**Hands-on Lab:**

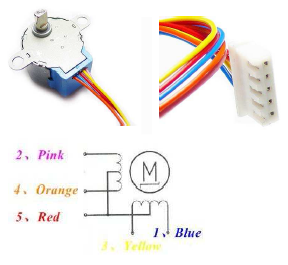
**LEGO NXT, NXC and Stepper Motors**

This lab uses NXC to actuate a stepper motor (4-phase, unipolar, 5-wire) through the LEGO NXT Brick.

**Preamble:** DC motor rotation is continuous. By contrast, steppers rotate discretely. Such stepping makes rotating a motor to fixed angle easier. Moreover, stepper rotation is practically instantaneous and accurate. This is significant because it eliminates (in a practical sense) rise-time and use of angle encoders. The net effect is that steppers are often used for precision movement applications. Examples include the positioning of 3D printer heads, CNC machine tool cutters, and robotic manipulators. Given the broad range and market for such positioning, steppers come in many different configurations to meet technical requirements such as a torque, rotational granularity, rotational speed, and size, weight, and power.

This lab introduces a commonly found 4-phase, 5-wire stepper. Specifically, the 28BYJ-48 stepper will be used. There are many vendors and online notes (e.g. using this motor with Arduino) but moreover, it often comes sold with a ULN2003 motor driver board (**Figure A left**).

**Figure A:** The 28BYJ-48 is commonly sold with a ULN2003 motor controller board with many vendors like [Amazon](https://www.amazon.com/HiLetgo-ULN2003-28BYJ-48-Stepper-4-phase/dp/B00LPK0E5A/ref=sr_1_5?crid=MBOE4C7K5CVL&keywords=28byj-48+stepper+motor+with+driver&qid=1552761082&s=gateway&sprefix=28BYJ-48+%2Caps%2C217&sr=8-5) (left). The wiring diagram from the manufacturing shows that 2 pairs of coils: Pink-and-Orange and Blue-and-Yellow, with power to the Red wire.





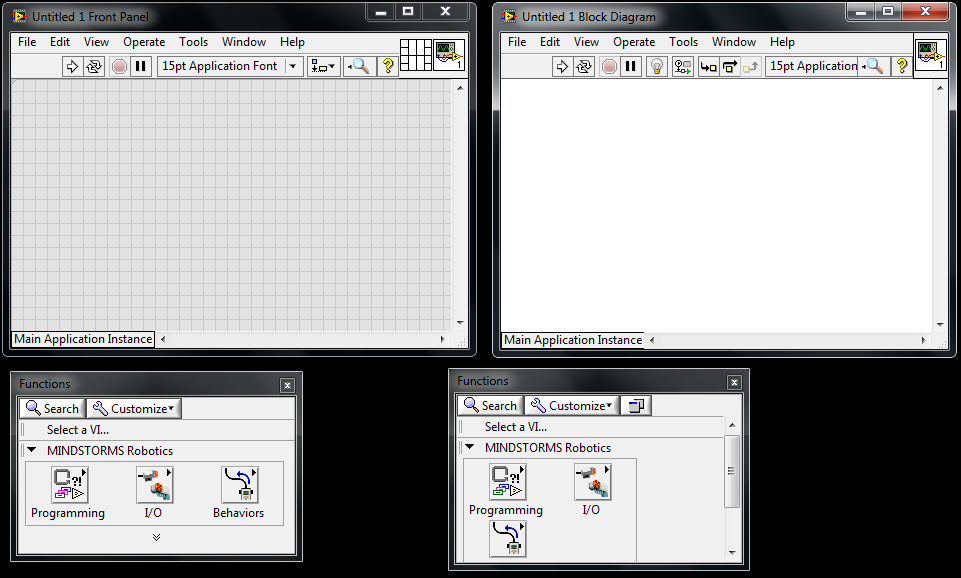
The [28BYJ-48 data sheet](https://www.mouser.com/ds/2/758/stepd-01-data-sheet-1143075.pdf) (**Figure A right**) shows there are 5-wires for this stepper. Moreover, there are 2 pairs of coils. One coil terminates in Pink and Orange wires, while the other coil terminates in Yellow and Blue. This coil configuration results in a 4-phase stepper; current must cycle through the 4 wires. Current is provided through the 5th wire (Red) which is connected the positive end of a voltage source (i.e. unipolar). The net effect is that the 28BYJ-48 is a *4-phase, 5-wire, unipolar* stepper motor.

