LabVIEW Lecture 1

Ertugrul Karademir

What is LabVIEW?

- Graphical programming environment
- Measurement, testing, control applications
- Hardware control
- Flowchart representation
- Object Oriented Approach

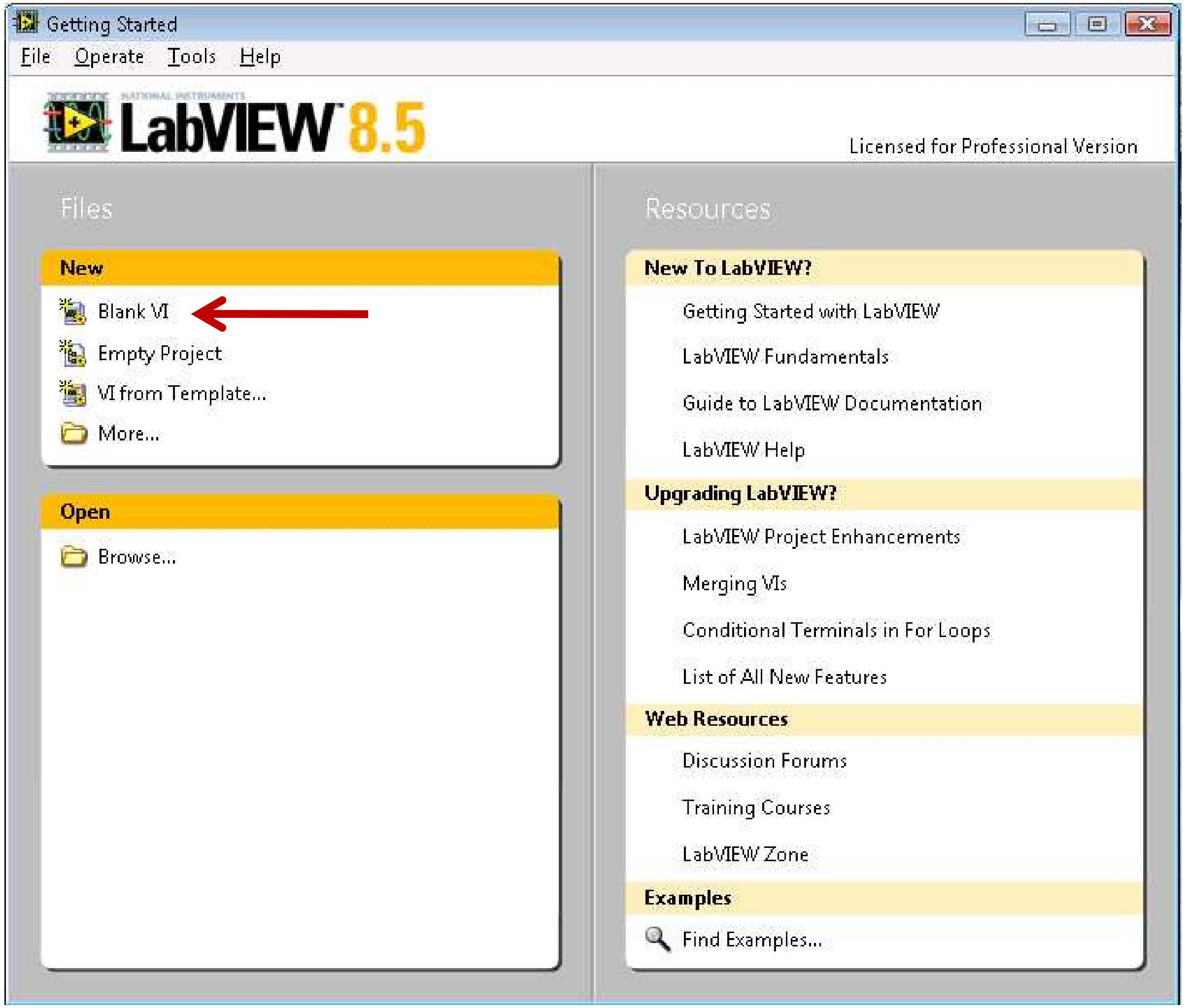
What is LabVIEW?

Ertugrul
Bİlkent
F44G44444
LabVIEW Professional Development System

Copyright (c) 2007 National Instruments. All rights reserved.

Version 8.5 - Initializing plug-ins

What is LabVIEW?



Getting Started

File Operate Tools Help

LabVIEW 8.5

Licensed for Professional Version

Files

New

- Blank VI
- Empty Project
- VI from Template...
- More...

Open

- Browse...

Resources

New To LabVIEW?

- Getting Started with LabVIEW
- LabVIEW Fundamentals
- Guide to LabVIEW Documentation
- LabVIEW Help

Upgrading LabVIEW?

- LabVIEW Project Enhancements
- Merging VIs
- Conditional Terminals in For Loops
- List of All New Features

Web Resources

- Discussion Forums
- Training Courses
- LabVIEW Zone

Examples

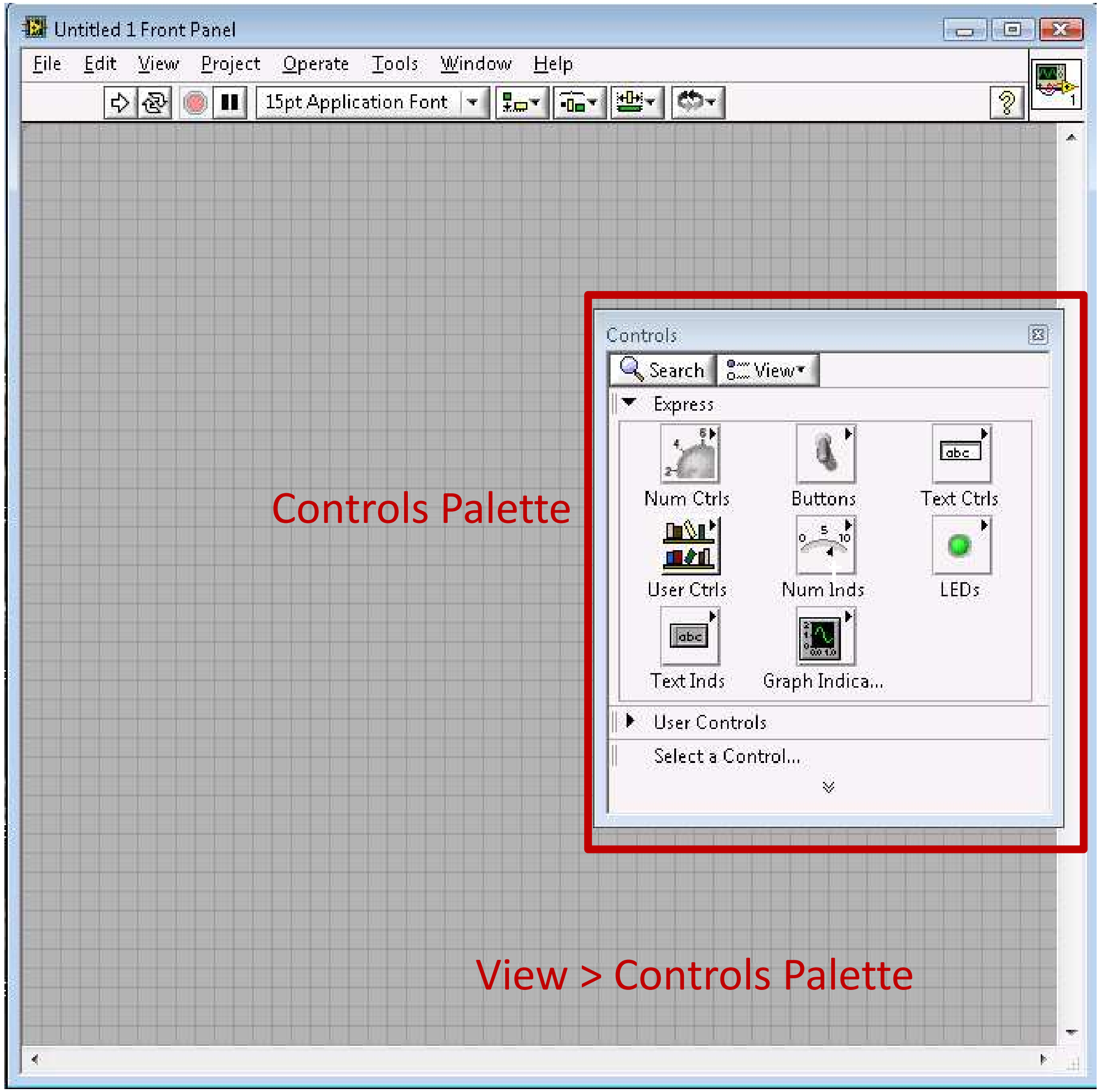
- Find Examples...

Virtual Instrument

- LabVIEW programming paradigm
    1. Program control with a front panel
    2. Algorithm with block diagram

