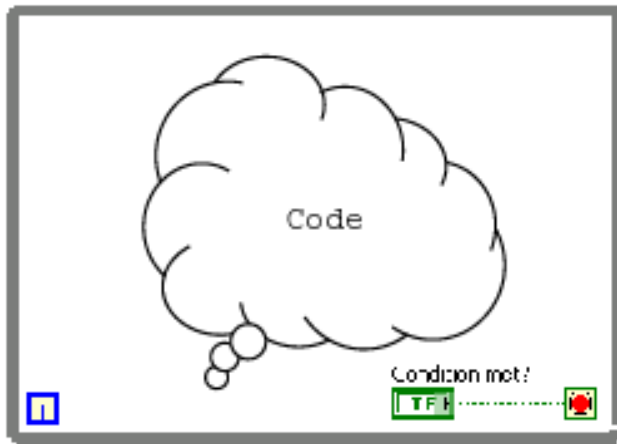


# Lecture 3-1

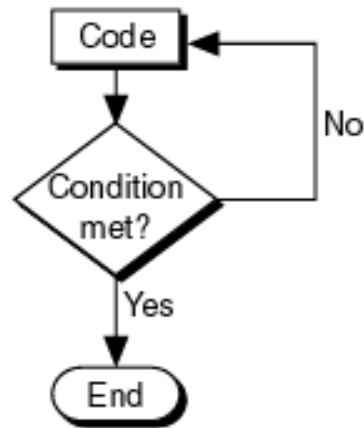
Loops and Charts,  
Arrays, Graphs, Clusters,  
Case, Sequence Structures,  
Local, Global Variables, and  
Attribute Nodes  
in Graphical Programming Language

吳文中

# While Loops



LabVIEW While Loop



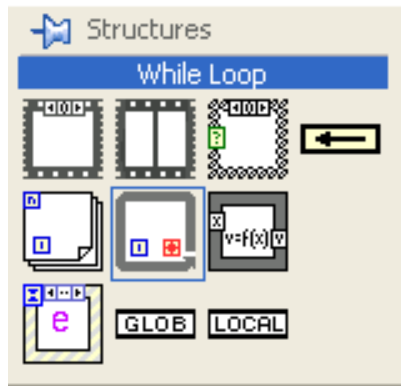
Flow Chart

```
Repeat (code) ;  
Until Condition met ;  
End ;
```

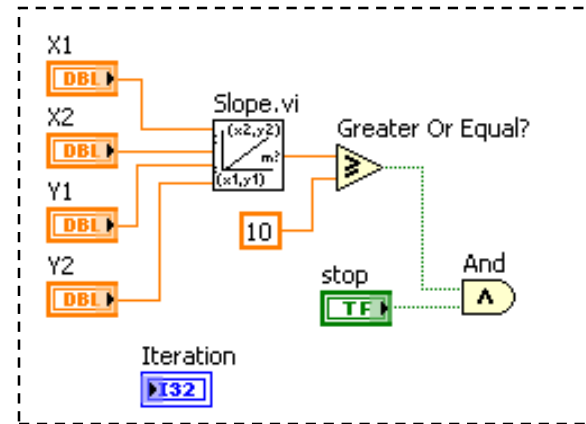
Pseudo Code

# While Loops

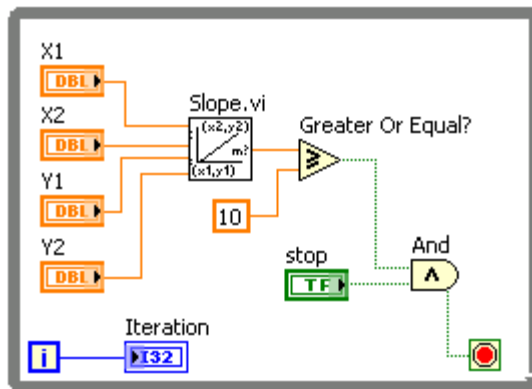
## 1. Select While Loop



## 2. Enclose code to be repeated



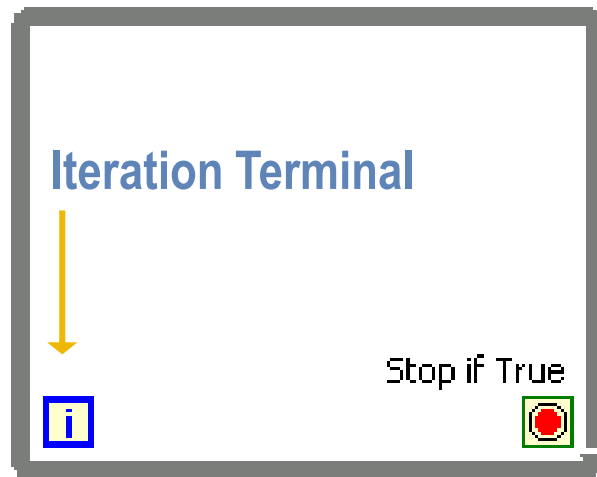
## 3. Drop or drag additional nodes and then wire



# While Loops

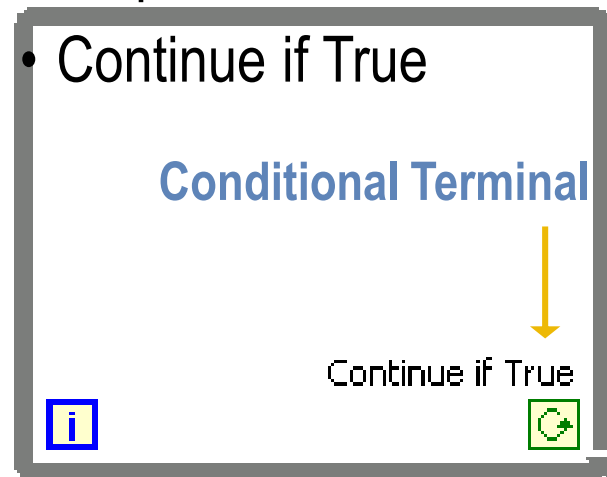
- Iteration terminal

- Returns number of times loop has executed.
- Is zero-indexed.



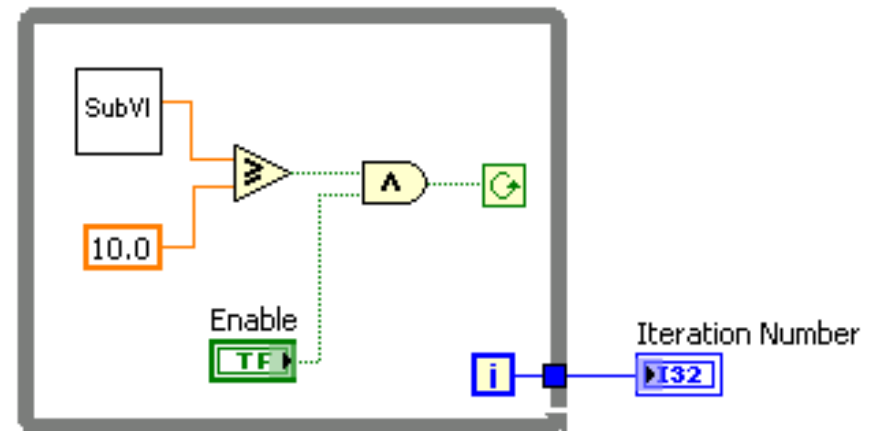
- Conditional terminal

- Defines when the loop stops.
- Has two options.
  - Stop if True



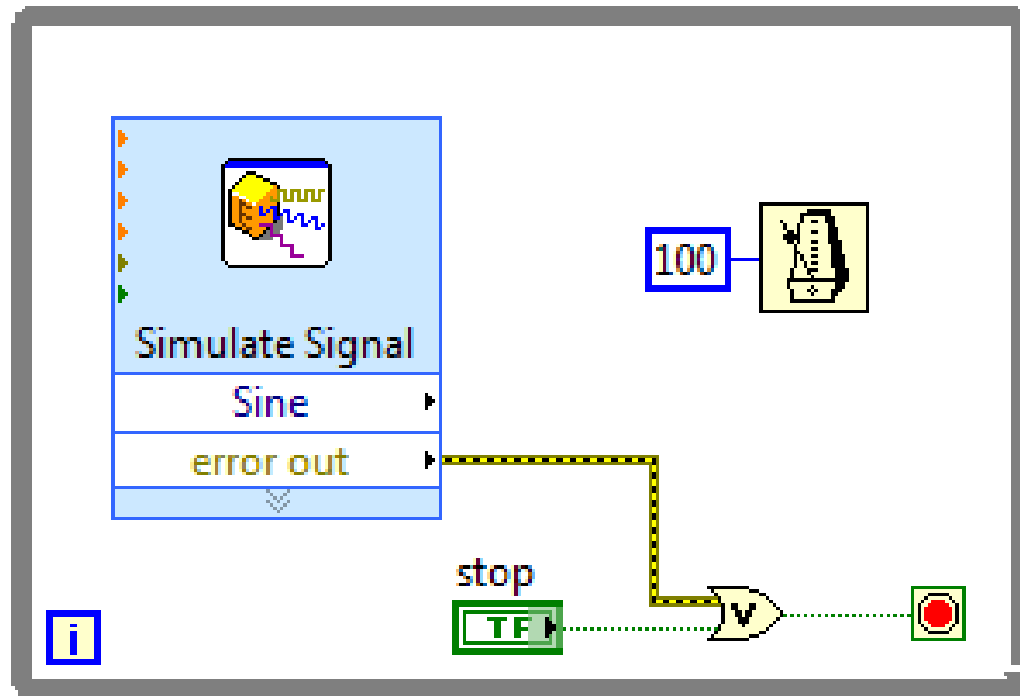
# Structure Tunnels

- Tunnels feed data into and out of structures.
- The tunnel is a block that appears on the border; the color of the block is related to the data type wired to the tunnel.
- When a tunnel passes data into a loop, the loop executes only after data arrive at the tunnel.
- Data pass out of a loop after the loop terminates.

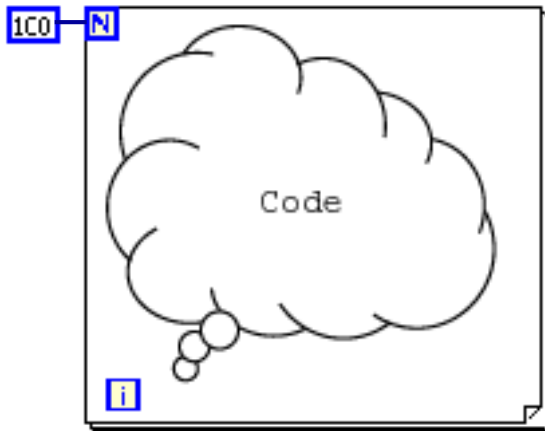


# While Loops – Error Checking and Error Handling

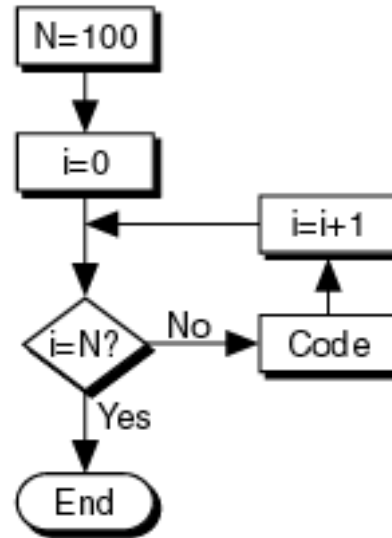
Use an error cluster in a While Loop to stop the While Loop if an error occurs.



