

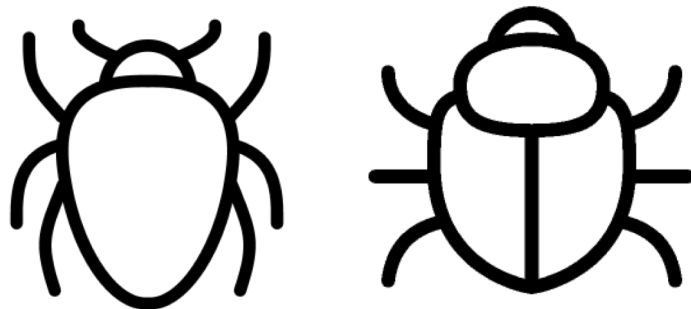


# Debugging in LabVIEW

Hans-Petter Halvorsen

# What is “Debugging”?

Debugging is the process of locating and fixing or bugs (errors) in your computer program code, in this case your LabVIEW program



# Debugging in LabVIEW

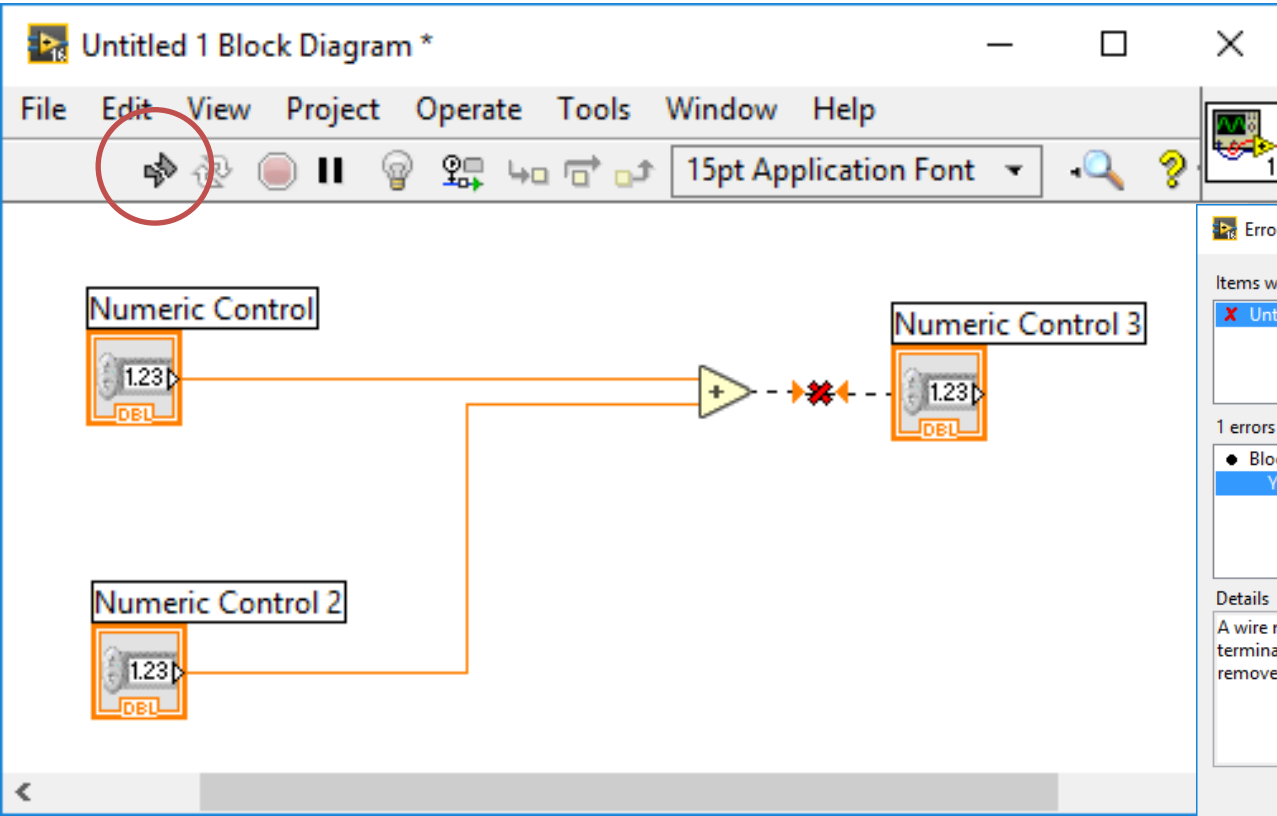
LabVIEW has powerful features for Debugging your Code, such as:

- Broken Run Arrow
- Highlight Execution
- Probes
- Breakpoints

# Broken Run Arrow

- Click the broken Run button to display the Error list window, which lists all the errors.
- Double-click an error description to display the relevant block diagram or front panel and highlight the object that contains the error.