Blah: Cycling table

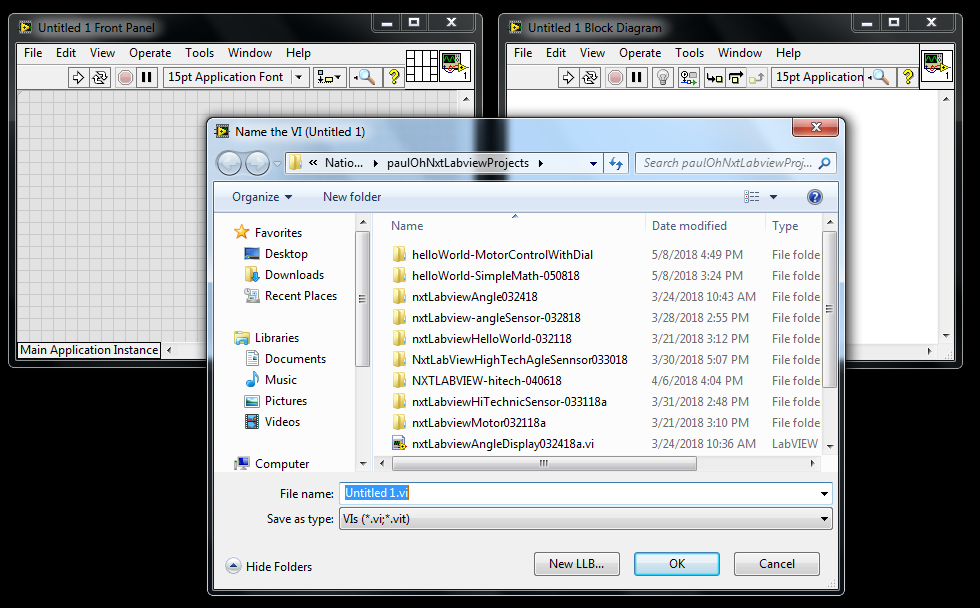


A front panel and block diagram will pop up. Depending on your monitor’s size, you may wish to resize them with your mouse as shown in **Figure B1**. The front panel will be your GUI and serves as your canvas to add virtual controls like knobs, indicators, and charts. Programming occurs in the block diagram by wiring up virtual controls.

Now is the time to name your program. Refer to **Figure B2** (File – Save As) to choose and create a suitable folder where you’ll keep all your future VI files (e.g. a USB thumb drive). Now you’re ready to create new NXT LabVIEW programs.



**Figure B1:** Front panel and Block Diagram



**Figure B2:** Chose File – Save As to store your VI programs e.g. to a USB thumb drive

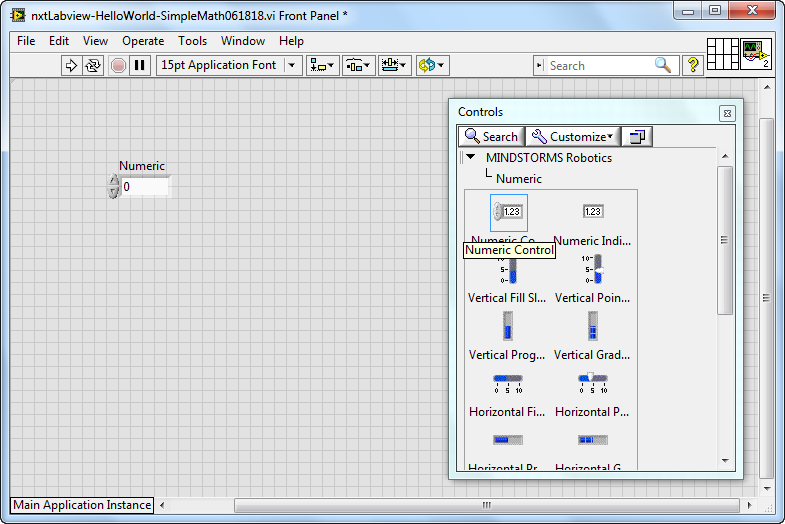
You’ll be repeating these Preamble steps for all your future VI programs.

# **Concept 1:** Creating your first VI: Front Panels and Block Diagrams

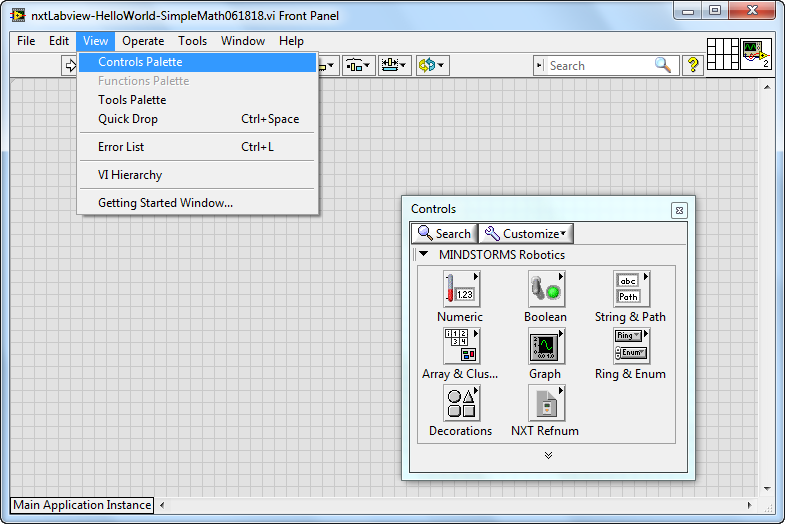
# LabVIEW stands for Laboratory Virtual Instrument Engineering Workbench. In LabVIEW, a virtual instrument has two parts, namely a front panel (the visual front-end) and a block diagram (underlying high-level graphics-based code). One creates the front panel and block diagram by clicking and dragging icons (called controls). The objective in this exercise is to create a simple program that adds displays the sum and product of two numbers. The LED glows when the two numbers are equal.

**Step 1:** Launch LabVIEW to create the following front panel (**Figure 1A** below). Save your VI as nxtLabview-HelloWorld-SimpleMath.vi.

