Code Smells:

Sniffing out Poorly Written Code

Charlotte Nicolaou PhD CLA

National Instruments

About the Session Author

Mark Ridgley
CLD, CTA, CPI, LabVIEW Champion
Owner, Radius Teknologies, LLC

What's in it for me?

You'll learn:

- What a code smell is
- What the most common code smells are
- Why code smells happen
- The good news about code smells
- Why you should be concerned about code smells
- What you can do when you encounter a code smell
- Tools to help eradicate code smells

What is a code smell?

Definition

Code Smell – A **symptom** in the source code that **possibly** indicates a deeper problem

Common Code Smells

- Poor naming
- Overly complicated
- Contrived complexity
- Too many modules
- Duplicate logic
- Disorganization
- Breaks dataflow
- Outdated practices
- Too many parameters
- Excessive use of literals
- Cyclomatic complexity
- Others???

Why do code smells happen?

- Time constraints
  - We are pressured to look for the **quickest** implementation instead of the **best** implementation
  - “If it runs, it’s done.”
- Start coding without requirements
- Cut/Paste/Rename
  - Easy and fast ‘reuse’
- Development team size
- No ownership of code

There is good news, though…

A code smell:

- Is **not** a Bug
- Is **not** technically incorrect
- **Does not** currently prevent the code from functioning

If these statements are true, then why should we be concerned?

Why be concerned?

- Code smells:
  - Make code less
    - Readable
    - Maintainable
    - Scalable
  - Affect all phases of the software lifecycle
    - Development
    - Support
    - Troubleshooting
    - Modification
    - Maintenance
  - May lead to bugs

What can we do?

- Be vigilant – Code smells are all around us
  - We just have to look for them
  - Can be in **any** code (LabVIEW VIs, TestStand sequences)
- Don’t look the other way when you discover a code smell
  - Take the initiative and time to **refactor** the code
    - If you don’t, who will?

Refactoring

Definition

Refactoring – The process of redesigning software to make it more readable and maintainable so that the cost of change does not increase over time

When to Refactor

- When you are adding features to or debugging your source code

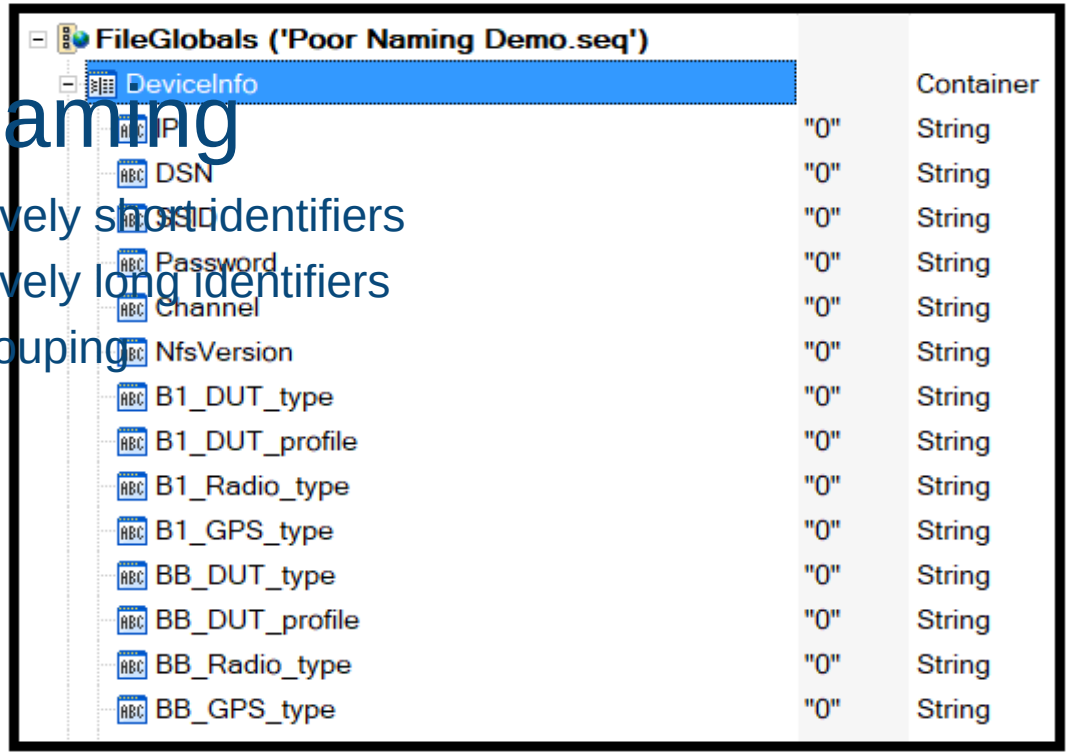
Demonstrations

Code From The Real World
(You can’t make this stuff up!)

