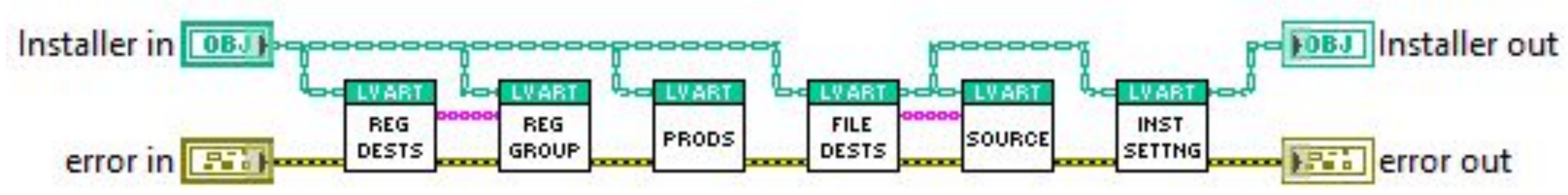
- Norm!: “This code is **noisy and distracting**”.



Dr. T to the Rescue!

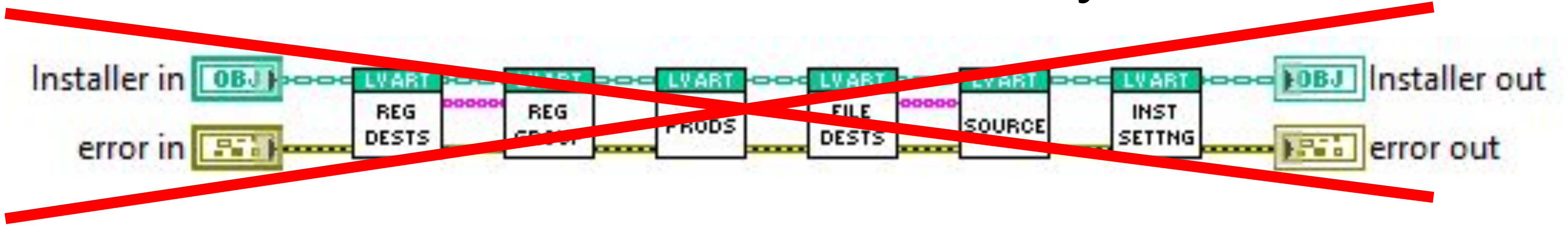
- THE GENIUS OF THE “AND”

- DNarm!: “This code is **readable, noisy, and distracting**”.

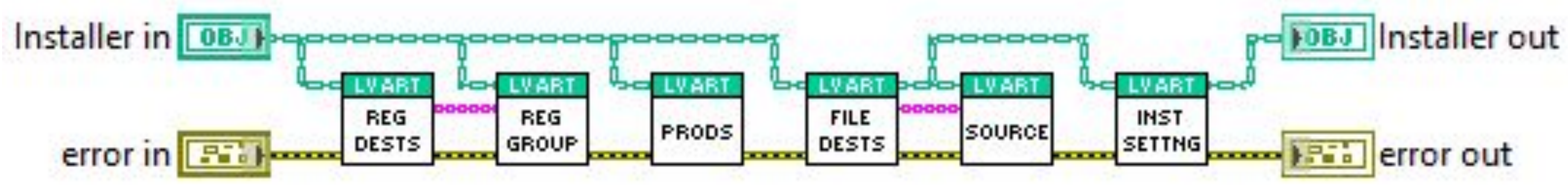


Where Does That Leave Us?

- As we create block diagrams, we need to make continual choices that **maximize readability** and **minimize distraction**.
- Zero distractions, mediocre readability



- Minor distractions, **maximum** readability



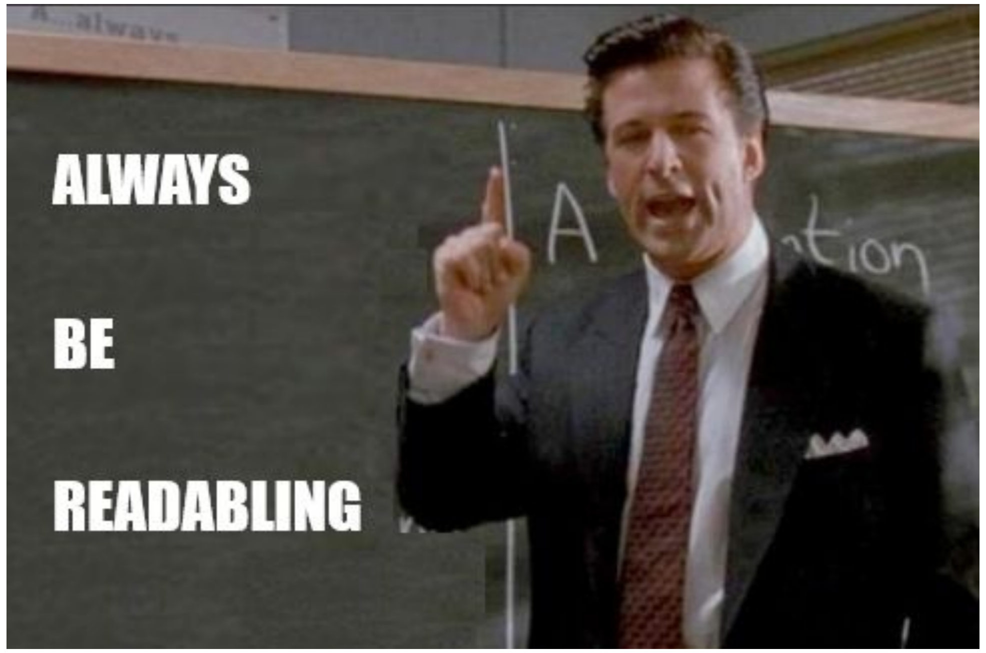
How do I write readable code?

How to Write Readable Code

With **every block diagram object** you create, ask yourself:

“Is there something I can do to make this code more readable?”

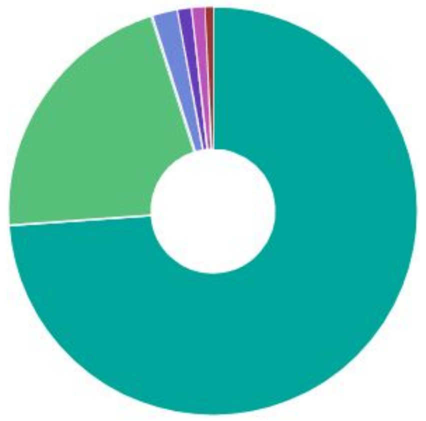
ABR:



Finally! The Outline of the Presentation!

- Maximize readability in the ways that you:
  - **Name objects**
  - **Utilize text**
  - **Craft block diagrams**

Two Disclaimers



The content in this presentation best applies when:

- Your LabVIEW team members all use the same written language when developing code.
- Members of your LabVIEW development team are not color blind.

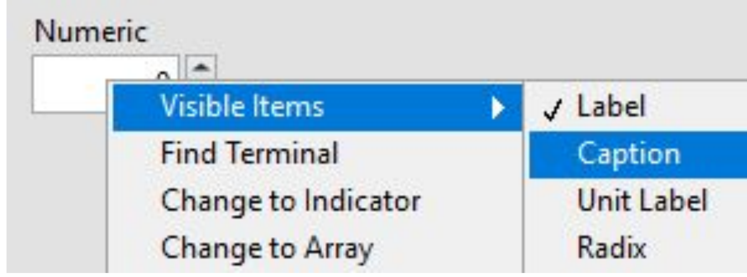
Firehose Incoming

https://bit.ly/labviewreadability

(This slide deck has 74 slides…)

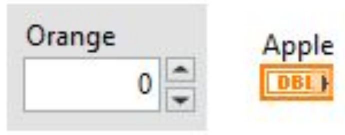
Content Warning

NAMING Objects

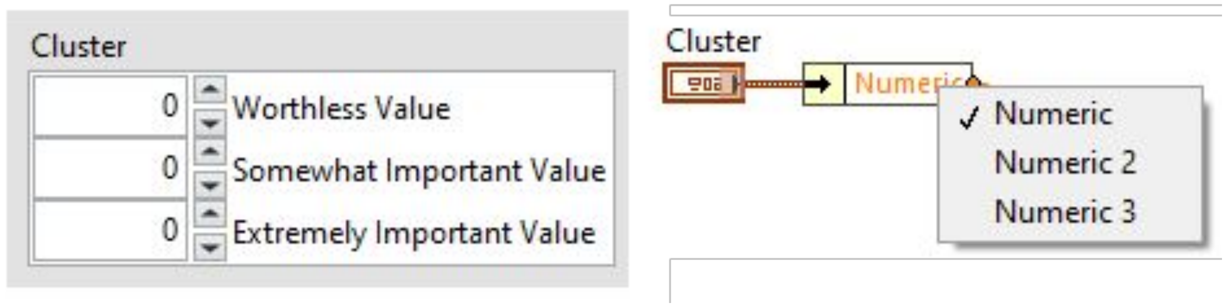


Minimize Use of Captions

- Captions break the *panel -> diagram* mapping for controls/indicators



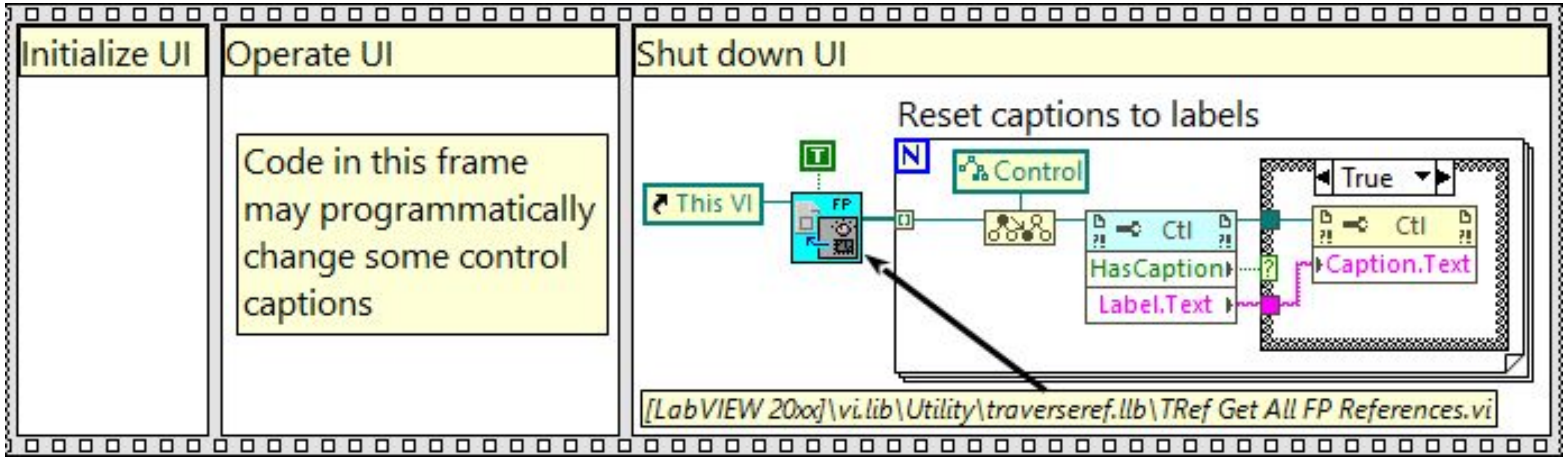
- Captions on cluster elements destroy readability



- "Caption on panel so label uses less space on the diagram" = 👎
  - Remember, we are maximizing **readability**, not horizontal density

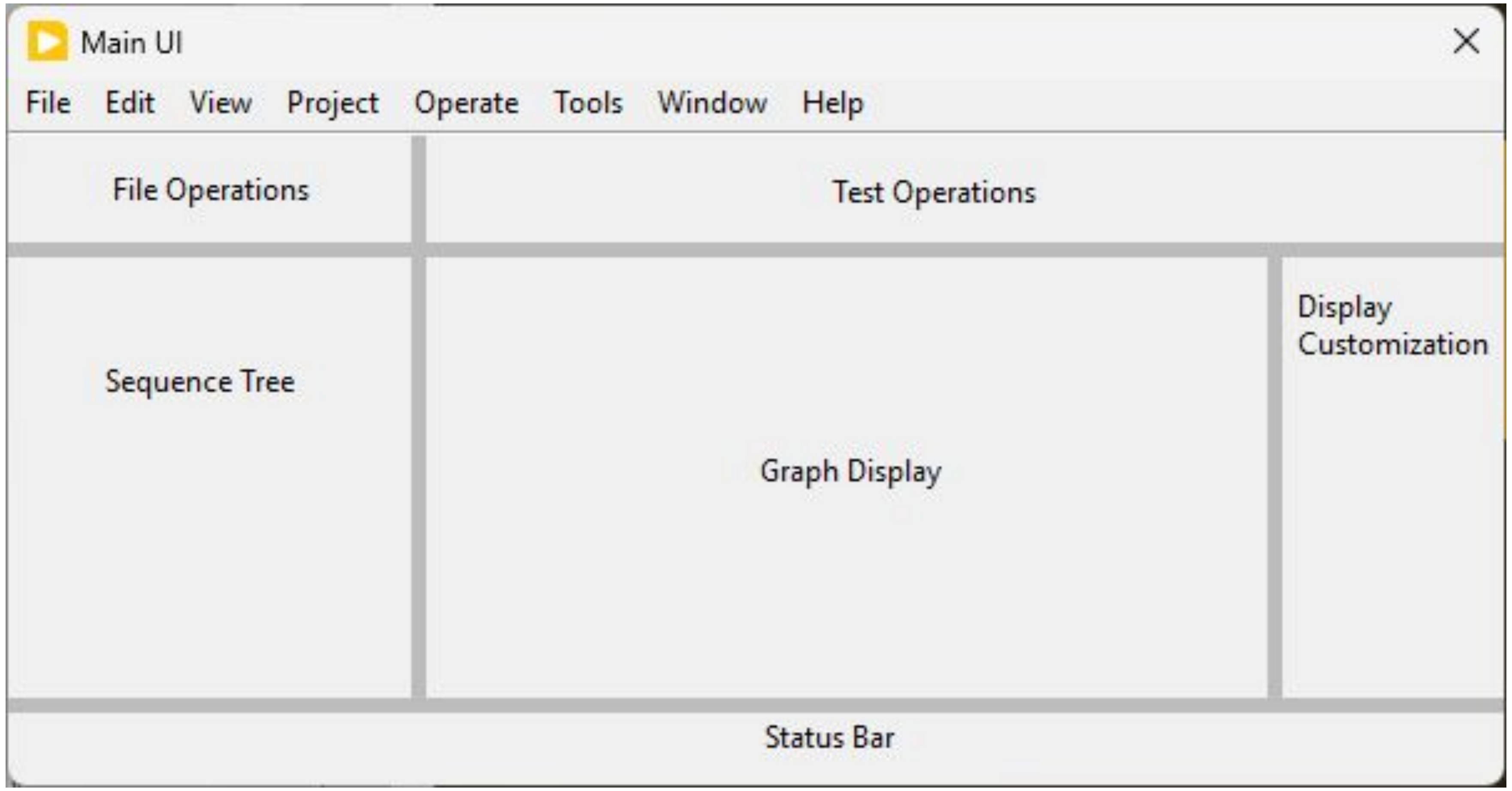
Improve DevX when Using Captions Legitimately