On your front panel, click View – Controls Palette as seen in **Figure 1A** below. This pops up a Controls palette. This palette categorizes GUI components like readout displays (numeric and alphanumeric), knobs and graph charts. Clicking Numeric reveals specific components (**Figure 1B**).



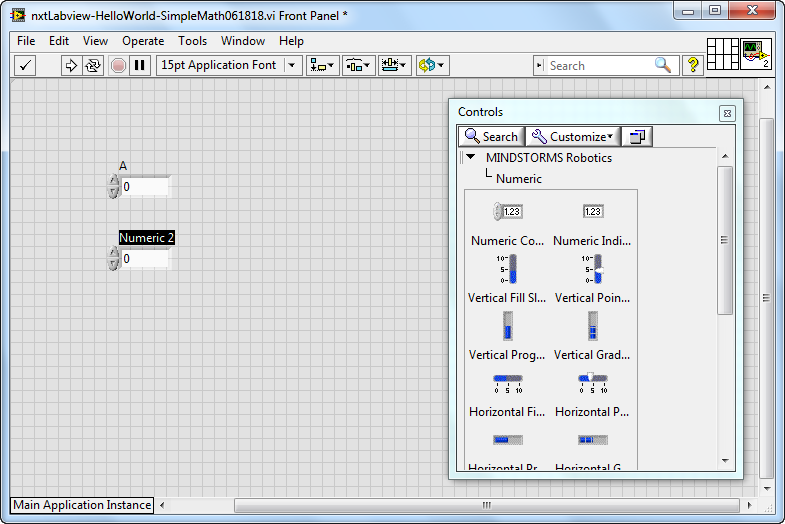
**Figure 1B:** In the Control Palette, clicking Numeric reveals specific components’



**Figure 1A:** On your Front Panel, click View-Controls Palette

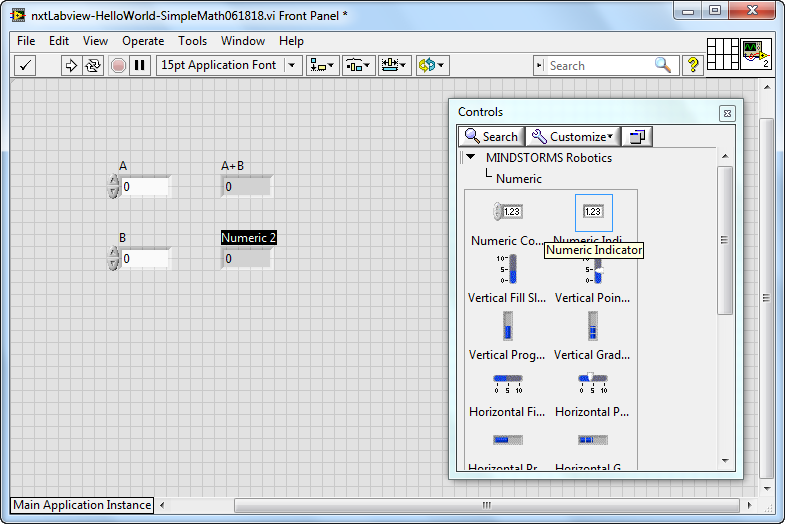
**Step 2:** Create your Front Panel

**Figure 1B** shows a numeric controller that has already been dragged to the Front Panel. Numeric controllers feature up/down rocker switches. These switches allow one to control the value. This numeric controller was renamed “A”. Create a second one. From the Control Palette, click-and-drag a numeric controller into the grey area of your front panel as shown in **Figure 1C**. Double-click the “Numeric” and rename it as “B”



**Figure 1C:** Two numeric controllers (labeled as A and B) were created by clicking-and-dragging “numeric controller” from the Controls palette

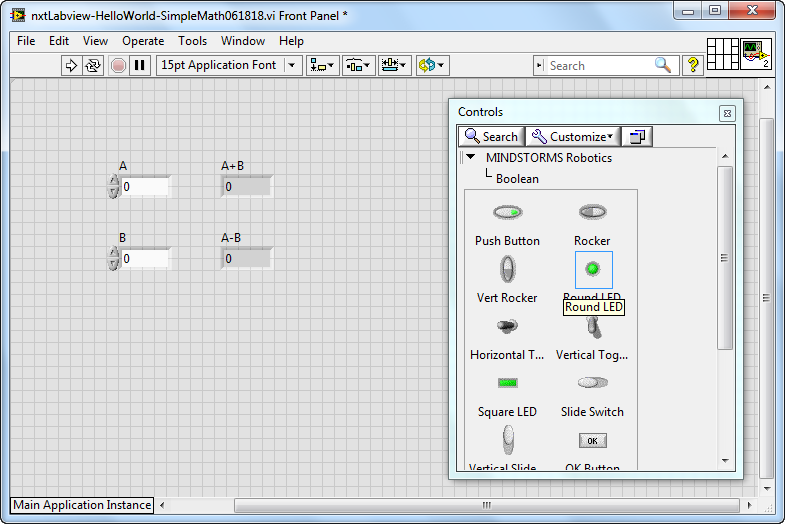
Complete your front panel by adding numeric indicators; click-and-drag a numeric indicator from the Controls Palette as seen in **Figure 1D**.



