Slide Numbers Index and Hyperlink (106 slides)

CAM83012E
Fall 2018 (Week 1)
Microcomputer Interfacing

My Contact Information

Michel Hanbury
Office Room – CA532 (ACCE building 5th floor)
Office Tel. - 613-727-4723 x3471
Labs in CA219B
hanburm@algonquincollege.com

Fall 2018 CAM8302E Microcomputer Interfacing

Microcomputer Interfacing:

During the course you will connect a microcomputer (Arduino) or a data acquisition device (myDAQ) to analog and digital inputs and outputs.

Use Brightspace LMS for labs, notes and quizzes.

Grading Information (2018)

Lab Section:

Lab Assignments

and Pre-Lab Quizzes ---------- 40%

Theory Section:

Quizzes (on-line -- in theory class) --- 10%

Midterms (2) ---------- 20%

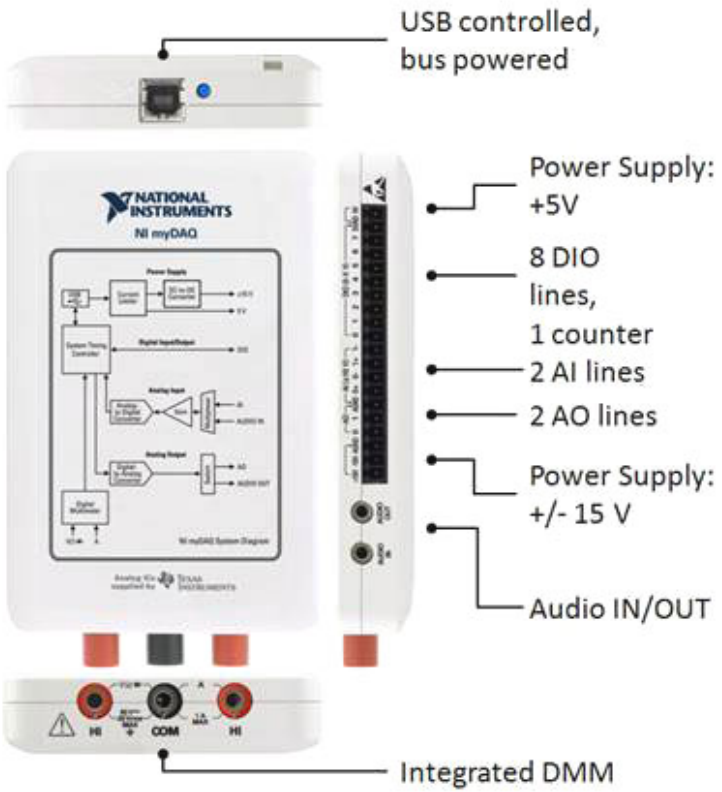
Exam ---------- 30%

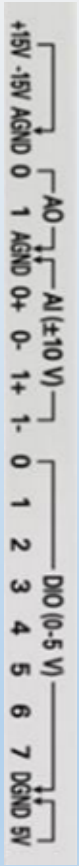
Important Note: You must pass both the theory and lab section to pass the course.