- If you are programmatically manipulating captions for UI controls, restore them to the label value at the end of execution
- This helps when you are editing the VI after running it


Initialize UI
Operate UI
Code in this frame may programmatically change some control captions
Shut down UI
Reset captions to labels
This VI
FP
N
T
Control
Ctl
HasCaption
Label.Text
True
Ctl
Caption.Text
[LabVIEW 20xx]\vi.lib\Utility\traverseref.llb\TRef Get All FP References.vi

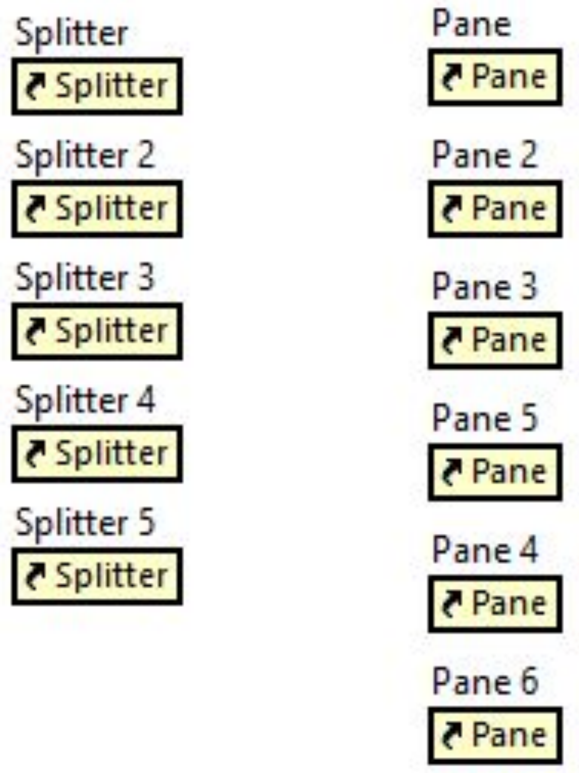
Splitters and Panes

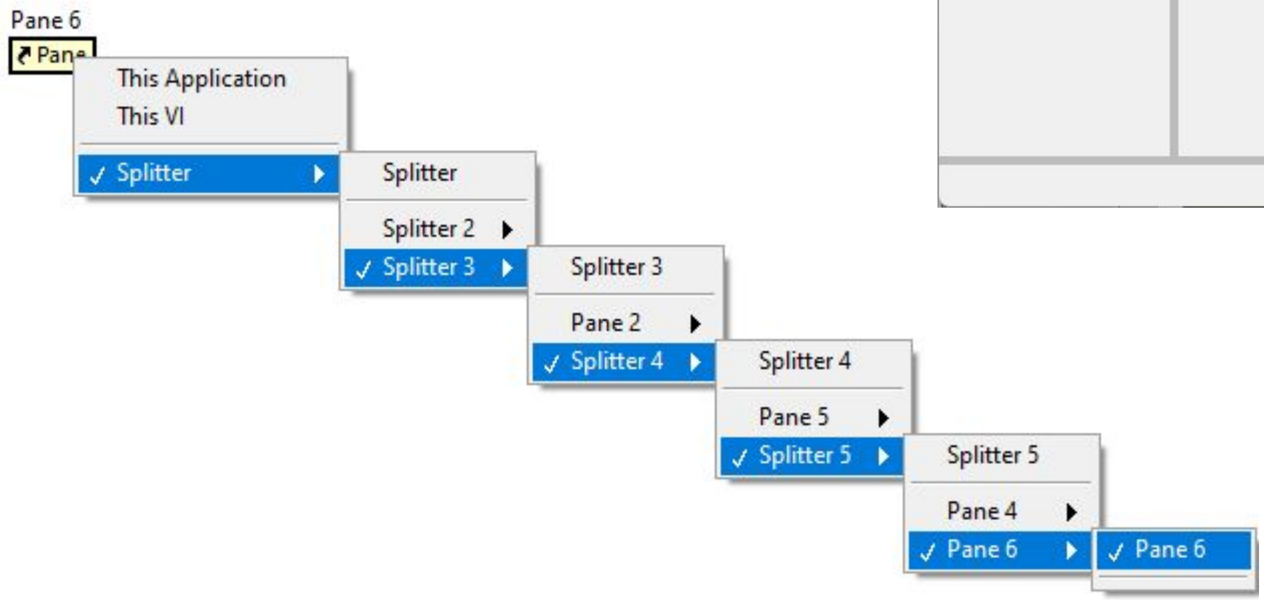
- Splitters are invaluable for developing sophisticated, resizable UIs

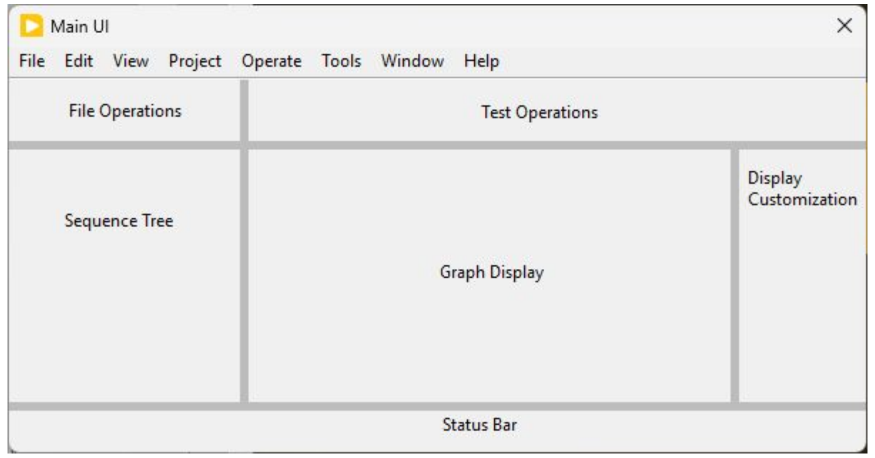


Developing with Splitters and Panes

- Splitters and Panes have a suboptimal DevX out of the box.

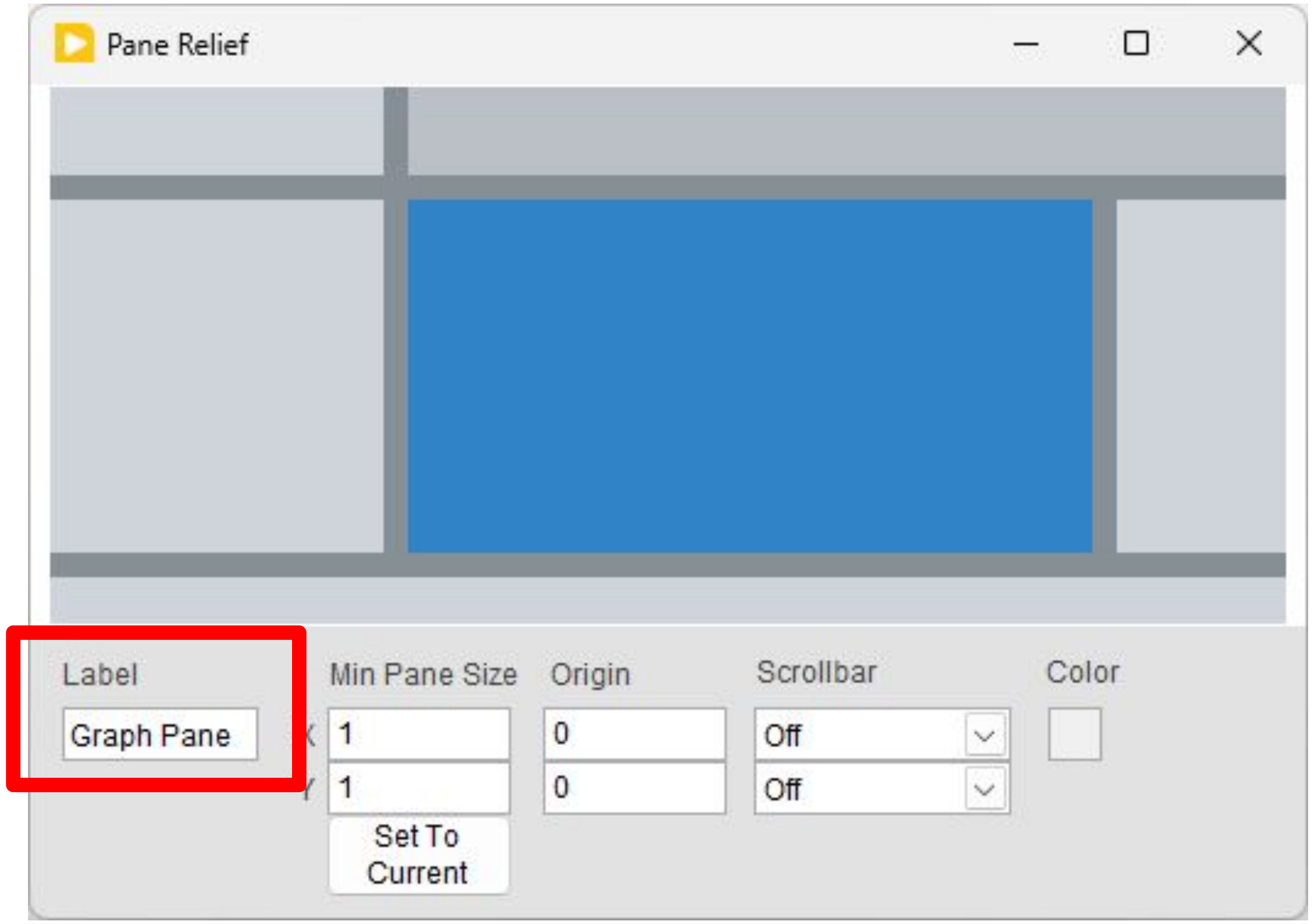






Name Splitters and Panes

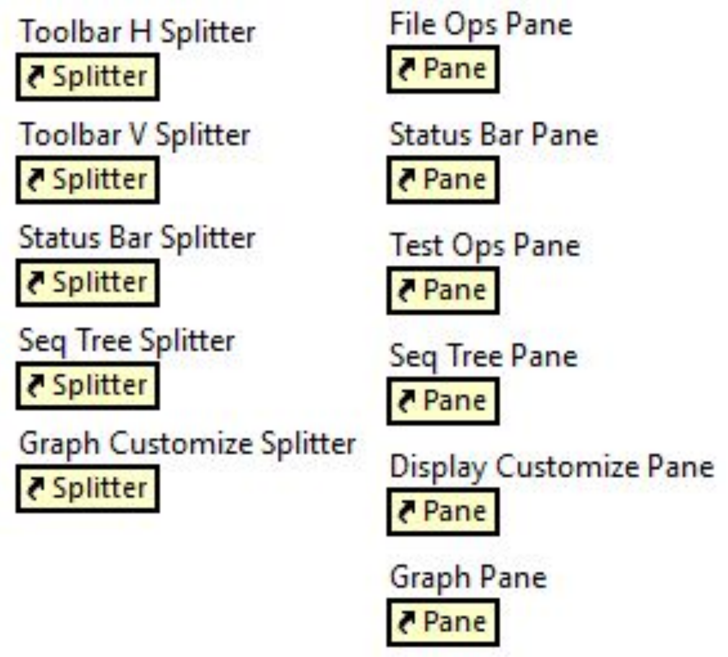
- For multi-pane UIs, take the time to name **all** your splitters and **all** your panes
- Use PaneRelief as a convenient naming tool



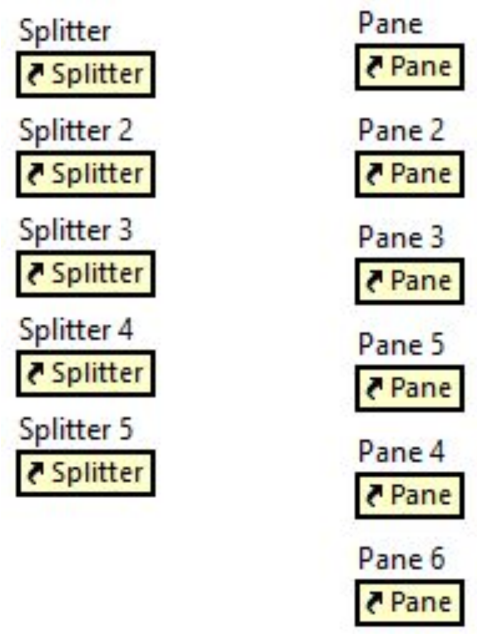
Label: Graph Pane
Min Pane Size: X 1, Y 1
Origin: 0, 0
Scrollbar: Off, Off

Name Splitters and Panes

- Code is much more readable with well-named Splitters and Panes



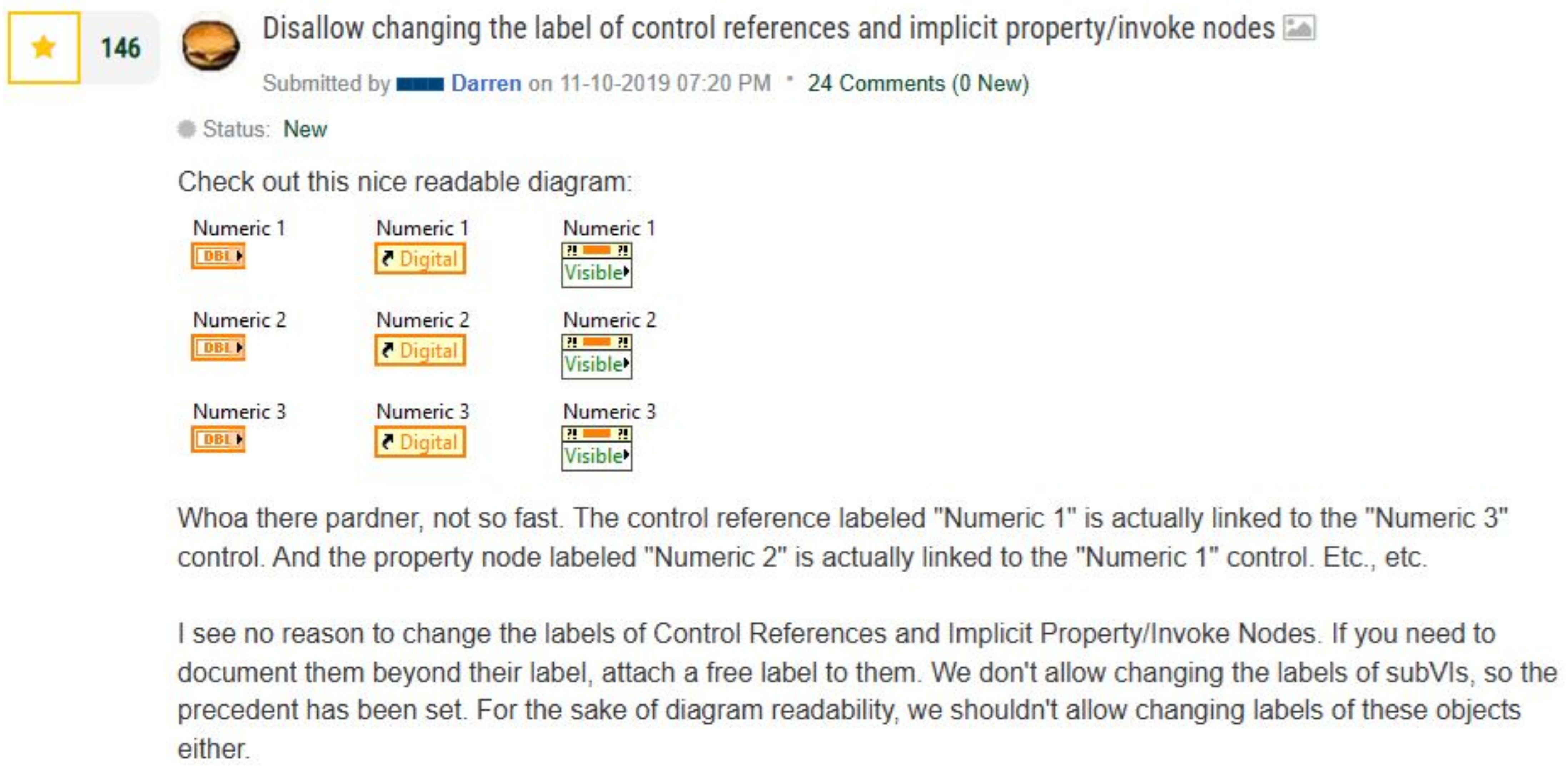
>>



- Use PaneRelief *before* creating Pane references!
  - Labels of existing references don’t update ☹