# Broken Run Arrow - Example



Double-click

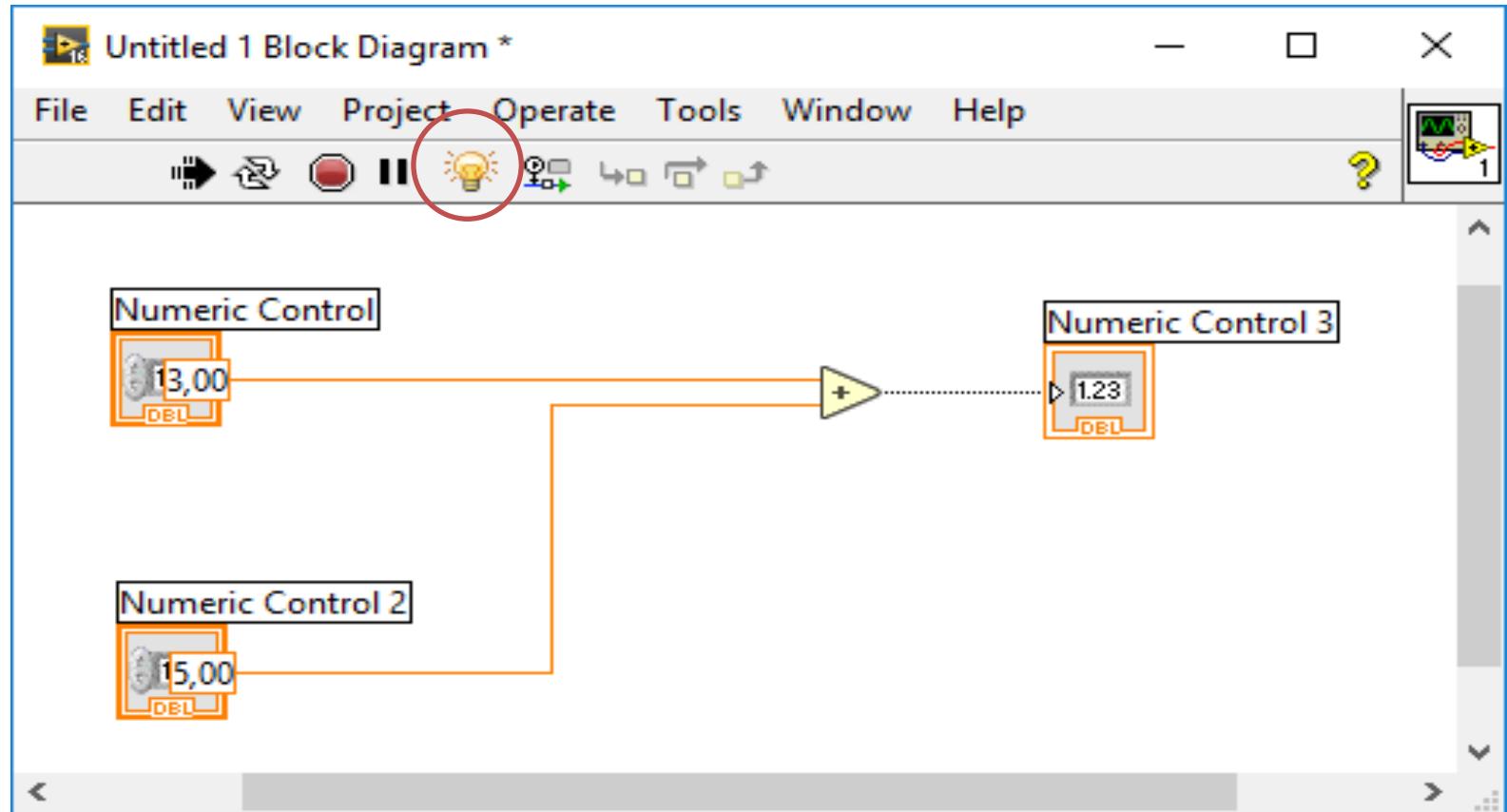
The screenshot shows the Error list window. The window title is "Error list". The "Items with errors" section shows "Untitled 1" with a red 'X' icon. The "1 errors and warnings" section shows "Block Diagram Errors" with a red dot icon. The error message is: "You have connected an output of Add to a Control. Change the control to an indicator." The "Details" section provides further information: "A wire must be connected to one and only one data source, such as a control's terminal or a function output. This wire is connected to two such sources. You must remove a source, perhaps by changing a control to an indicator." The "Show Error" button is highlighted in blue.