NI myDAQ USB Interface

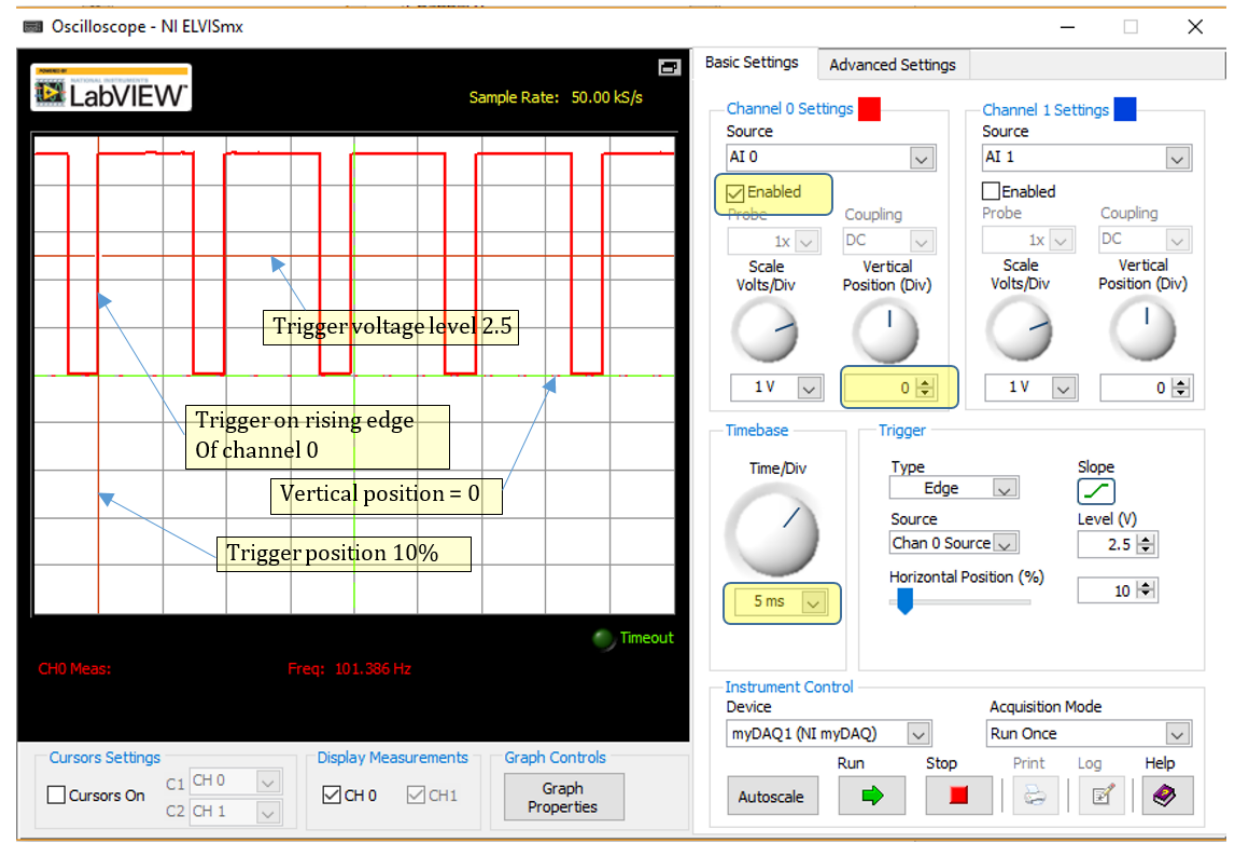
myDAQ Connections

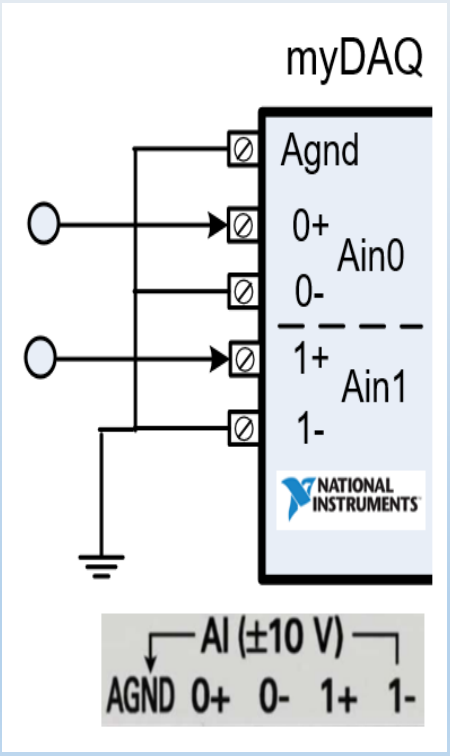
Analog Input:
2 channels, 200kS/s, 16-bit
Analog Output:
2 channels, 200kS/s, 16-bit
3.5mm stereo audio jacks
Digital I/O: 8 LVTTL lines
Counter: 1 counter/timer
Integrated DMM: V, A, Ohm
Power Supply: +5V, +/-15V
Screw term + mass term option
Bus Powered (USB) operation