- Easy to debug
- Hard to program

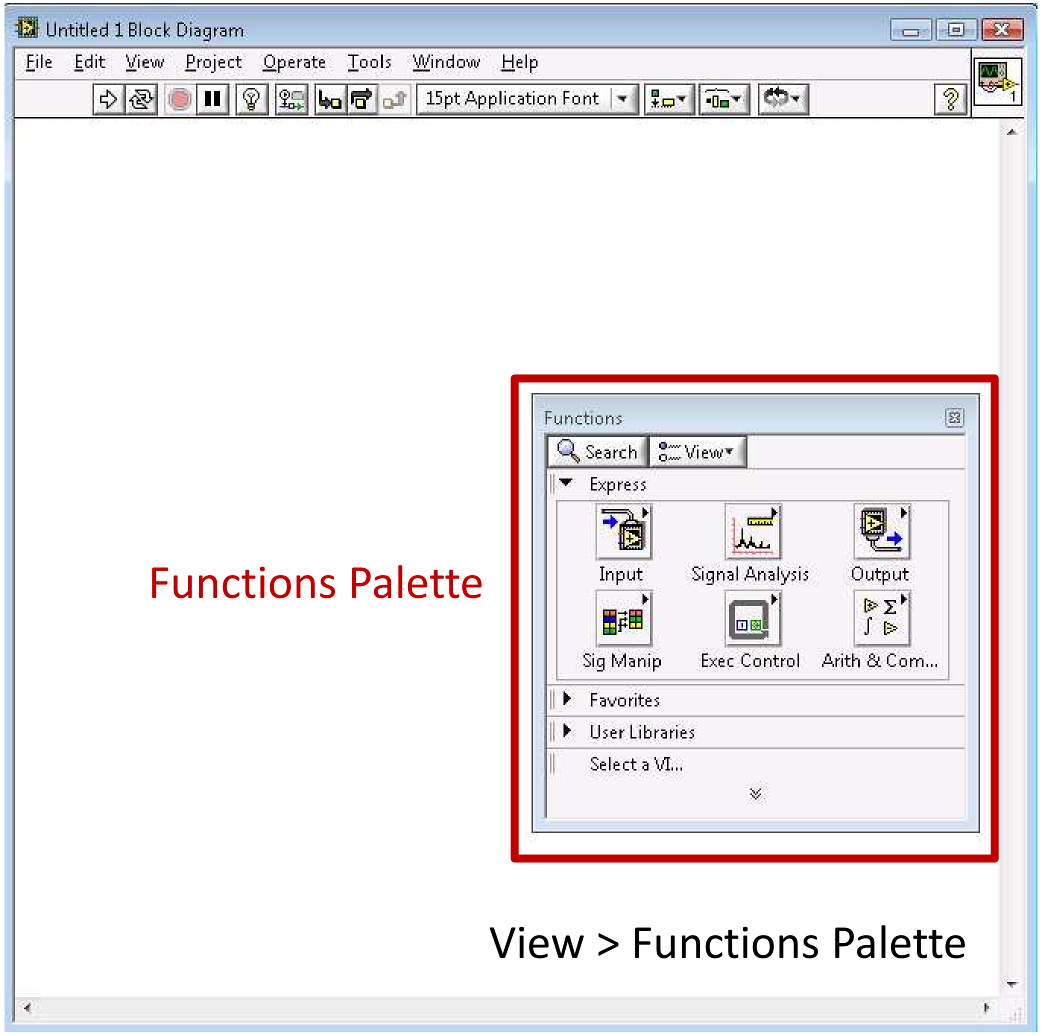
Front Panel



Controls Palette

View > Controls Palette

Block Diagram



Functions Palette

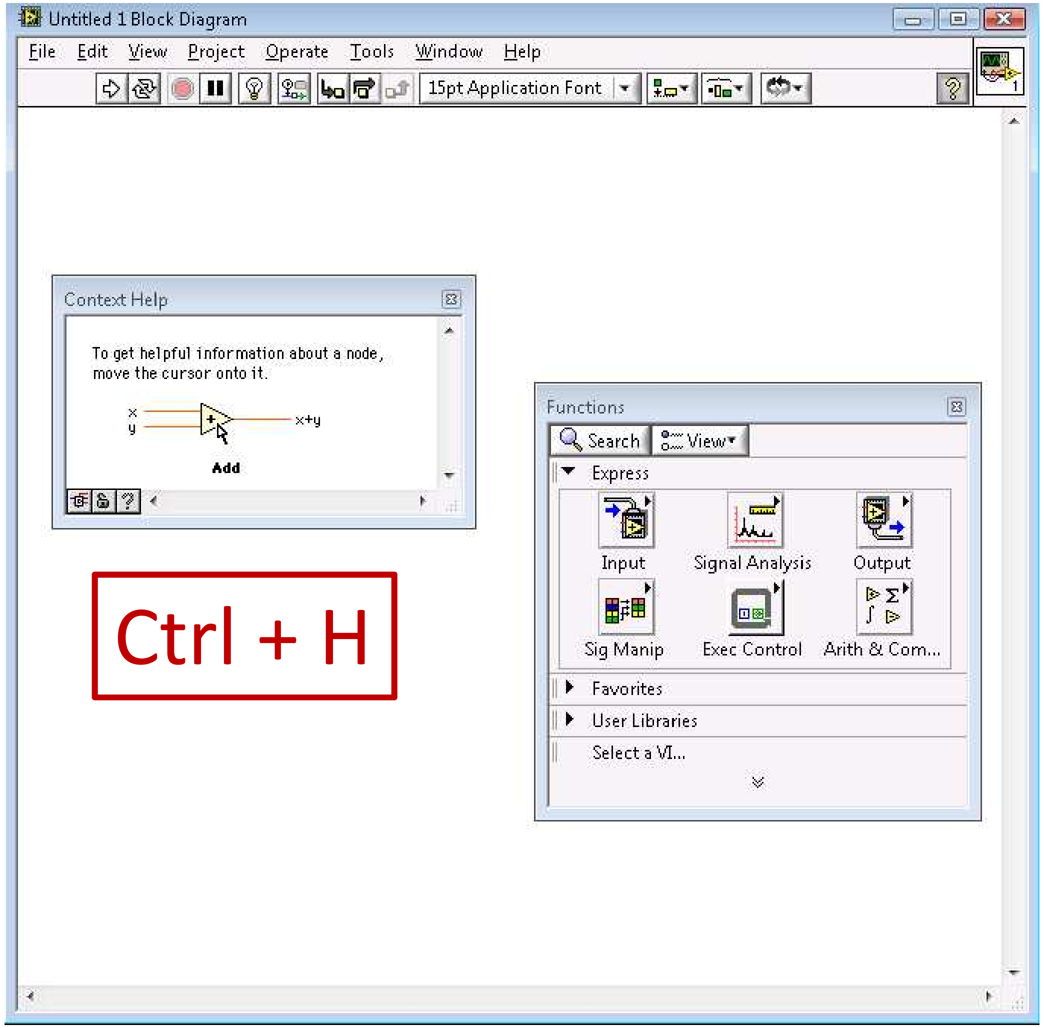
View > Functions Palette

Front Panel – Block Diagram

- To toggle between two panels

Ctrl + E

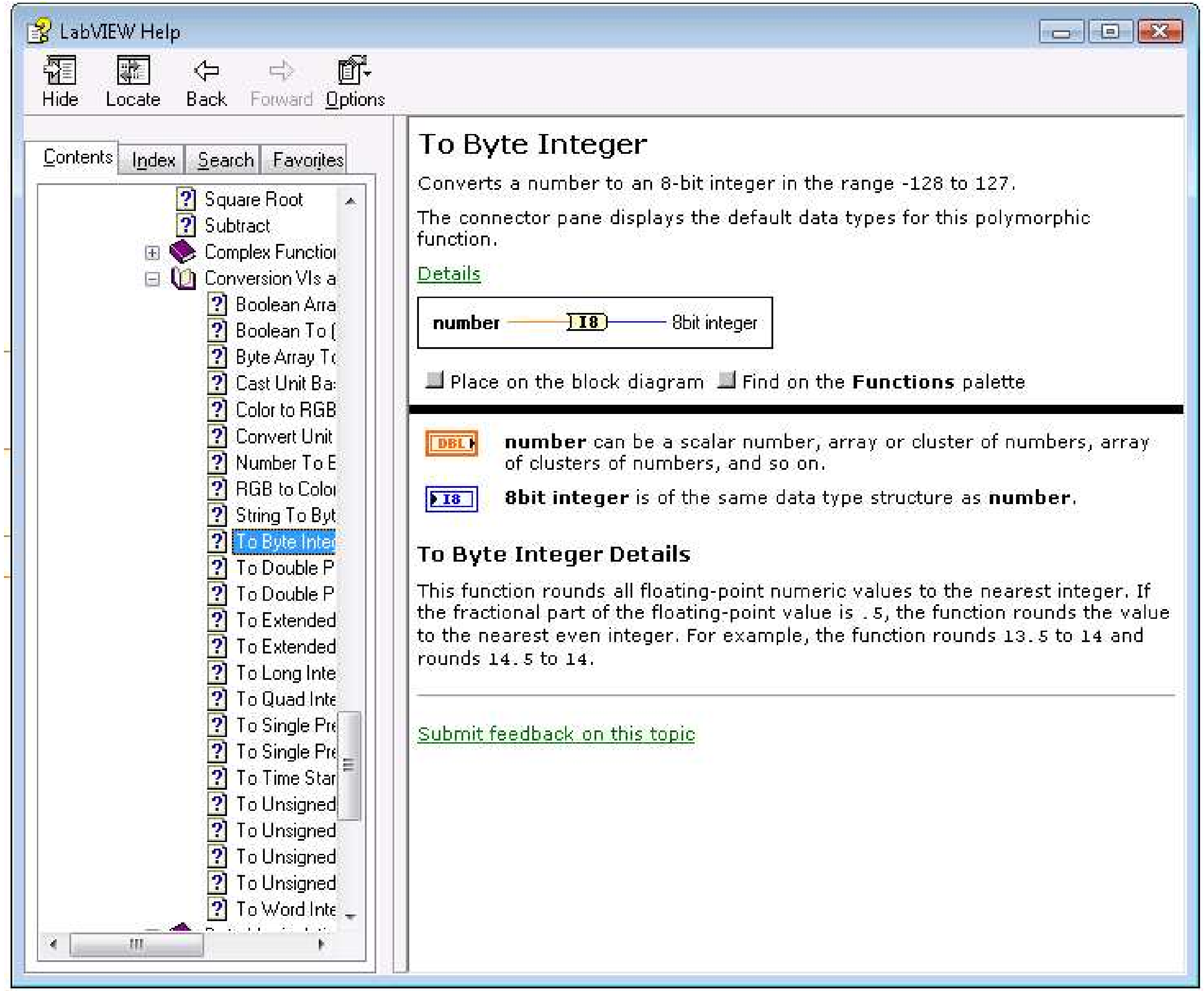
Context Help



To get helpful information about a node, move the cursor onto it.

Ctrl + H

Detailed help



To Byte Integer

Converts a number to an 8-bit integer in the range -128 to 127.

The connector pane displays the default data types for this polymorphic function.

Details

□ Place on the block diagram □ Find on the **Functions** palette

number can be a scalar number, array or cluster of numbers, array of clusters of numbers, and so on.

8bit integer is of the same data type structure as **number**.

To Byte Integer Details

This function rounds all floating-point numeric values to the nearest integer. If the fractional part of the floating-point value is `.5`, the function rounds the value to the nearest even integer. For example, the function rounds `13.5` to `14` and rounds `14.5` to `14`.

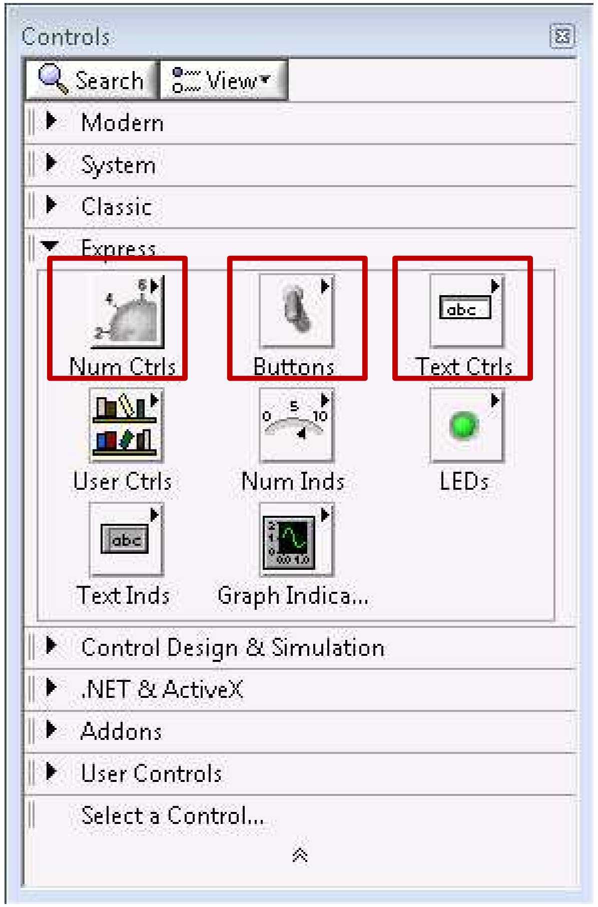
Submit feedback on this topic

Three fashions of variables

- Mixture of GUI and console

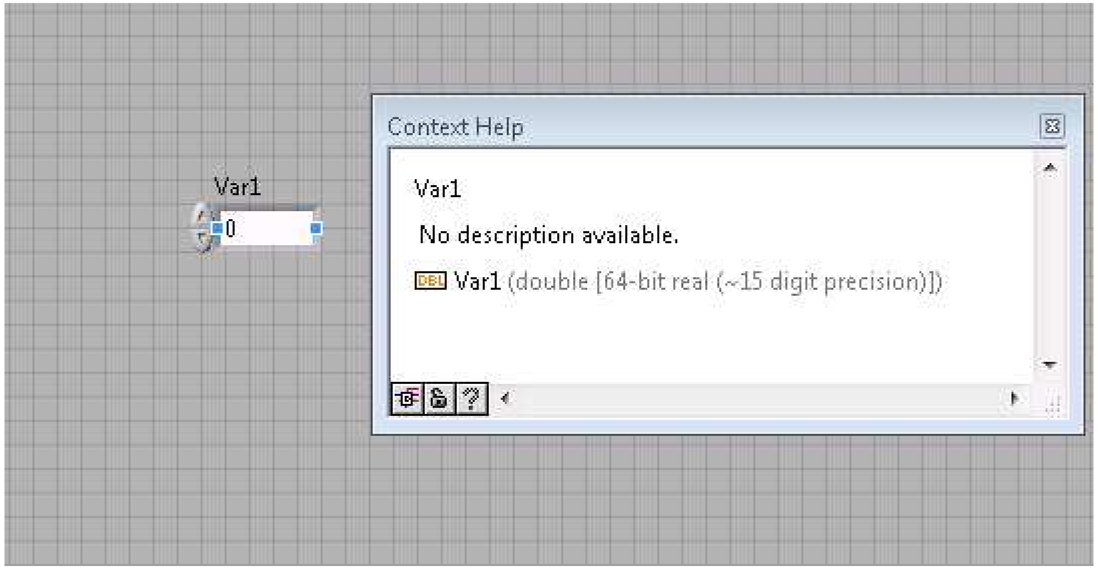
1. Controls
   - User input (In Java: TextBox GUI Component)
2. Indicators
   - Program output (In Java: Label GUI Component)
3. Constants
   - Predefined constants (In Java: Class variables with `const` definer)

Controls



- Selected from Controls Palette
- Usually Express Palette is enough

Controls



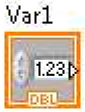
Var1

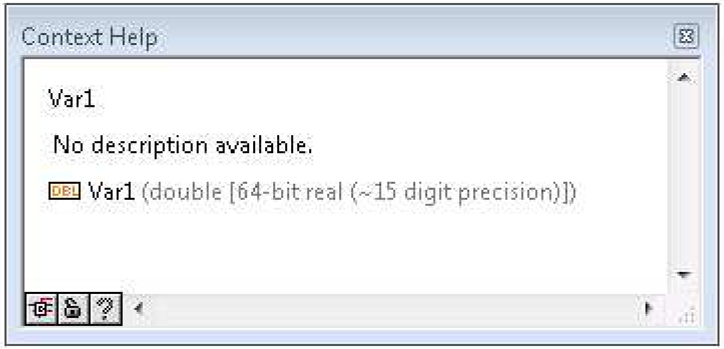
No description available.