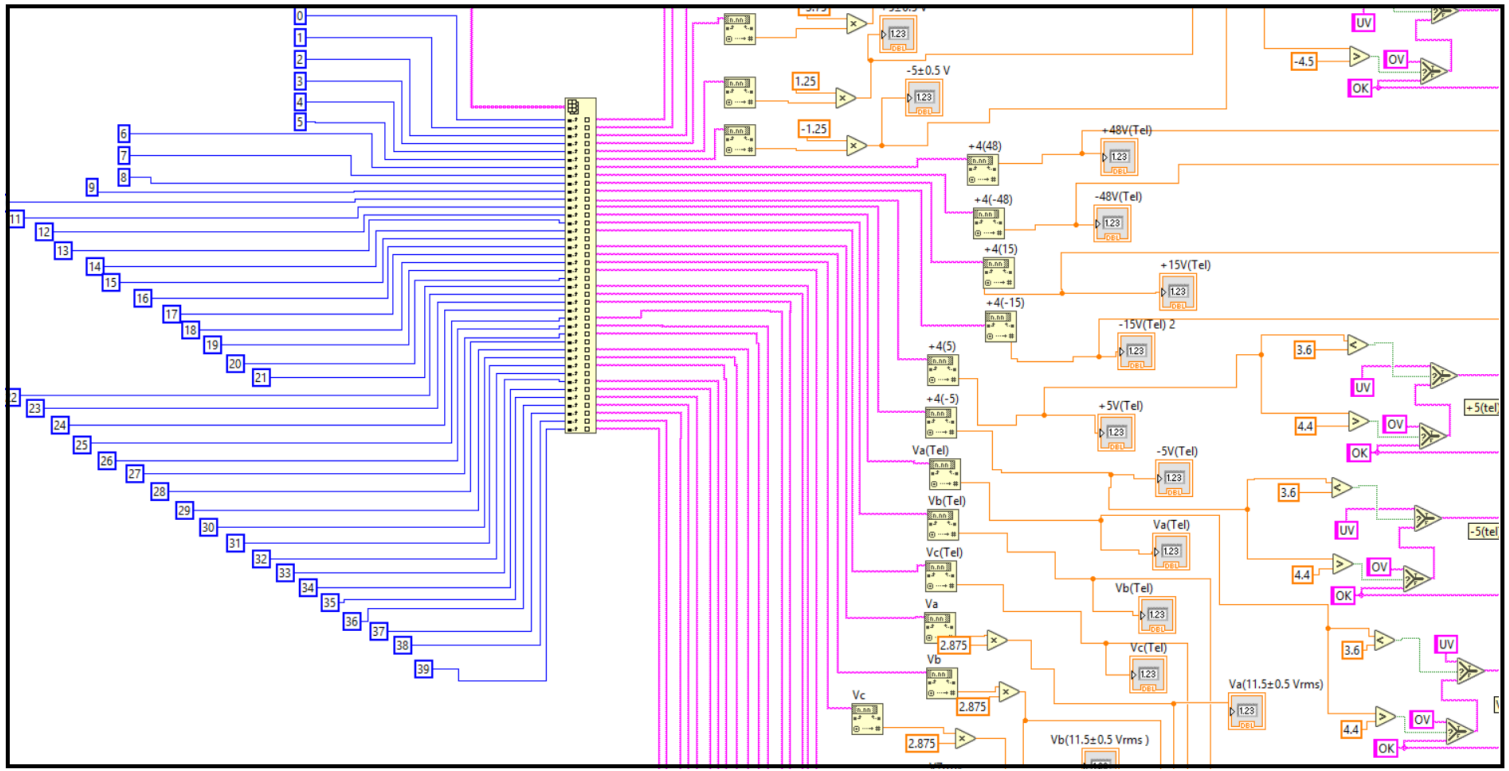
Demonstration

Poor Naming

- Excessively short identifiers
- Excessively long identifiers
- Poor grouping

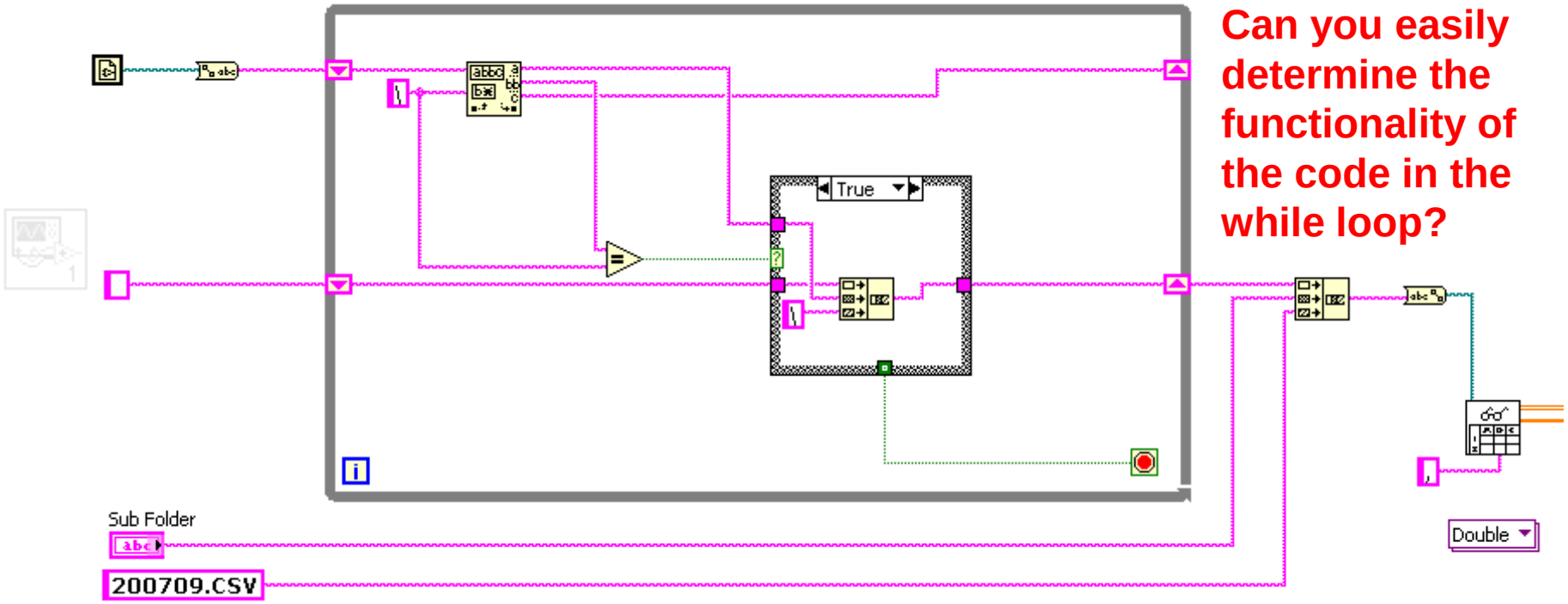