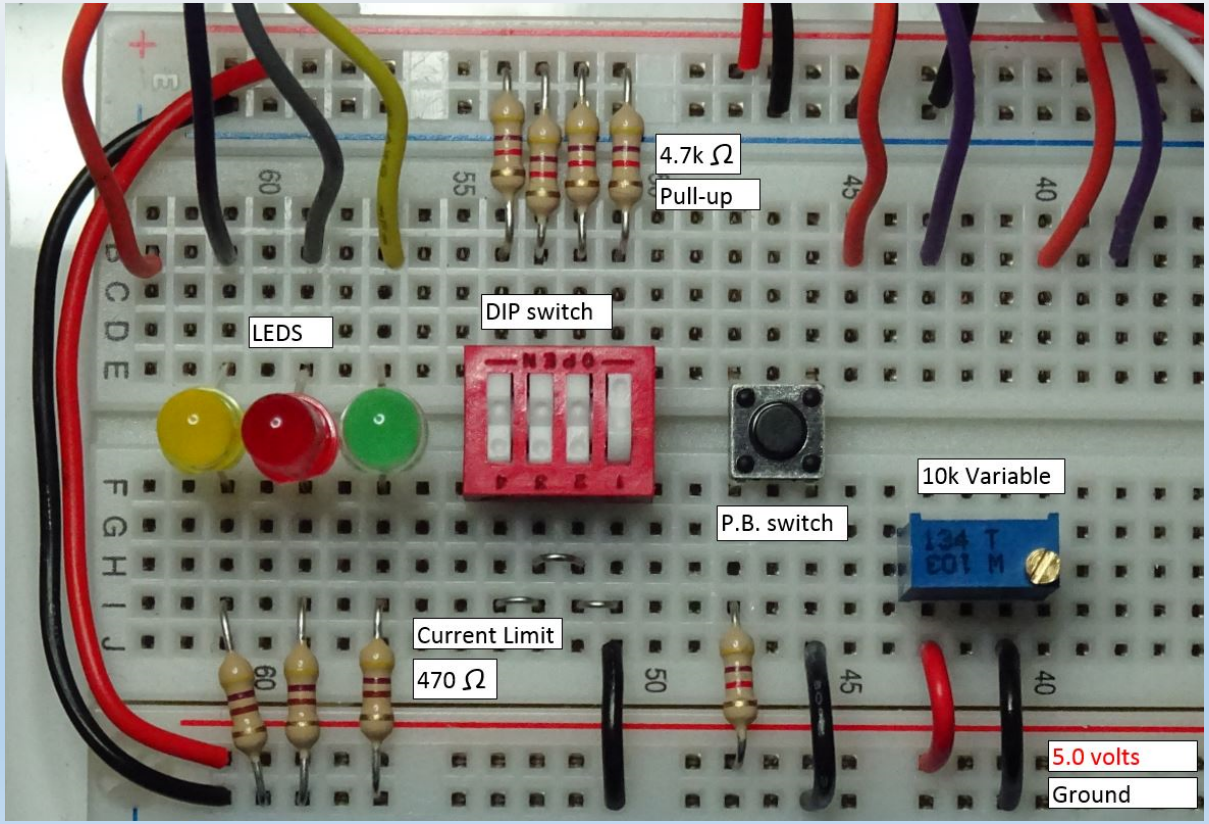


NI myDAQ USB Oscilloscope





myDAQ Interface Circuit



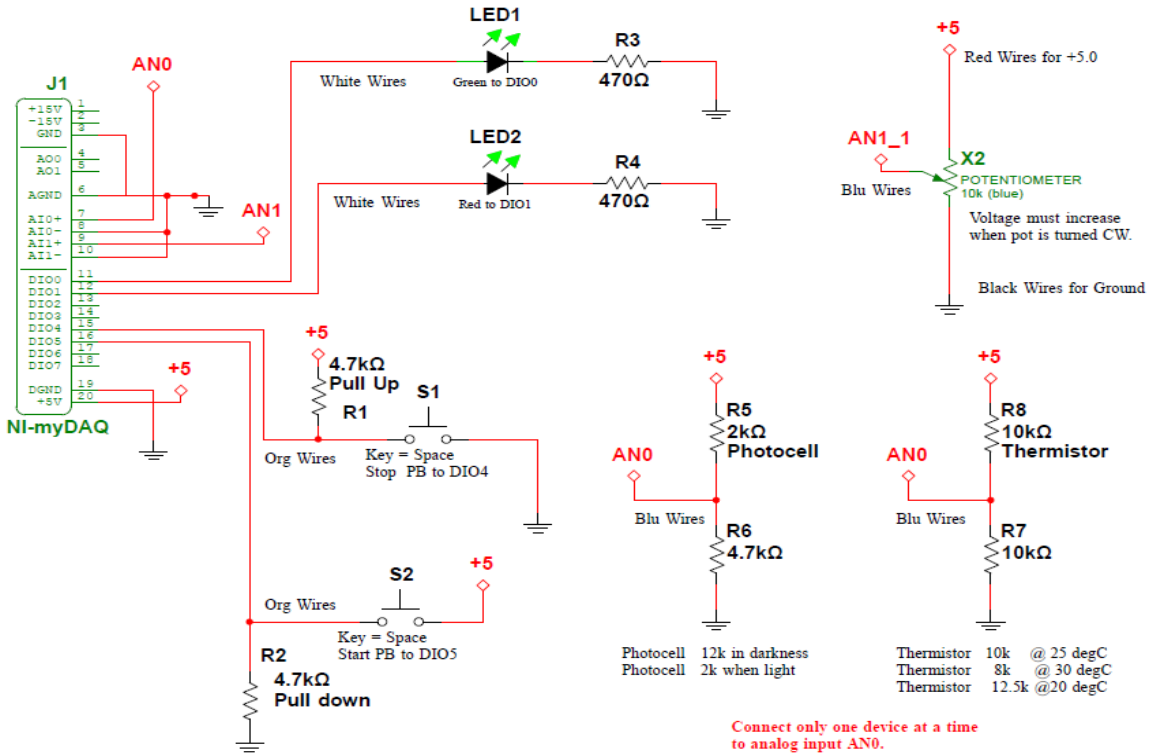
Wiring:

Do not place wires over components.

Use only red and back wires for connections to ground and the supply voltage.

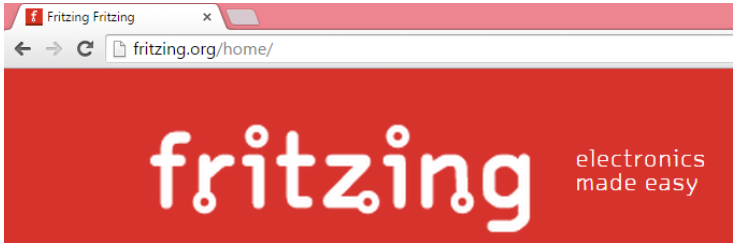
Keep the components organized.