# For Loops



LabVIEW For Loop



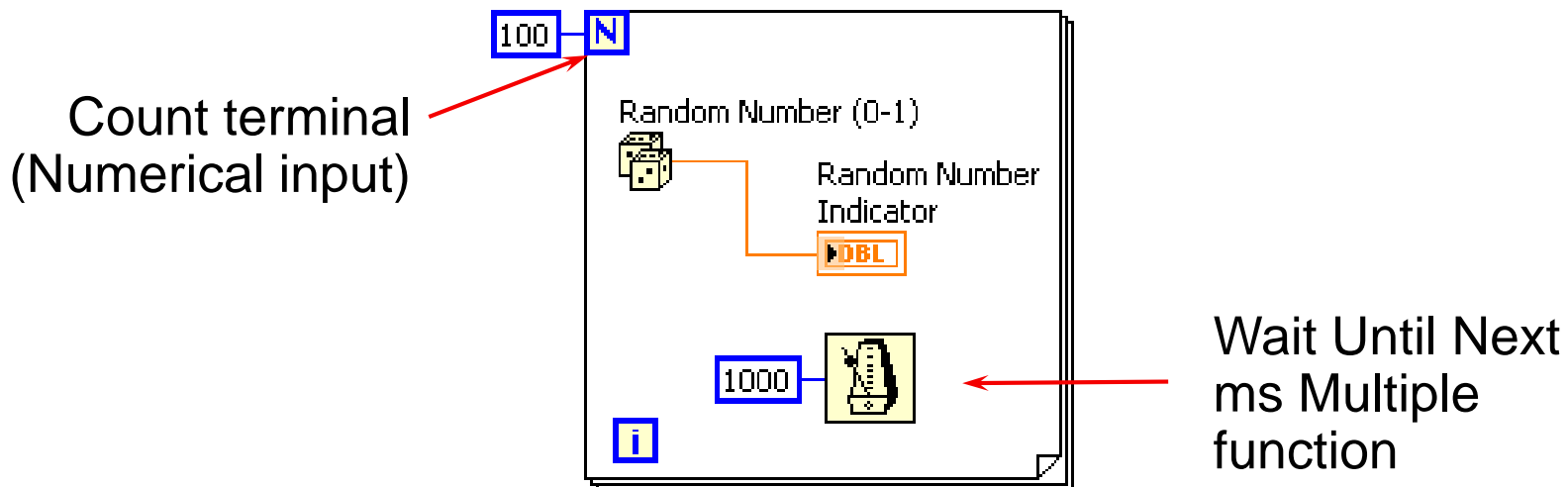
Flow Chart

```
N=100;  
i=0;  
Until i=N:  
    Repeat (code; i=i+1);  
End;
```

Pseudo Code

# For Loops

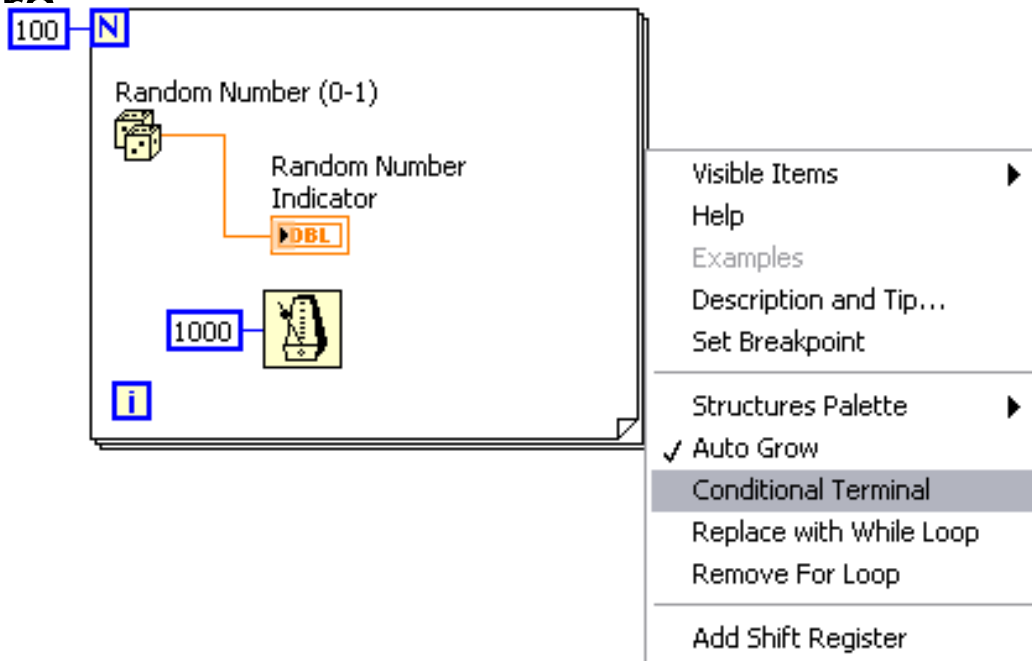
- In Structures subpalette of Functions palette
- Enclose code to be repeated and/or resize and add nodes inside boundary
- Executes diagram inside of loop a predetermined number of times





# For Loops – Conditional Terminal

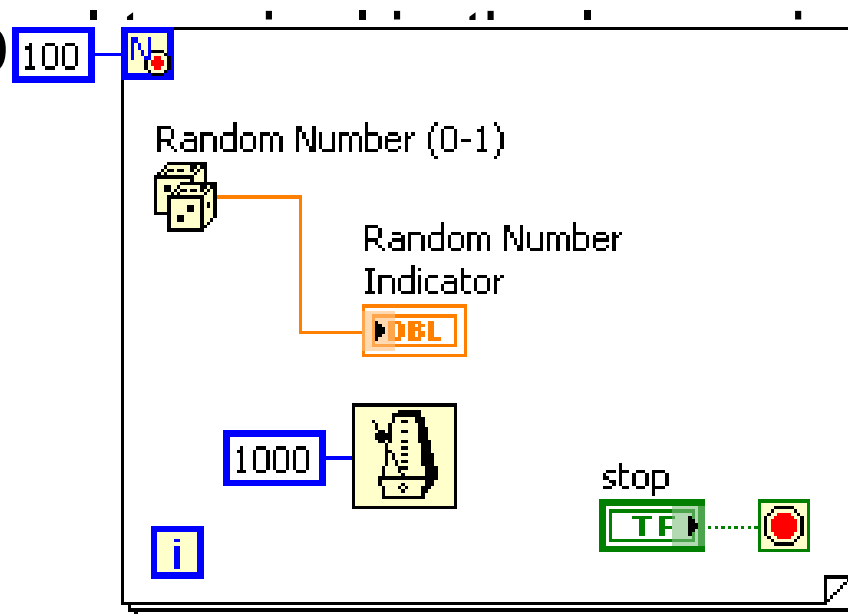
- You can add a conditional terminal to configure a For Loop to stop when a Boolean condition is true or an error occurs



# For Loops – Conditional Terminal

- For Loops configured with a conditional terminal have:

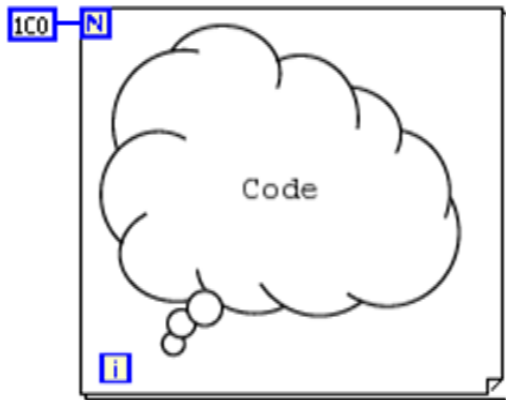
- A red glyph next to the count terminal.
- A condition



# For Loop/While Loop Comparison

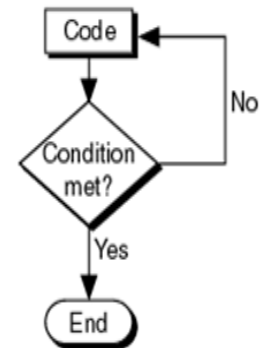
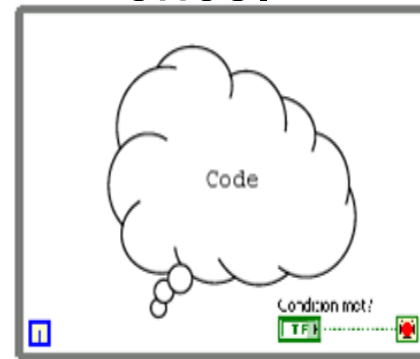
## For Loop

- Executes a set number of times unless a conditional terminal is added.
- Can execute zero times.
- Tunnels automatically output an array of data.



## While Loop

- Stops executing only if the value at the conditional terminal meets the condition.
- Must execute at least once.



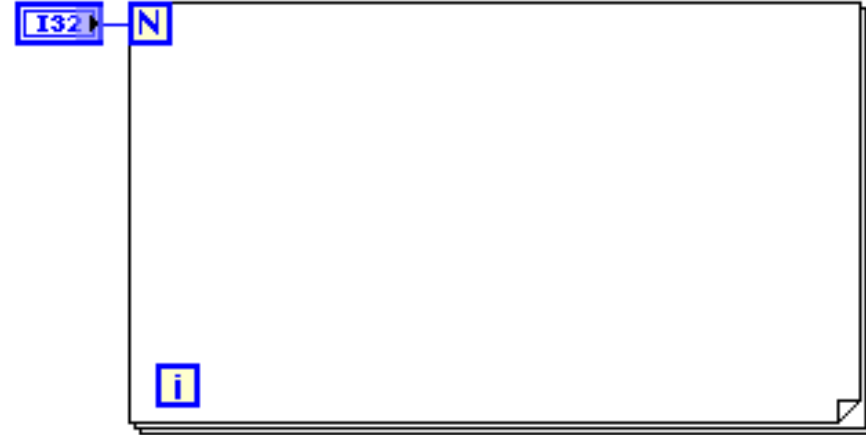
# For Loops – Numeric Conversion

- The number of iterations a For Loop executes must be specified in non-negative integers.
- If you wire a double-precision, floating-point numeric value to the count terminal, LabVIEW converts the numeric value to a 32-bit signed integer.

Double-Precision  
Floating Point

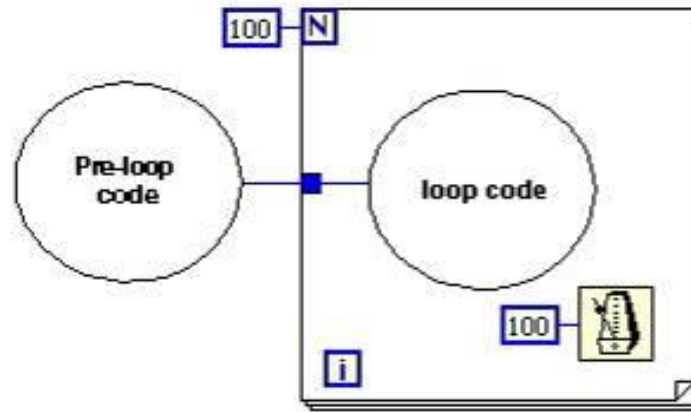
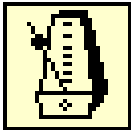