Demonstration



NATIONAL INSTRUMENTS

Demonstration

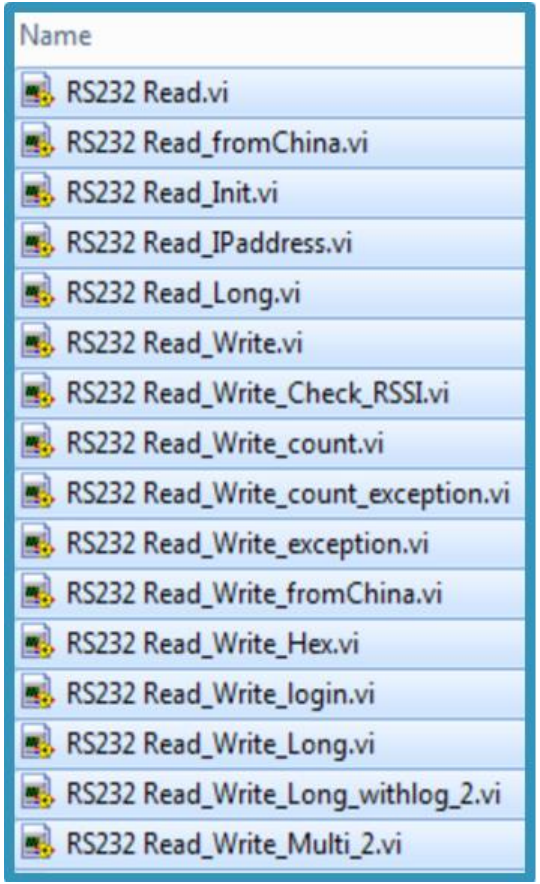


Can you easily determine the functionality of the code in the while loop?