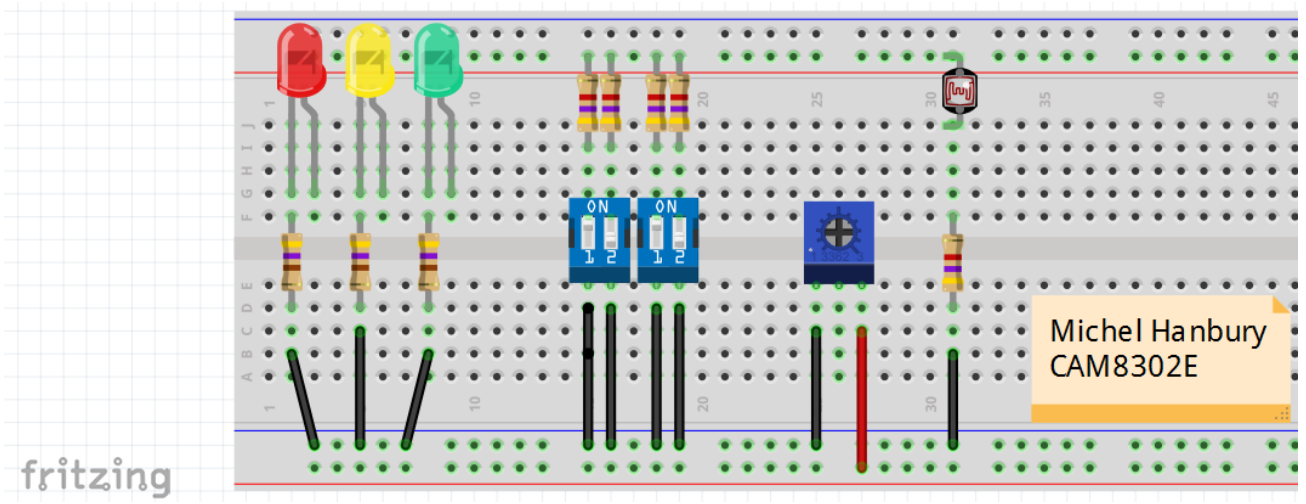
myDAQ Basic I/O Circuit (2018):