32-Bit Signed Integer

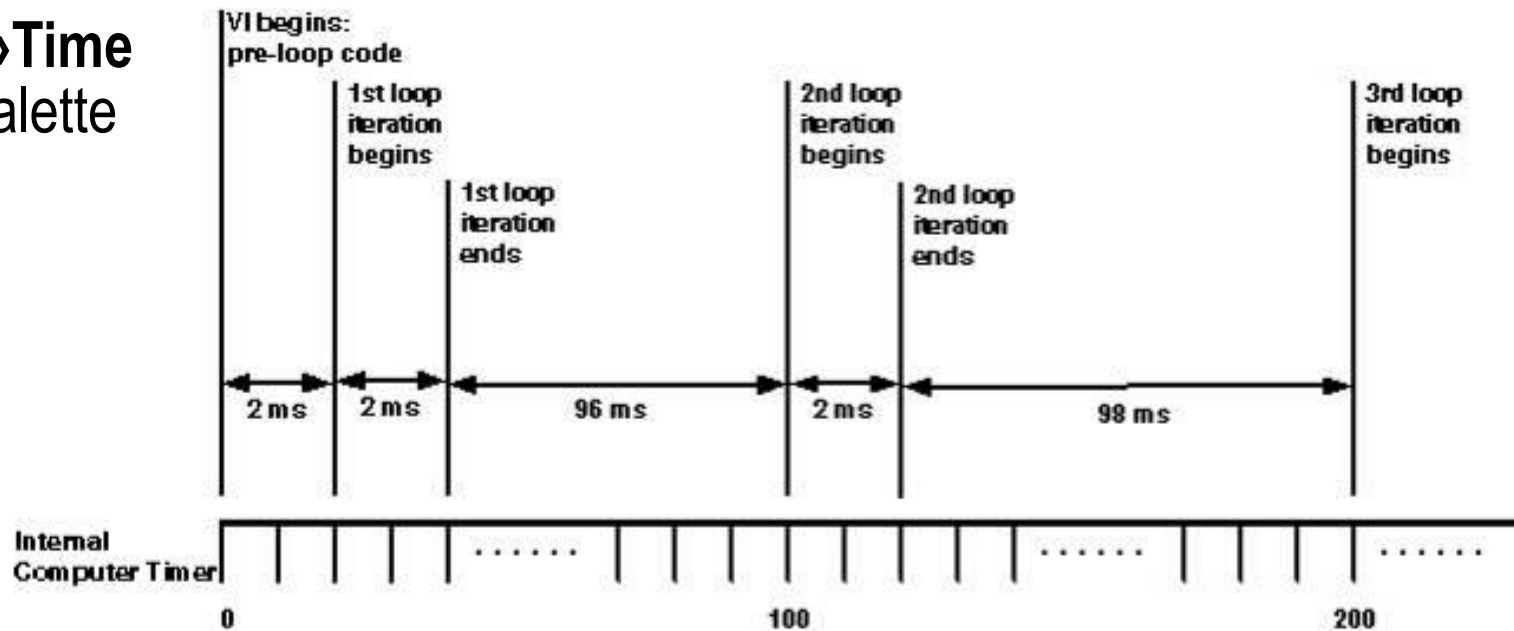


# Wait Functions

Wait Until Next  
ms Multiple

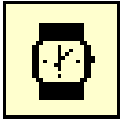


Functions»Time  
& Dialog palette



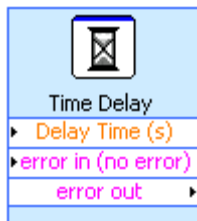
# Wait Functions

Wait (ms)

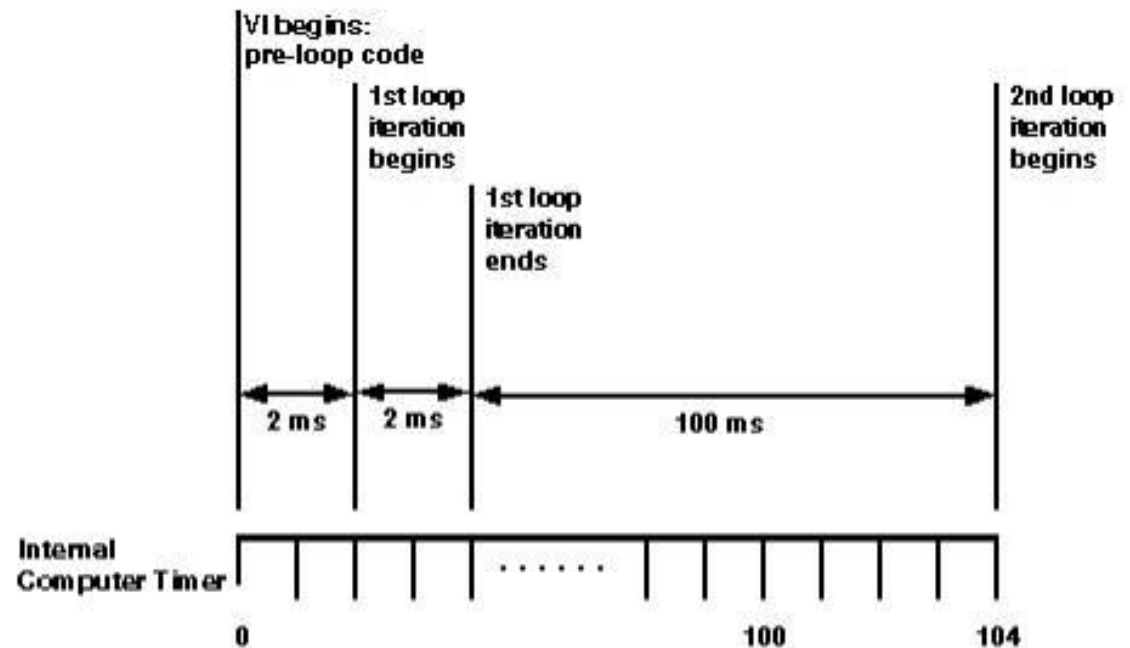
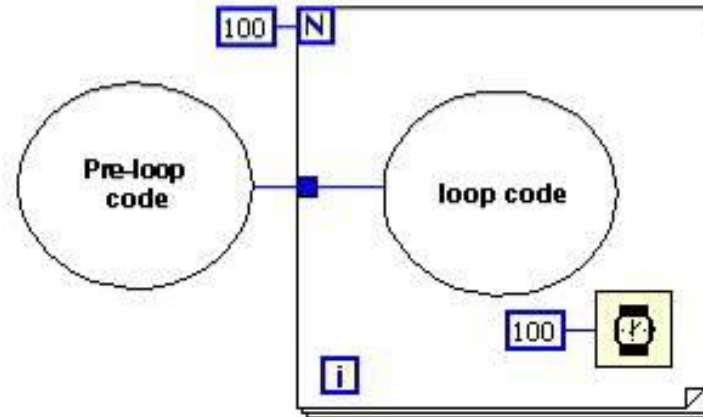


Functions»Time  
& Dialog palette

Time Delay



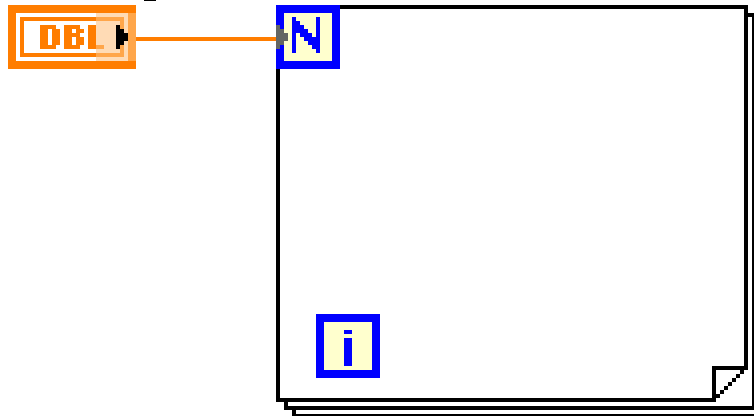
Functions»Time  
& Dialog palette



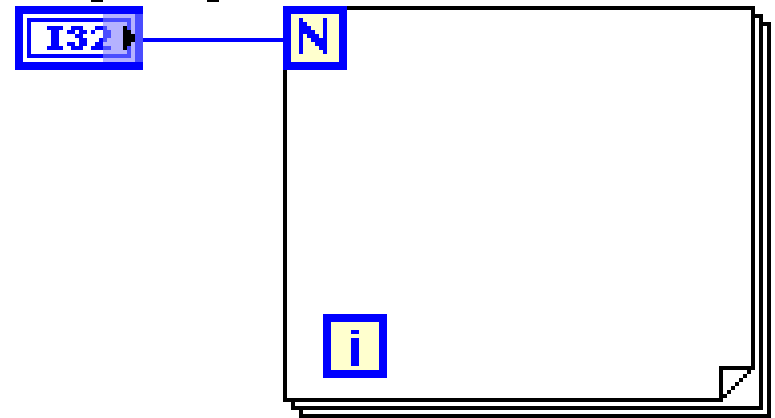
# Numeric Conversion

- Numerics default to double-precision (8 bytes) or long integer (4 bytes)
- LabVIEW automatically converts to different representations
- For Loop count terminal always converts to a long integer
- Gray *coercion* dot on terminal indicates conversion

Double-Precision,  
Floating-Point Numeric

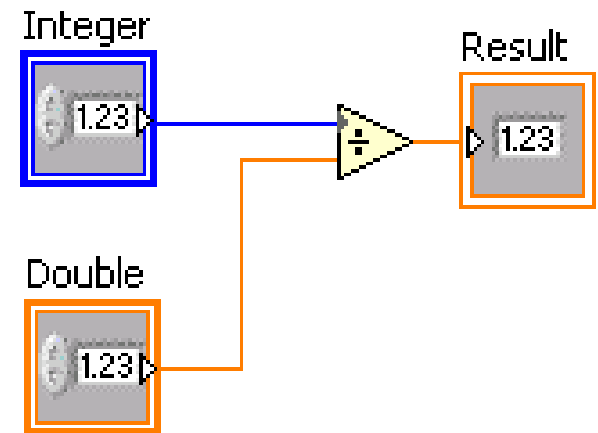


Long Integer



# Numeric Conversion

- LabVIEW chooses the representation that uses more bits.
- If the number of bits is the same, LabVIEW chooses unsigned over signed.
- To choose the representation, right-click on the terminal and select Representation.

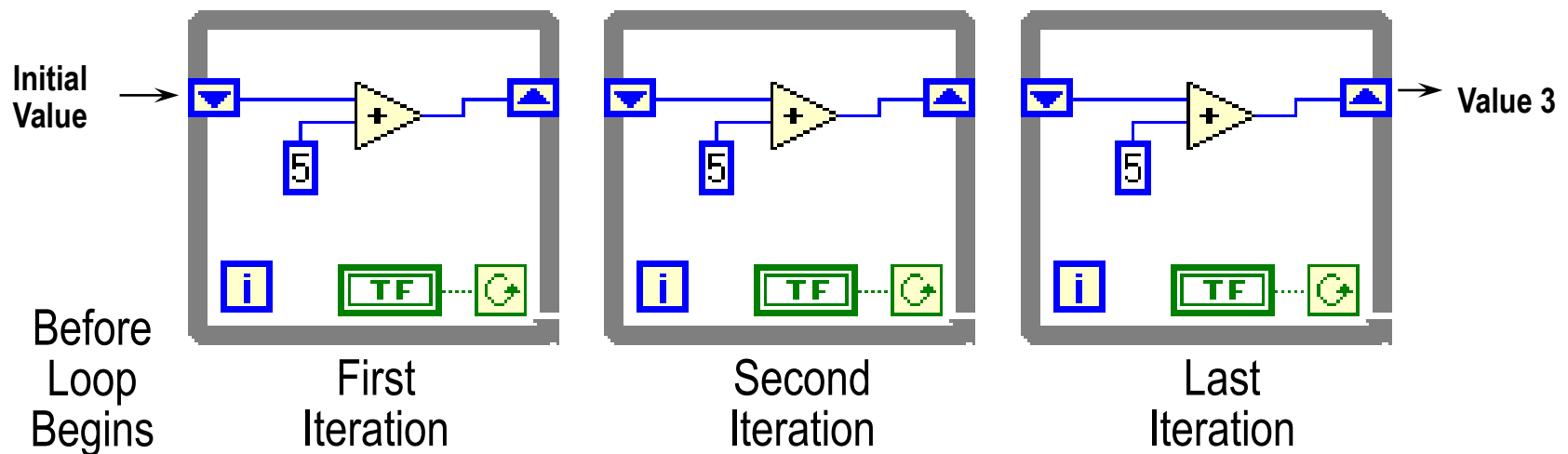


- When LabVIEW converts floating-point numerics to integers, it rounds to the nearest integer. LabVIEW rounds x.5 to the nearest even integer.  
For example, LabVIEW rounds 2.5 to 2 and 3.5 to 4.