Aside: Kudo this idea

- Disallow changing the label of control references and implicit property/invoke nodes



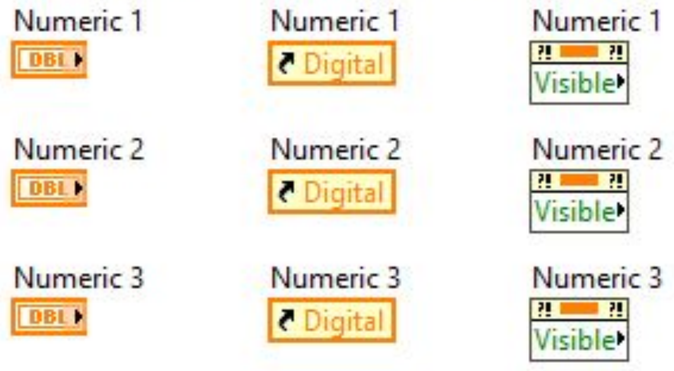
146

Disallow changing the label of control references and implicit property/invoke nodes

Submitted by [redacted] Darren on 11-10-2019 07:20 PM • 24 Comments (0 New)

Status: New

Check out this nice readable diagram:

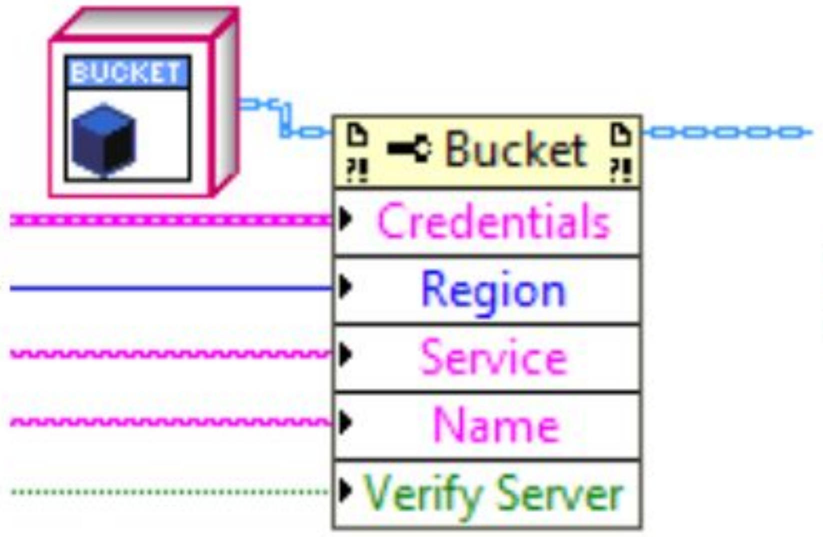


Whoa there pardner, not so fast. The control reference labeled "Numeric 1" is actually linked to the "Numeric 3" control. And the property node labeled "Numeric 2" is actually linked to the "Numeric 1" control. Etc., etc.

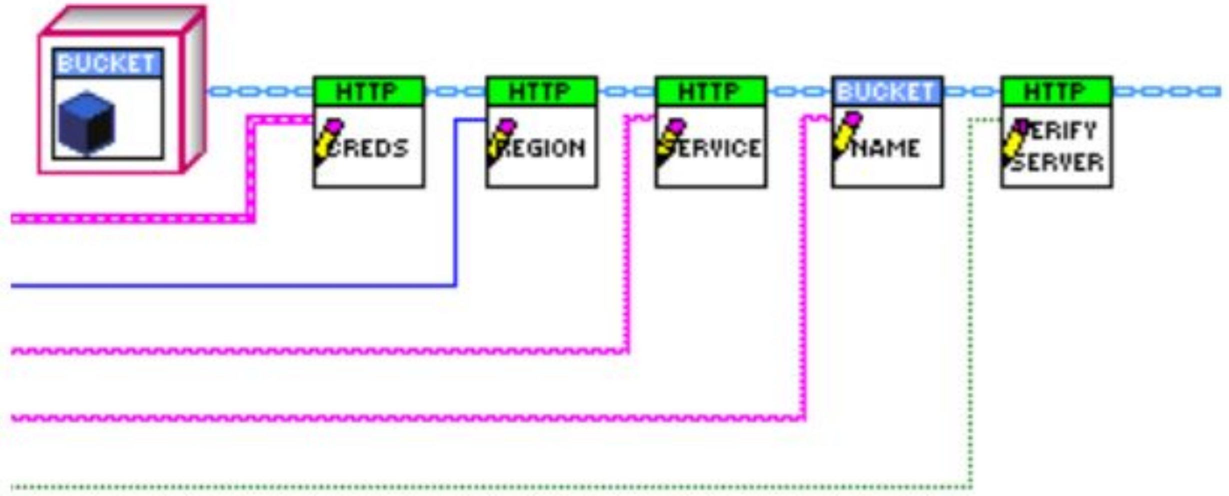
I see no reason to change the labels of Control References and Implicit Property/Invoke Nodes. If you need to document them beyond their label, attach a free label to them. We don't allow changing the labels of subVIs, so the precedent has been set. For the sake of diagram readability, we shouldn't allow changing labels of these objects either.

LabVIEW Class Properties

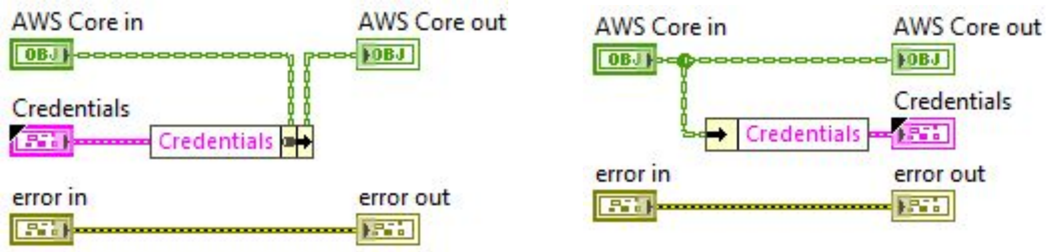
- Class Property Nodes are more readable than Accessor SubVIs
  - https://bit.ly/ludicrouslabview



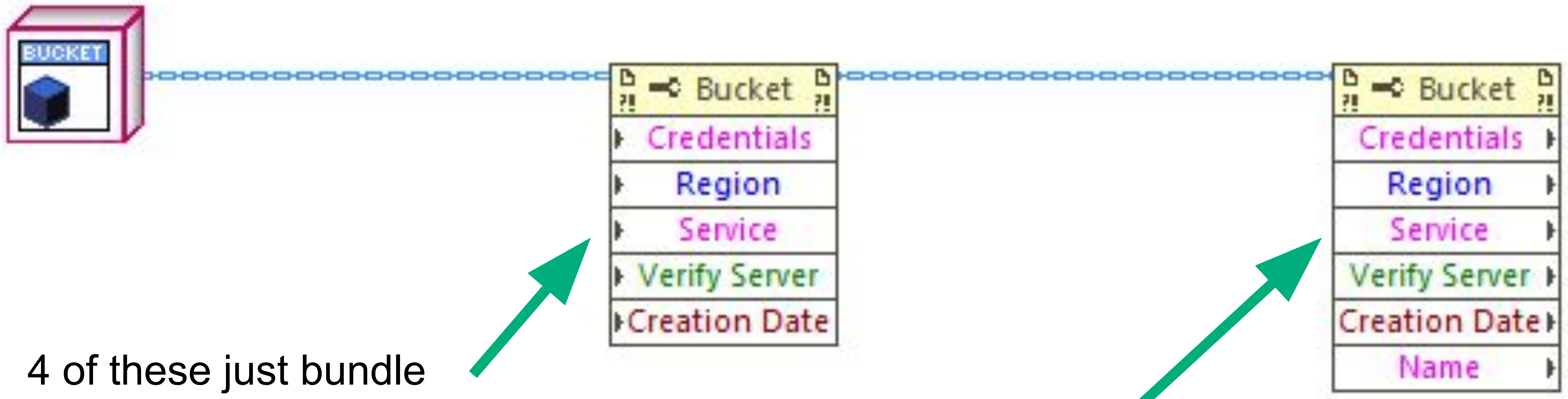
>>



LabVIEW Class Properties



- Class Property VIs should only bundle/unbundle class data…
- … until they shouldn't.