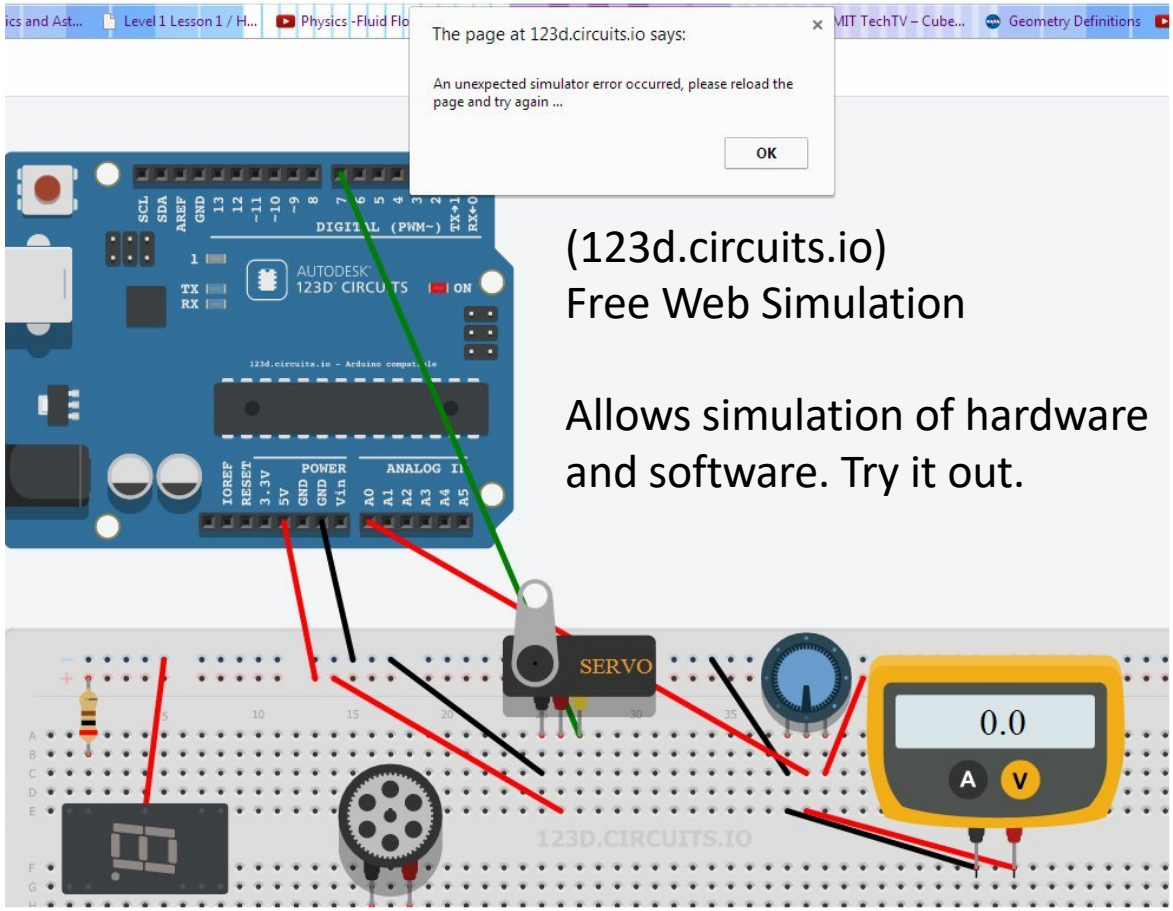


This is a typical schematic using Multisim 14. The schematic shows all the components, component values and their connection to ground, the supply, other parts of the circuit or to the myDAQ interface device.

Fritzing – free software to draw circuits and board layouts. You may also use NI Multisim 14

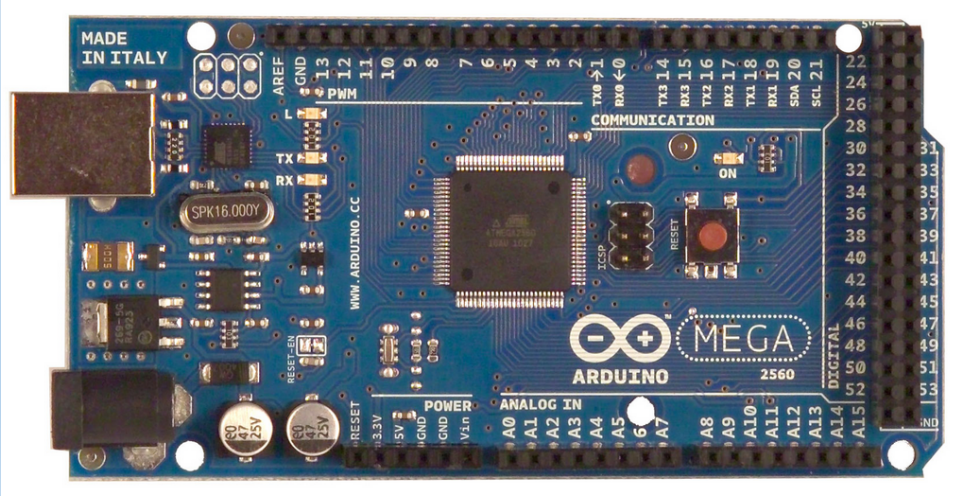





The page at 123d.circuits.io says:
An unexpected simulator error occurred, please reload the page and try again ...
OK
(123d.circuits.io)
Free Web Simulation
Allows simulation of hardware and software. Try it out.
SERVO
0.0
A
V
123D.CIRCUITS.IO