# Accessing Previous Loop Data – Shift Register

- Available at left or right border of loop structures
- Right-click the border and select Add Shift Register
- Right terminal stores data on completion of iteration
- Left terminal provides stored data at beginning of next iteration

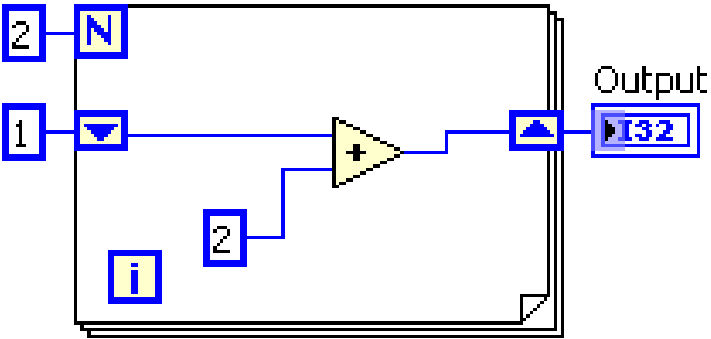
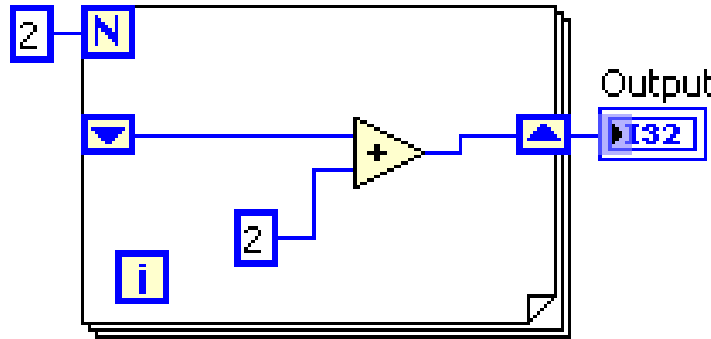


# Initializing Shift Registers

Run once

VI finishes

Run again

Block Diagram	1st run	2nd run
<p>Initialized Shift Register</p> 	<p>Output = 5</p>	<p>Output = 5</p>
<p>Not Initialized Shift Register</p> 	<p>Output = 4</p>	<p>Output = 8</p>

# Use Default if Unwired

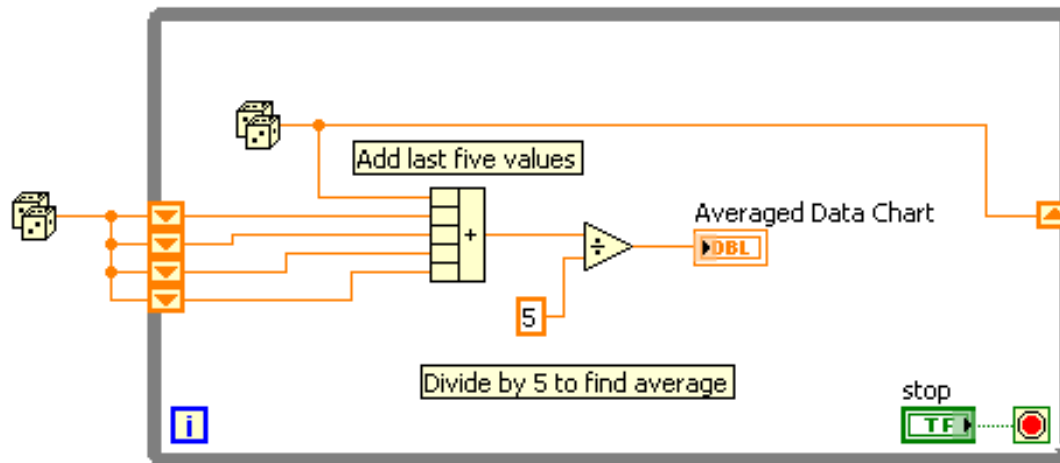
Default values vary by data type:

Data Type	Default Value
Numeric	0
Boolean	FALSE
String	Empty

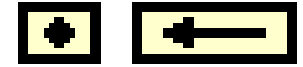
Uninitialized shift registers use default values for first run.

# Multiple Previous Iterations

- Stacked shift registers remember values from multiple previous iterations and carry those values to the next iterations.
- Right-click the left shift register and select **Add Element** from the shortcut menu to stack a shift register.



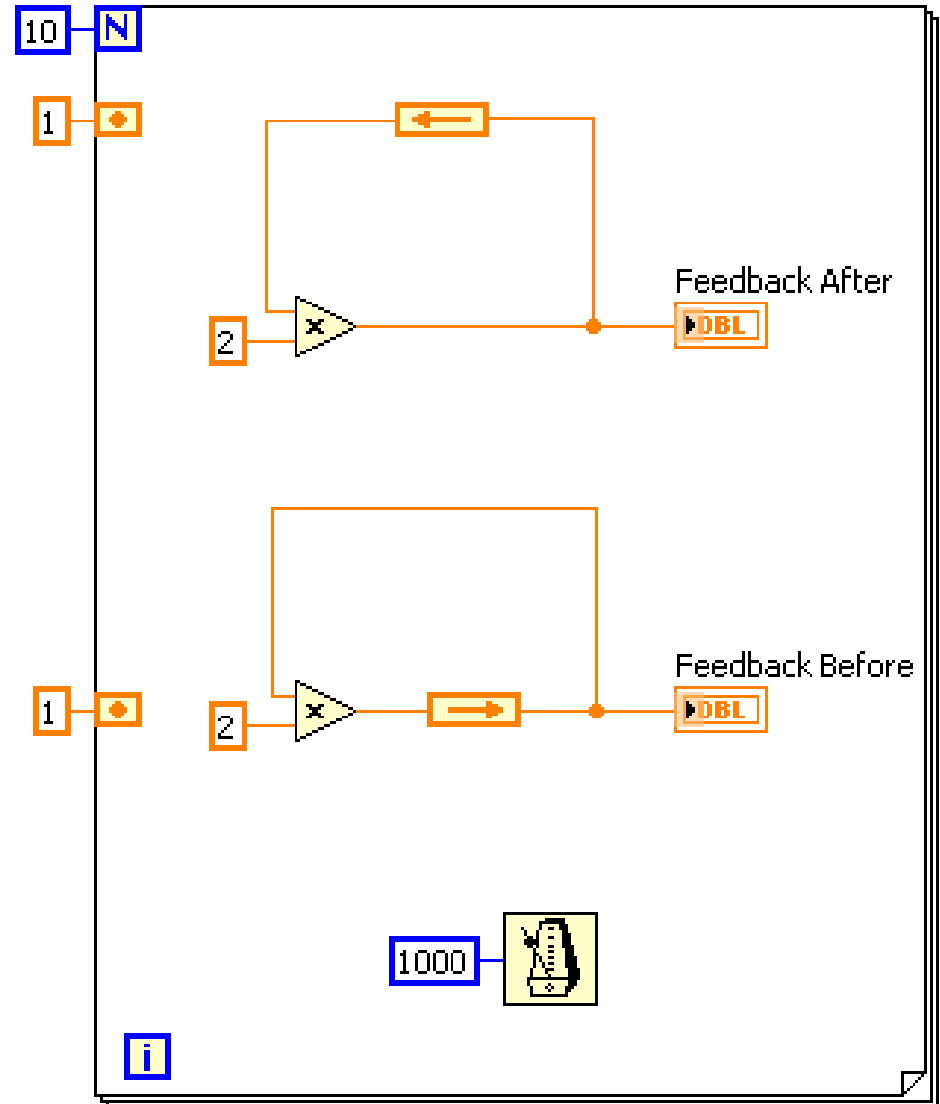
# Feedback Nodes



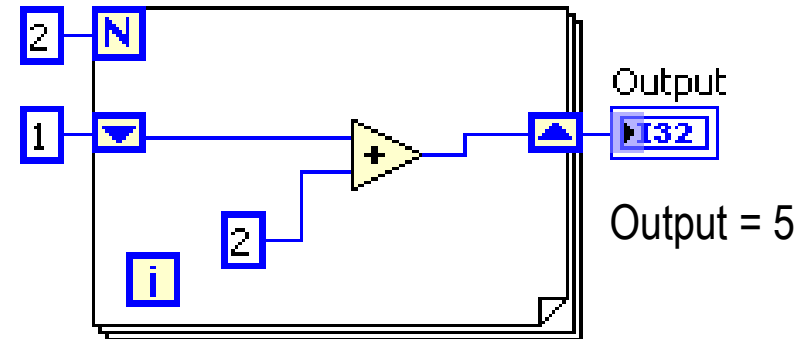
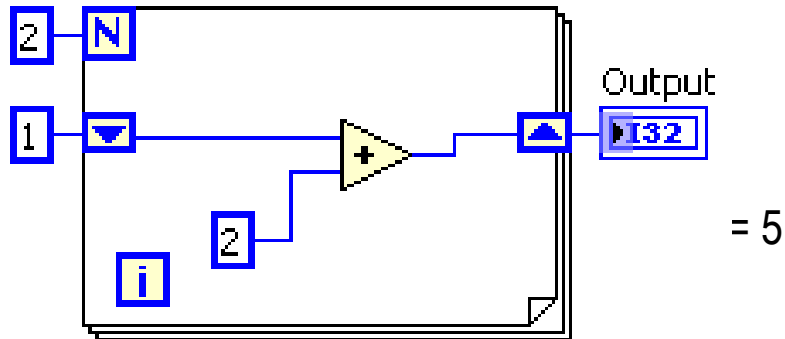
- Appears automatically in a For Loop or While Loop if you wire the output of a subVI, function, or group of subVIs and functions to the input of that same VI, function, or group.
- Stores data when the loop completes an iteration, sends that value to the next iteration of the loop, and transfers any data type

# Feedback Node

- Wire from output to input to automatically create a feedback node  
<OR>
- Place a feedback node from the **Functions»Structures** palette



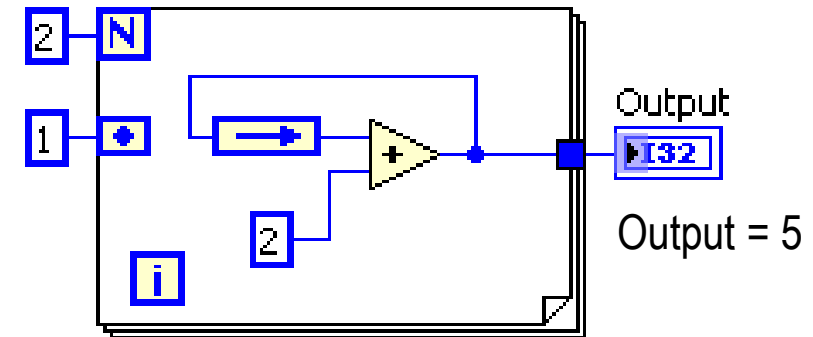
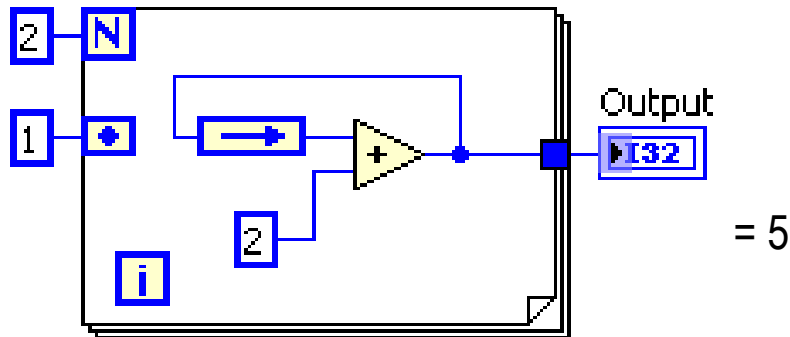
# Initialized Shift Registers & Feedback Nodes