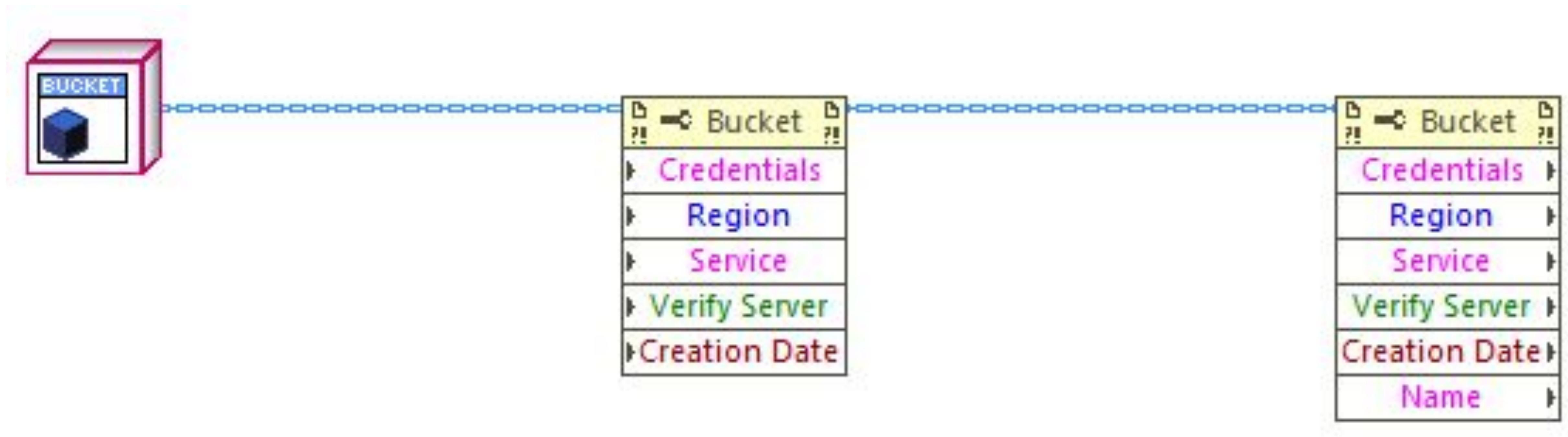


4 of these just bundle
1 of these has an extra bundle

5 of these just unbundle
1 of these does a bunch of extra stuff

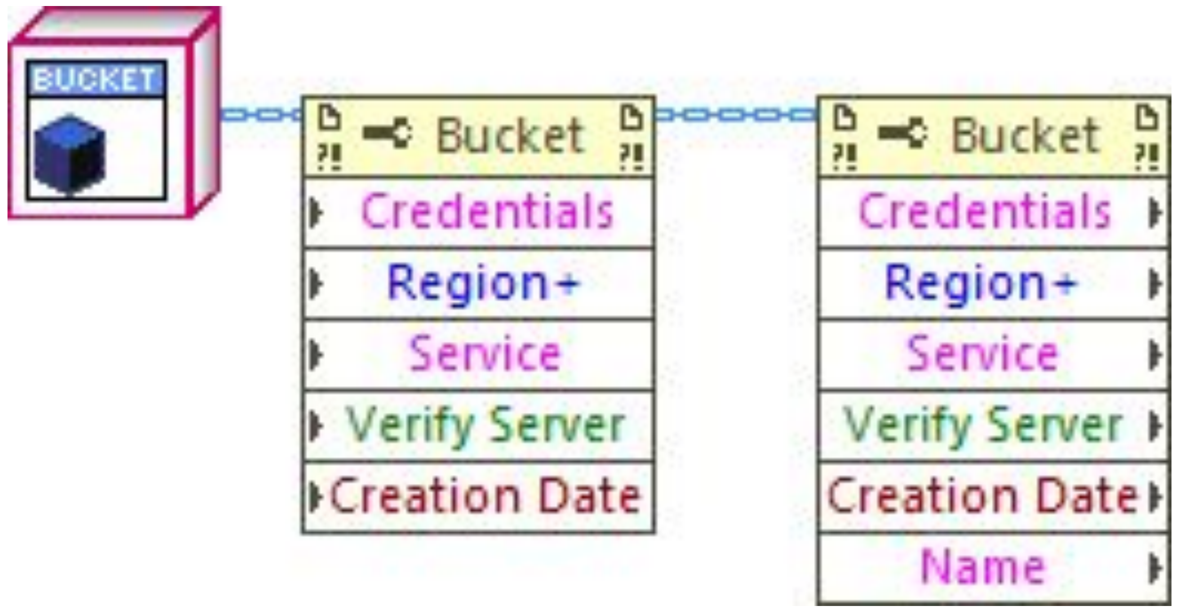
LabVIEW Class Properties DevX

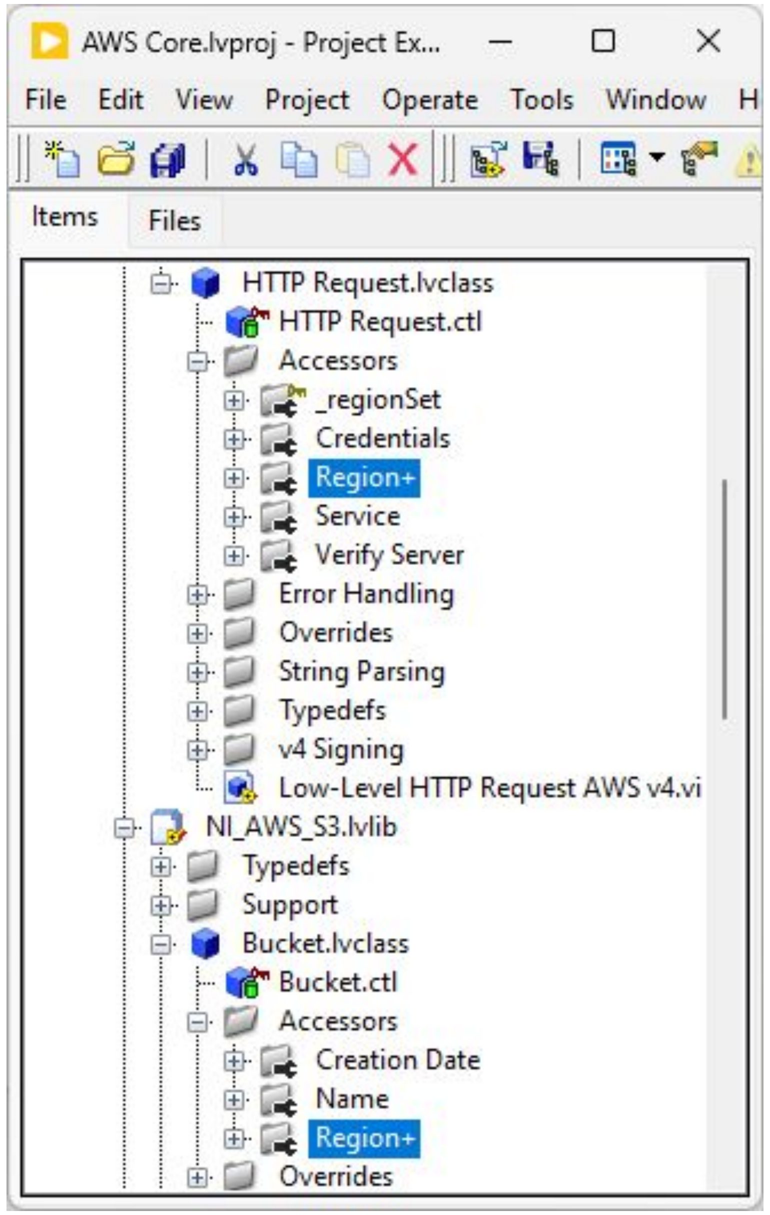
- As a **user** of this API, you shouldn't care what the Property VIs are doing under the hood.
- As a **debugger** of this API, you **absolutely do care** what the Property VIs are doing under the hood.



Experimental Idea: Use “+” in Prop Folder Name

- Simple, visual, readable indication that a class property VI does “extra” stuff
- Try it out, let me know what you think



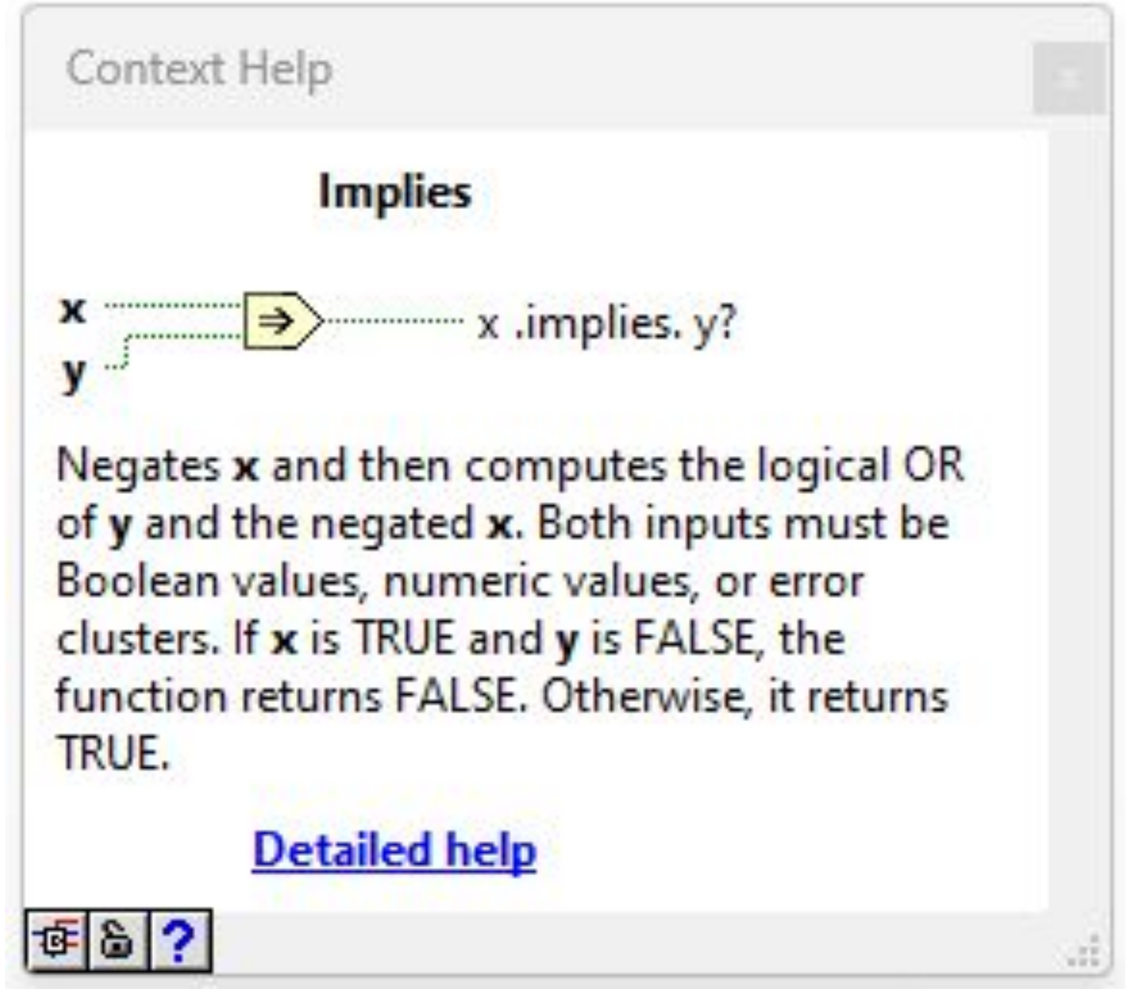

AWS Core.lvproj - Project Ex...
File Edit View Project Operate Tools Window H
Items Files

```
HTTP Request.lvclass
    HTTP Request.ctl
    Accessors
        _regionSet
        Credentials
        Region+
        Service
        Verify Server
    Error Handling
    Overrides
    String Parsing
    Typedefs
    v4 Signing
    Low-Level HTTP Request AWS v4.vi
NI_AWS_S3.lvlib
    Typedefs
    Support
    Bucket.lvclass
        Bucket.ctl
        Accessors
            Creation Date
            Name
            Region+
        Overrides
```

Utilizing TEXT

Context Help

- The Context Help Window is a great tool!


Context Help

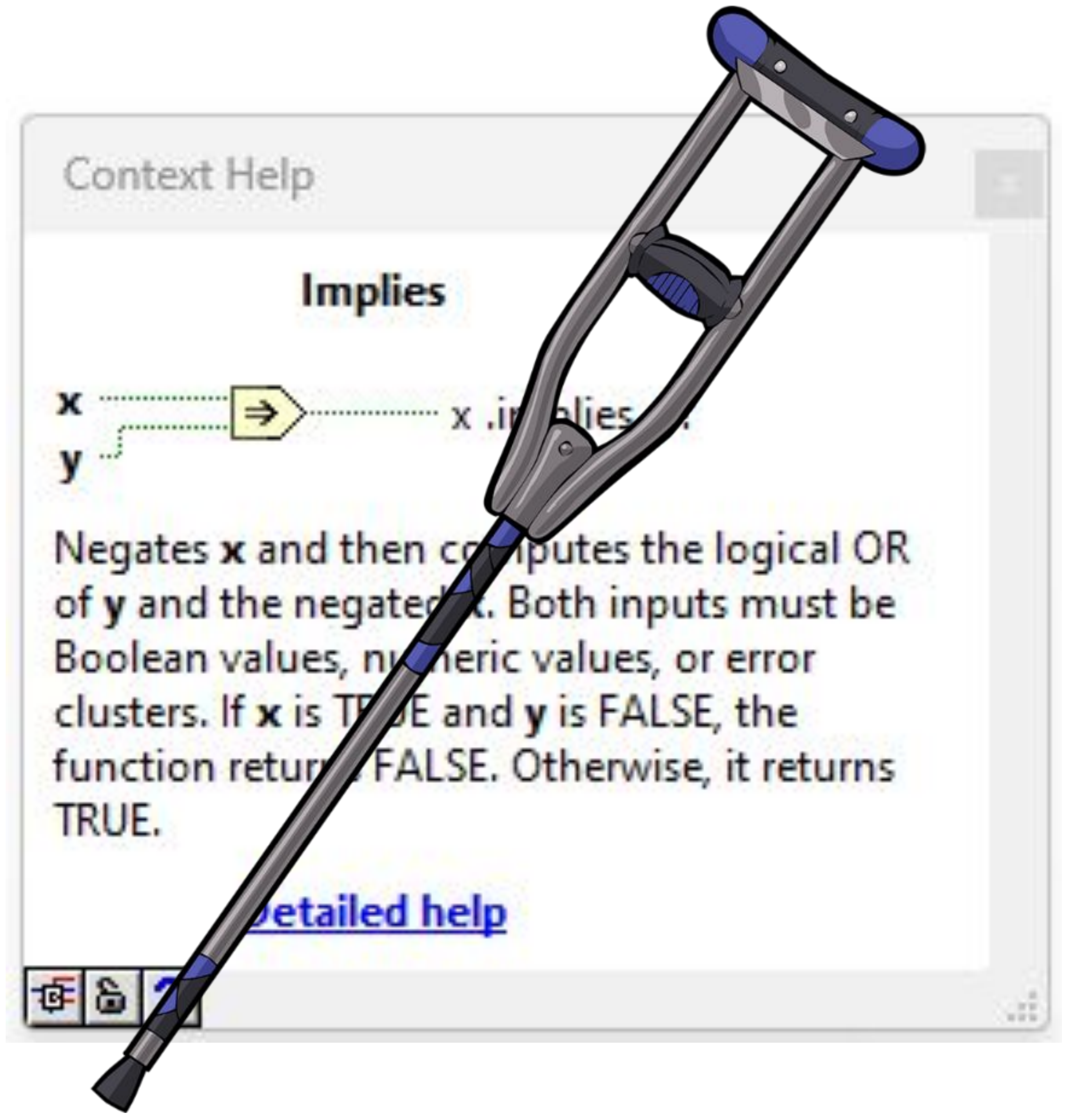
Implies

Negates **x** and then computes the logical OR of **y** and the negated **x**. Both inputs must be Boolean values, numeric values, or error clusters. If **x** is TRUE and **y** is FALSE, the function returns FALSE. Otherwise, it returns TRUE.

Detailed help

Context Help

- The Context Help Window is a CRUTCH!



Negates **x** and then computes the logical OR of **y** and the negated **x**. Both inputs must be Boolean values, numeric values, or error clusters. If **x** is TRUE and **y** is FALSE, the function returns FALSE. Otherwise, it returns TRUE.

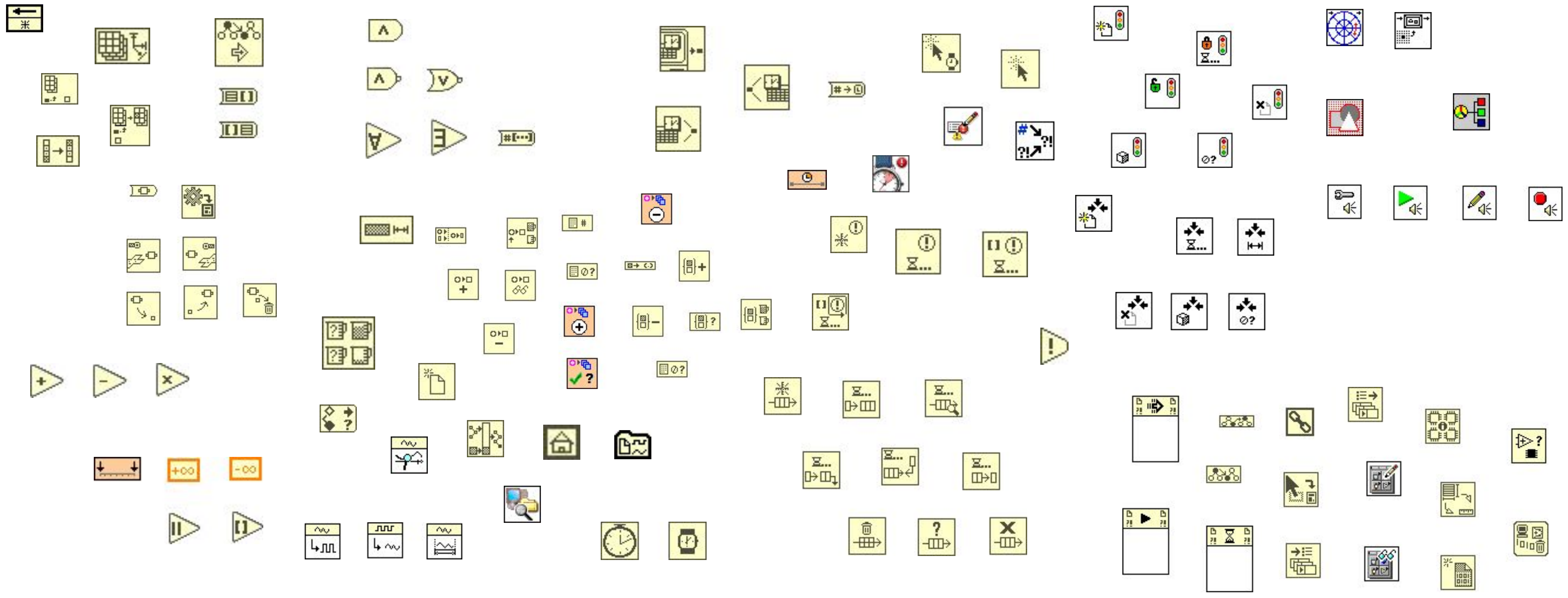
Detailed help

Context Help

- If you are using the Context Help Window, it's because you **need help** reading a block diagram.
- If you are using the Context Help Window, it is because the block diagram **insufficiently conveys** its behavior.
- Said another way, the diagram is **not readable**.
- I'm not saying we should never use Context Help.
- I am saying that the parts of drawing the diagram **that we control** should **minimize** the need to use Context Help.