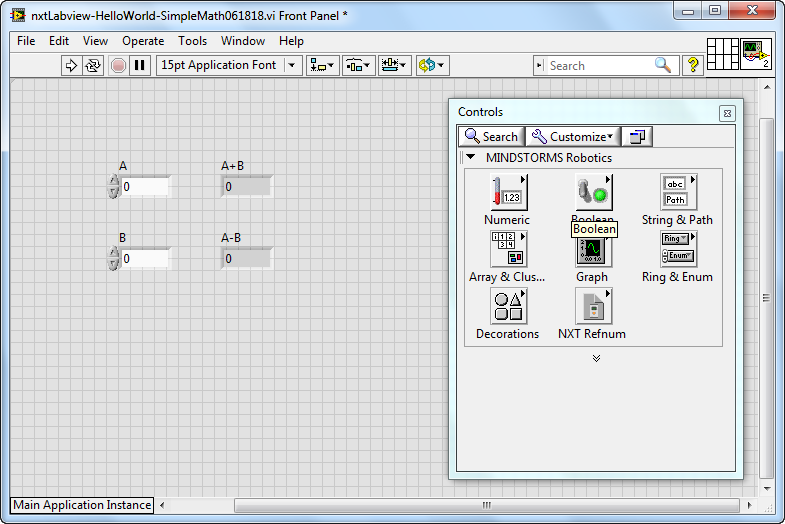
**Figure 1D:** Two numeric indicators, labeled “A+B” and “AxB” were dragged from the Control Palette.

Figure 1D also the 2 numeric indicators. Double-click their labels and rename as “A+B” (for the top one) and “AxB” (for the bottom one).

Next, add a round LED. From the Controls Palette, select Boolean (**Figure 1E**). Click-and-drag a “Round LED” into the front panel (**Figure 1F**).

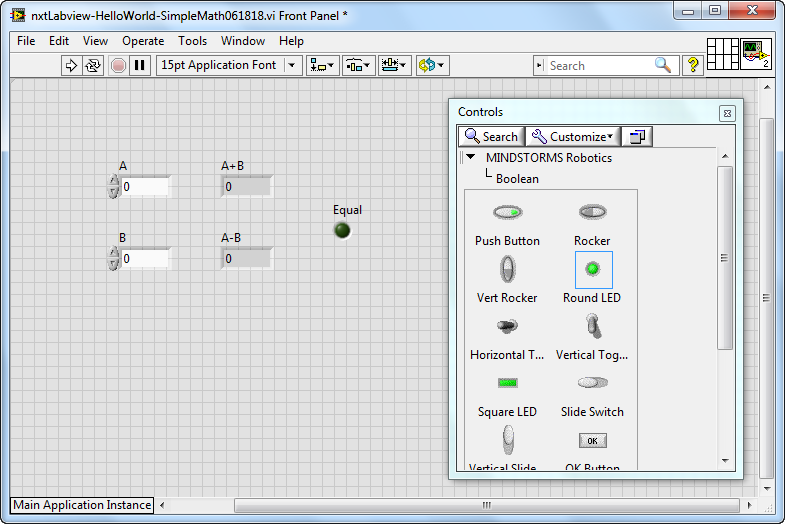


**Figure 1F:** Select Round LED from the Boolean category



**Figure 1E:** Select Boolean from Controls Palette

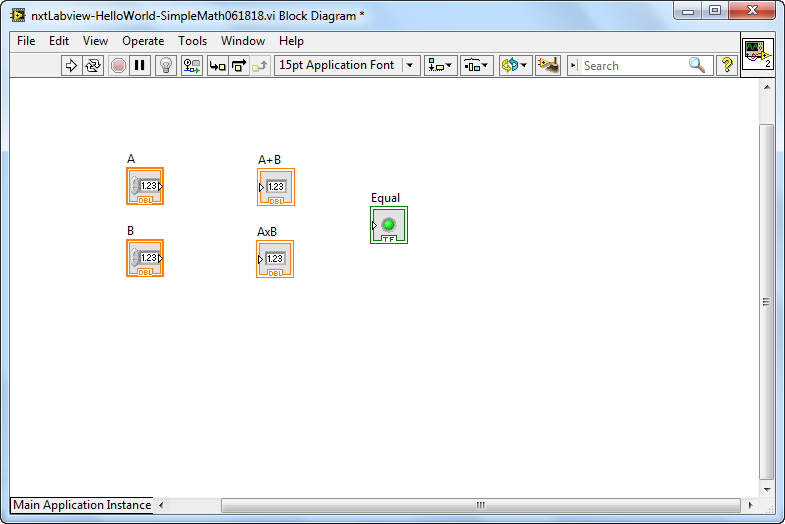
Re-label the round LED as “Equal” and verify that your completed front panel looks like **Figure 1G**. Click File-Save All to ensure your program is saved.



**Figure 1G:** Completed front panel with 2 numeric controls, 2 numeric indicators, and 1 round LED

**Step 3:** Create the Block Diagram

With your front panel completed, it’s time to program the functions of all the controls. From your front panel, click Window – Show Block Diagram (**Figure 1H**). This will pop up the Block Diagram. Click-and-drag components so that the block diagram looks like **Figure 1I**.