DBL Var1 (double [64-bit real (~15 digit precision)])

Controls

- Each control has a proxy in the Block diagram
- Note that proxy has only OUTPUT node



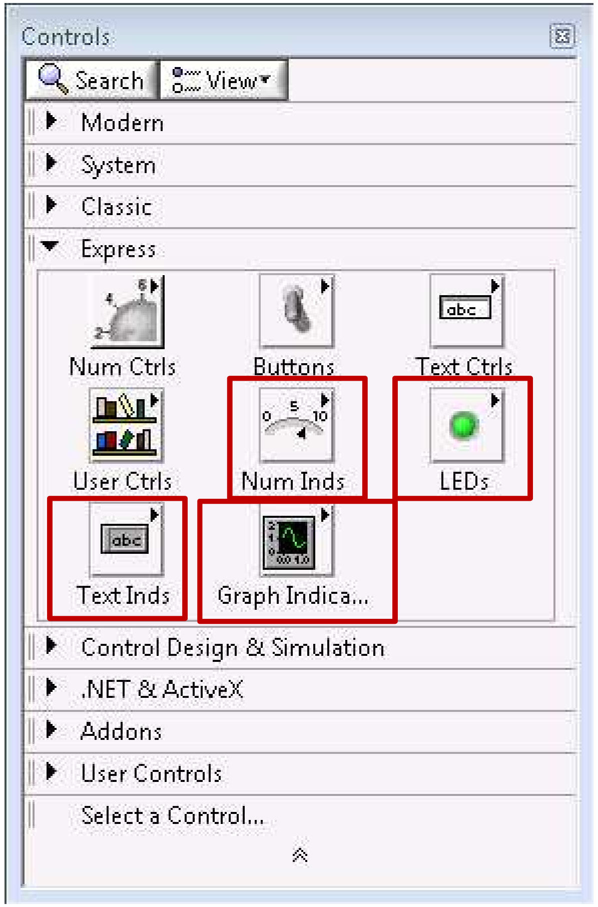


Var1

No description available.

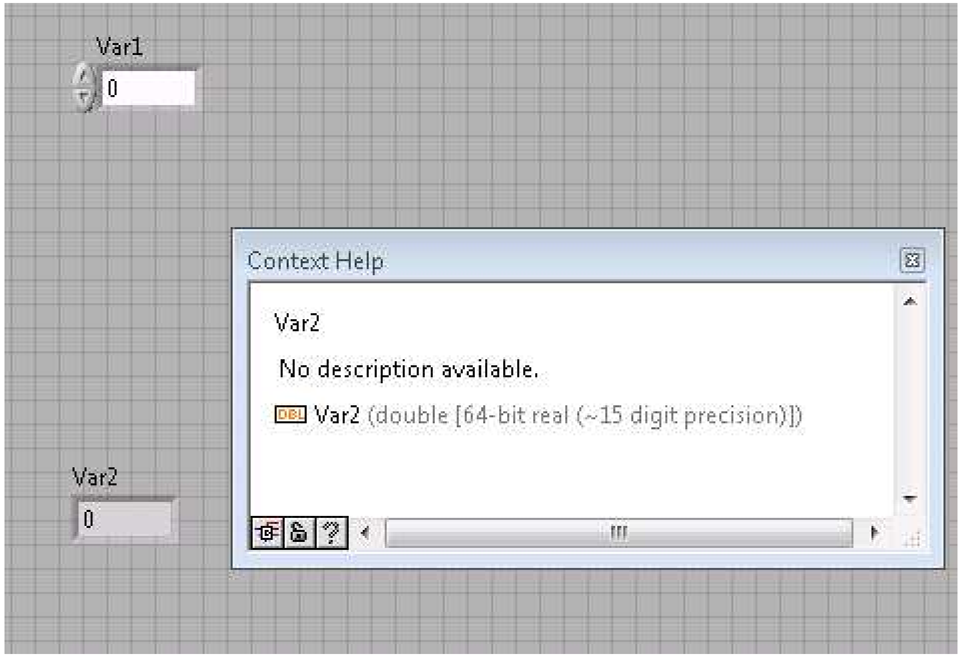
DBL Var1 (double [64-bit real (~15 digit precision)])

Indicators



- Also Selected from Controls Palette
- Usually Express Palette is enough

Indicators



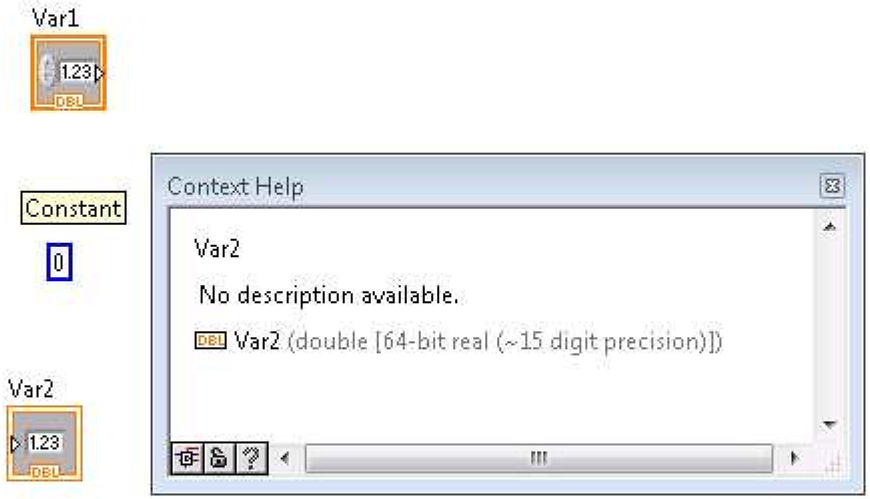
Var2

No description available.

DBL Var2 (double [64-bit real (~15 digit precision)])

Indicators

- Each indicator also has a proxy in the Block diagram
- Note that proxy has only INPUT node



Var2

No description available.

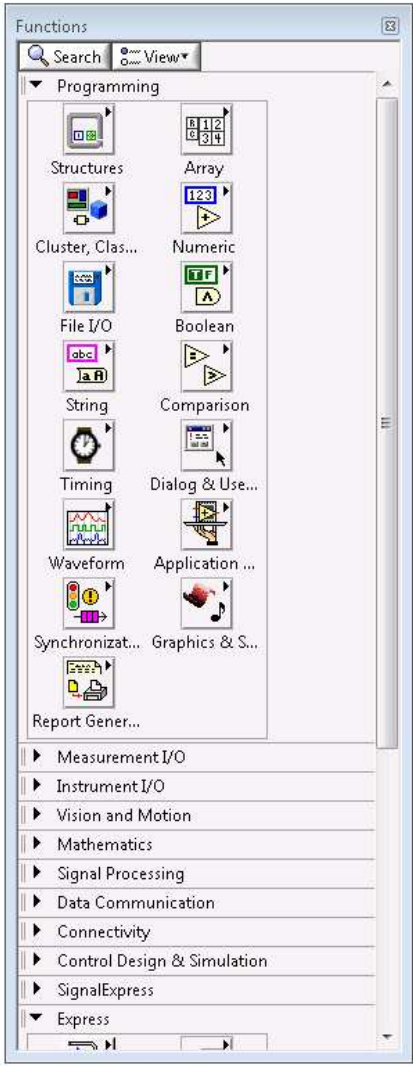
DBL Var2 (double [64-bit real (~15 digit precision)])

Constants

Programming

Use the Programming VIs and functions as the basic building blocks for a VI.

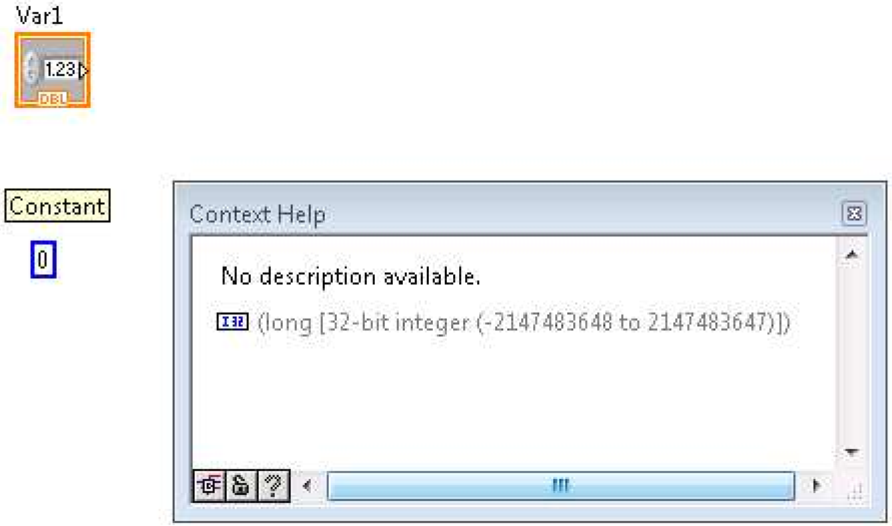
Detailed help



- You can use functions palette to insert constants

Constants

- Constants only live in Block diagram

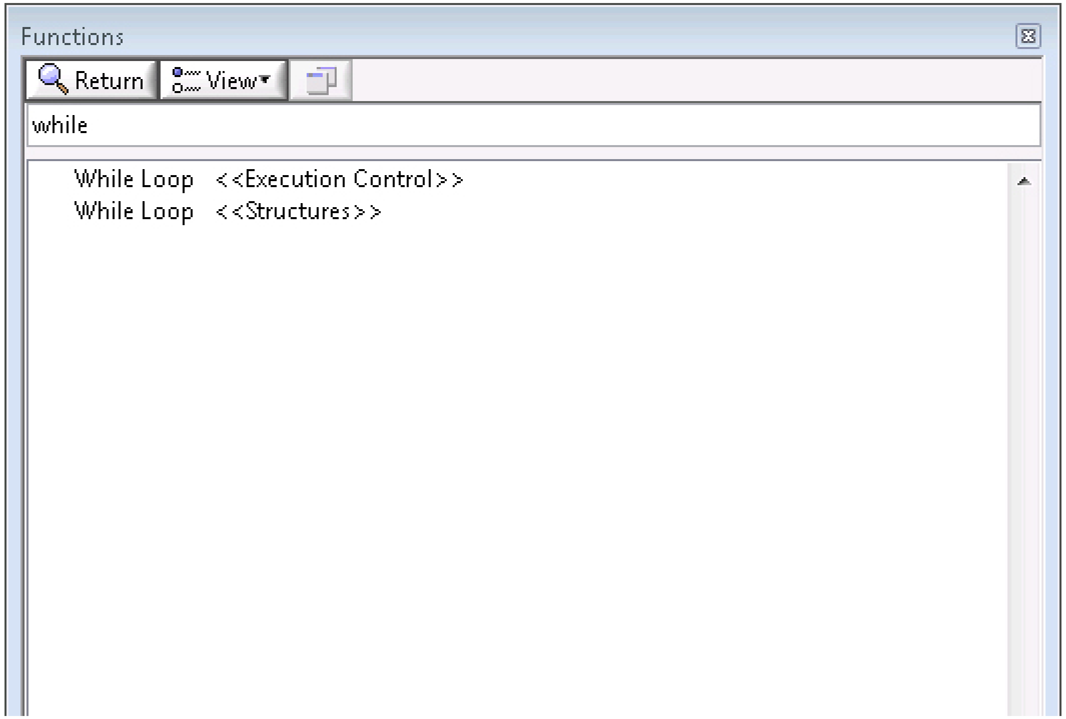


No description available.