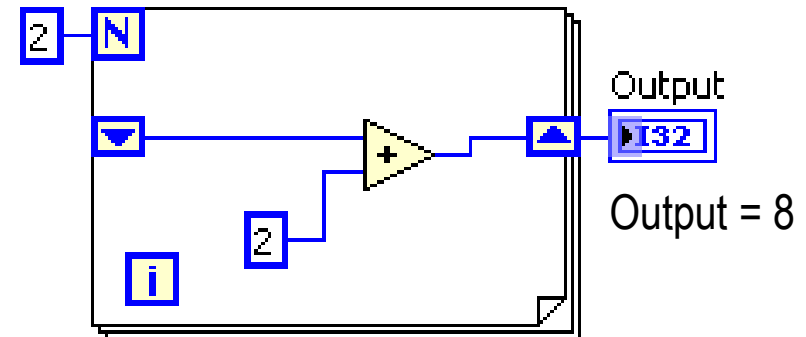
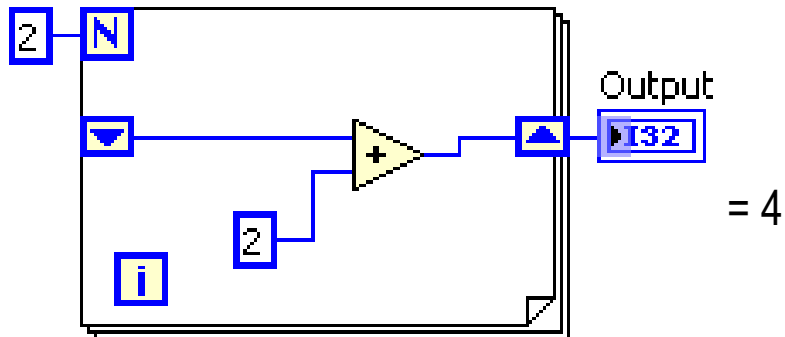
Run Once

VI stops execution

Run Again



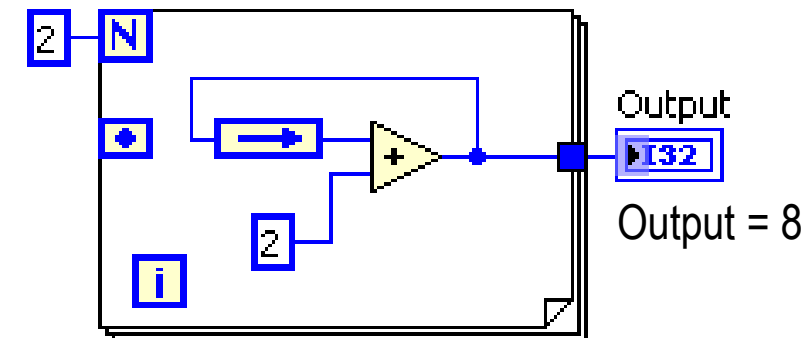
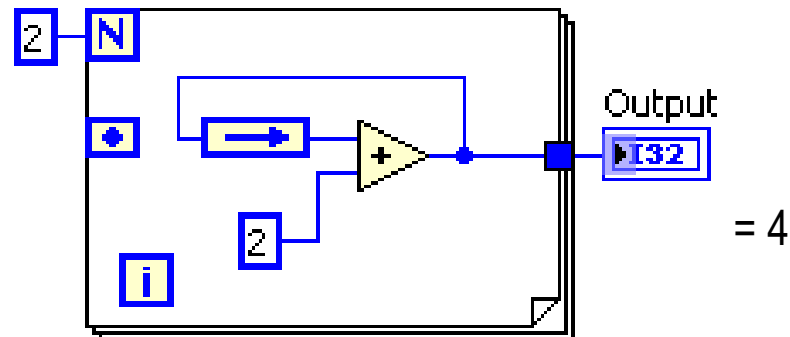
# Uninitialized Shift Registers & Feedback Nodes



Run Once

VI stops execution

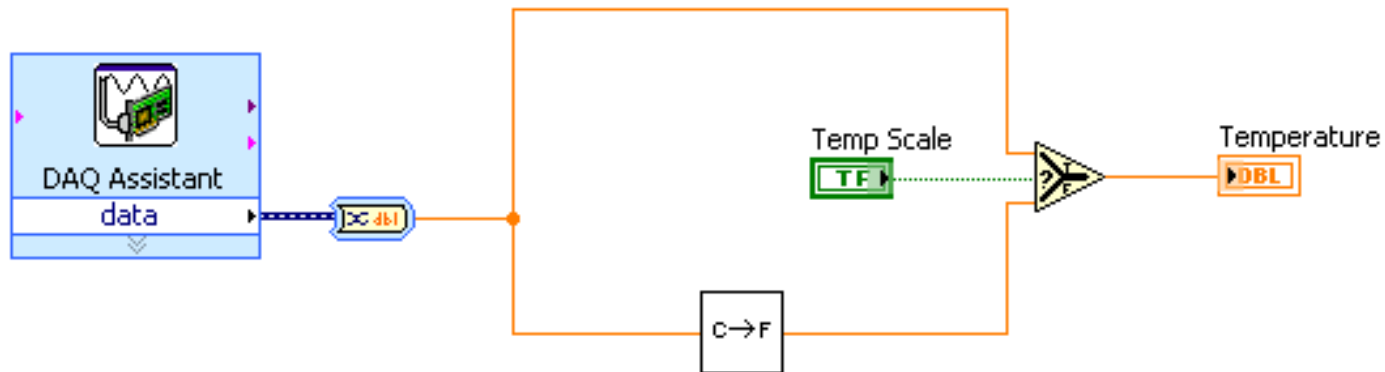
Run Again





# Simple Decision: Select Function

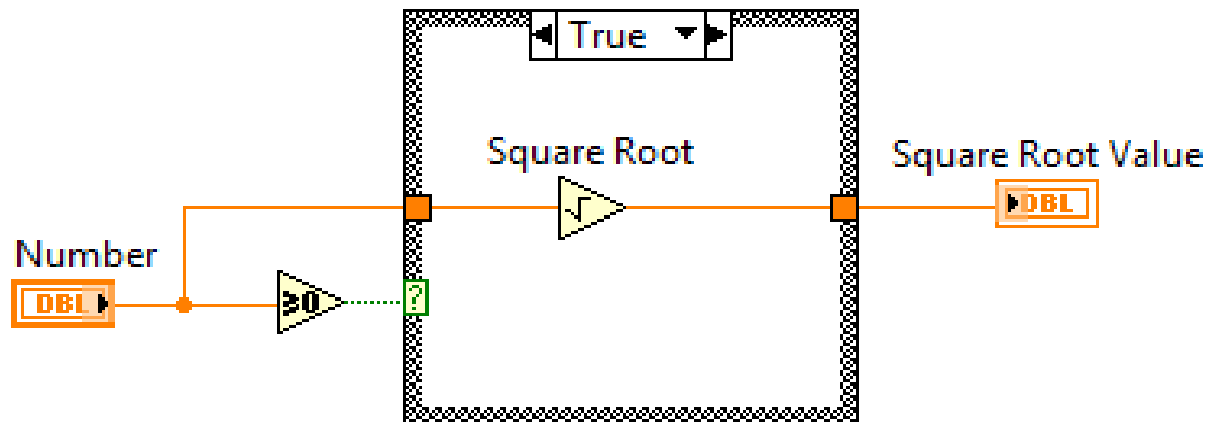
- If Temp Scale is TRUE, pass top input;  
if temp scale is FALSE, pass bottom input.



- If the decision to be made is more complex than a Select function can execute, a Case Structure may be what is required.

# Case Structures

- Have two or more subdiagrams or cases.
- Use an input value to determine which case to execute.
- Execute and display only one case at a time.
- Are similar to **case** statements or **if...then...else** statements in text-based programming languages.



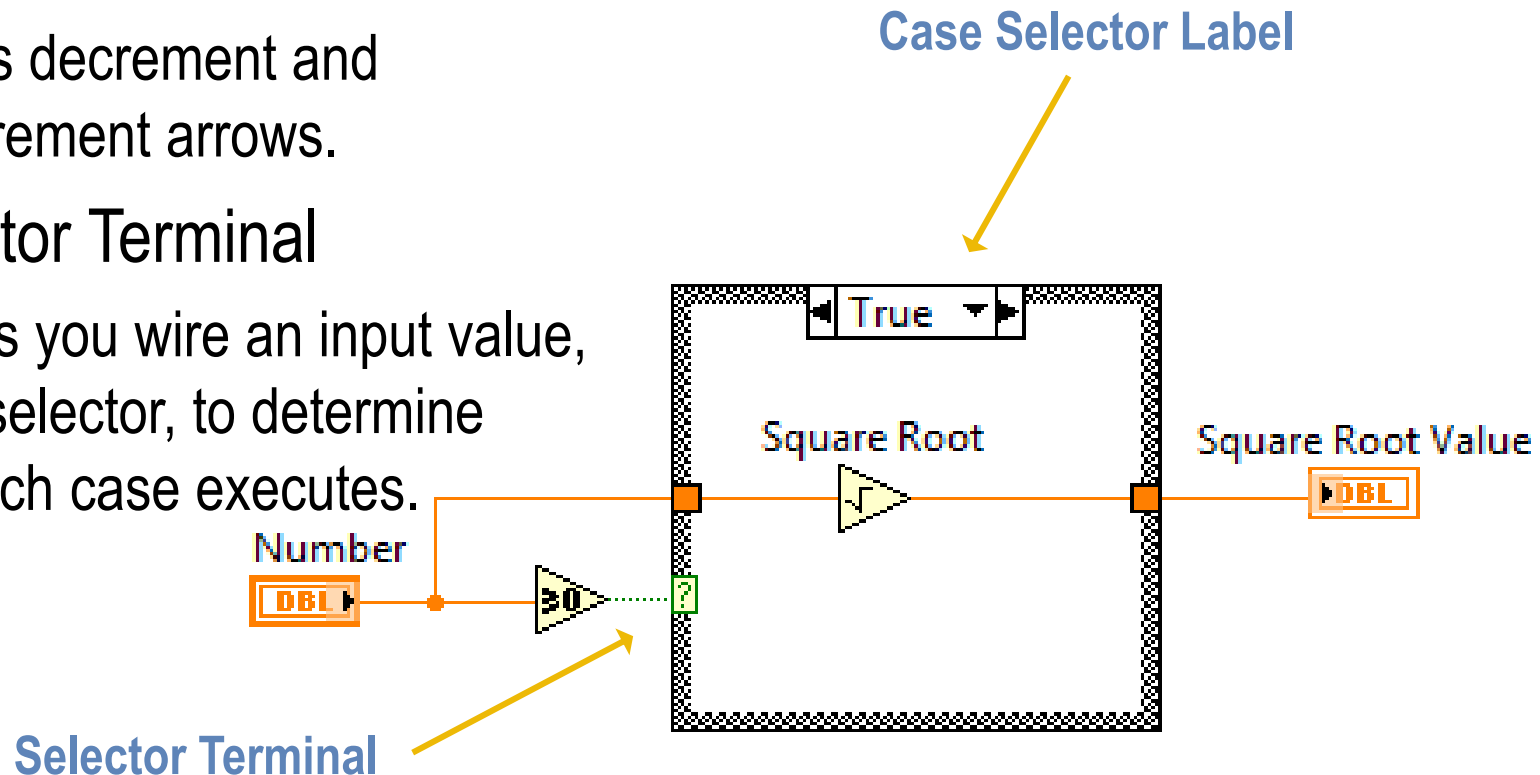
# Case Structures

## – Case Selector Label

- Contains the name of the current case.
- Has decrement and increment arrows.