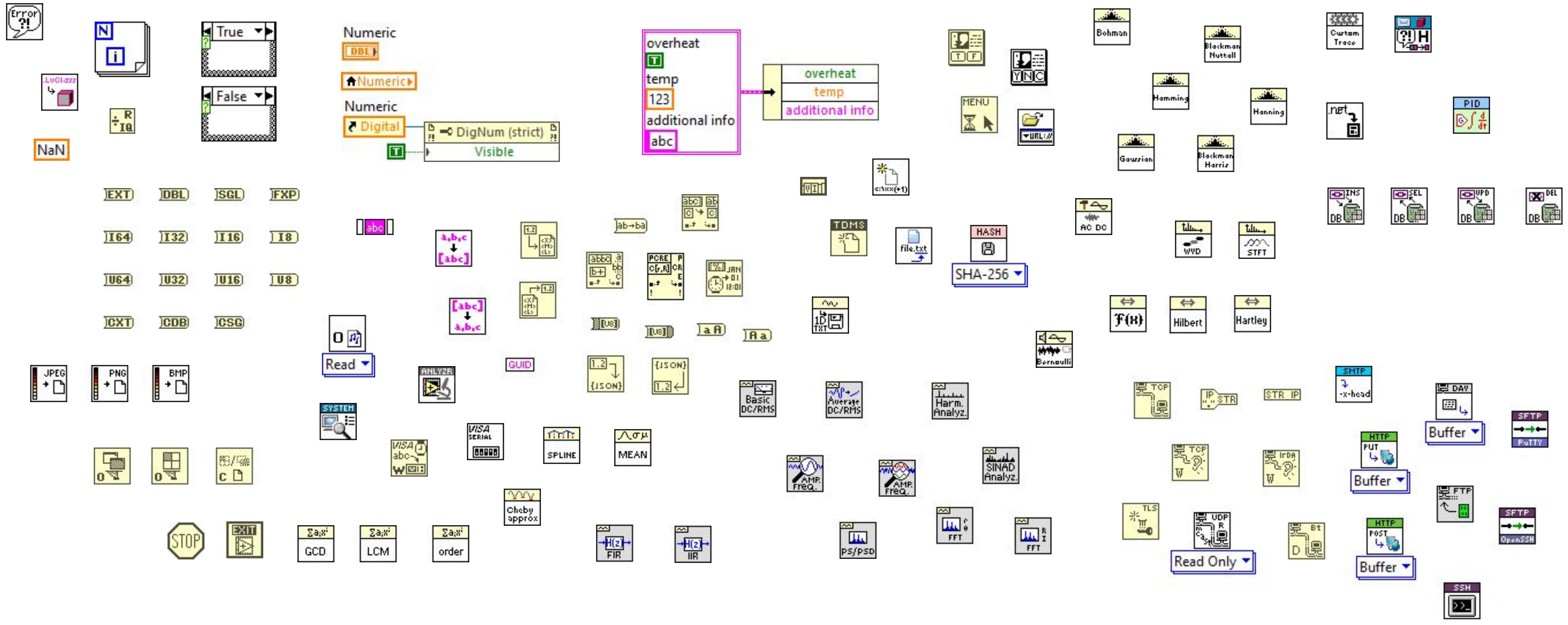
Graphical Programming

For a graphical language, we obviously use symbols a lot.



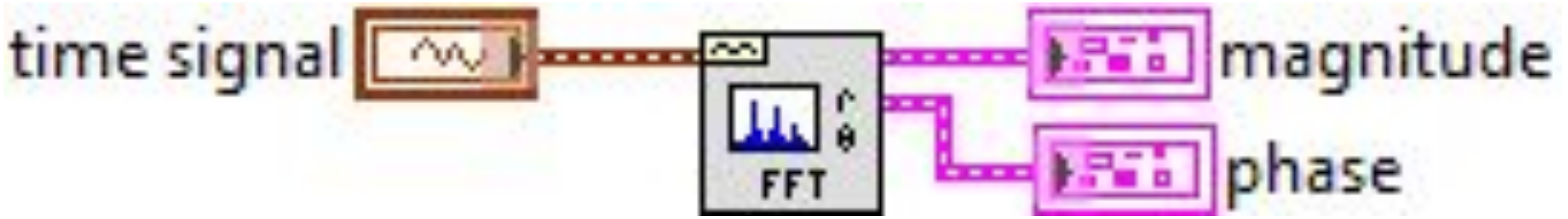
Graphical Programming?

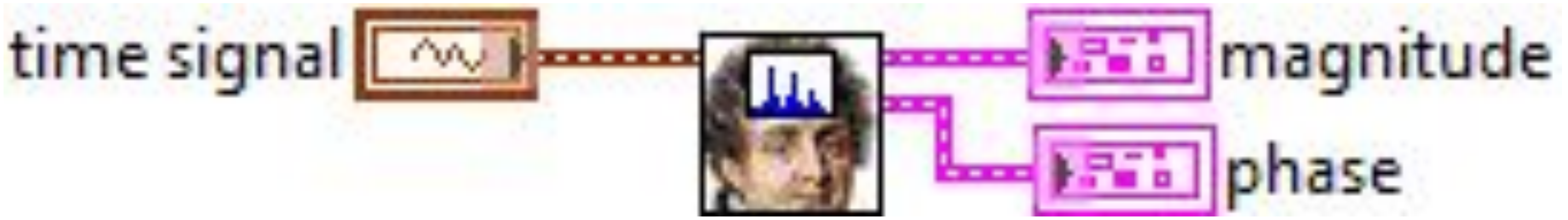
For a graphical language, we also use words and letters a lot.



Graphical Programming… Now With Text!

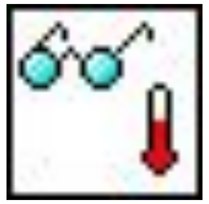
- And that's a good thing!





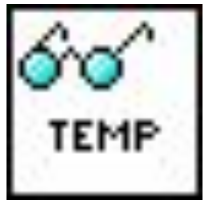
So Many Options!

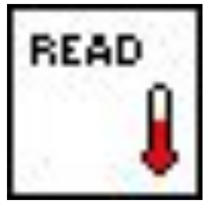
- A completely graphical icon?



- A completely textual icon?

- Somewhere in between?





It’s Time For Everybody’s Favorite Game…

It’s Time For Everybody’s Favorite Game…

HOW MANY
PEOPLE CAN I
OFFEND WITH
ONE SLIDE?
WHEEL OF FORTUNE
GDEVCON NA 2024
THE WORD FINDER

How Many People Can I Offend With One Slide?

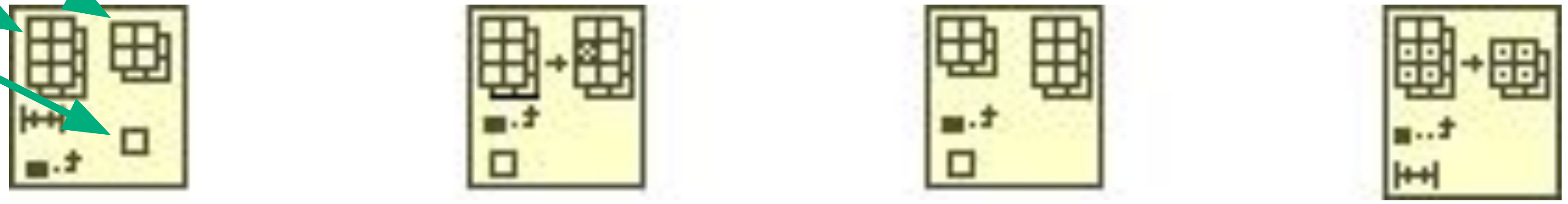
DELETE FROM ARRAY

REPLAC ARRAY SUBSET

INSERT INTO ARRAY

ARRAY SUBSET

… are more readable than …

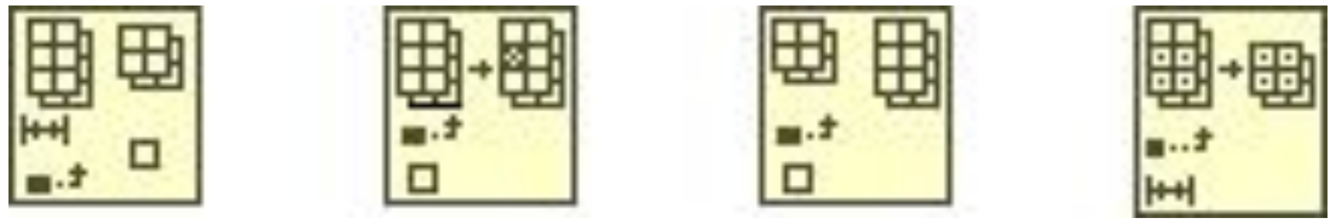


You don’t need Context Help to know what those subVIs are doing.

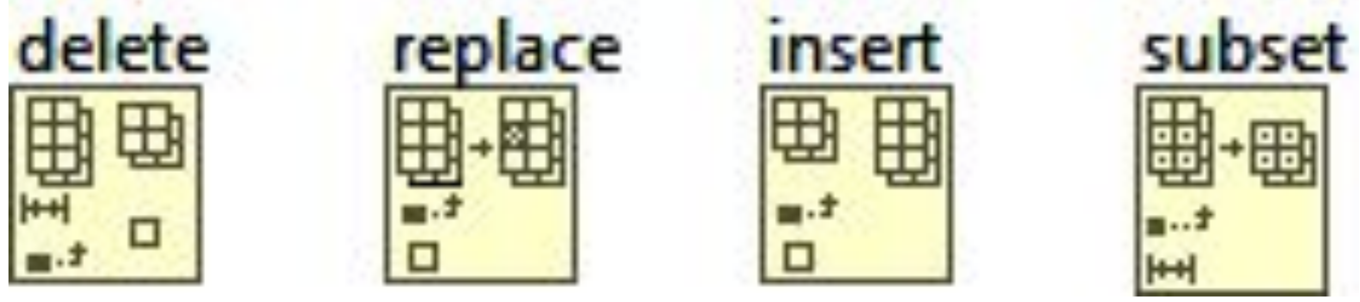
You don’t need Art Appreciation class to know what those subVIs are doing.

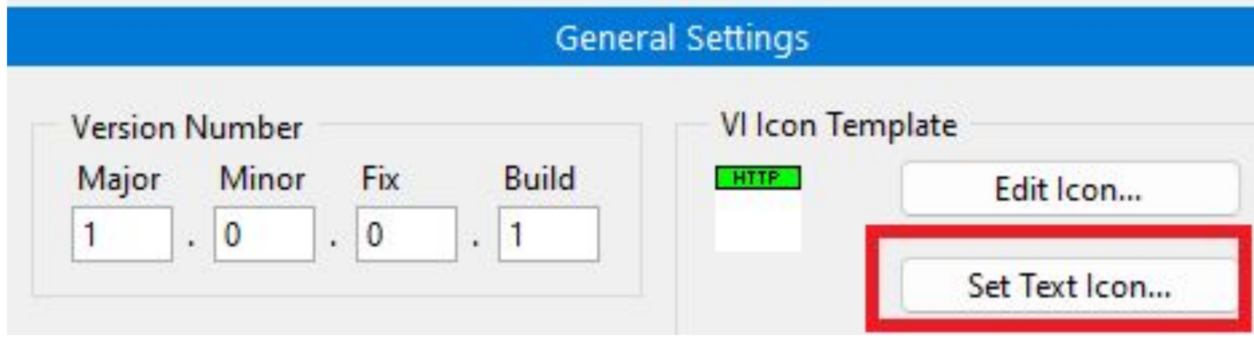
Function Icons

- We can’t change the way built-in functions and subVIs are drawn.



- But we can improve DevX with labels where it makes sense.




General Settings

Version Number

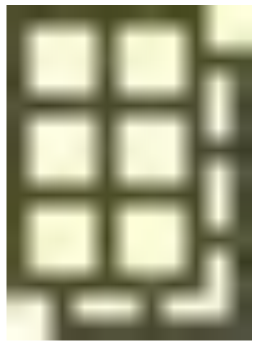
Major	Minor	Fix	Build
1	0	0	1

VI Icon Template

For the VIs We Write...

Use **text-based banners** to relate groups of VIs.

= Array



= Array? Matrix? Grid? Window? Building?

Use **text-based icon body text** to describe VI functionality.

ARRAY
DELETE

ARRAY
REPLAC

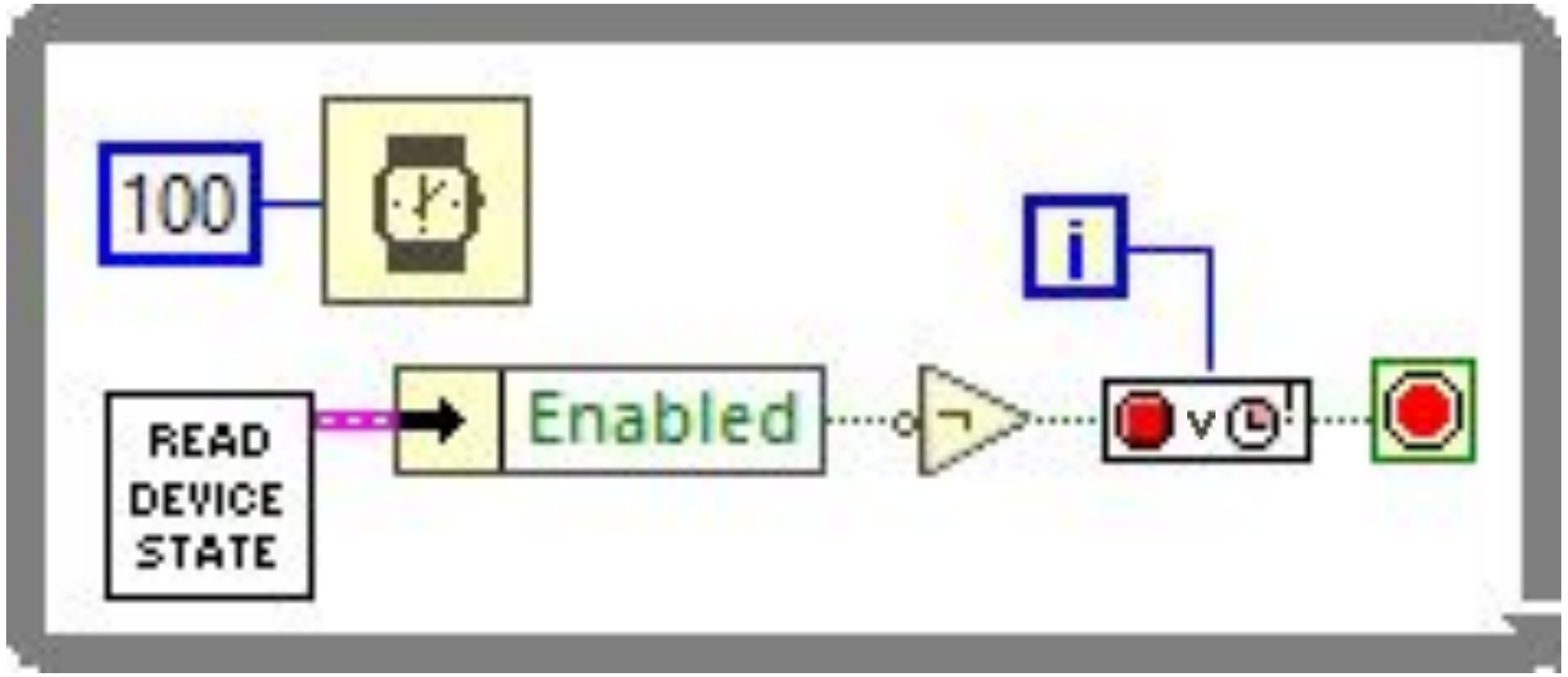
ARRAY
INSERT

ARRAY
SUBSET

1. *Ctrl-Space*
2. *[type text]*
3. *Ctrl-K*

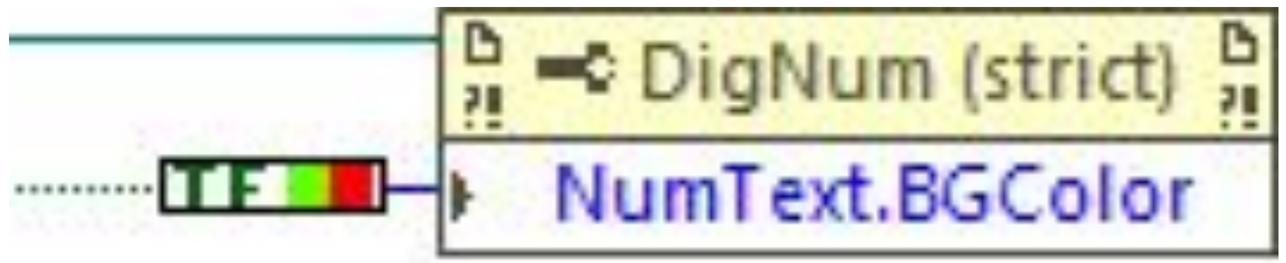
For the *skinny* VIs We Write...