I32 (long [32-bit integer (-2147483648 to 2147483647)])

Search

- You can always seach for the item
- Click on the search button on the palettes



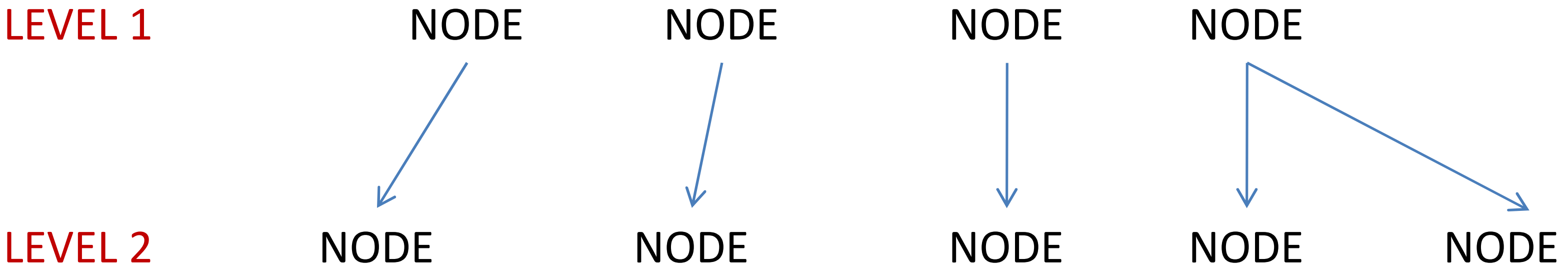
While Loop <<Execution Control>>
While Loop <<Structures>>

Types

- All standard types are present
  - Integer (signed, unsigned, long, word, byte, quad)
  - Floating point (single precision, double precision, extended precision)
  - Boolean
  - String
- All of above can be arranged in arrays, matrices, clusters

Flow Chart Paradigm

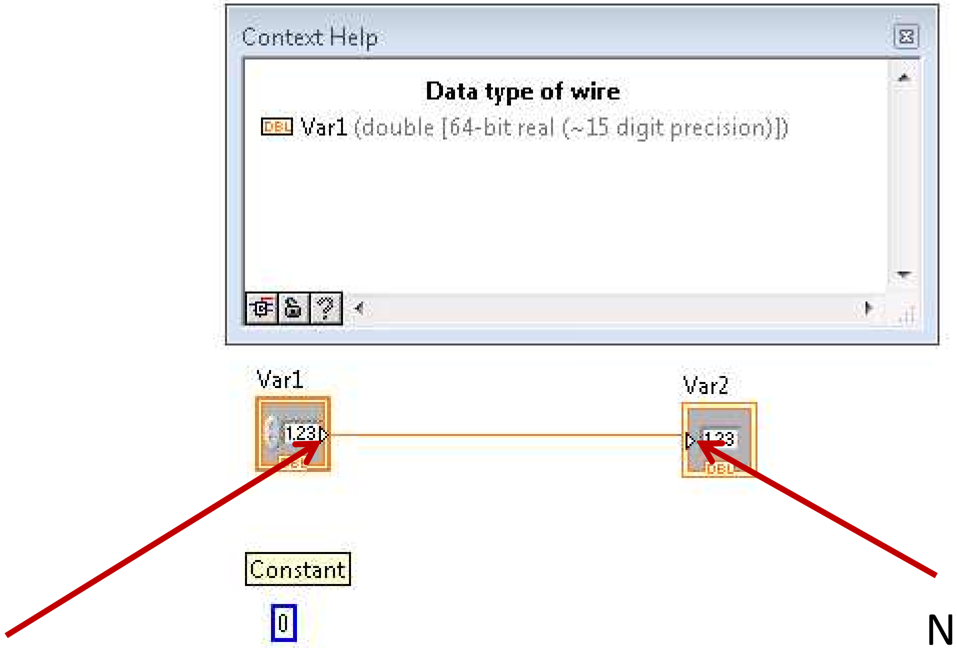
- Flow of execution is done by following nodes in a flow diagram



And so on...

Wiring

- Defines the direction of flow



Context Help

Data type of wire

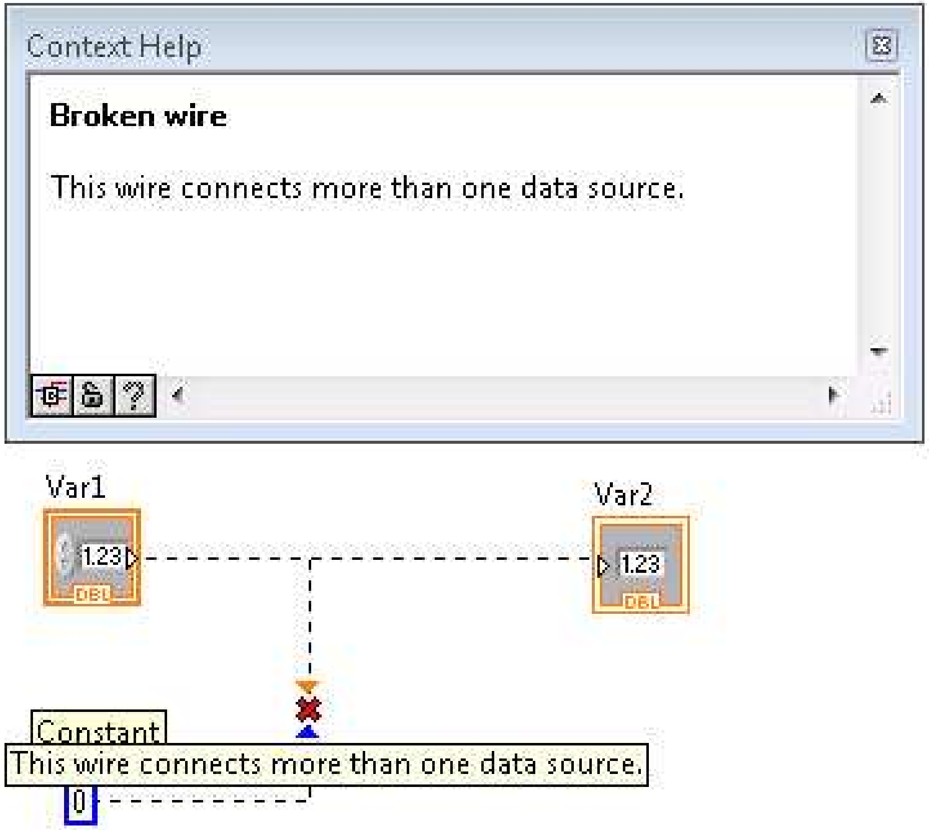
DBL Var1 (double [64-bit real (~15 digit precision)])

NODE

NODE

Wiring

- One to many connection is acceptable
- Many to one connection is illegal



Context Help

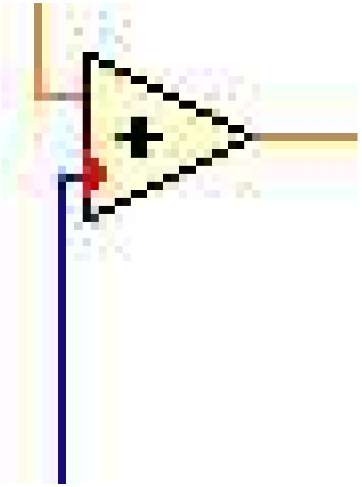
Broken wire

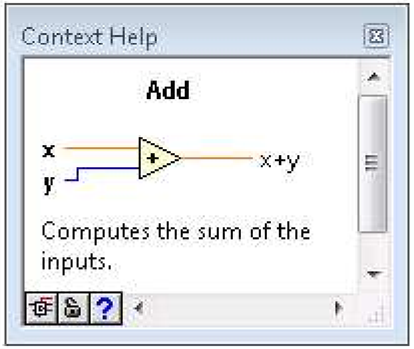
This wire connects more than one data source.

This wire connects more than one data source.

Wiring

- Application of algorithm is done by wiring
- Color of the wire indicates type
  - Blue: Integer , Orange: Floating Point
  - Purple: String, Green: Boolean

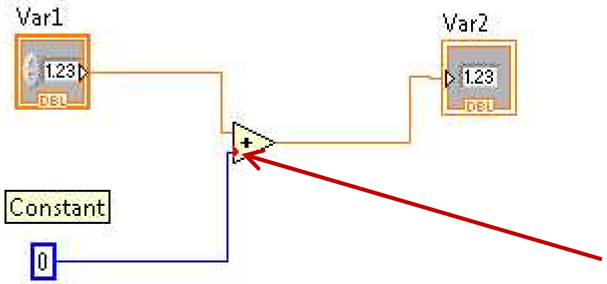




Context Help

Add

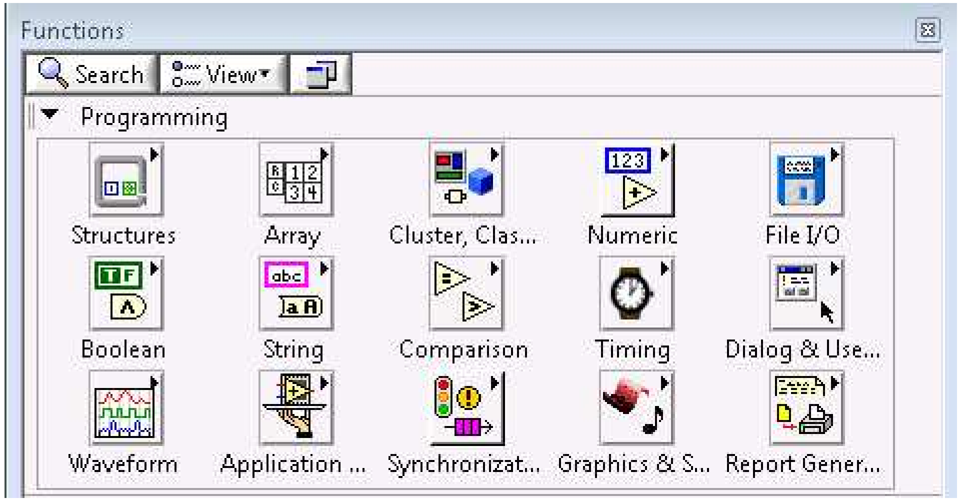
Computes the sum of the inputs.



Red dot indicates that “wrong type has wired but it’s OK, he has made the type-casting”.