NATIONAL INSTRUMENTS

Demonstration

Too Many Modules



NATIONAL INSTRUMENTS

Demonstration

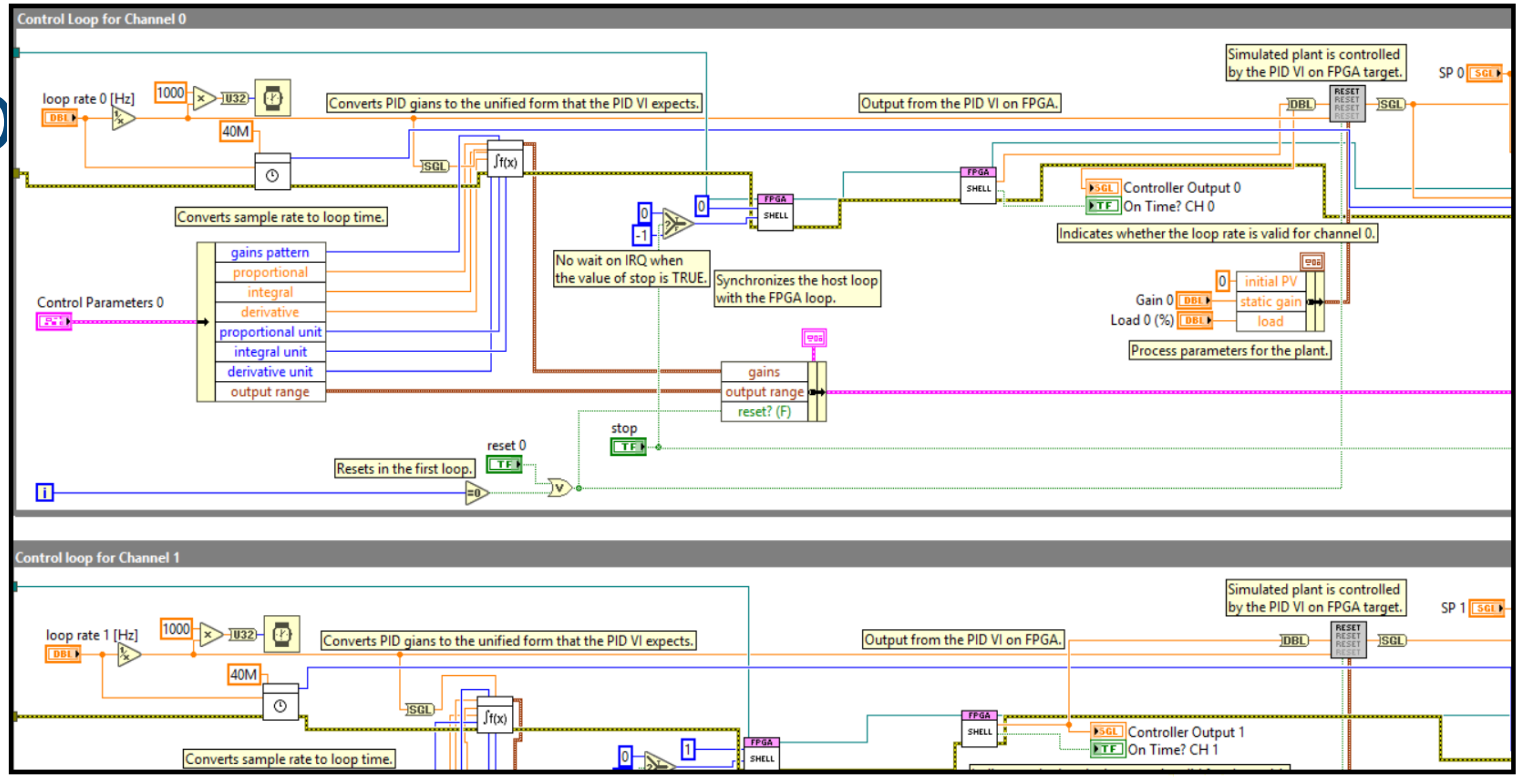
Evolut

VI Name	Total Terminals	Connected Terminals
RS232 Read_Write_Multi_withlog_count_exception.vi	16	14
RS232 Read_Write_Multi_withlog_2.vi	16	13
RS232 Read_Write_Multi_count_exception.vi	16	13
RS232 Read_Write_Multi_count.vi	16	13
RS232 Read_Write_Multi_2.vi	16	12
RS232 Read_Write_Reboot_withlog_count.vi	12	9
RS232 Read_ZigBee.vi	12	8
RS232 Read_Write_withlog_exception.vi	12	8
RS232 Read_Write_withlog_count_exception.vi	12	8
RS232 Read_Write_withlog_count.vi	12	8
RS232 Read_Write_withlog_2_froChina.vi	12	8
RS232 Read_Write_withlog_2.vi	12	8
RS232 Read_Write_Reboot_withlog_2.vi	12	8
RS232 Read_Write_Reboot_withlog.vi	12	8
RS232 Read_Write_Reboot_count_exception.vi	12	8
RS232 Read_Write_Reboot_count.vi	12	8
RS232 Read_Write_Long_withlog_2.vi	12	8
RS232 Read_Write_count_exception.vi	12	8
RS232 Read_Write_count.vi	12	8
RS232 Read_Write_withlog.vi	12	7