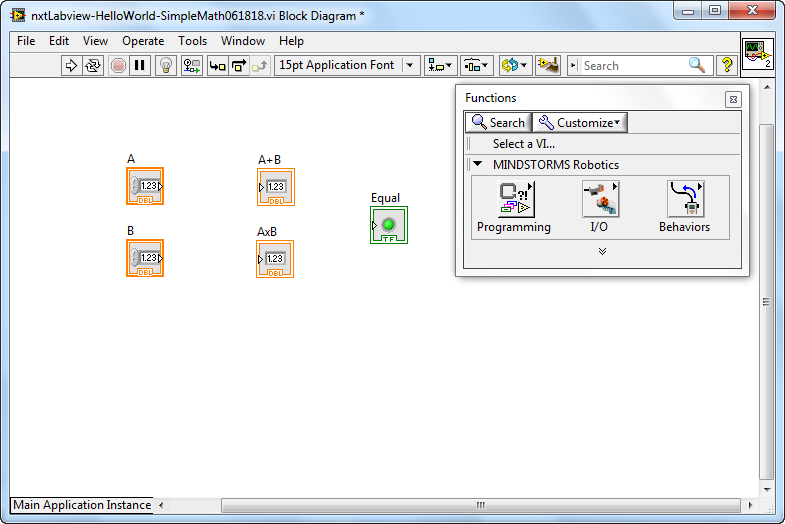


**Figure 1I:** Rearrange components. This will make wiring easier in the next steps to see

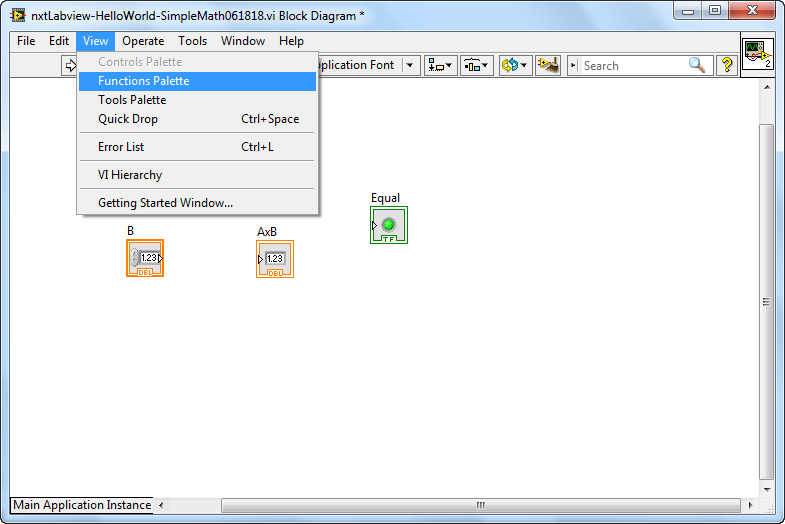


**Figure 1H:** Click Window-Show Block Diagram

Now one must program the relationship between components used in your front panel. Click View-Functions Palette (**Figure 1J**) will reveal sub-categories like Programming, I/O and Behaviors (**Figure 1K**).

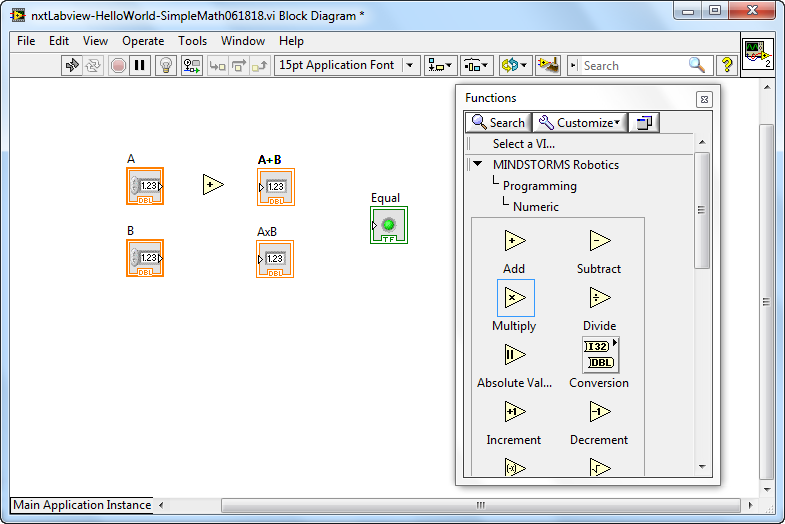


**Figure 1K:** Functions palette shows 3 options.

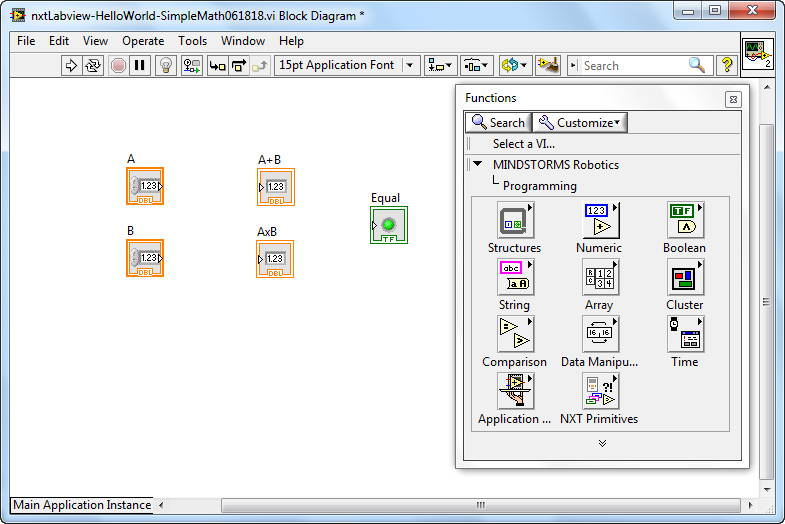


**Figure 1J:** Click View-Functions Palette

Click Programming reveals sub-categories like Structures, Numeric, and Boolean (**Figure 1L**). Click Numeric reveals functions like Add, Subtract and Multiply (**Figure 1M**).