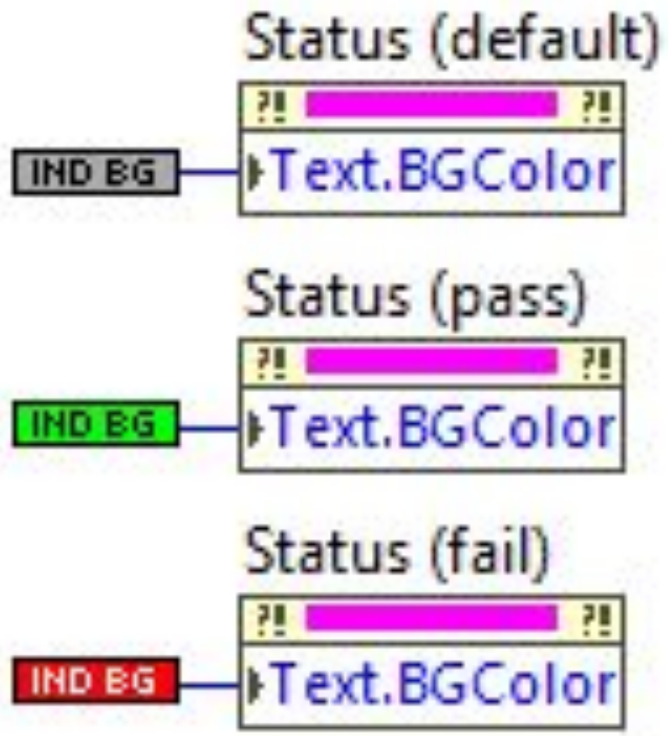
Graphical icons *sometimes* convey functionality more effectively than text.

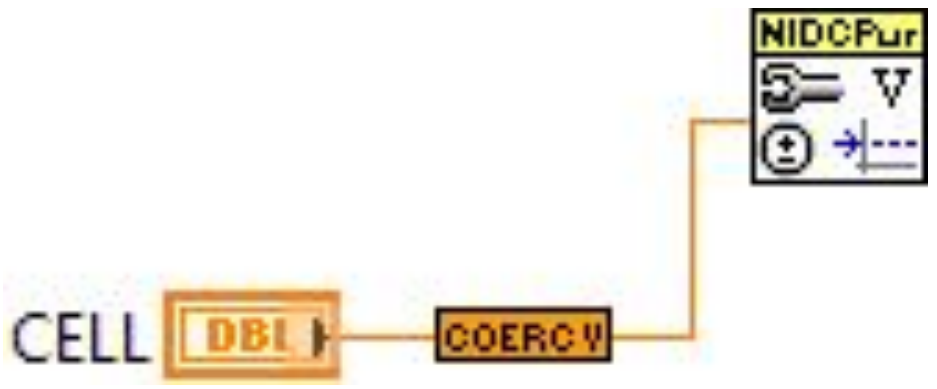


For the *skinny* VIs We Write…

…and sometimes they don't.



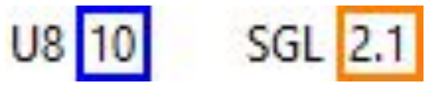




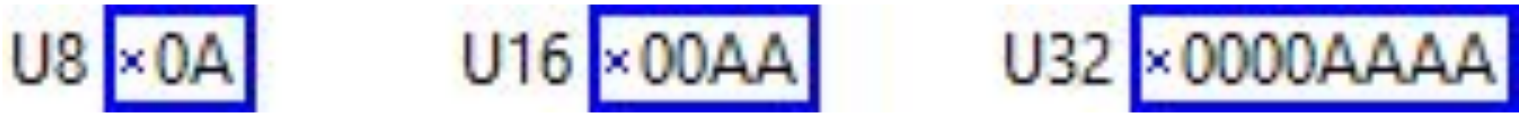
CRAFTING Block Diagrams

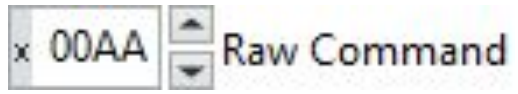
Readable Numeric Constants

- The “default” numeric constant types are **DBL** and **I32**
- If your numeric constant is **any other data type**, label it:



- Format hex numerics in the most readable manner:

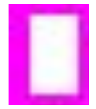


  - Show radix, pad with zeros on left, minimum field width of 2, 4, or 8
  - This goes for the front panel as well 

  - Use the Quick Format Quick Drop Keyboard Shortcut

Readable String Constants

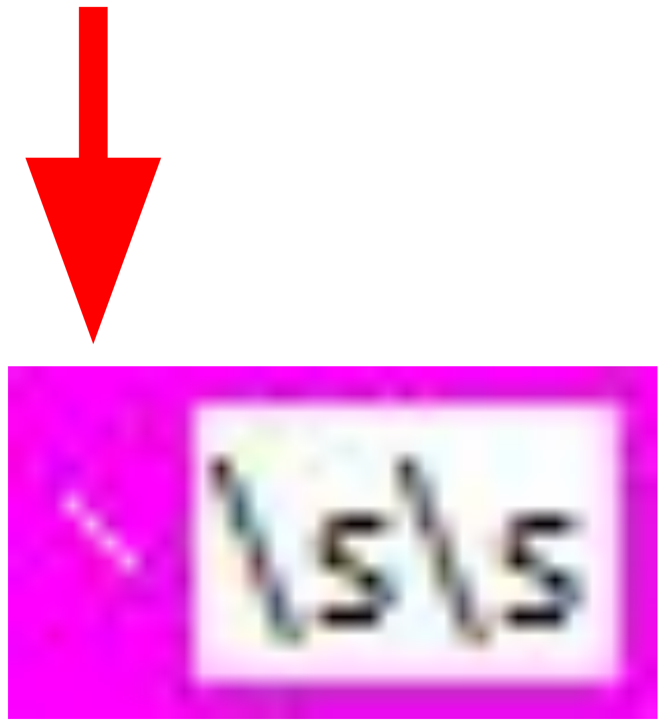
- What is the value of this string constant?

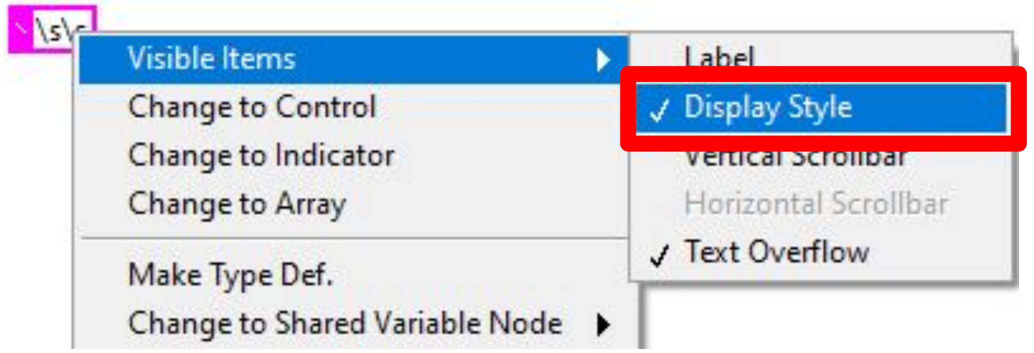


- What is the value of this string constant?

\s\s

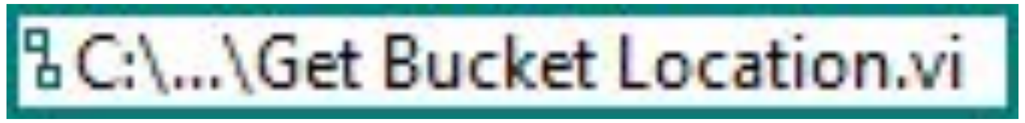
- Follow these general guidelines:
  - Empty string: “”
  - String containing **only** human-readable text: File not found.
  - Format string containing whitespace: %d\s%f\s%s\n
  - String containing **only** whitespace: \s\s \n
  - Hex string (usually for device comms): DEAD BEEF
  - Use the Quick Format Quick Drop Keyboard Shortcut





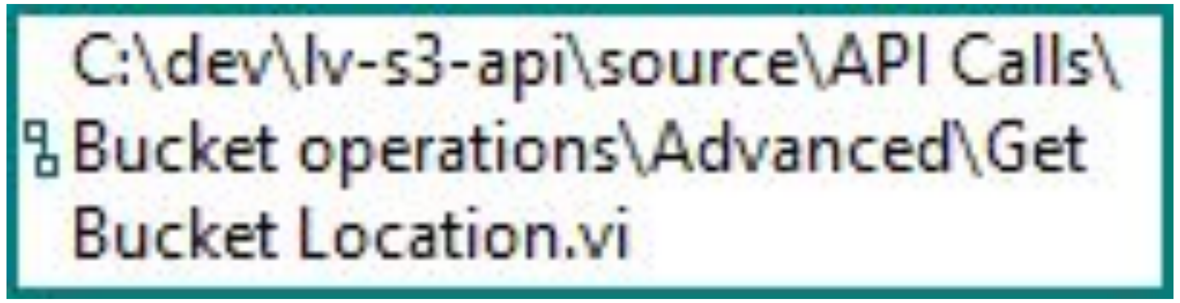
Readable Path Constants

- I know the VI file name, and that it's on my C: drive… and not much else:



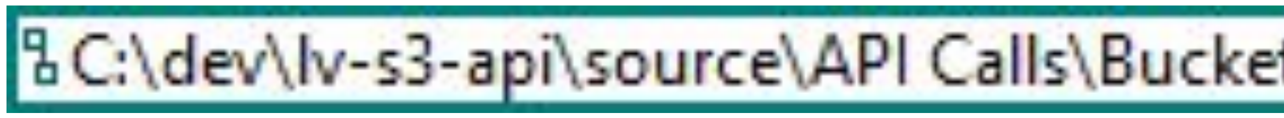
C:\...\Get Bucket Location.vi

- Expand the path constant to display the entire path… don't be afraid to make it multi-line:



C:\dev\lv-s3-api\source\API Calls\
Bucket operations\Advanced\Get
Bucket Location.vi

>>

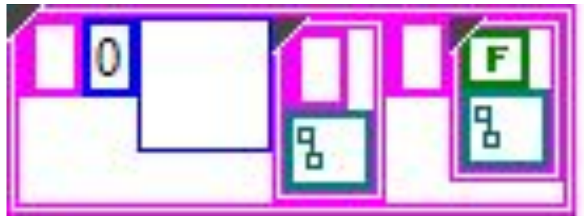


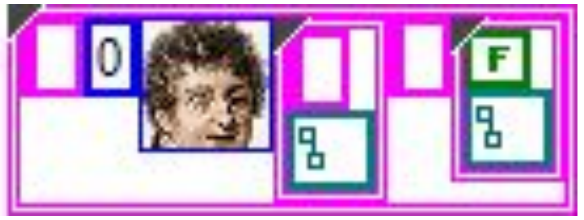
C:\dev\lv-s3-api\source\API Calls\Bucke

(this goes for relative path values as well)

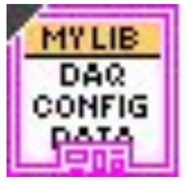
Readable Cluster Constants

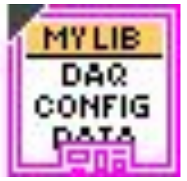
- Which of these cluster constants contains a non-default value?



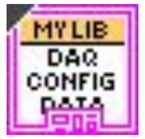


- Which of these cluster constants contains a non-default value?

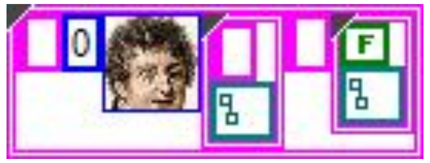




- When defining a cluster *data type*, use **Icon View**

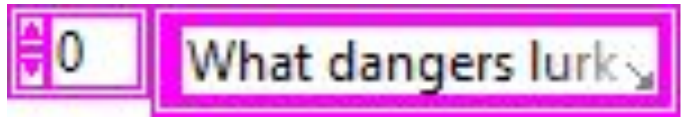


- When defining a cluster *value*, use **Default View**

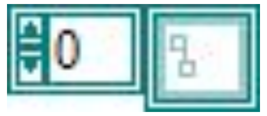


Readable Array Constants

- How many elements are in this array (and what are their values?):

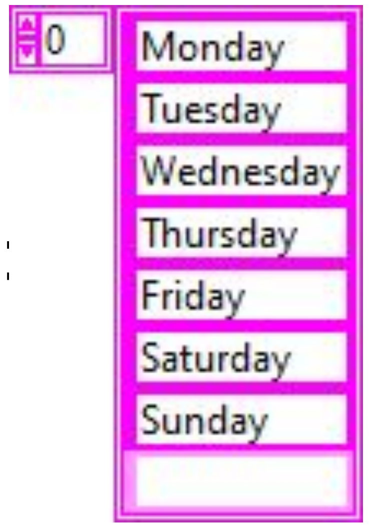


- If I can only see one element, it better be an empty array.