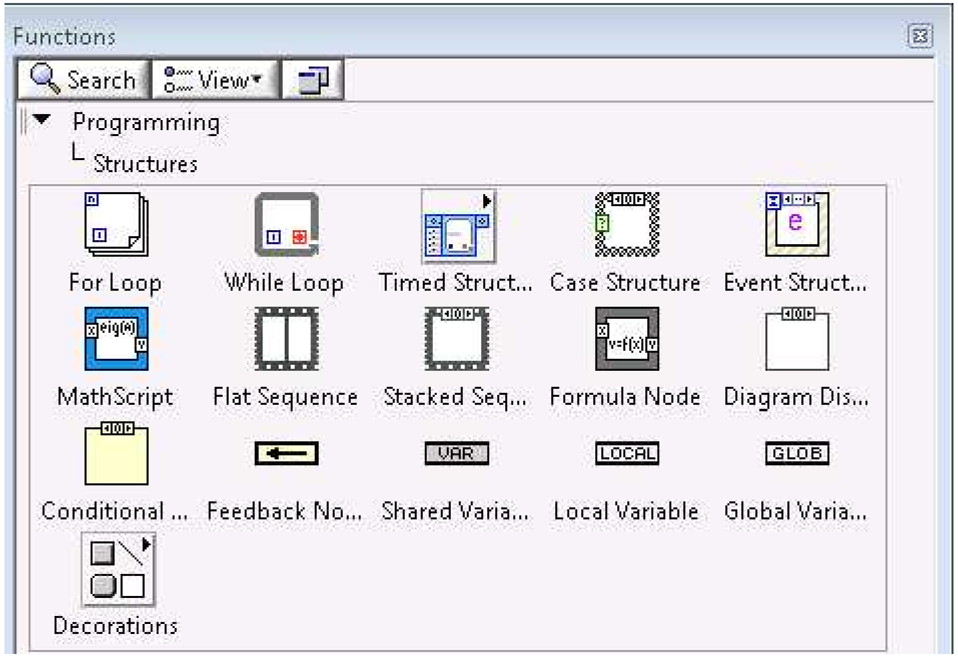
Algorithm Construction

All algoritm structures lies in Programming Sub-Palette

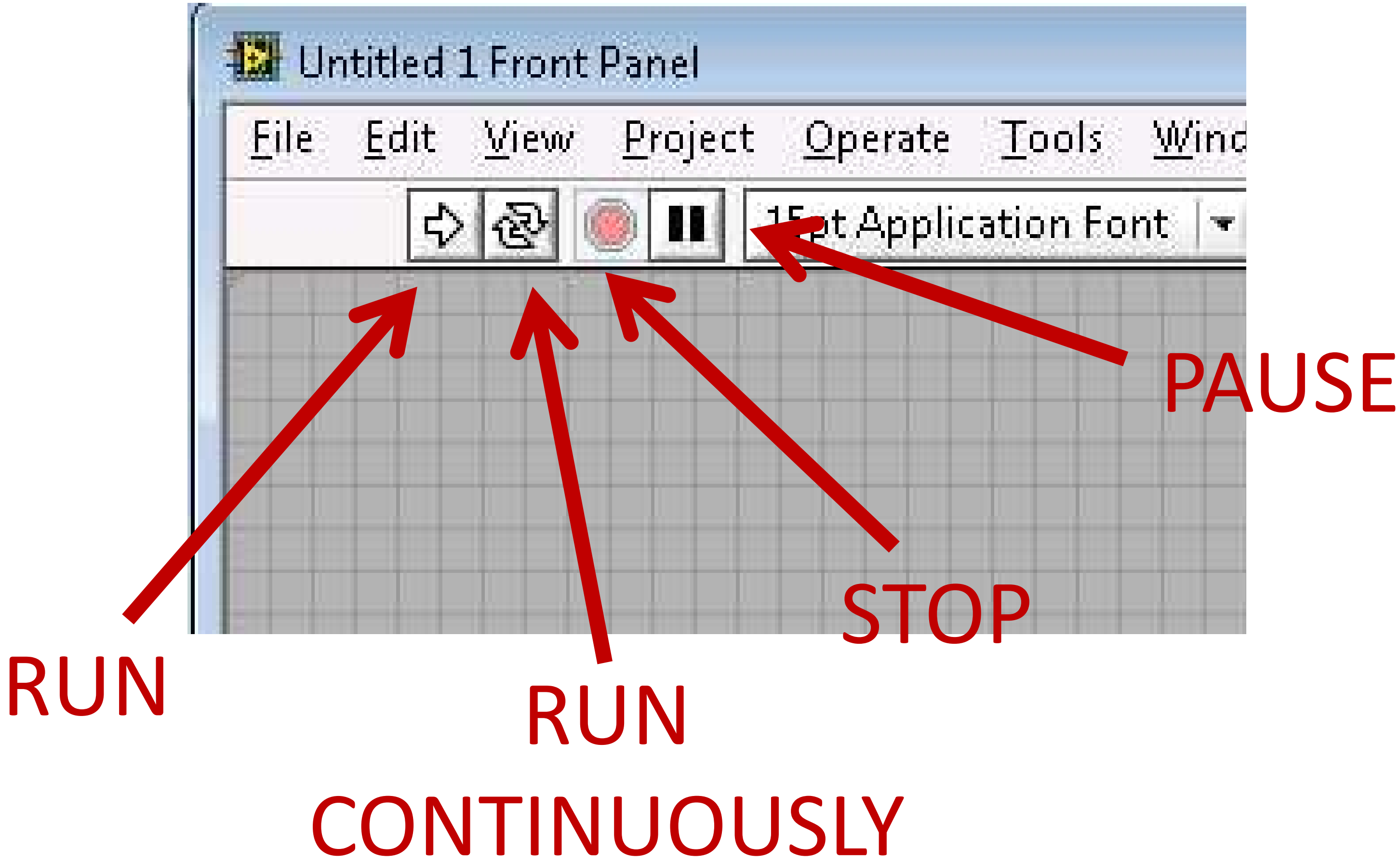


Algorithm Construction

Program flow structures are under Structures Sub-sub-palette



Program Control



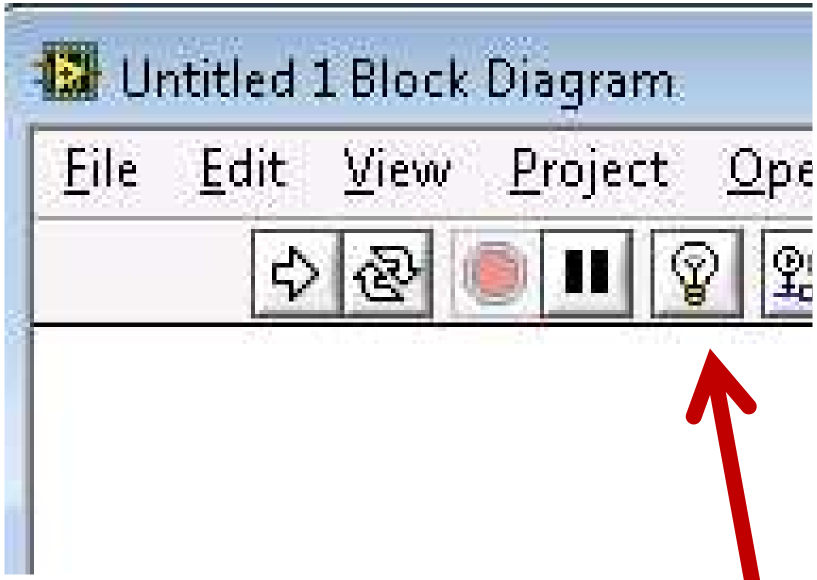
PAUSE

STOP

RUN

RUN CONTINUOUSLY

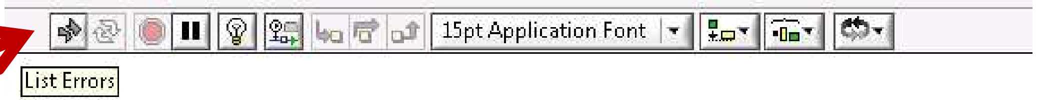
Debugging

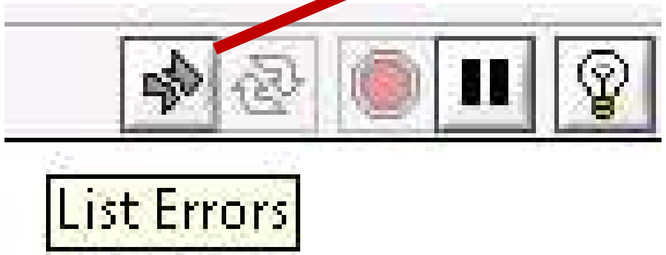


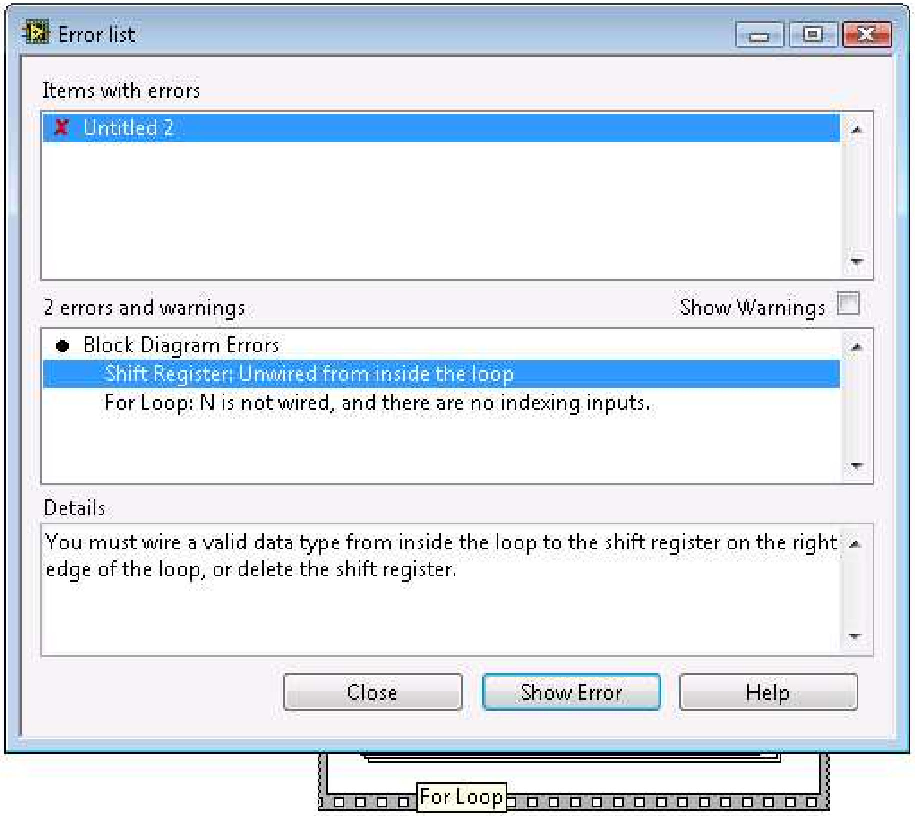
Untitled 1 Block Diagram

HIGHLIGHT EXECUTION

Debugging







Items with errors

Untitled 2

2 errors and warnings

Show Warnings □

- Block Diagram Errors
  - Shift Register: Unwired from inside the loop
  - For Loop: N is not wired, and there are no indexing inputs.

Details

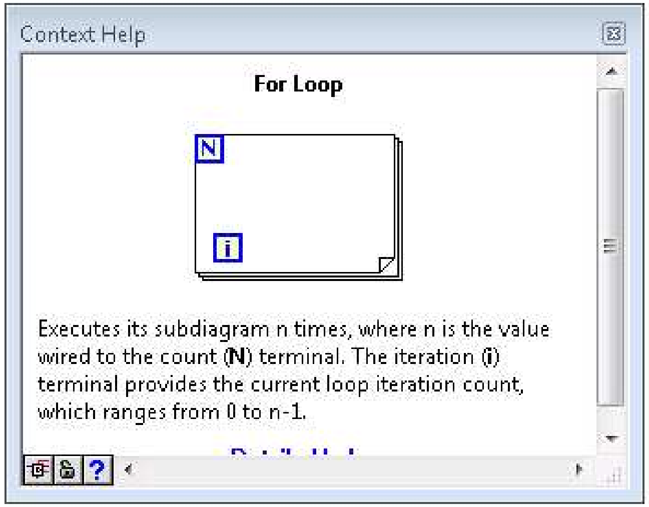
You must wire a valid data type from inside the loop to the shift register on the right edge of the loop, or delete the shift register.

- Broken Arrow Means ERROR
- Click to see Error List

Program Flow Control

- For loops
- While loops
- Sequences
  - Flat sequence
  - Stacked sequence
- And many more
- Compansates for: Event handling, Top-to-bottom execution, OOP, etc.

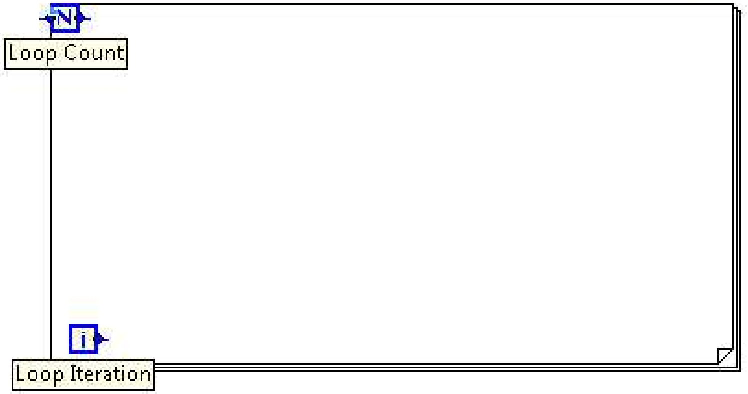
For loop



Context Help

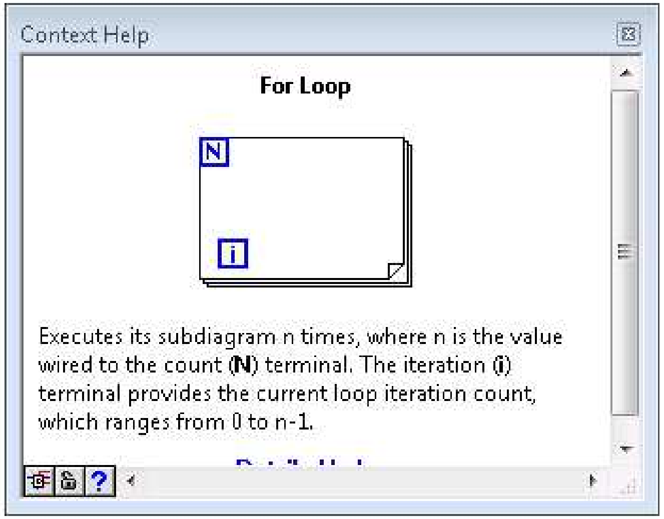
For Loop

Executes its subdiagram n times, where n is the value wired to the count (N) terminal. The iteration (i) terminal provides the current loop iteration count, which ranges from 0 to n-1.