## – Selector Terminal

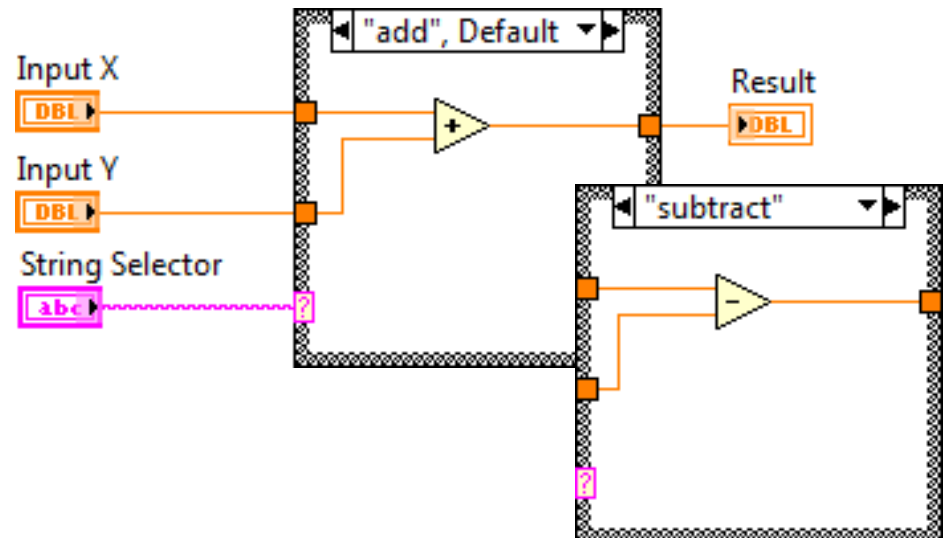
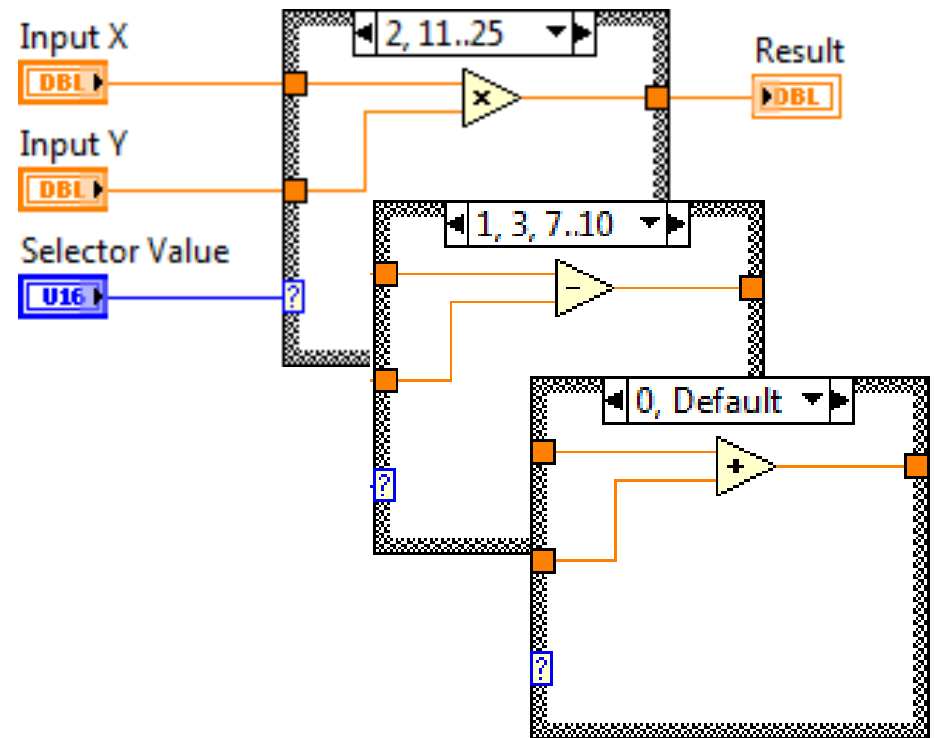
- Lets you wire an input value, or selector, to determine which case executes.



# Case Structures

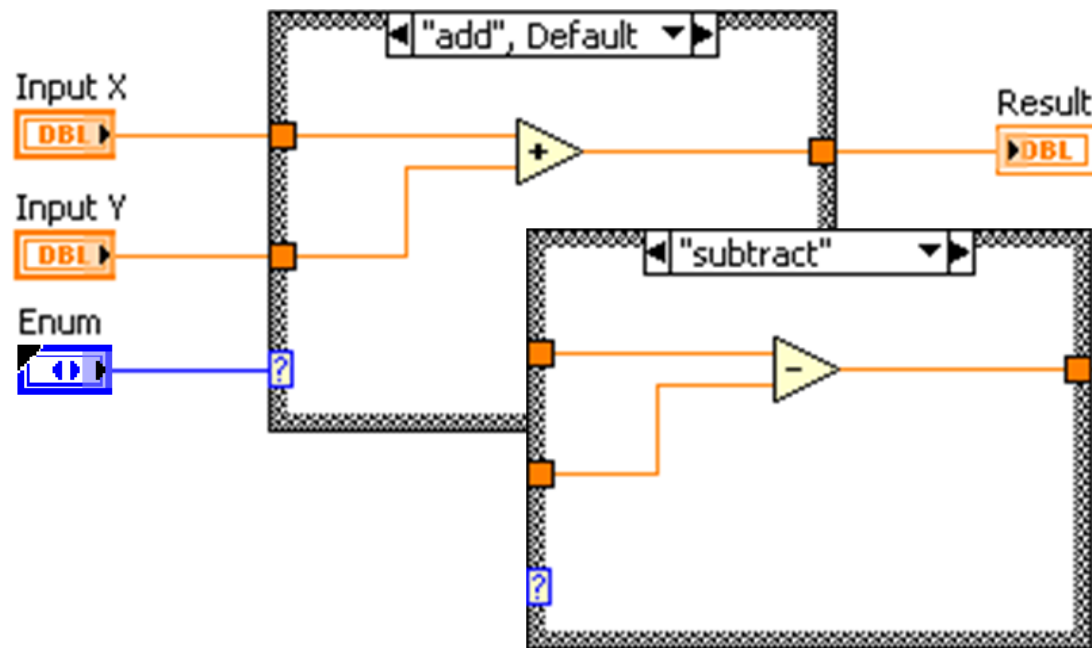
Selector terminal data types:

- Boolean
  - True case and False Case
- Error Cluster
  - Error Case and No Error Case
- Integer, string, or enum
  - Structure can have any number of cases.
  - Include a Default diagram to avoid listing every possible input value.



# Enum Case Structure

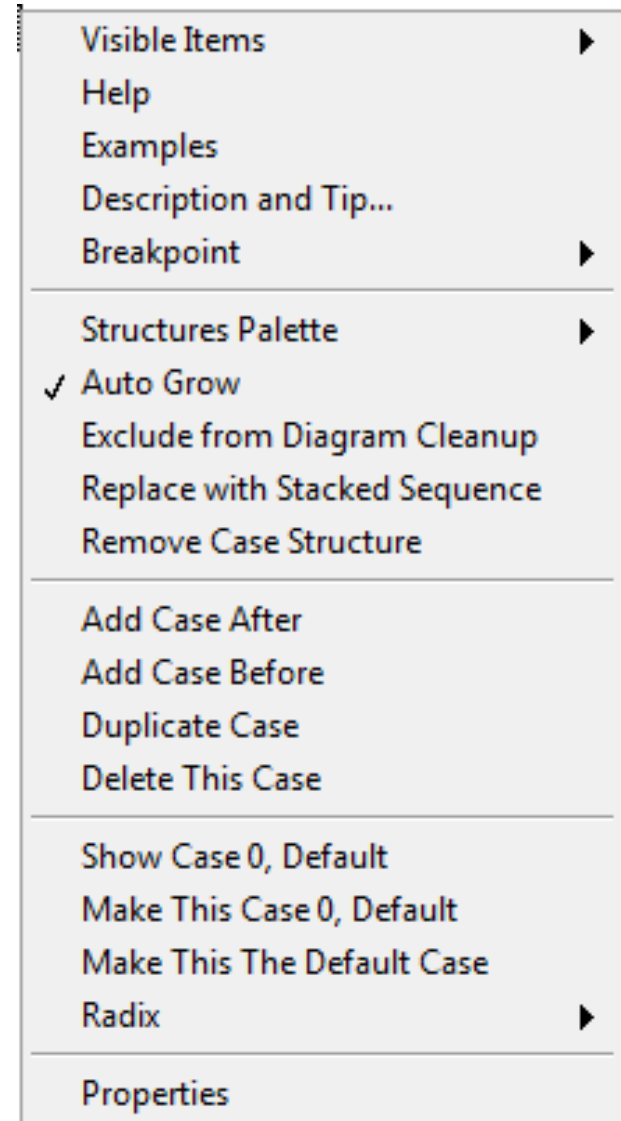
- Gives users a list of items from which to select
- The case selector displays a case for each item in the enumerated type control



# Shortcut Menu

Use the shortcut menu of a Case structure to:

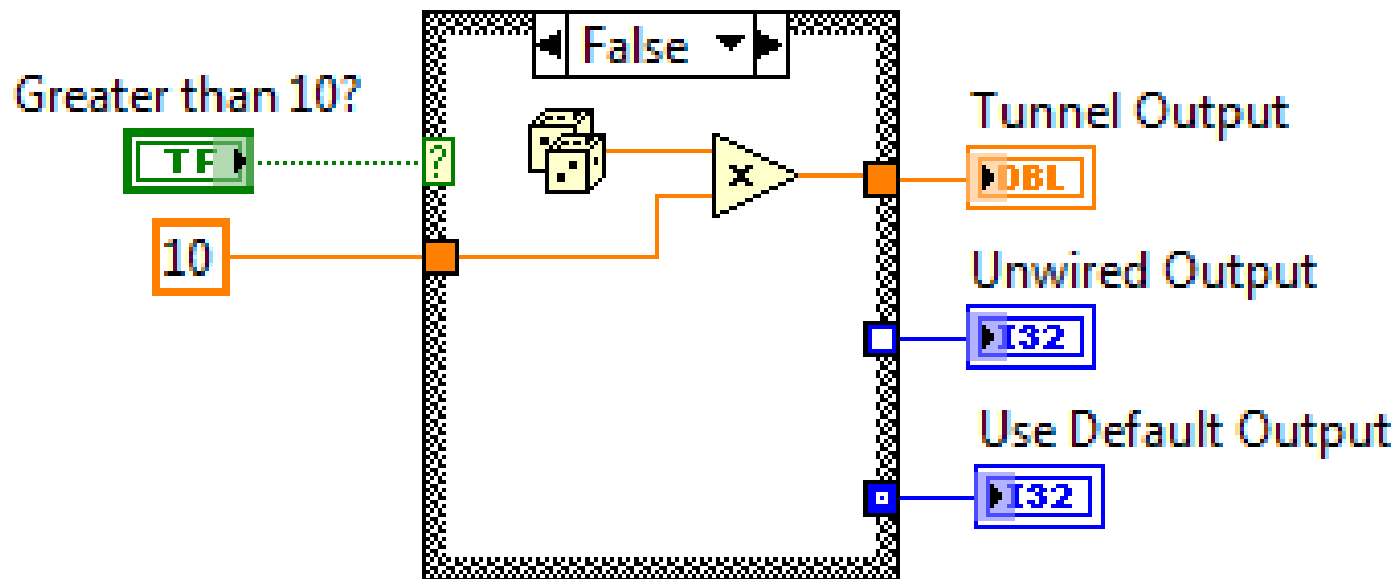
- Customize the structure and diagrams.
- Remove or replace the structure.
- Add, duplicate, remove, or rearrange cases.
- Specify the Default case.
- Switch cases.



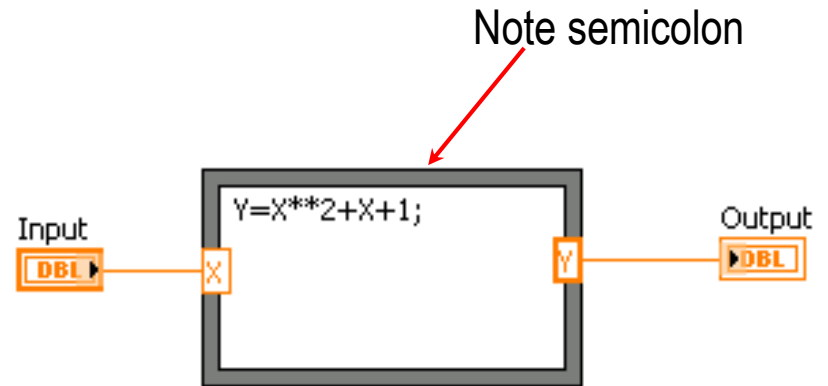
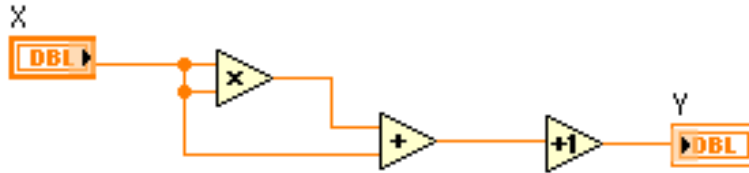
# Input and Output Tunnels

You can create multiple input and output tunnels.

- Inputs tunnels are available to all cases if needed.
- You must define each output tunnel for each case.