**DEMO**

# Highlight Execution

- View an animation of the execution of the block diagram by clicking the Highlight Execution button.
- Execution highlighting shows the flow of data on the block diagram from one node to another using bubbles that move along the wires.
- Note! Execution highlighting greatly reduces the speed at which the VI runs.

# Highlight Execution



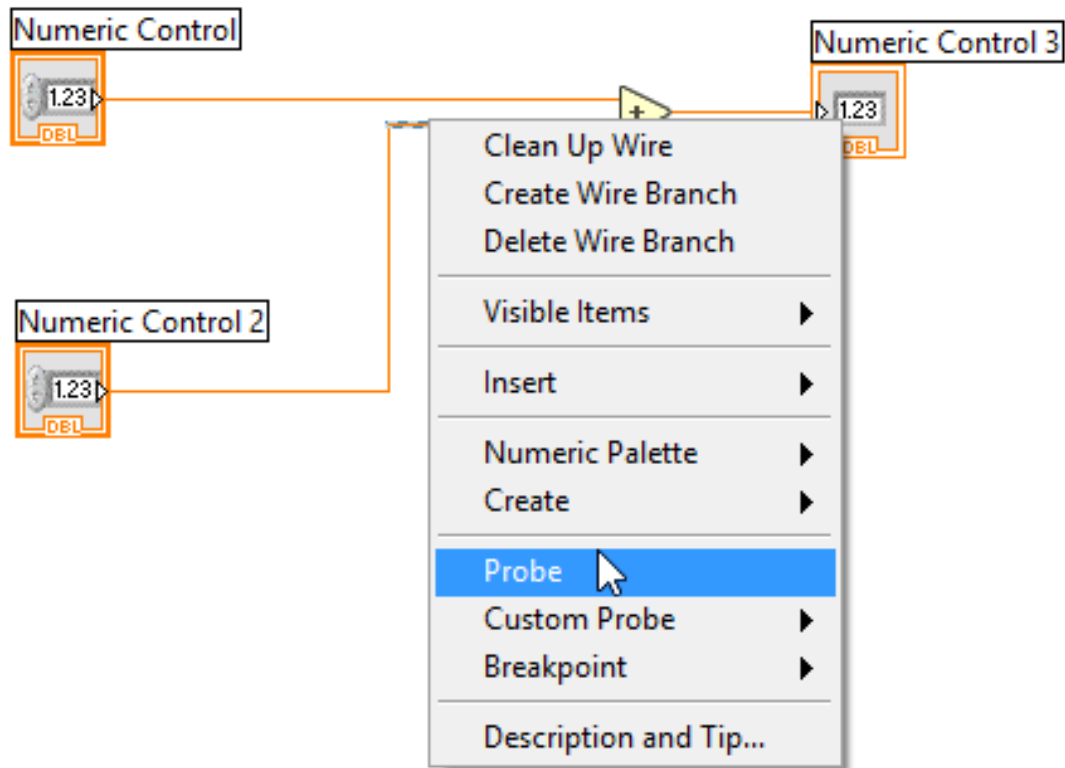


**DEMO**

# Probes

- Use the Probe tool to check intermediate values on a wire as a VI runs.
- Probe Watch Window

# Probes



# Probe Watch Window

The screenshot shows a LabVIEW block diagram titled "Untitled 1 Block Diagram \*". The diagram consists of three numeric control blocks: "Numeric Control", "Numeric Control 2", and "Numeric Control 3". Each block contains the value "1.23". The "Numeric Control" and "Numeric Control 2" blocks are connected to a junction point labeled "2". From this junction, one path goes to an addition block (+) and another path goes to a junction point labeled "3". The addition block is also connected to a junction point labeled "4", which is then connected to "Numeric Control 3".

The "Probe Watch Window" is open in the foreground, displaying a table of probe data. The table has three columns: "Probe(s)", "Value", and "Last Update". The data is as follows:

Probe(s)	Value	Last Update
Untitled 1		
[2] Numeric	3.000E+0	07.09.2016 12.34.10
[3] Numeric	5.000E+0	07.09.2016 12.34.10
[4] Probe	8.000E+0	07.09.2016 12.34.10

The "Probe Display" area on the right side of the window shows the value "8".