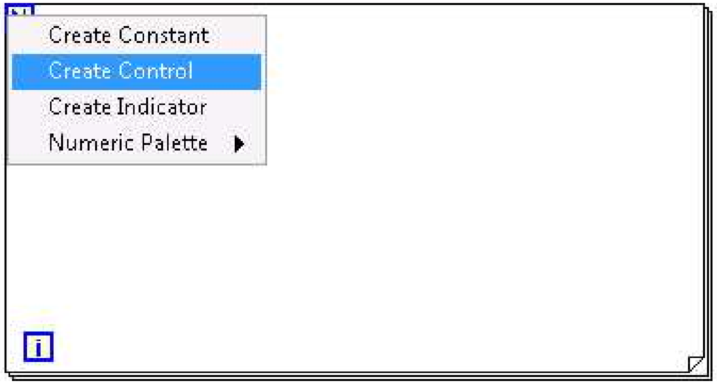
- Loop for limited iterations
- Must know the iteration amount before-hand
- Loop count can-not be change once set

For loop



For Loop

Executes its subdiagram n times, where n is the value wired to the count (N) terminal. The iteration (i) terminal provides the current loop iteration count, which ranges from 0 to n-1.

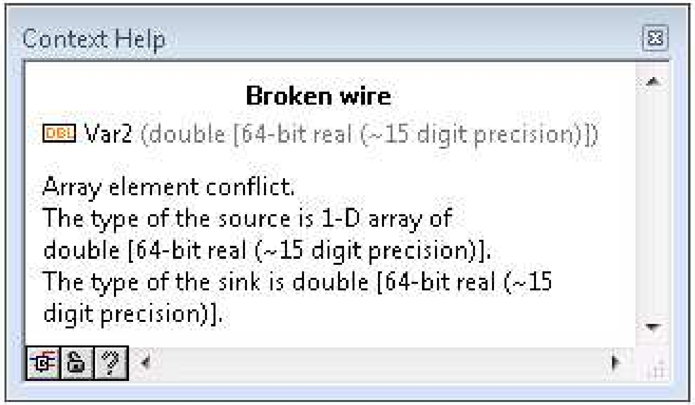


Another way to create variables: Right clicking onto the node

For loop example

- Get some number from the user
- Add 3 to it 10 times
- Display the result

For loop example

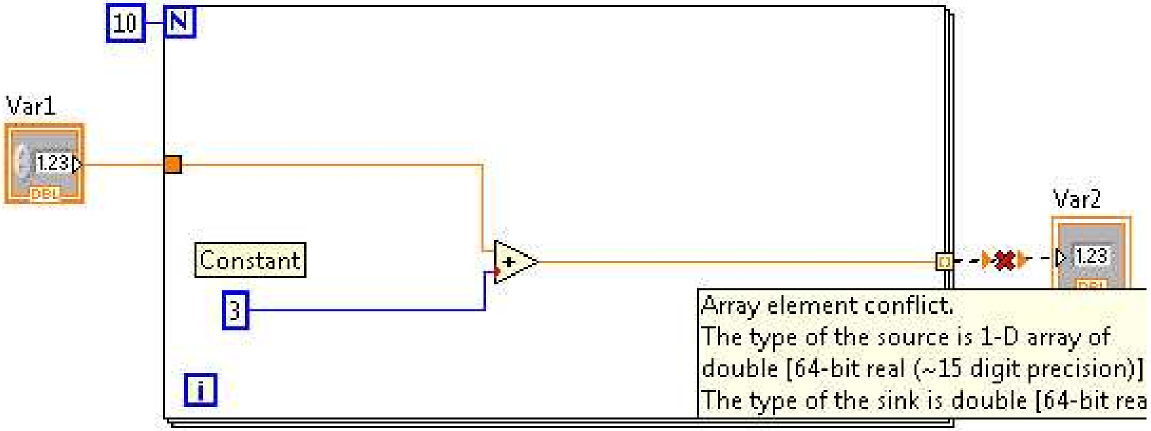


Context Help

Broken wire

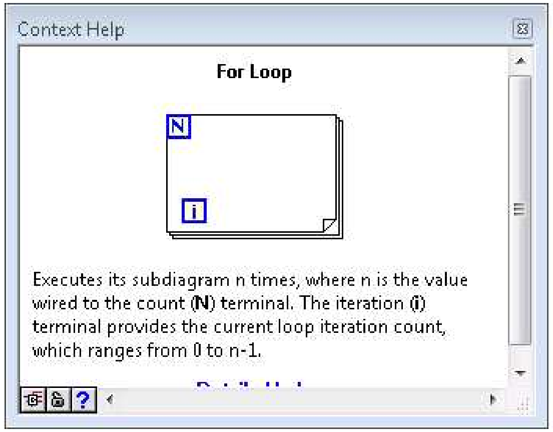
DBL Var2 (double [64-bit real (~15 digit precision)])

Array element conflict.
The type of the source is 1-D array of double [64-bit real (~15 digit precision)].
The type of the sink is double [64-bit real (~15 digit precision)].



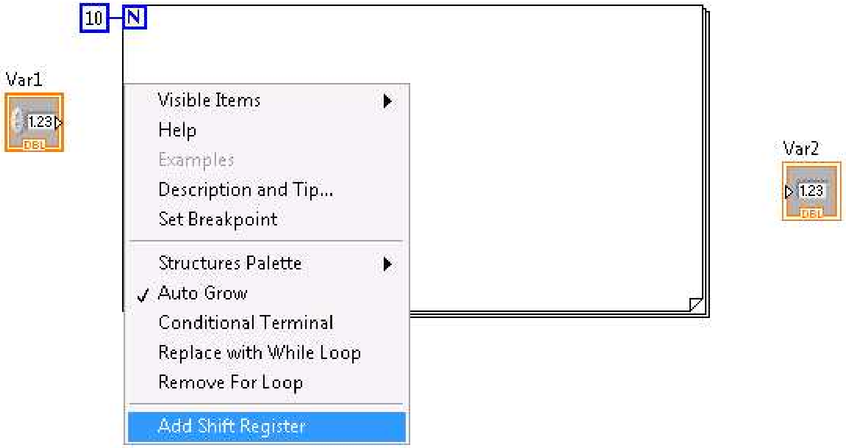
Array element conflict.
The type of the source is 1-D array of
double [64-bit real (~15 digit precision)]
The type of the sink is double [64-bit rea

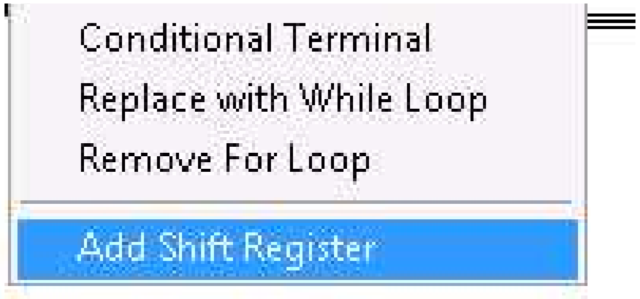
For loop example



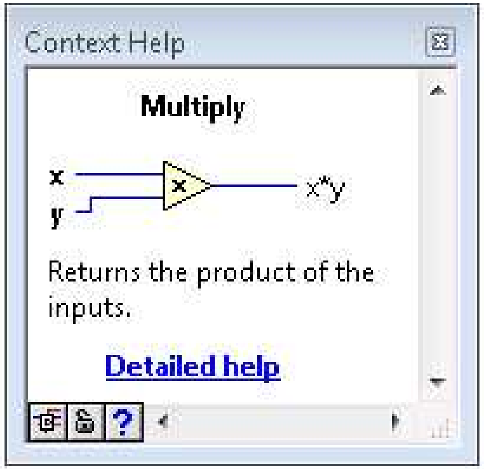
Executes its subdiagram n times, where n is the value wired to the count (N) terminal. The iteration (i) terminal provides the current loop iteration count, which ranges from 0 to n-1.

Shift registers convey result of one iteration to the next iteration





For loop example

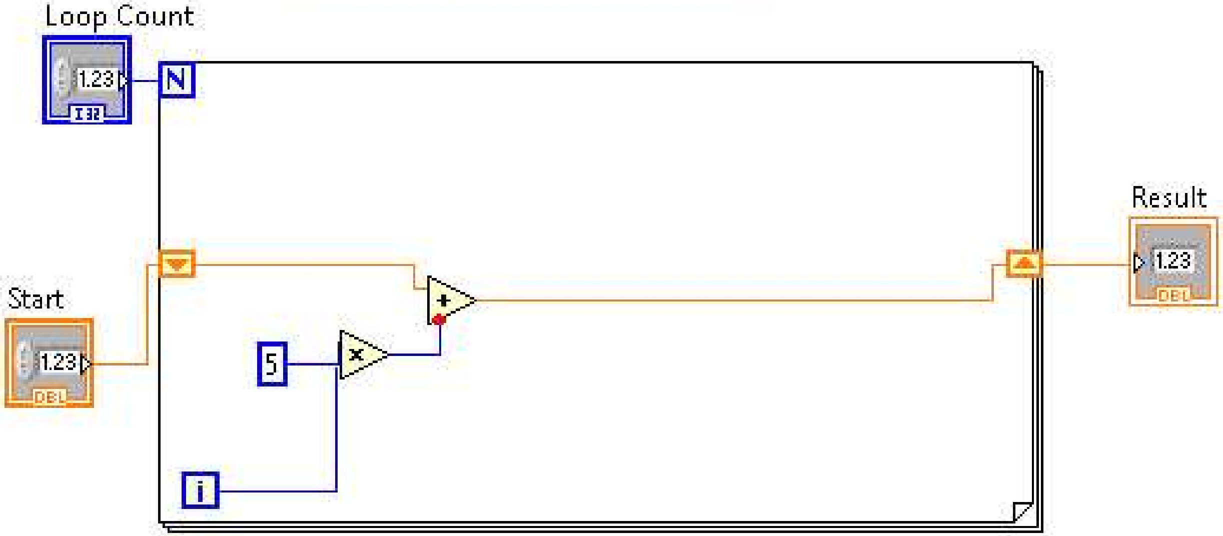


Context Help

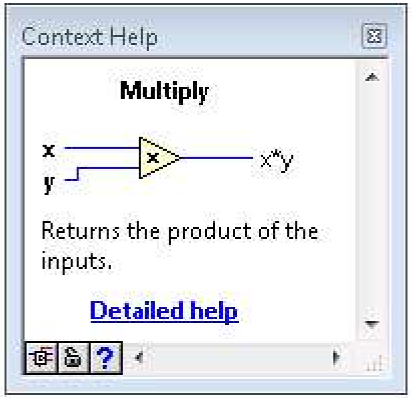
Multiply

Returns the product of the inputs.

Detailed help



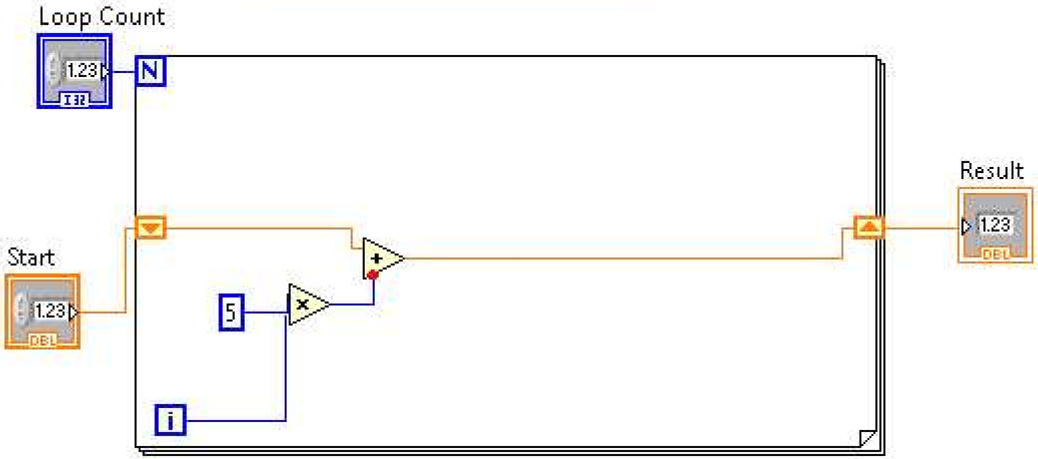
For loop example



Multiply

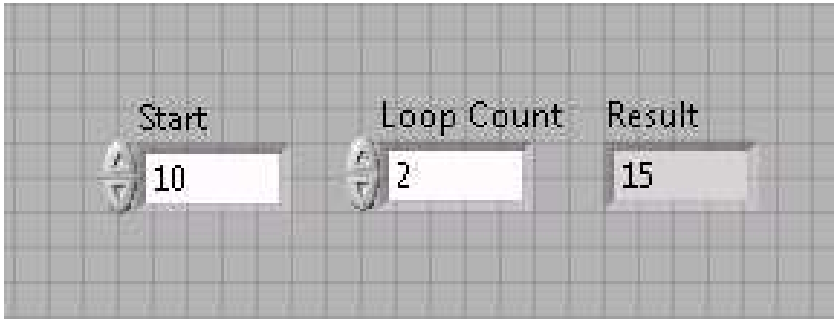
Returns the product of the inputs.