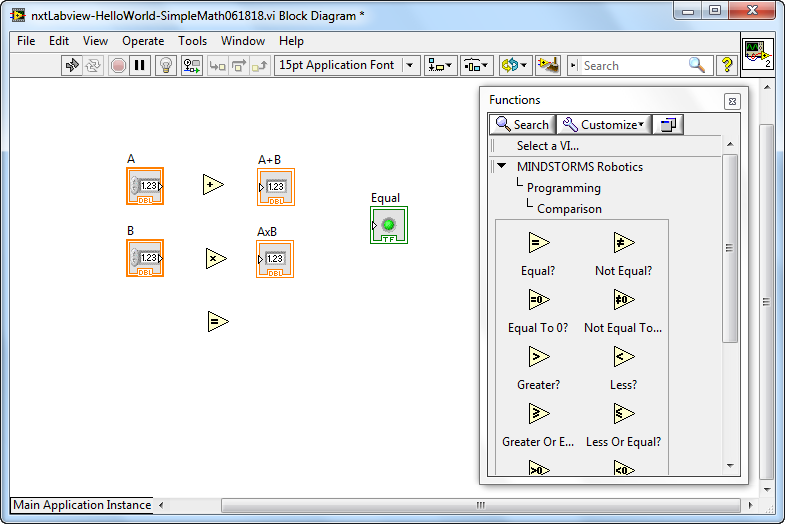


**Figure 1M:** Numeric categories show functions like Add and Subtract



**Figure 1L:** Programming categories reveal components like Structure and Numeric

**Figure 1M** shows that an “Add” function already dragged between the numeric control (A) and numeric indicator (A+B). Select the “Multiply” function so that your block diagram looks like **Figure 1N**. During run-time, when the two values in the numeric controls are equal, the round LED will glow. Thus, under “Comparison” select the “Equal” function and drag into your block diagram (**Figure 1N**).

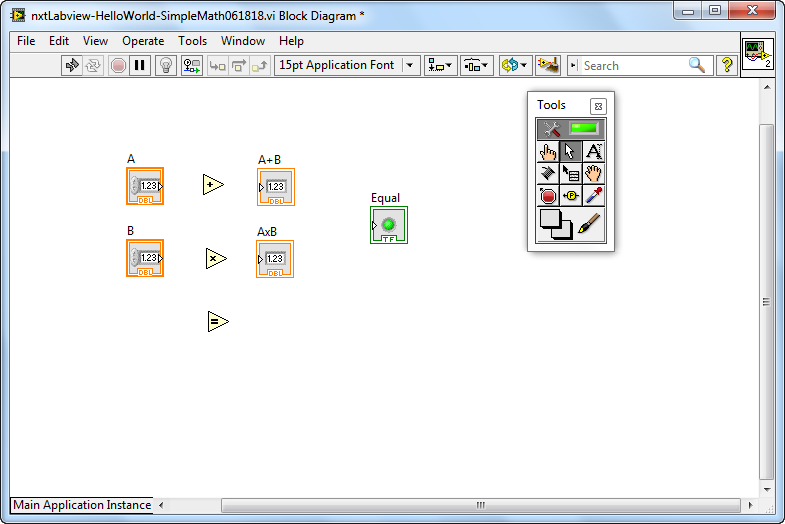


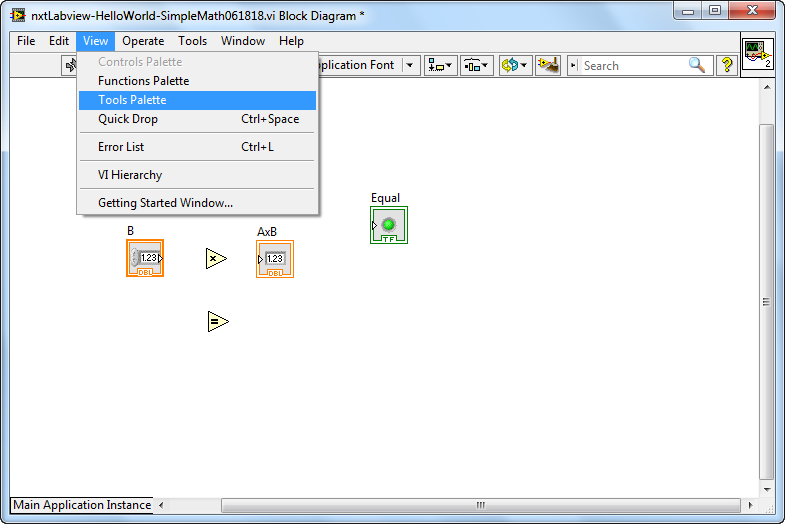
**Figure 1N:** Under the Programming category, select Comparison. Find the “Equal” function and click-and-drag to the block diagram.