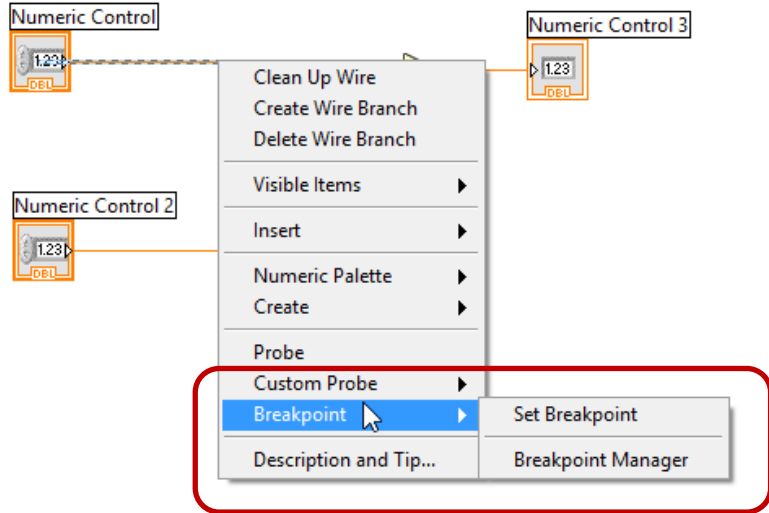
**DEMO**

# Breakpoints

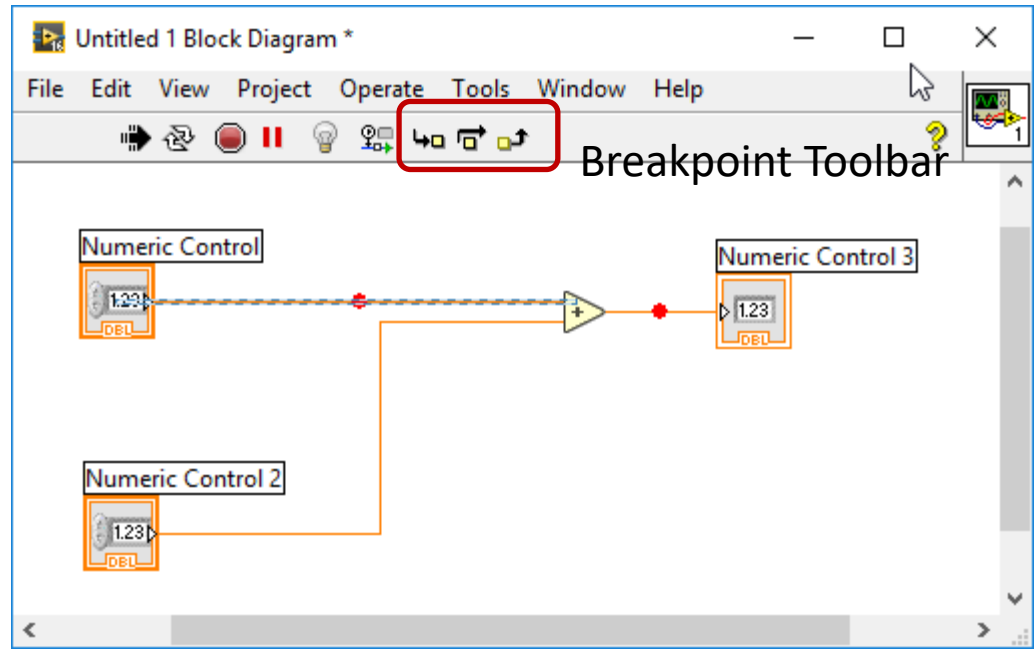
- Use the Breakpoint tool to place a breakpoint on a VI, node, or wire on the block diagram and pause execution at that location.
- When you set a breakpoint on a wire, execution pauses after data pass through the wire.

Tools Palette:

# Breakpoints



Set/Clear Breakpoint →



# Breakpoint Manager

The screenshot shows the LabVIEW interface with a block diagram titled "Untitled 1 Block Diagram \*". The diagram contains three numeric controls: "Numeric Control", "Numeric Control 2", and "Numeric Control 3". Each control displays the value "1.23" and has a "DEL" button below it. The controls are connected to a central adder (+) block. A wire from "Numeric Control" and a wire from "Numeric Control 2" connect to the left input of the adder. The output of the adder connects to "Numeric Control 3". Red dots on the wires indicate that breakpoints are enabled at these locations. The "Breakpoint Manager" window is open in the foreground, displaying a table of breakpoint information.