Detailed help



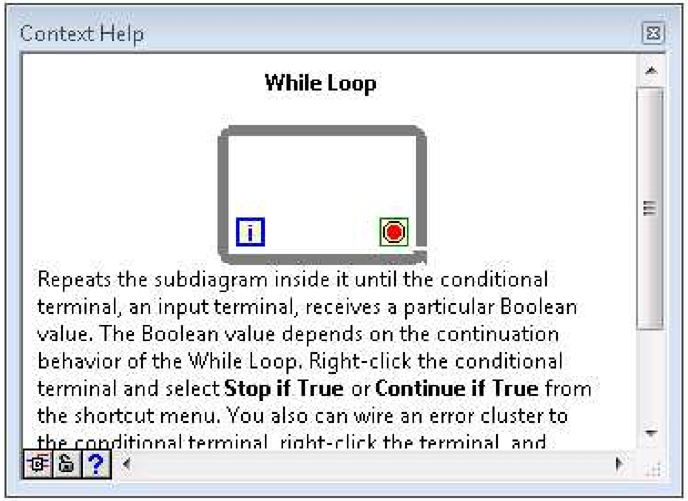
Do not limit your imagination

For loop example



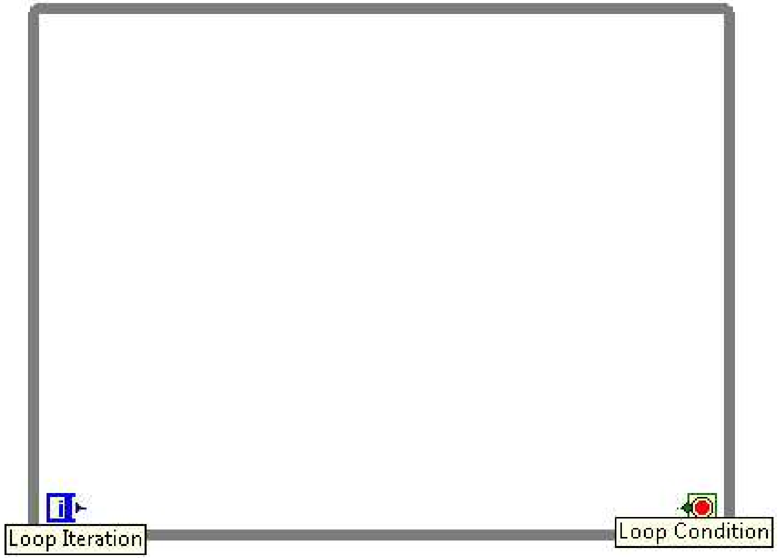
User only interacts with three objects

While loop

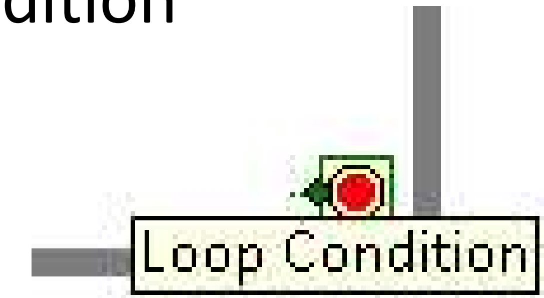


While Loop

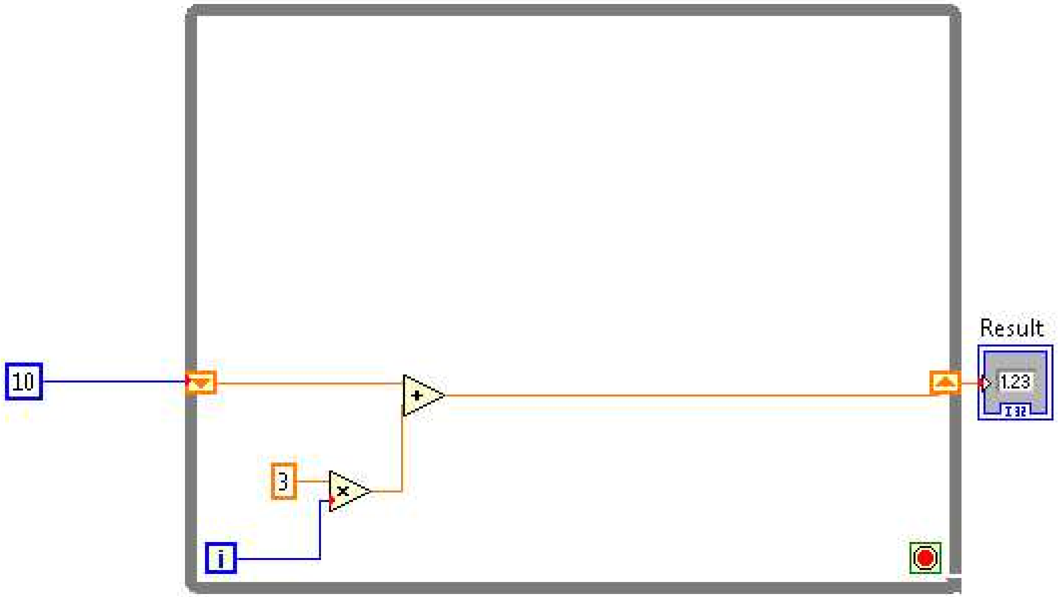
Repeats the subdiagram inside it until the conditional terminal, an input terminal, receives a particular Boolean value. The Boolean value depends on the continuation behavior of the While Loop. Right-click the conditional terminal and select **Stop if True** or **Continue if True** from the shortcut menu. You also can wire an error cluster to the conditional terminal, right-click the terminal, and



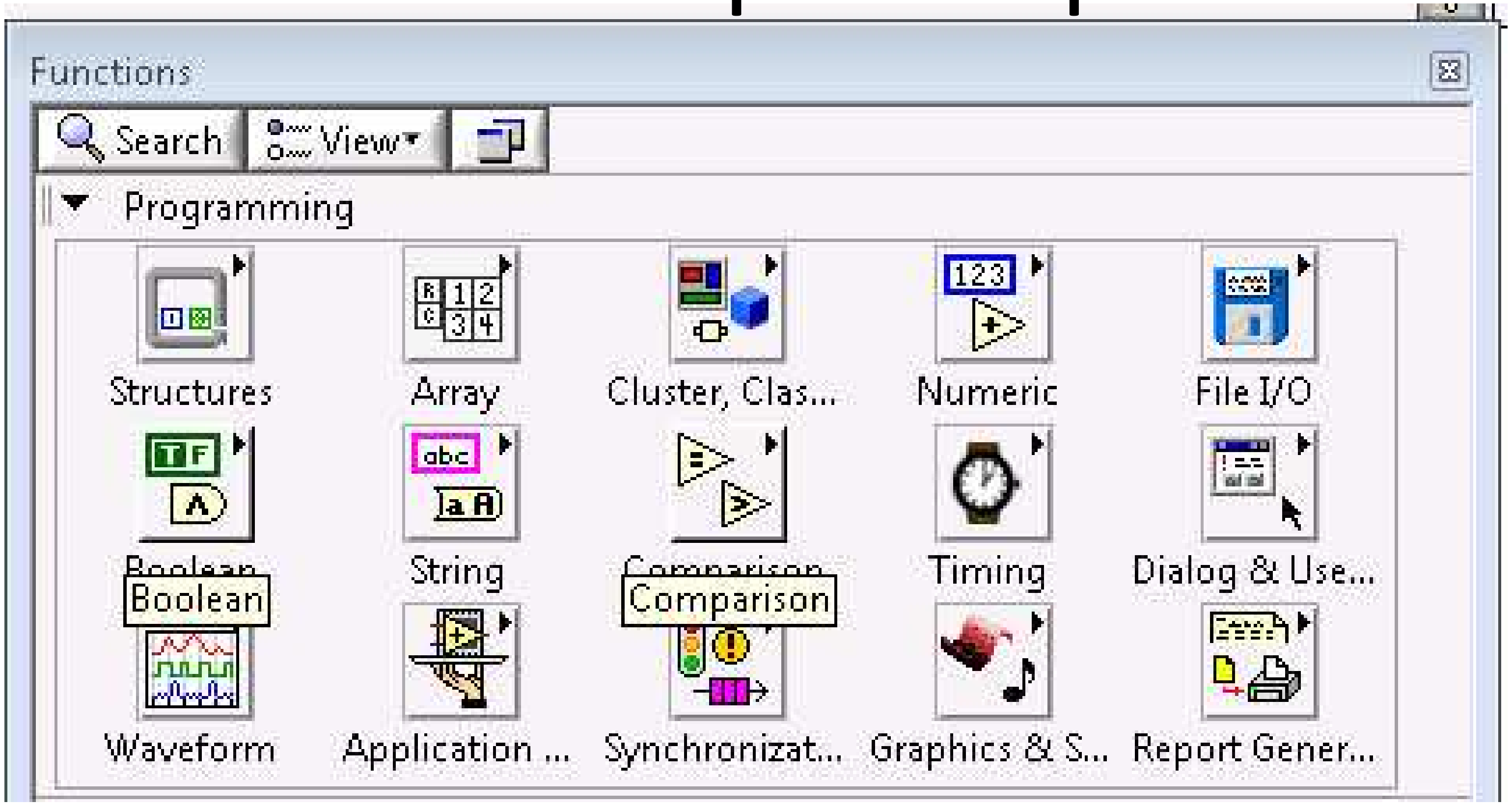
- •Loops until the loop condition is satisfied
- •Or while the loop condition is not satisfied
  - •Select by clicking on the loop condition



While loop example

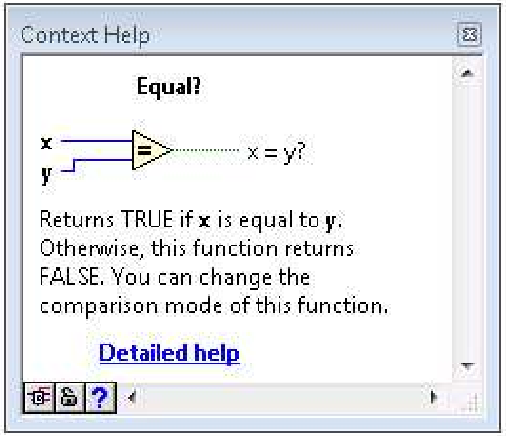


While loop example



Employ boolean and comparison palette controls to manage loop condition

While loop example

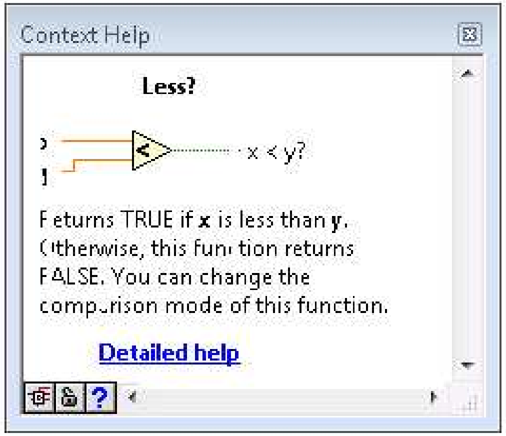


Context Help

Equal?

Returns TRUE if **x** is equal to **y**. Otherwise, this function returns FALSE. You can change the comparison mode of this function.