**Step 4:** Wiring up the block diagram

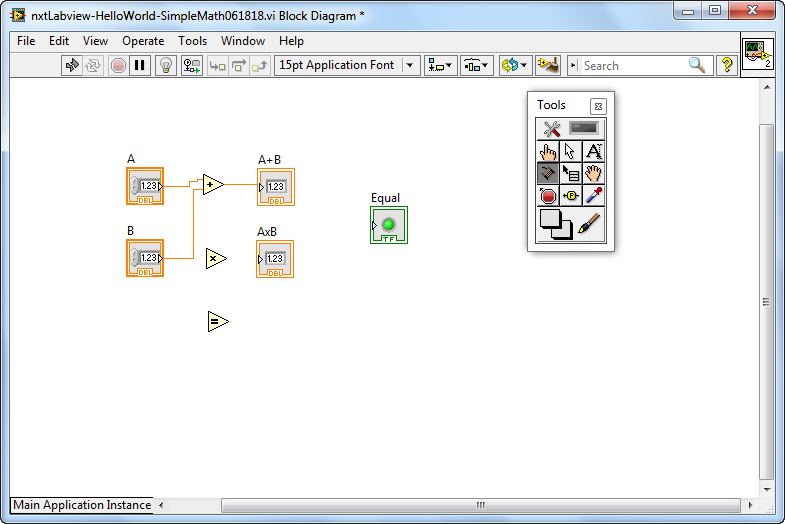
To final step to complete the block diagram, is to wire connections between the functions and controls. Click View – Tools Palette and select the wiring tool (spool icon) as seen in **Figure 1O**.

**Figure 1O:** Selecting View – Tools Palette (left) brings up Tools (right). One can select the wiring tool (which is the spool icon).



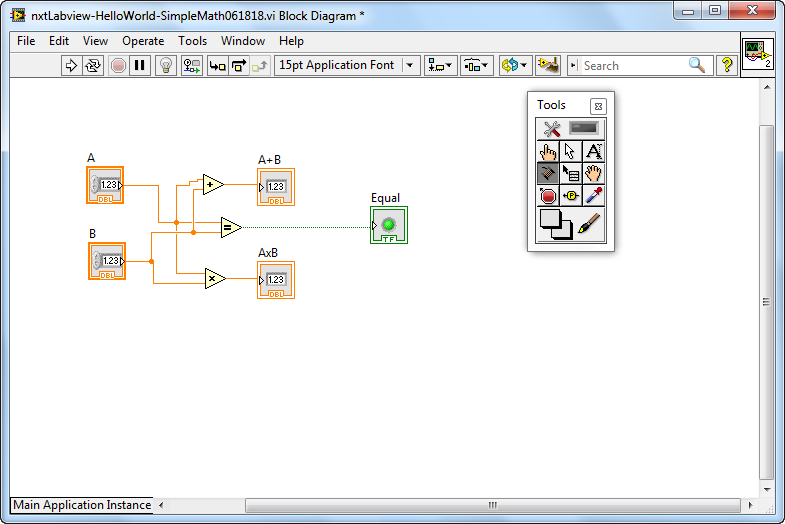


With the wiring tool, one mouse-clicks nodes on the controls. **Figure 1P**, shows wiring of the two numeric controls (A and B), into the left side of the Add function. The output of the Add function is then wired to the input of the A+B numeric indicator.



**Figure 1P:** Wiring of the Add function

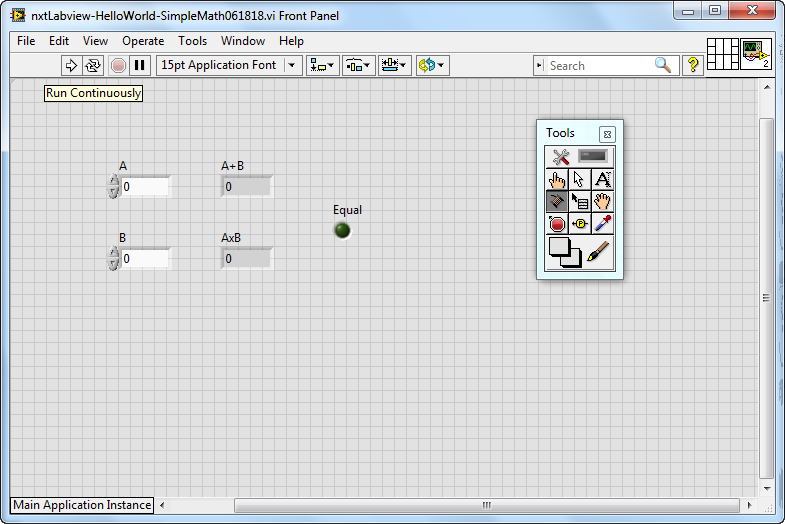
In a similar fashion, wire up the Multiply and Equal functions as shown in **Figure 1Q**.



**Figure 1Q:** One can click a function or controls and move it in the block diagram. This helps make “wiring” easier and cleaner to view and use.

**Step 5:** Execute Program

Return back to your front panel. Click the Run Continuously icon (**Figure 1R**). One can now increment or decrement values in the numeric controls. The numeric indicators will show their sum and product. When the two values of the numeric indicators are equal, the LED will glow green.