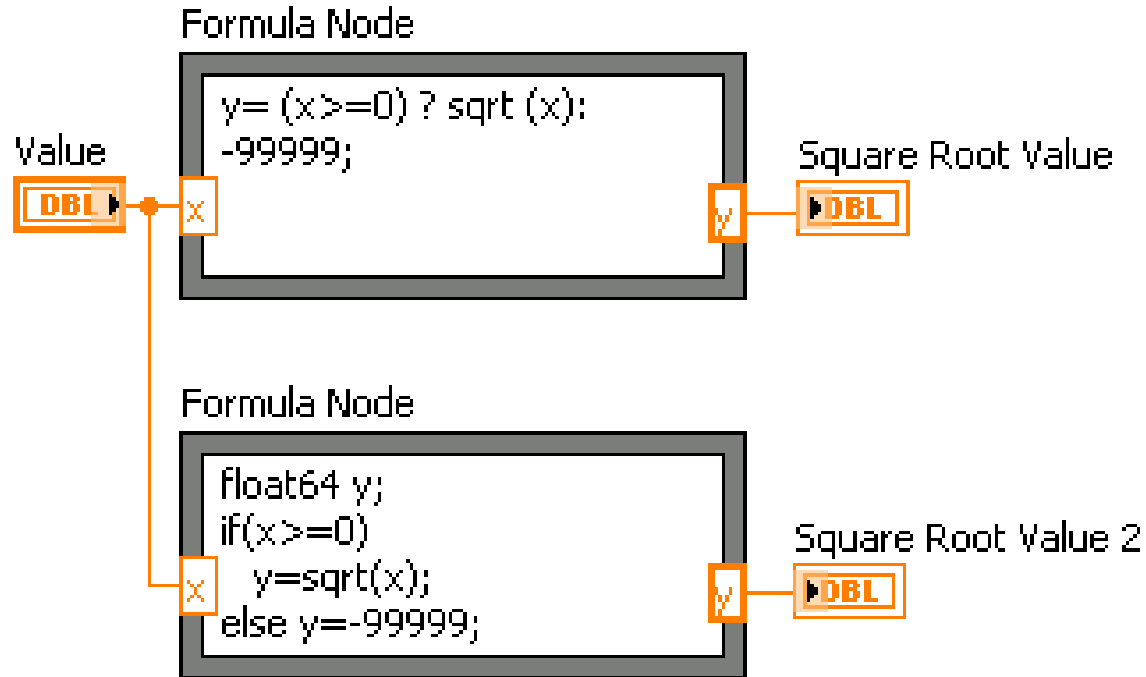
# Formula Node



- In the Structures subpalette
- Implement complicated equations
- Variables created at border
- Variable names are case sensitive
- Each statement must terminate with a semicolon (;)
- Context Help Window shows available functions



# Decision Making with Formula Nodes

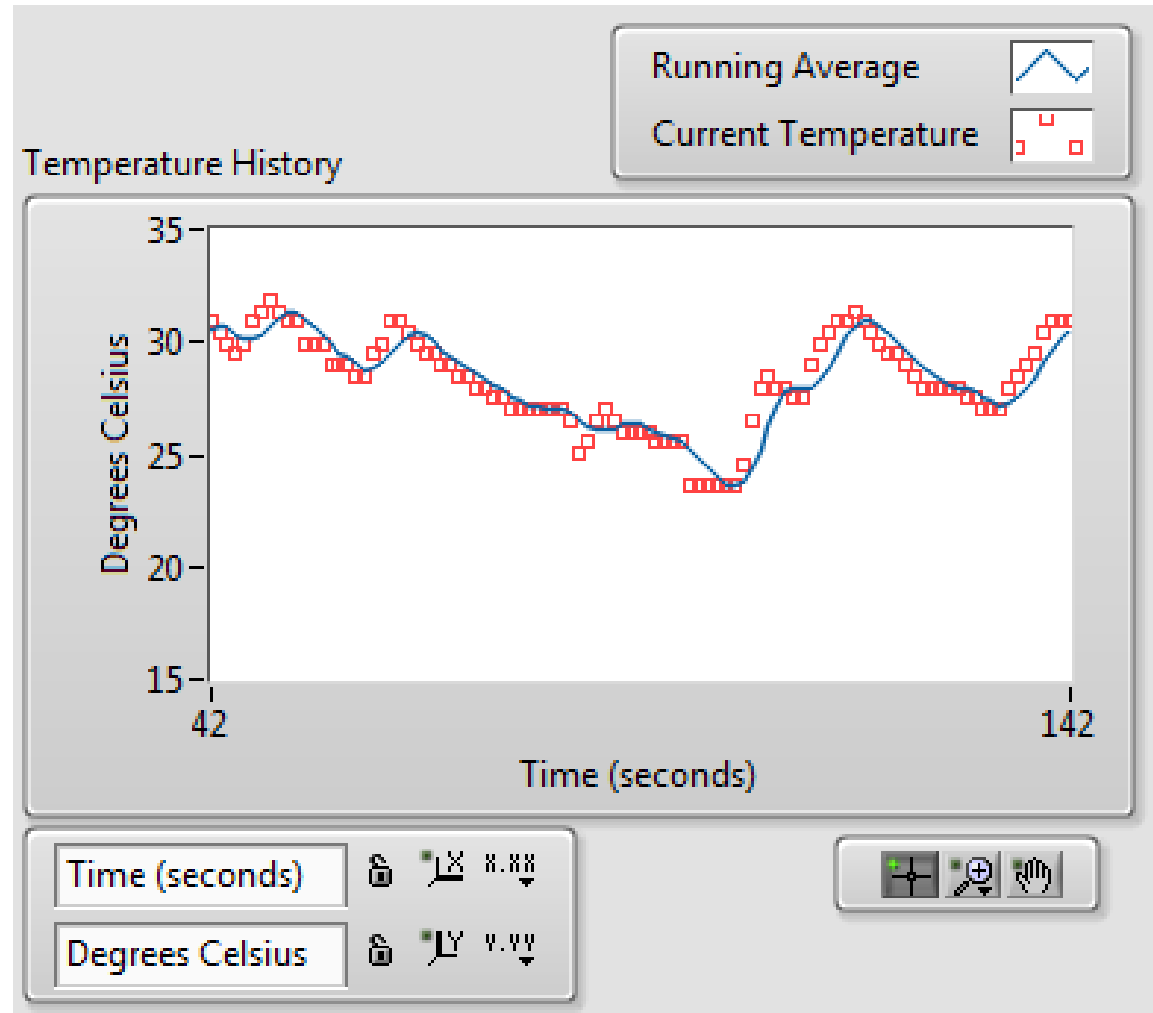


Two different ways of using an if-then statement in a Formula Node

Both structures produce the same result

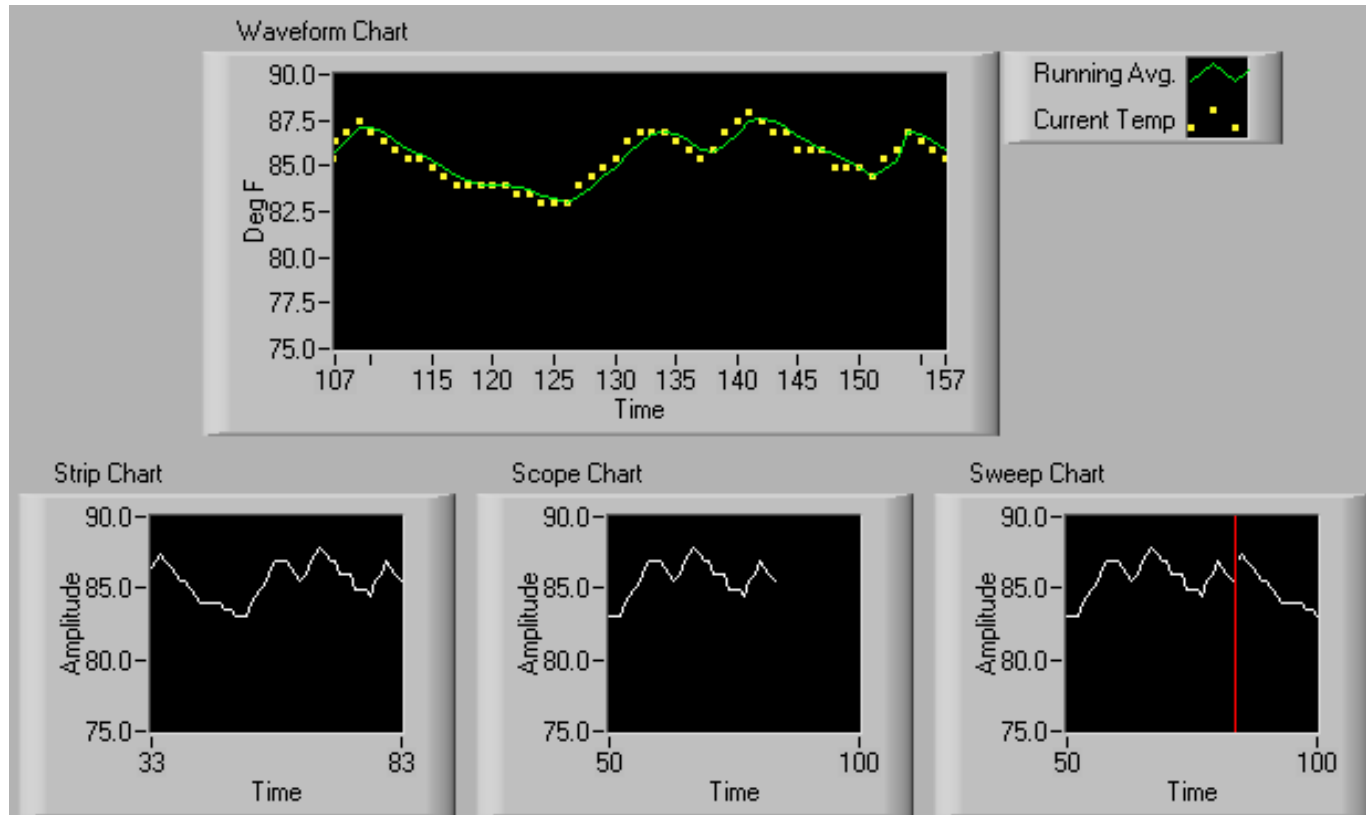
# Plotting Data – Waveform Chart

- Waveform chart is a special type of numeric indicator.
- Waveform charts display single or multiple plots.