NATIONAL INSTRUMENTS

Demonstration

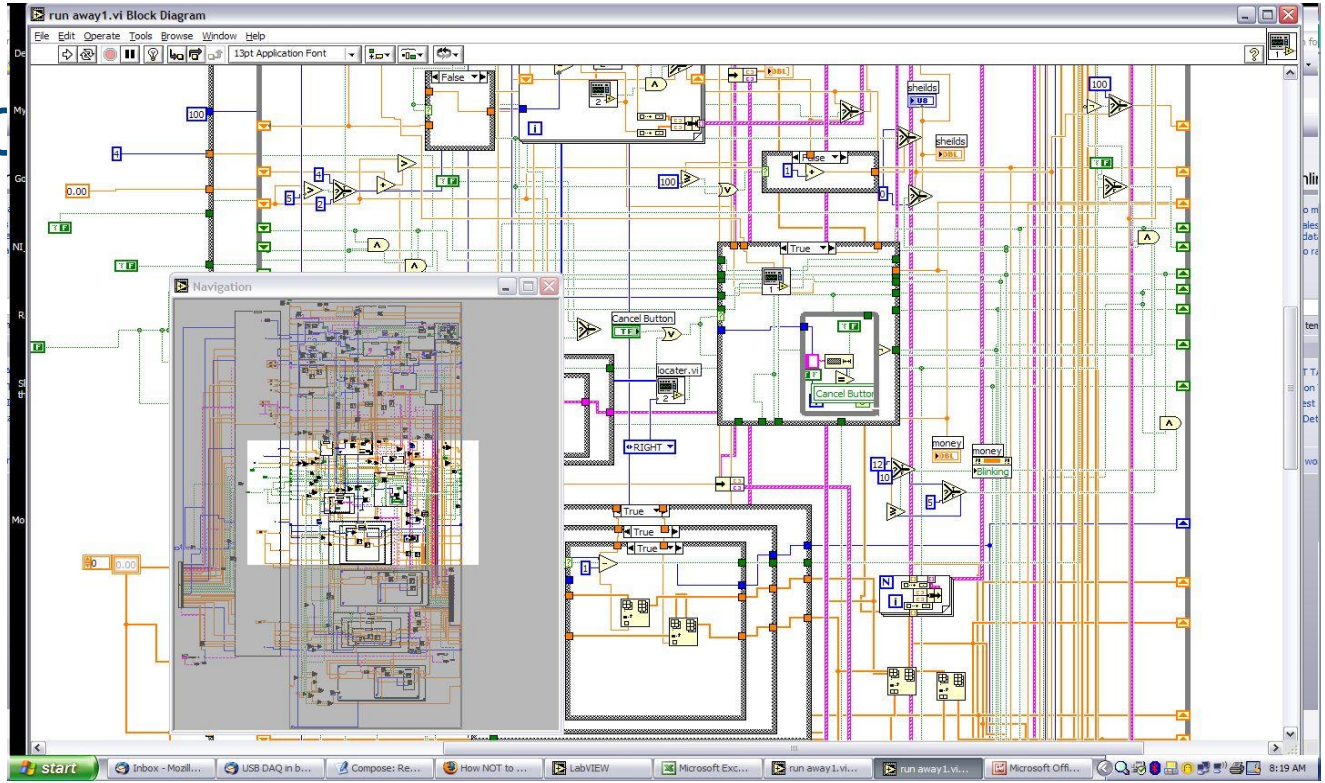
D



NATIONAL INSTRUMENTS

Demonstration

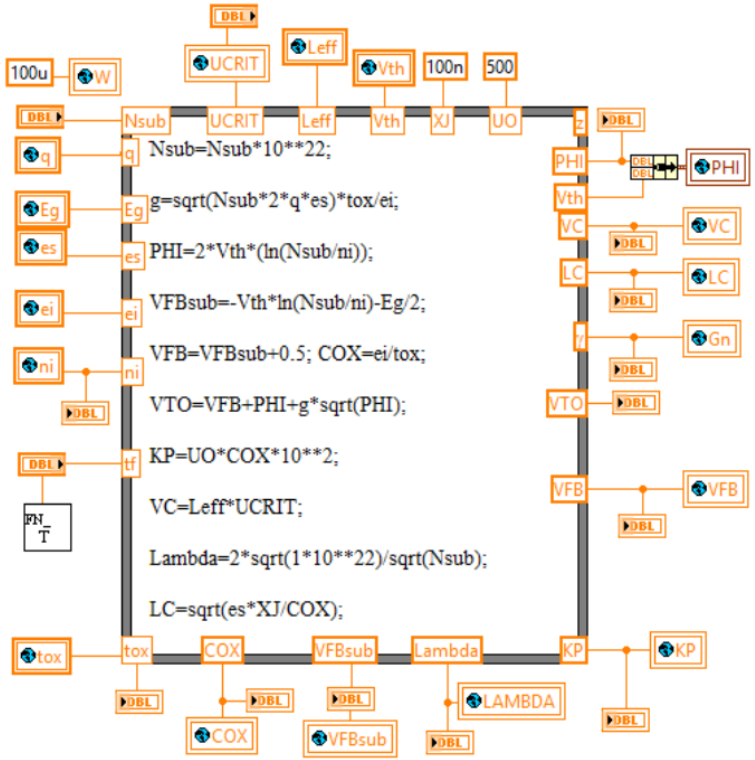
Disc



NATIONAL INSTRUMENTS

Demonstration

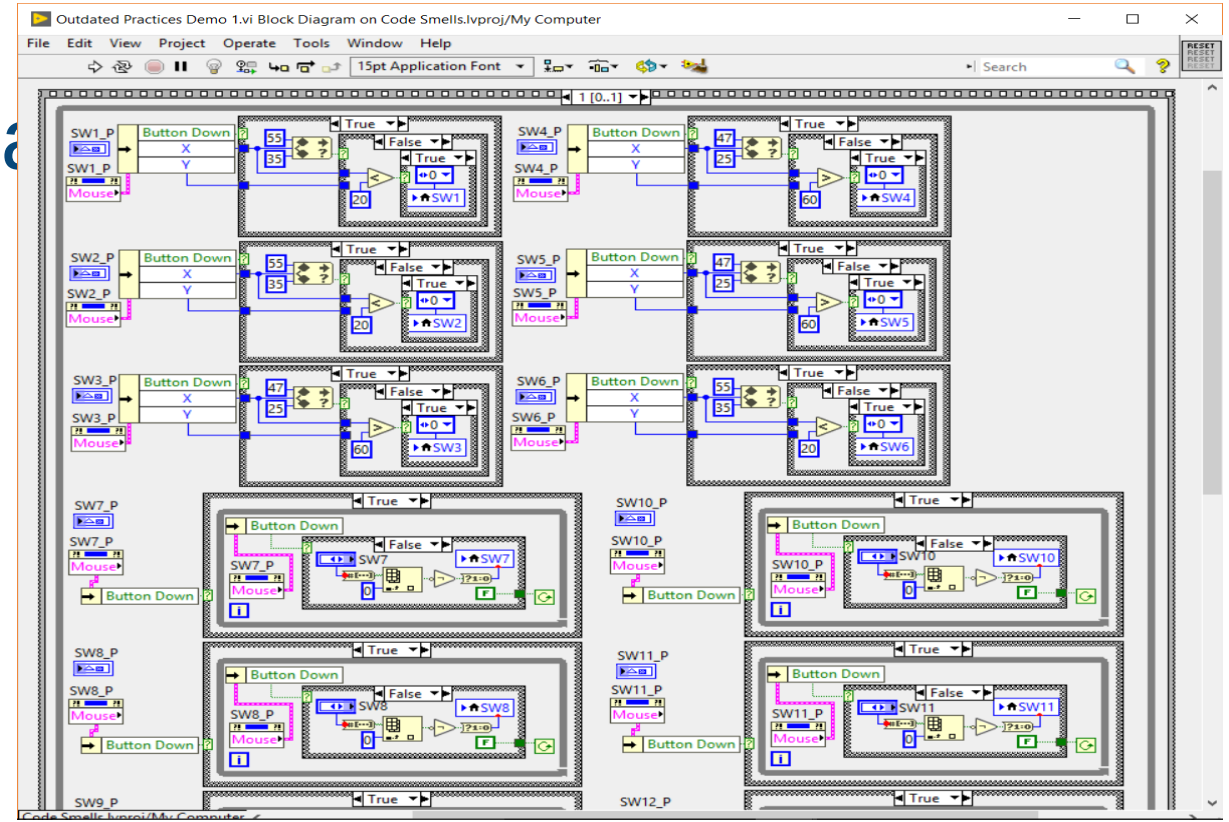
Breaks Da



NATIONAL INSTRUMENTS

Demonstration

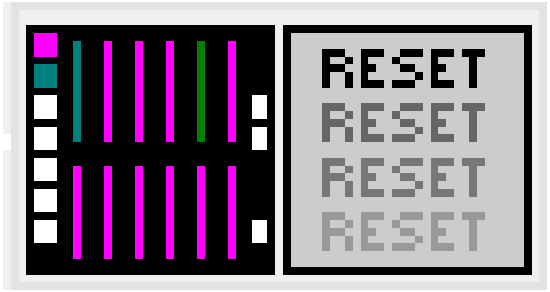
Outda

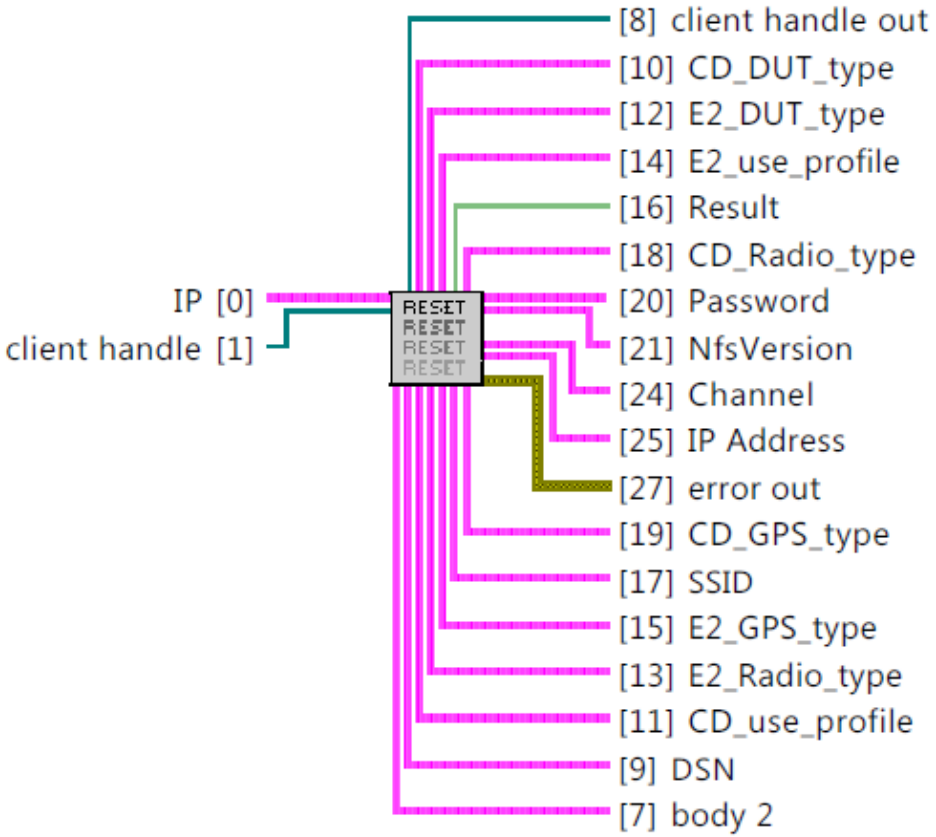


NATIONAL INSTRUMENTS