**Figure 1R:** The Run Continuously button (double arrows in a circle) is clicked to execute the program.

Congratulations! You’ve written your first LEGO Labview Program!

**Exercise 1:** In LabVIEW create programs for the following:

* 1. Accepts temperatures in Celsius and displays its Fahrenheit equivalent (try pointer slides)
  2. Accepts 3 numbers and displays the average
  3. Accepts 2 numbers and displays which is the maximum

**Concept 2:** Case Structures

Branch statements are those that test a condition (true or false) and execute commands based on those conditions. LabVIEW has several controls for branching. In the block diagram programming, the Case Structure control is invoked under the Executive Control category.

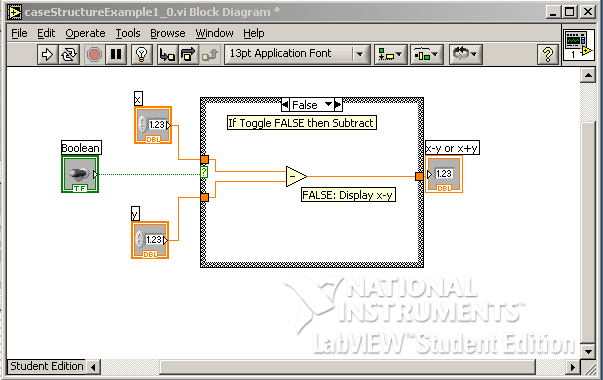
**Step 1:** Create the following LabVIEW front end. Save your VI as nxtLabviewCaseStatement.vi – Note that toggle switch by default is FALSE by with a “Switch When Pressed” mechanical action.

..\..\..\nationalInstruments\paulOh\figures\caseStructureExample1_0-frontPanelWithText.wmf

**Step 2:** Create the Block Diagram (True Case: adds the two numbers). Note: clicking the down arrow in the Condition Terminal sets the True condition.

..\..\..\nationalInstruments\paulOh\figures\caseStructureExample1_0-blockDiagramTrueWithText.wmf

**Step 3:** Create the False Case condition (False case: subtracts the two numbers)



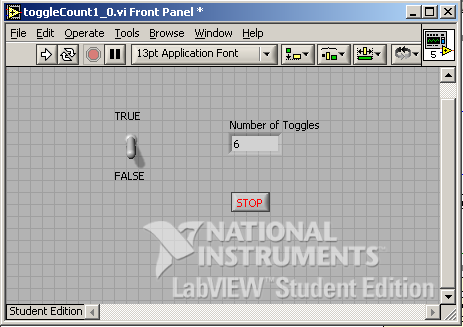
**Exercise 2:** In LabVIEW create programs for the following:

* 1. Given an input in inches, the equivalent in centimeters is displayed (by default) and the equivalent in feet is displayed (if the switch is toggled)
  2. The maximum or minimum of two numbers is displayed based on toggle position

**Concept 3:** While Loops

Loops that repeat until a condition is met are found in every programming language and LabVIEW is no exception. The While-loop repeats commands in its block control while the condition is TRUE. The While-loop control also has an iteration terminal that increases each time the block is executed.

**Step 1:** This LabVIEW VI counts the number of times the switch is toggled. Create the following front panel. Save it as nxtLabviewWhileStatement.vi.

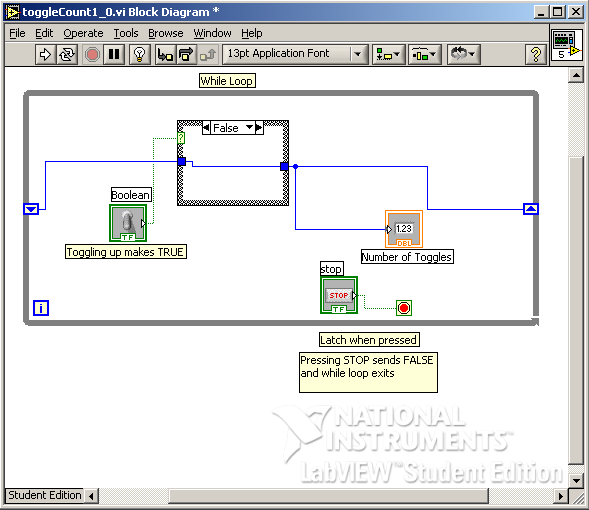


Note: the toggle switch should have its mechanical action set to “Latch when released”.

**Step 2:** Create the following block diagram. A case structure control is inside a while-loop control. When the case is TRUE, a variable (shift register) increases, thereby displaying the number of toggles. The while-loop control stops when the user hits the Stop button.

..\..\..\nationalInstruments\paulOh\figures\toggleCount1_0-blockDiagramTrueWithText.wmf

**Step 3:** Add the False condition in the conditional terminal. Note that there are no contents (i.e. do nothing) in the case structure



**Step 4:** Run your program and toggle the switch. Also, try clicking on the Highlight Execution (light bulb icon) button from the Block Diagram menu bar and notice what happens when you execute your program.

**Exercise 3:** In LabVIEW create programs for the following:

* 1. The display increases by 2 each time the switch is toggled
  2. Increments or decrements depending on the users choice (i.e. using a 2nd toggle switch)
  3. Create a for-loop example.