Detailed help

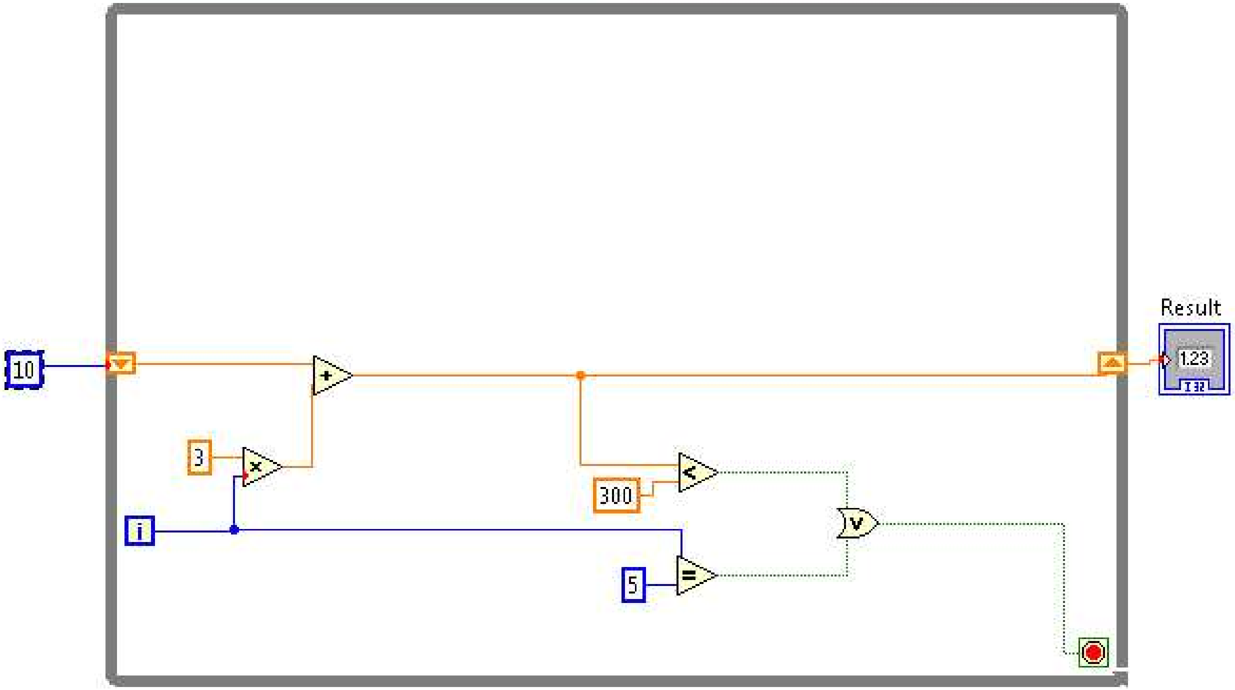


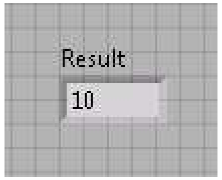
Context Help

Less?

Returns TRUE if **x** is less than **y**. Otherwise, this function returns FALSE. You can change the comparison mode of this function.

Detailed help

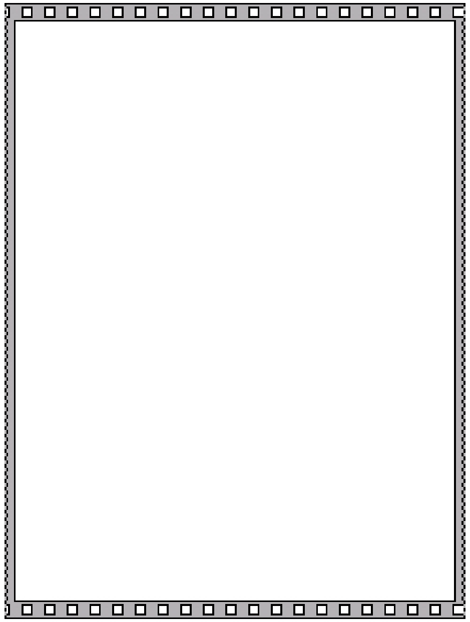




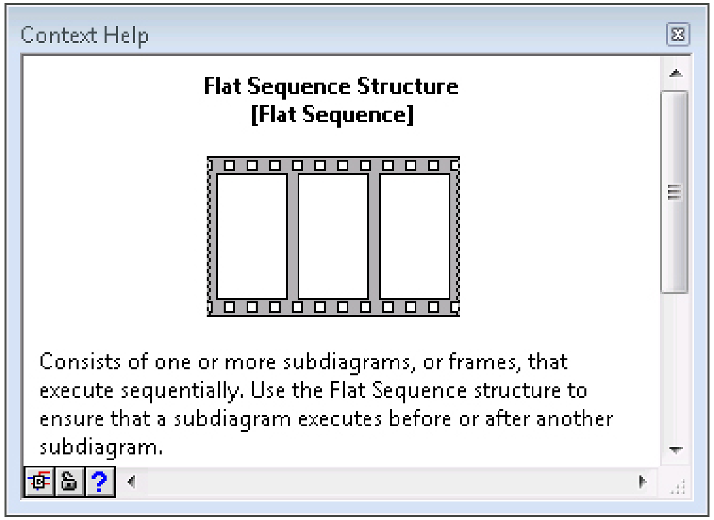
Result

10

Flat sequence



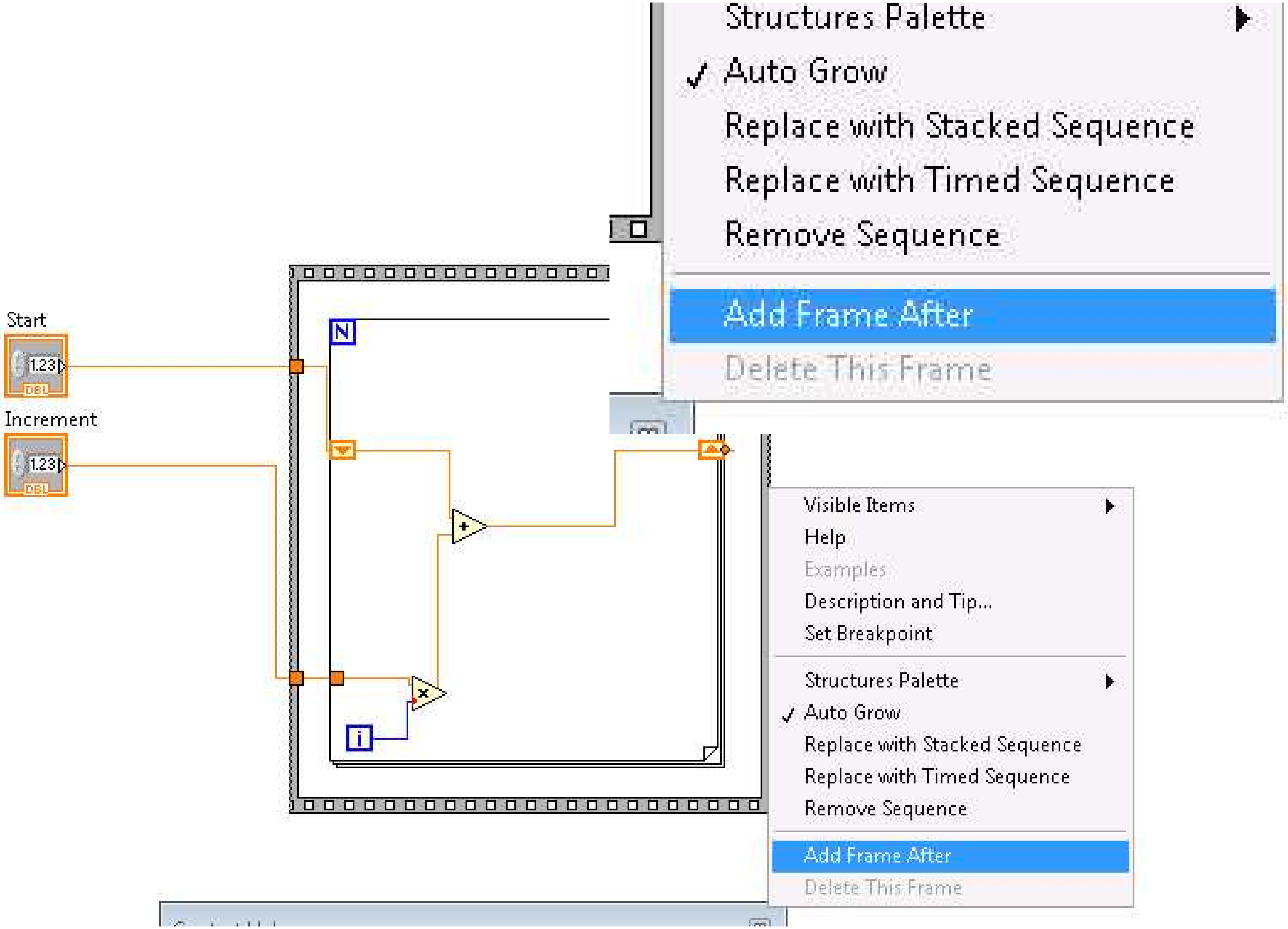
Executes contents of each frame one by one



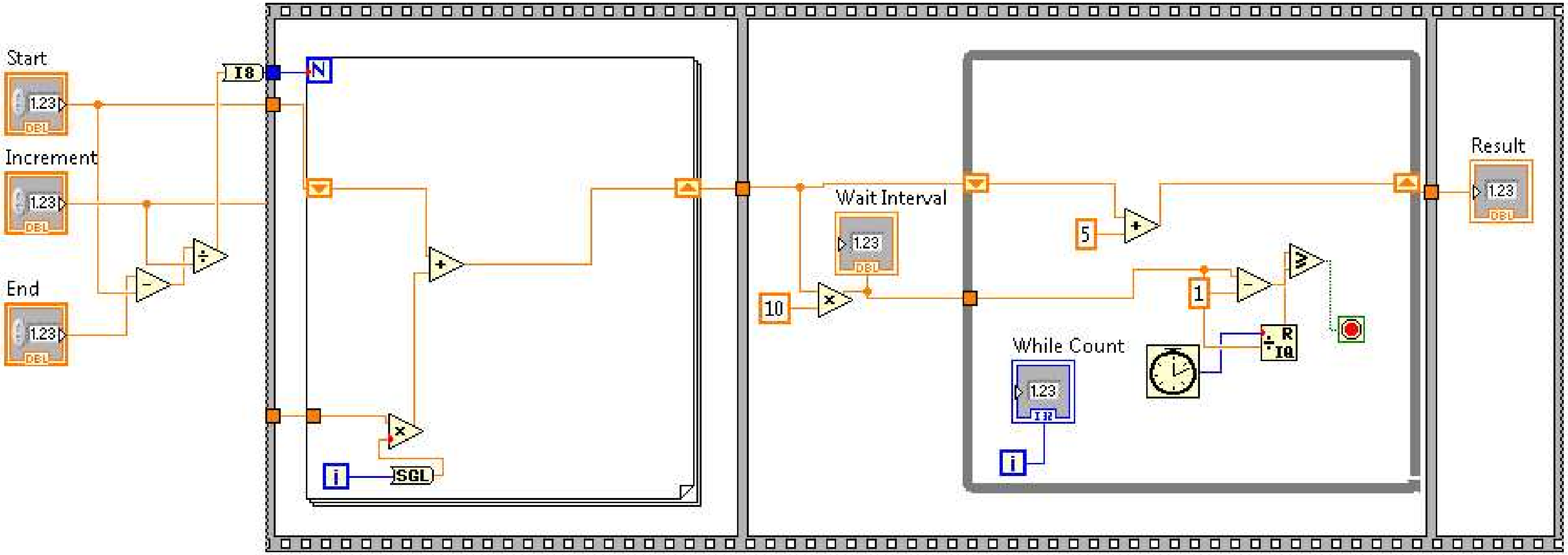
Flat Sequence Structure
[Flat Sequence]

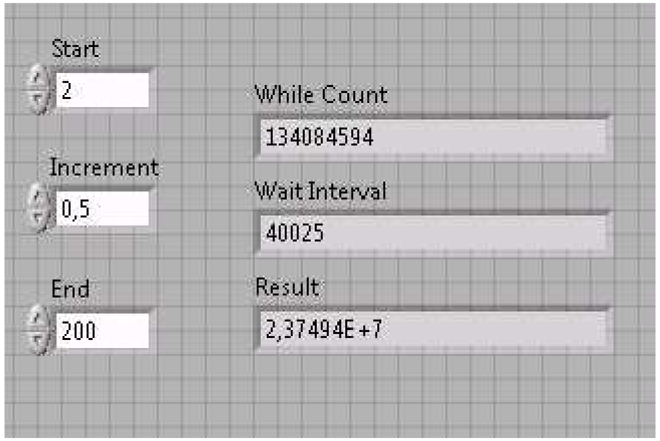
Consists of one or more subdiagrams, or frames, that execute sequentially. Use the Flat Sequence structure to ensure that a subdiagram executes before or after another subdiagram.

Flat sequence



Flat sequence





Start
2

Increment
0,5

End
200

While Count
134084594

Wait Interval
40025

Result
2,37494E+7