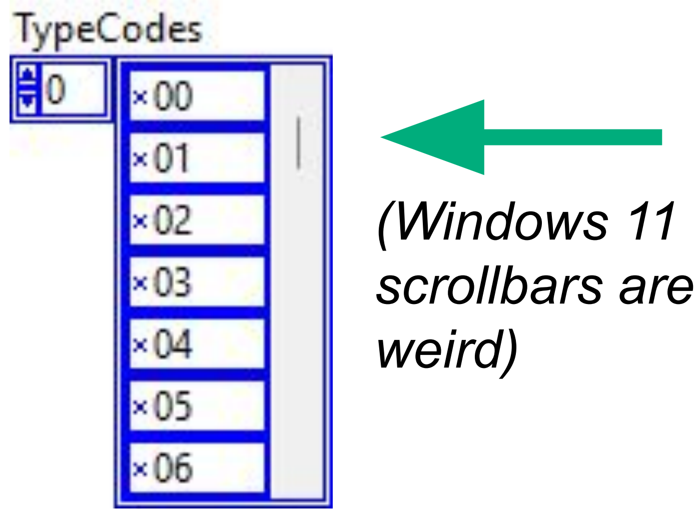


Readable Array Constants

- Show all elements (**and an empty one at the end**) if the array constant can reasonably fit on the containing diagram:

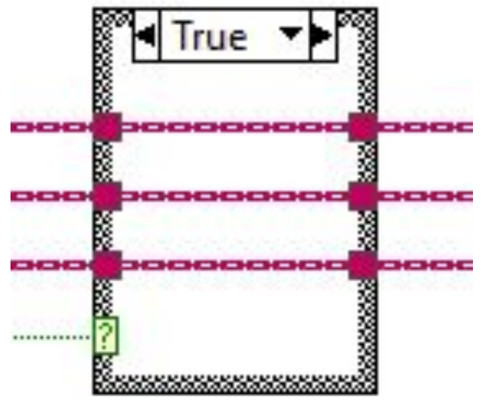


- If displaying all elements isn't feasible, show as many as possible, and show the scrollbar:

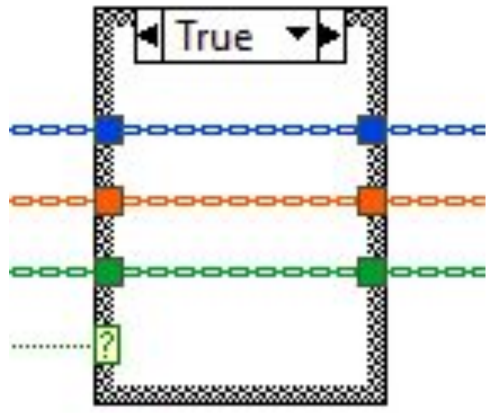


(Windows 11 scrollbars are weird)

Readable Class Wires

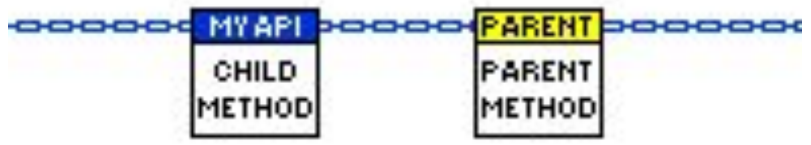
- A class wire with default appearance tells me nothing:
- 

- A customized class wire gives me more information:



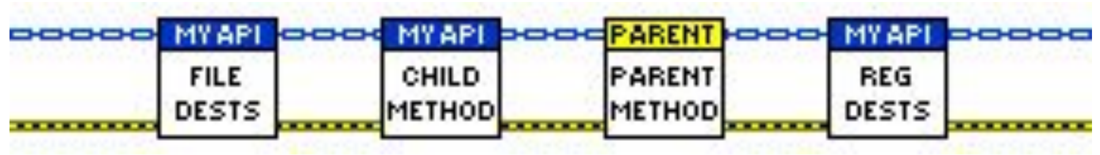
"The Blue Wire"
"The Orange Wire"
"The Green Wire"

Make banners match wire colors to further enhance readability

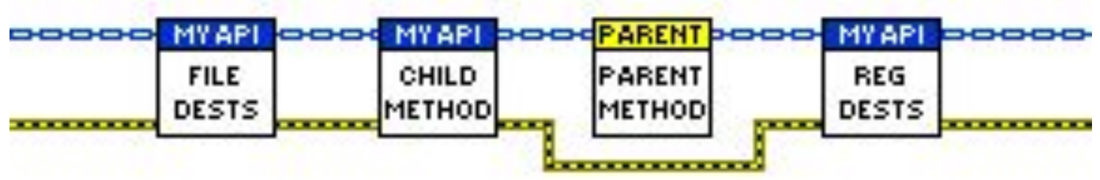


Readable Error Wires

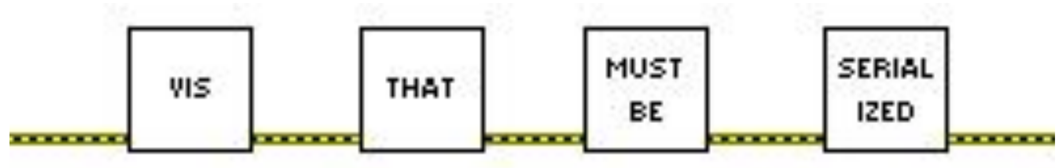
- An ‘error out’ terminal indicates that a VI or function could return an error:



- If **your** subVI can’t return an error, don’t add an ‘error out’ terminal to your subVI:

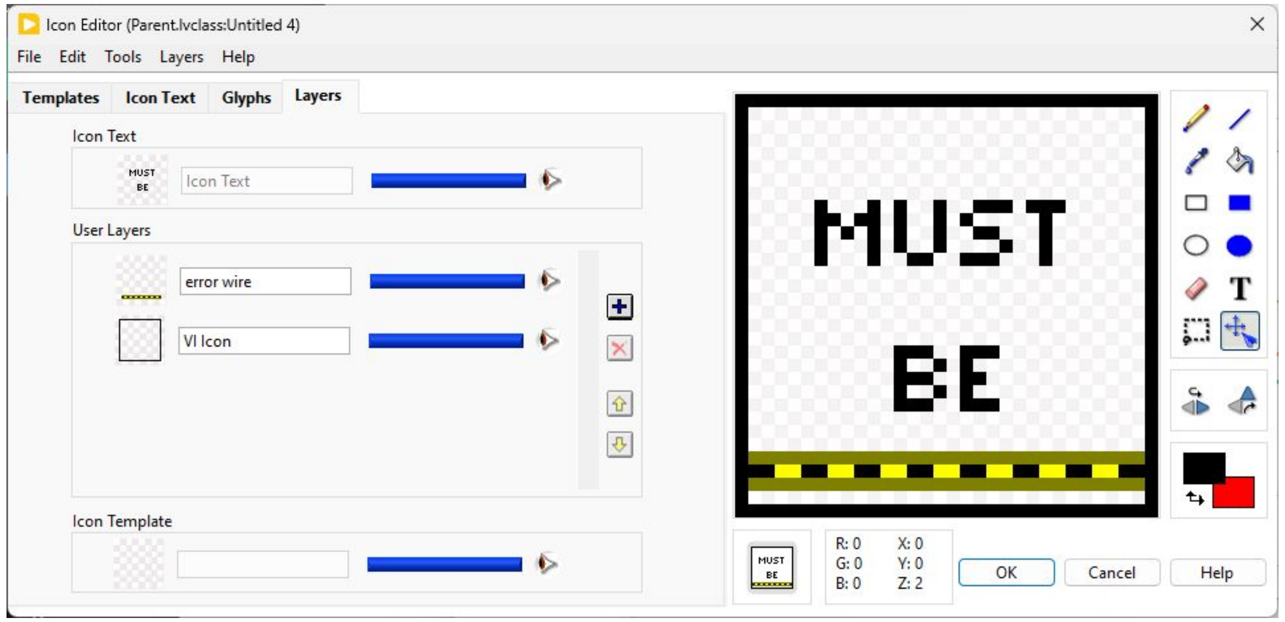


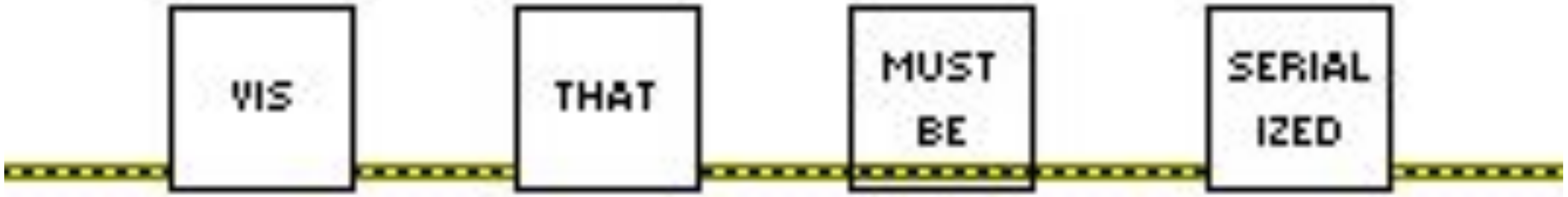
- “But what if I’m using the error wire for serialization?”



Experimental Idea: Error Pass-through on Icon

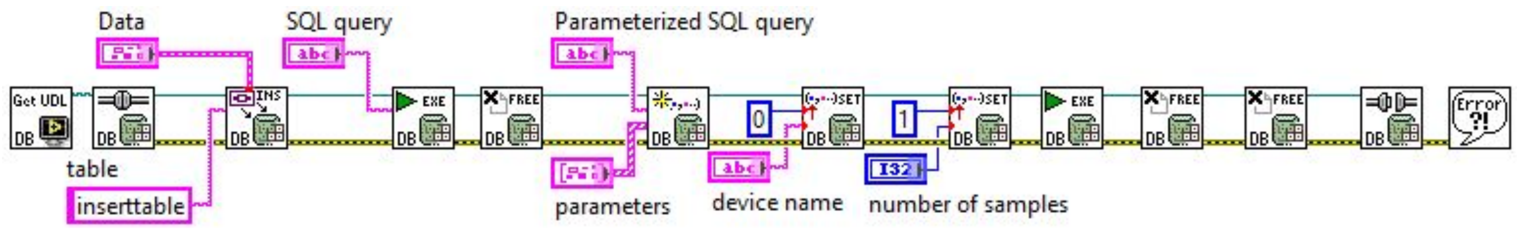
- Simple, visual, readable indication that an error wire passes through a subVI
- Try it out, let me know what you think



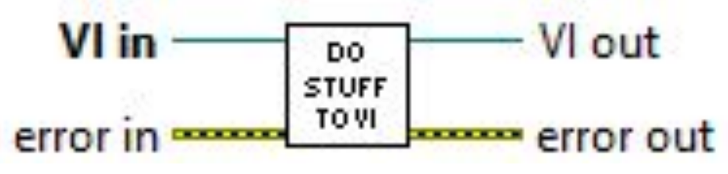


Readable Reference Wires

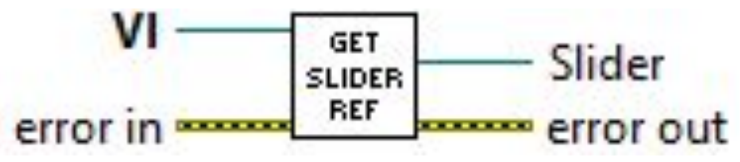
- Train-track reference wires are convenient:



- The fact that different reference wires are the same color is **inconvenient**
- **Always** position pass-through reference wires at the same height:

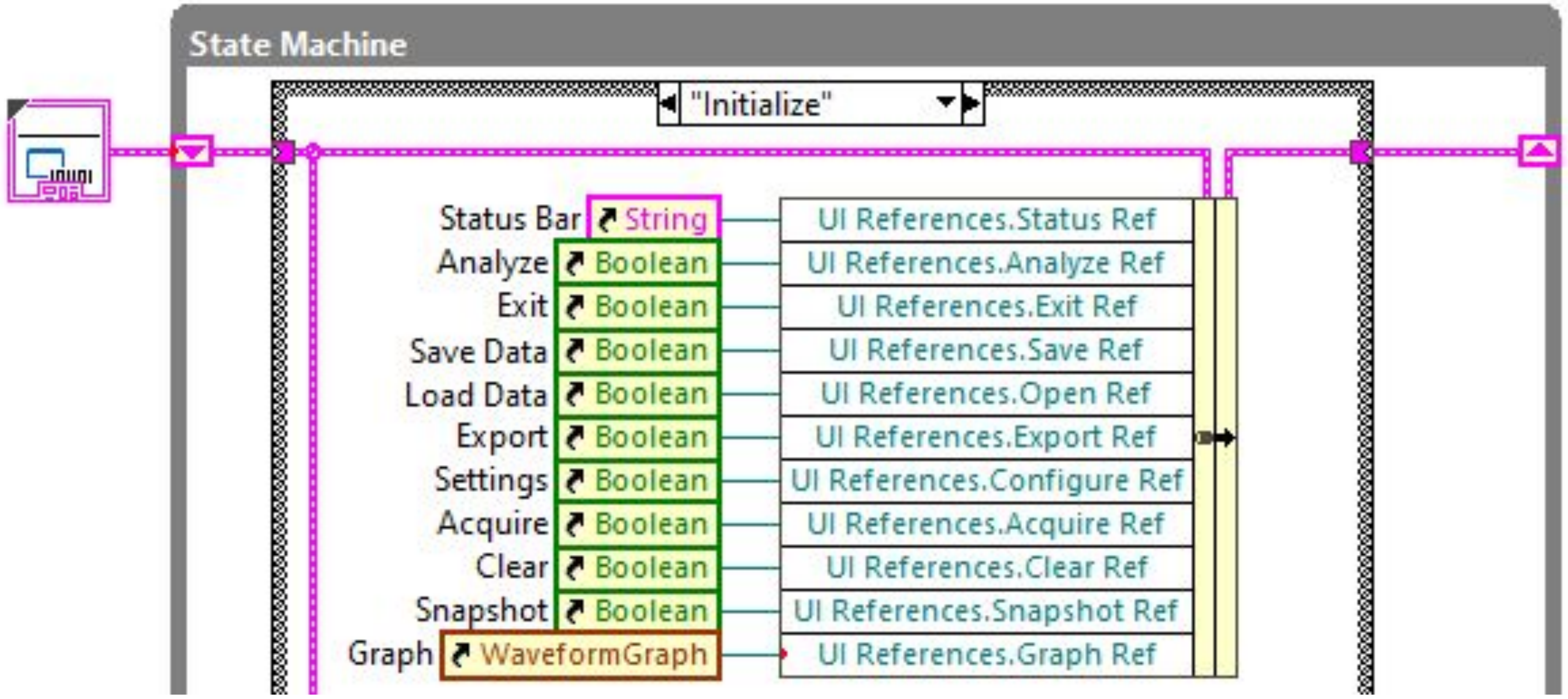


- **Always** position different references at different heights:



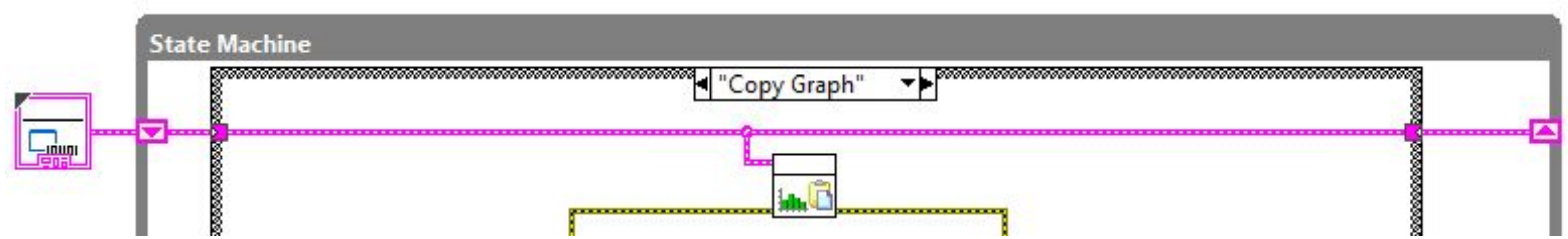
Readable UI Code in General

- It’s time we all say goodbye to this pattern:



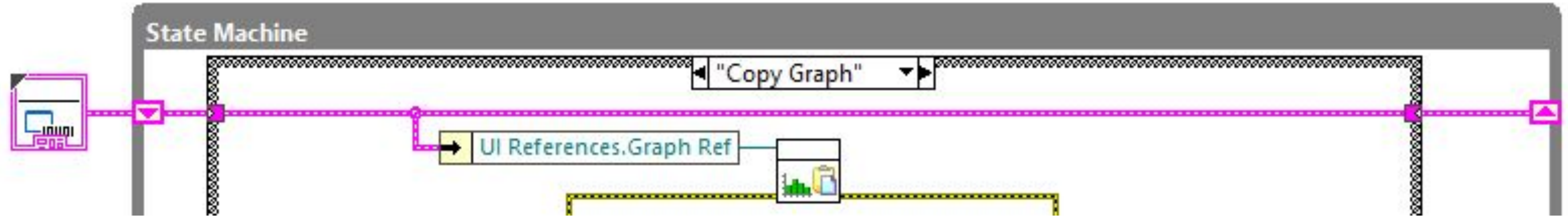
Status Bar	String	UI References.Status Ref
Analyze	Boolean	UI References.Analyze Ref
Exit	Boolean	UI References.Exit Ref
Save Data	Boolean	UI References.Save Ref
Load Data	Boolean	UI References.Open Ref
Export	Boolean	UI References.Export Ref
Settings	Boolean	UI References.Configure Ref
Acquire	Boolean	UI References.Acquire Ref
Clear	Boolean	UI References.Clear Ref
Snapshot	Boolean	UI References.Snapshot Ref
Graph	WaveformGraph	UI References.Graph Ref

Bundled Control References Aren’t Readable



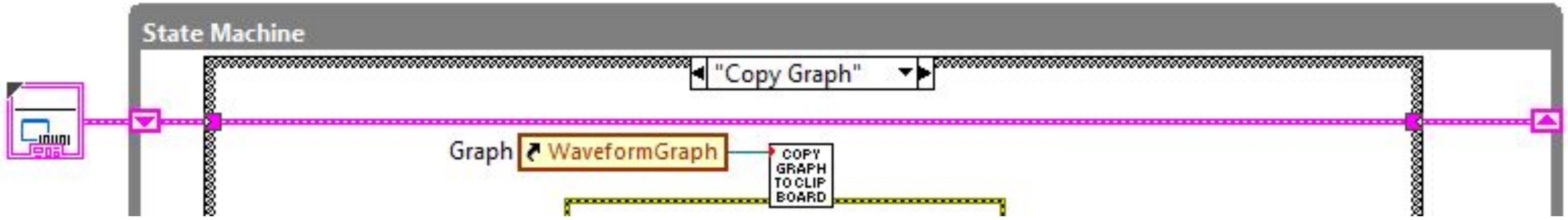
- *Maybe* that subVI is copying a graph… (to the clipboard)?
- Is the subVI messing with anything else on the panel besides the graph?

Unbundled Control References Are... Better?



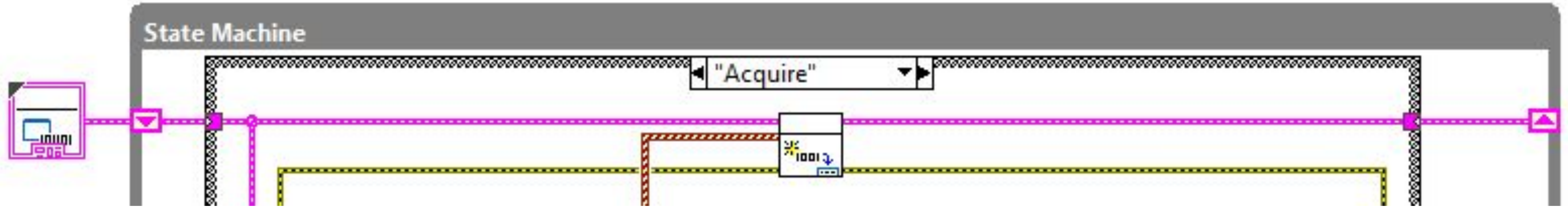
- Ok, that's a little better, now we know which control the subVI is acting on
- But there's *still* a disconnect between the control and the unbundled reference

Readable and Debuggable UI Code in General



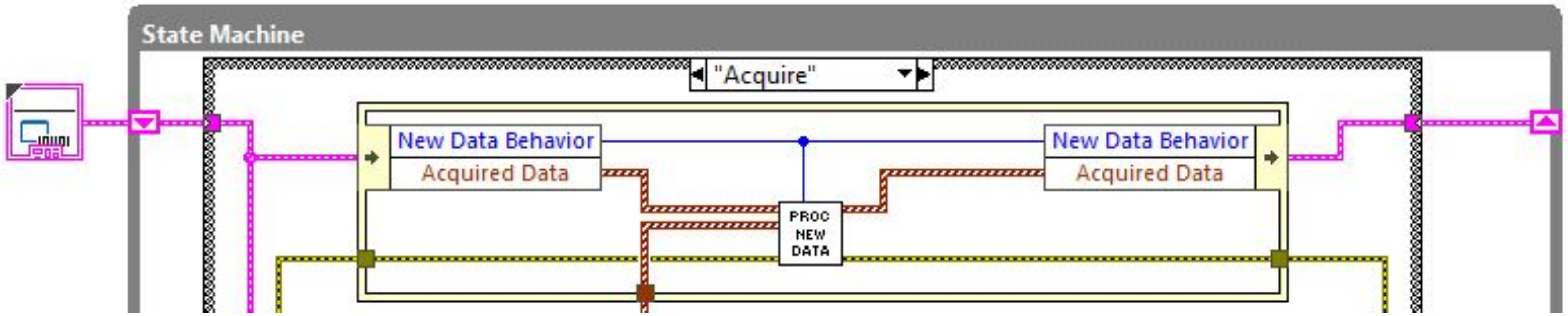
- That subVI is *definitely* copying the graph to the clipboard
- And not messing with anything else
- And is easy to find when **debugging** code pertaining to the graph

Yippee Ki-Yay, Mothercluster



- That subVI could be doing motherclusting ANYTHING!
  - *Seriously… what does that (professionally-developed) icon even mean?*
- And it could be doing it to ANY of your motherclusting data!

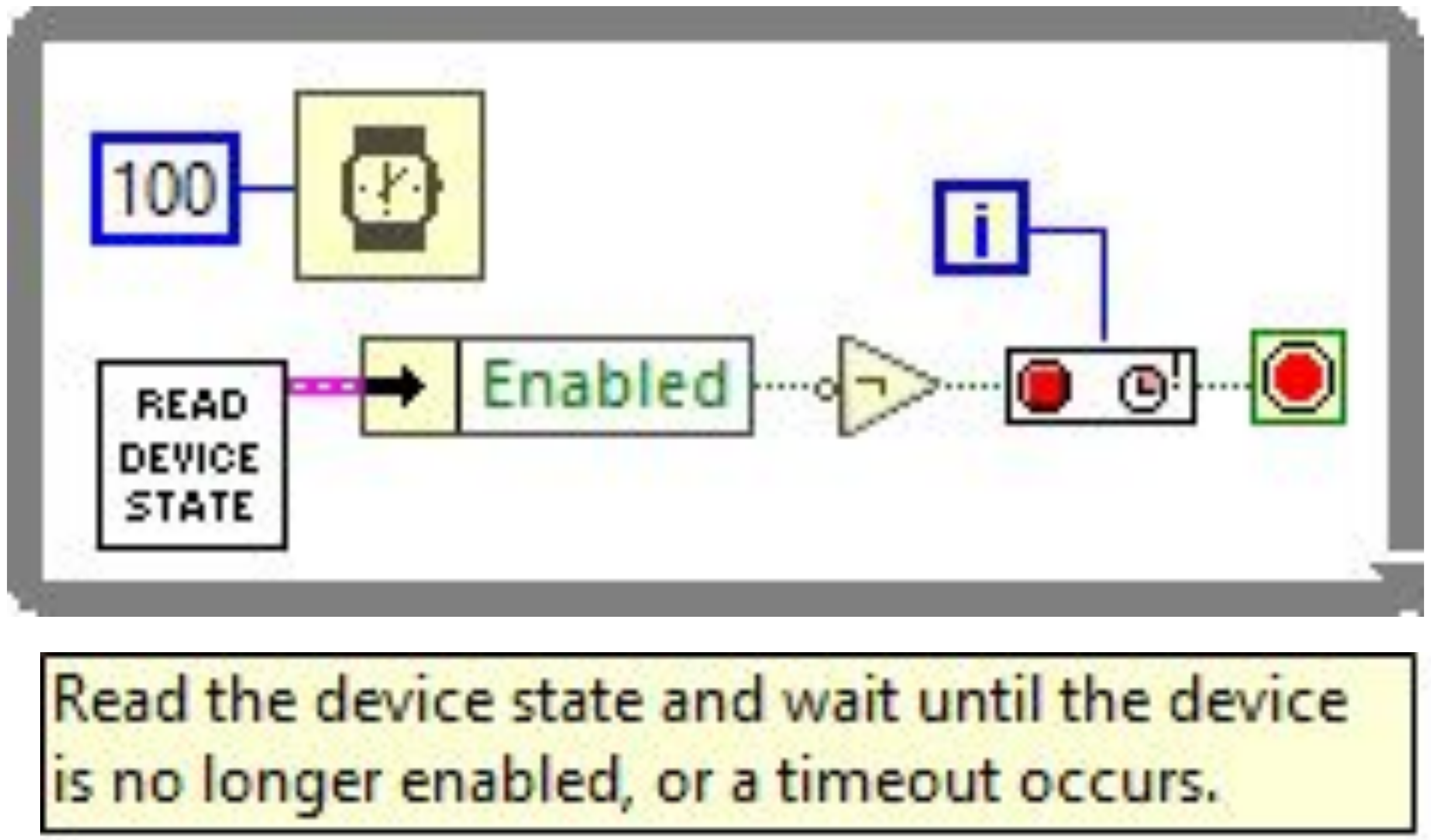
If You're a Mothercluster Lover…



- That subVI is processing new data
- That subVI is only modifying 'Acquired Data' in the mothercluster, and nothing else

A Quick Word About Free Labels

None of My Readable Diagrams Had Comments



Read the device state and wait until the device is no longer enabled, or a timeout occurs.

It's Time For Everybody's 2nd Favorite Game…

It’s Time For Everybody’s 2nd Favorite Game…

CAN I
POSSIBLY
OFFEND MORE
PEOPLE?
WHEEL OF FORTUNE
GDEVCON NA 2024
THE WORD FINDER

Can I Possibly Offend More People?

- Free labels **should not be used** to generally describe the behavior of a diagram.

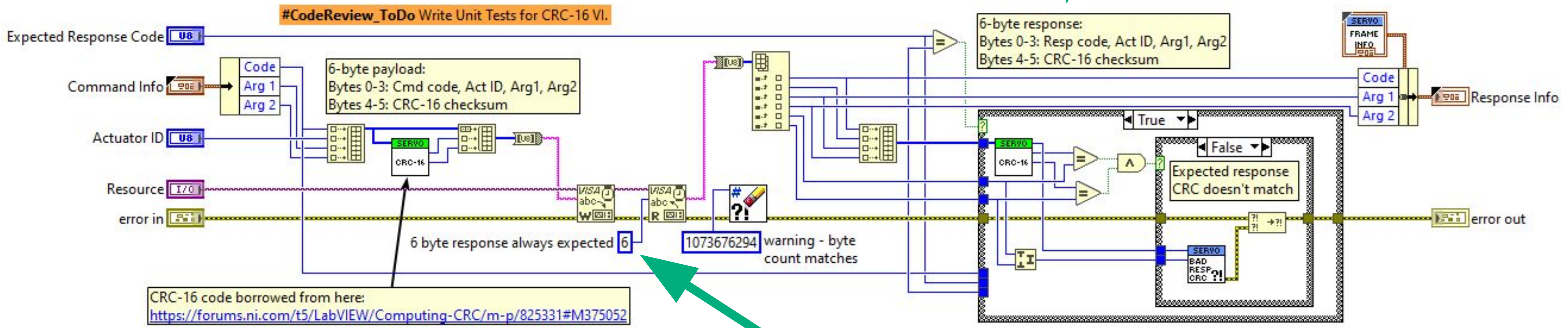
- That’s what diagrams are for.

- That’s what **readable** diagrams are for.

When Do Labels Make Sense?

#Bookmarks for code reviews/to-dos/etc.

Explaining *non-obvious* details of an algorithm



#CodeReview_ToDo Write Unit Tests for CRC-16 VI.

6-byte payload:
Bytes 0-3: Cmd code, Act ID, Arg1, Arg2
Bytes 4-5: CRC-16 checksum

6-byte response:
Bytes 0-3: Resp code, Act ID, Arg1, Arg2
Bytes 4-5: CRC-16 checksum

6 byte response always expected 6

1073676294 warning - byte count matches

Expected response CRC doesn't match

CRC-16 code borrowed from here:
https://forums.ni.com/t5/LabVIEW/Computing-CRC/m-p/825331#M375052

Describing the *non-obvious* purpose of a constant value

Citing an external source (website, e.g.)

Command-Response.vi

Summary

Summary

- Maximize readability in the ways that you:
  - NAME things
  - Utilize TEXT
  - CONSTRUCT block diagrams
- Don’t be afraid to question “best practices”
  - Text on a block diagram is good! And so, so *readable*.
- **Always Be Readabling**

dnatt.org

bit.ly/labviewreadability