VI Name	Object Name	State
Untitled 1	Wire	●
Untitled 1	Wire	●

Close Help

Enable/Disable  
Breakpoints

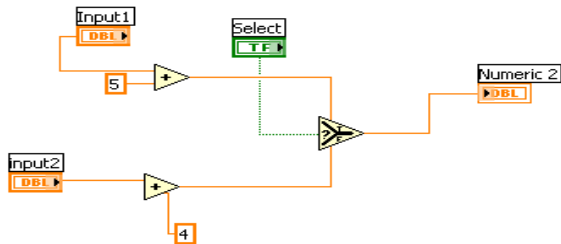


**DEMO**

# How To Avoid Bugs

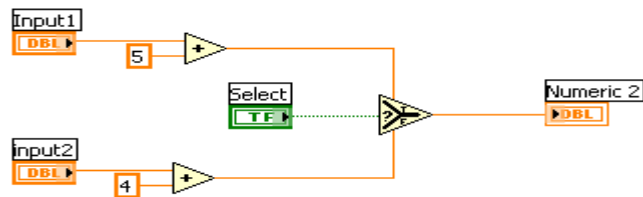
- Structure your Code properly, i.e., avoid so-called «Spaghetti Code»
- Flow from left to right
- Use SubVIs
- Use the State Machine Programming technique
- Make it simple
- Keep in mind that others should understand your Program
- ..

# Bad vs. Good Code



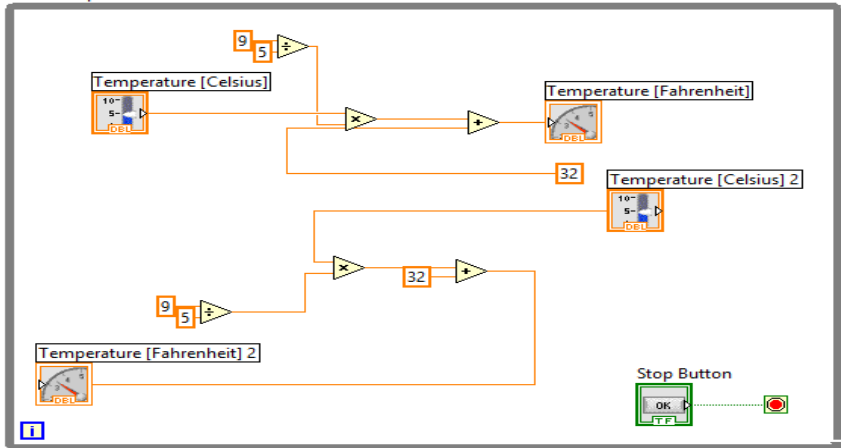
Avoid Spaghetti Code!

The Flow should go from left to right

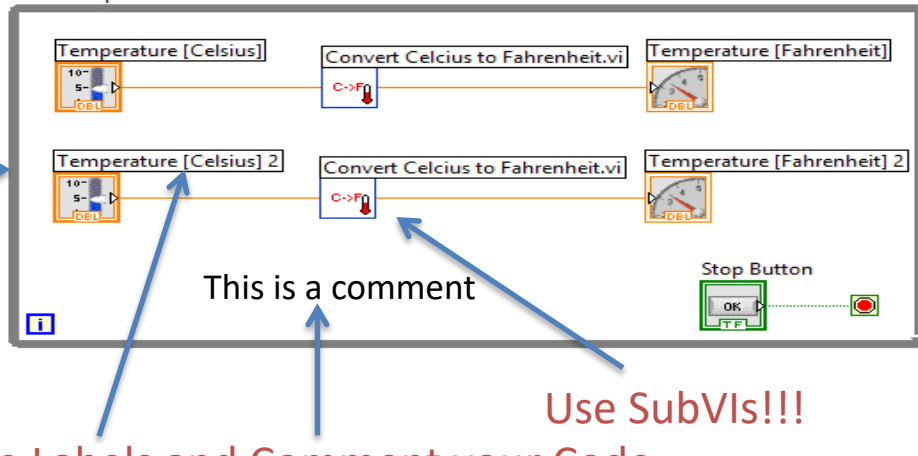


Make your code readable for others!

While Loop



While Loop



This is a comment

Use SubVIs!!!

Use Labels and Comment your Code

# Hans-Petter Halvorsen

University of South-Eastern Norway

[www.usn.no](http://www.usn.no)

E-mail: [hans.p.halvorsen@usn.no](mailto:hans.p.halvorsen@usn.no)

Web: <https://www.halvorsen.blog>