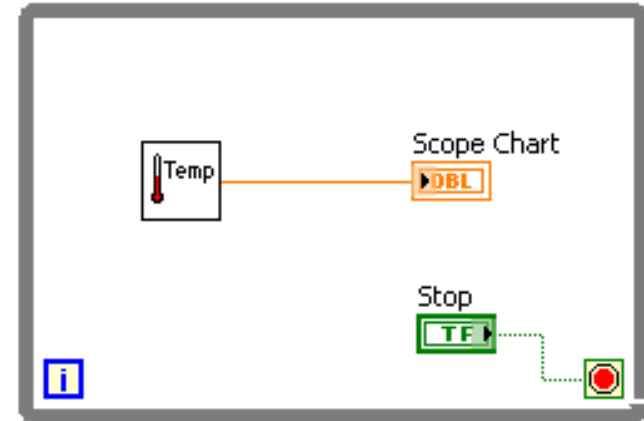
# Waveform Charts

Selected from the **Controls»Graphs and Charts** palette

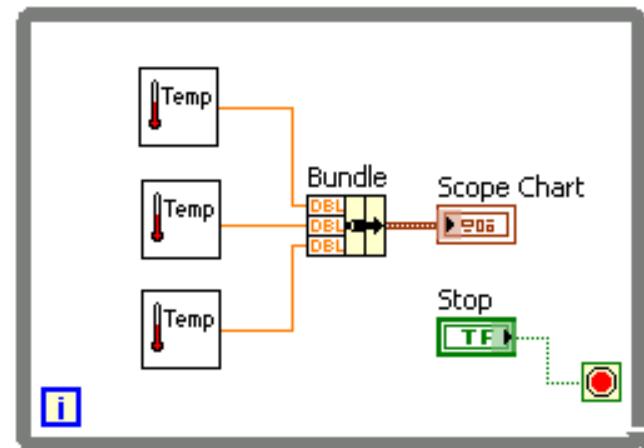


# Wiring to Charts

## Single-Plot Chart

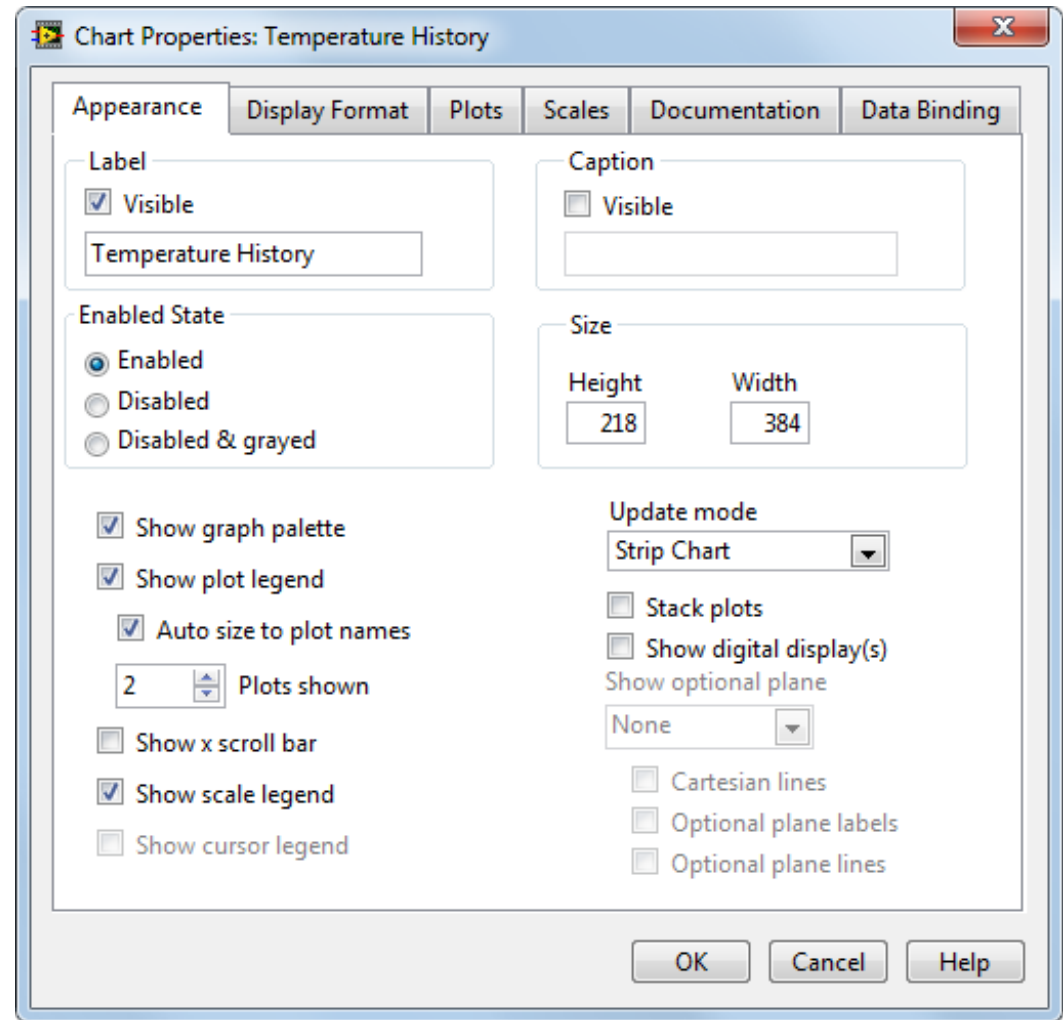


## Multiple-Plot Chart



# Waveform Chart Properties

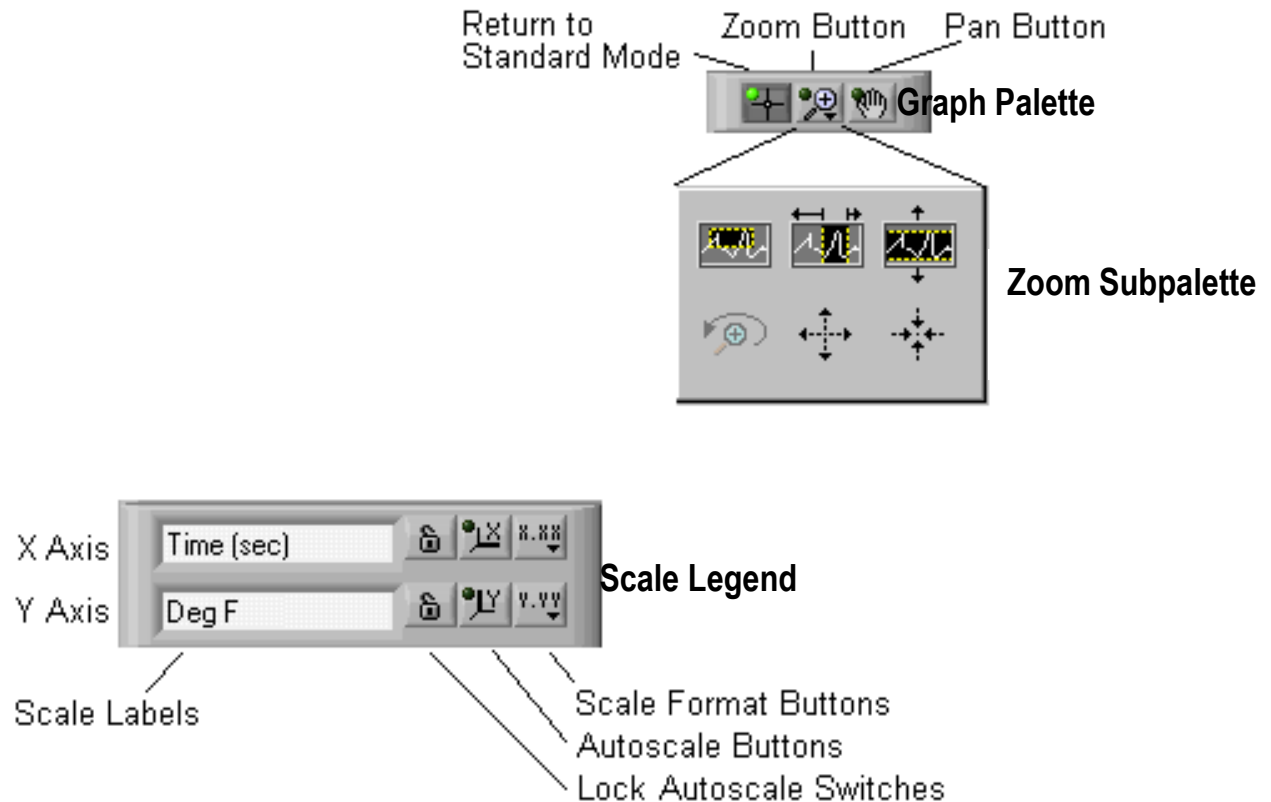
- Extensive plot customization lets you:
  - Show or hide legends.
  - Change color and line styles.
  - Change interpolation styles.



# Customizing Charts and Graphs

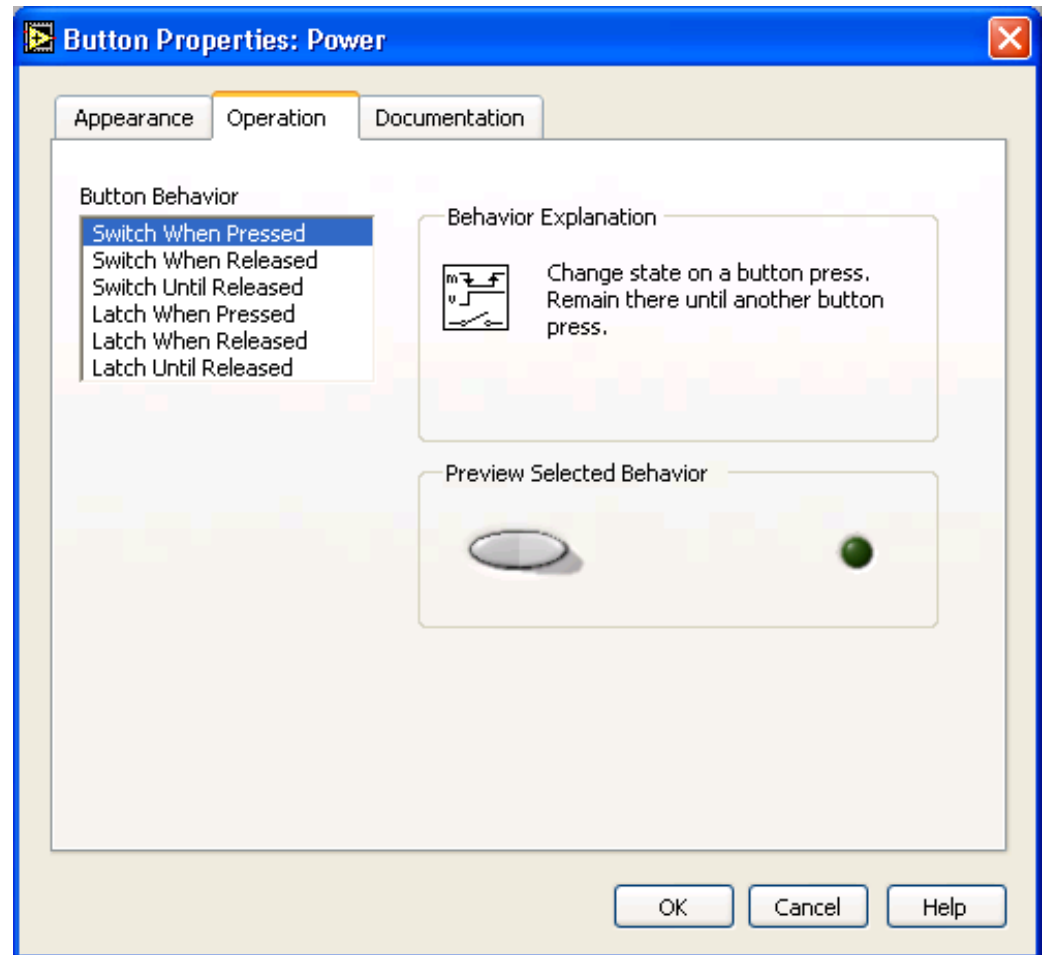
Right-click and select **Visible Items** to view the following items:

- Plot Legend
- Digital Display
- Scrollbar
- X and Y Scale
- Graph Palette
- Scale Legend



# Mechanical Action

- Switch action — Control is toggled until changed by hand
- Latch action — Control reverts to default state when read by diagram



# Summary

- Two structures to repeat execution: While Loop and For Loop
- Loop timing controlled using Wait Until Next ms Multiple function, the Wait (ms) function, or the Time Delay Express VI.
- Coercion dots appear where LabVIEW coerces a numeric representation of one terminal to match the numeric representation of another terminal
- Feedback nodes and shift registers transfer data values from one iteration to the next
- Use shift registers only when more than one past iteration is needed



# Summary

- The Select function is used to choose between two inputs dependant on a boolean input.
- A Case structure has two or more cases. Only one case is visible at a time, and the structure executes only one case at a time.
- If the case selector terminal is Boolean, the structure has a TRUE case and a FALSE case. If the selector terminal is an integer, string, or enumerated type value, the structure can have up to  $2^{31}-1$  cases.
- Inputs are available to all cases, but cases do not need to use each input. If at least one output tunnel is not defined, all output tunnels on the structure appear as white squares.
- Formula Nodes are useful for complicated equations and for using existing text-based code. A semicolon (;) must terminate each statement.

# Summary

- The waveform chart is a special numeric indicator that displays one or more plots. The waveform chart has the following three update modes:
  - A strip chart shows running data continuously scrolling from left to right across the chart.
  - A scope chart shows one item of data, such as a pulse or wave, scrolling partway across the chart from left to the right.
  - A sweep works similarly to a scope except it shows the old data on the right and the new data on the left separated by a vertical line.