Demonstration

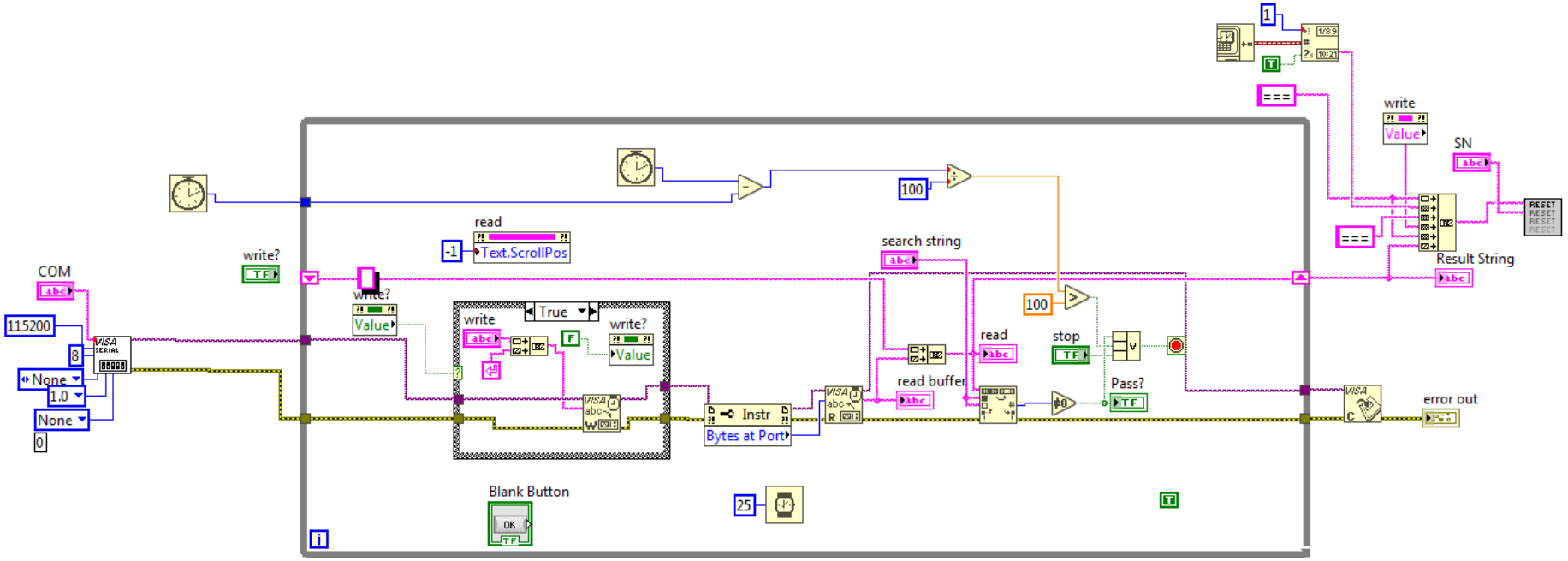
Too Many Parame





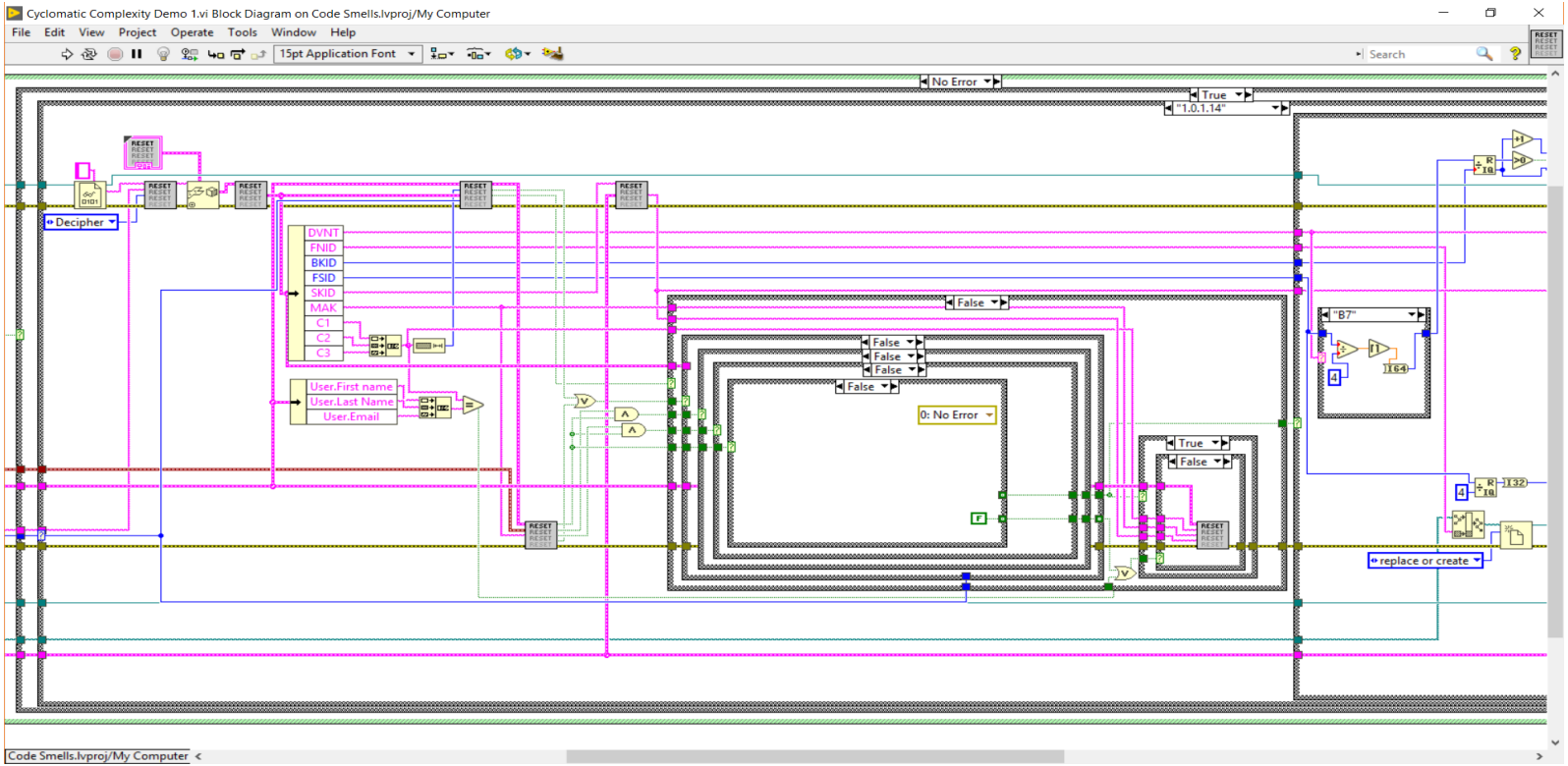
NATIONAL INSTRUMENTS

Demonstration



NATIONAL INSTRUMENTS

Demonstration



NATIONAL INSTRUMENTS

Tools to Help Eradicate Code Smells

- LabVIEW: VI Analyzer
  - Tools >> VI Analyzer >> Analyze VIs…
- TestStand: Sequence Analyzer
  - Debug >> Sequence Analyzer >> Analyze <Sequence Name>

Topics Covered

- Poor naming
- Overly complicated
- Contrived complexity
- Too many modules
- Duplicate logic
- Disorganization
- Breaks dataflow
- Outdated practices
- Too many parameters
- Excessive use of literals
- Cyclomatic complexity
- Tools to help eradicate code smells

Next Steps

- Learn more about detecting and correcting smelly code
  - LabVIEW Help
    - On the Index tab, type “VI Analyzer” into the keywords to find field, then double-click the “VI Analyzer” topic and click the link to open the LabVIEW VI Analyzer Toolkit User Guide.
  - TestStand Help
    - On the Index tab, type “Sequence Analyzer” into the keywords to find field, then double-click the “analyzing sequence file” topic.

Questions / Comments

Mark Ridgley
CLD, CTA, CPI, LabVIEW Champion
Owner, Radius Teknologies, LLC

Thank You!

NATIONAL INSTRUMENTS