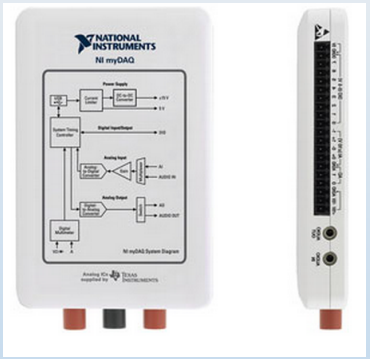
Arduino Microcontroller

Powerful high speed (16 MHz) Atmega 2560 processor
52 I/O pins
8k RAM, 256k of FLASH, and 4k EEPROM
Free IDE (Integrated Development Environment) – easy to use and program.
Low power device. Automatically starts program when powered.
A/D converters, digital I/O, serial I/O, counters, timers, communications

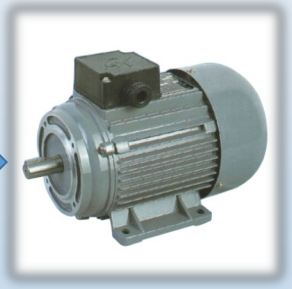


myDAQ USB Output Interface

Computer

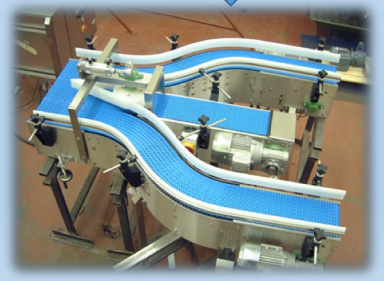


Interface



Motor

High current, high voltage signals must connect to the myDAQ using an electronic interface circuit to provide the device with voltage and current level which can be safely handled by the myDAQ.

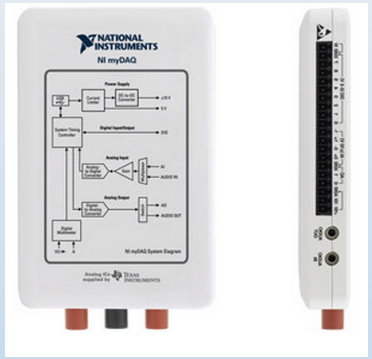


Conveyor

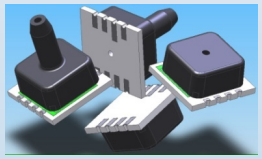
Hoist

myDAQ USB Input Interface

Computer



Interface

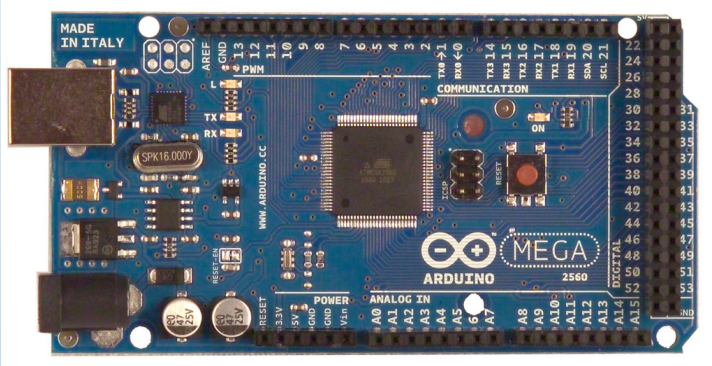


Pressure Sensors

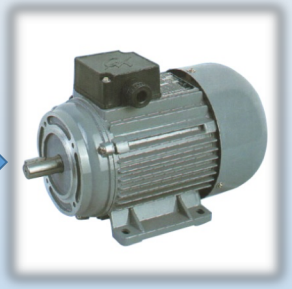
Limit switches

Temperature Sensors

Arduino Embedded Output Interface

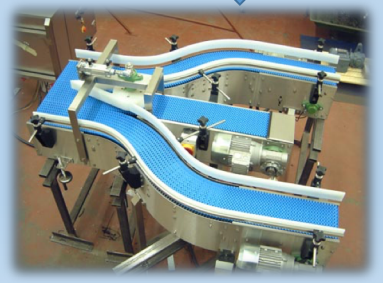


Interface



Motor

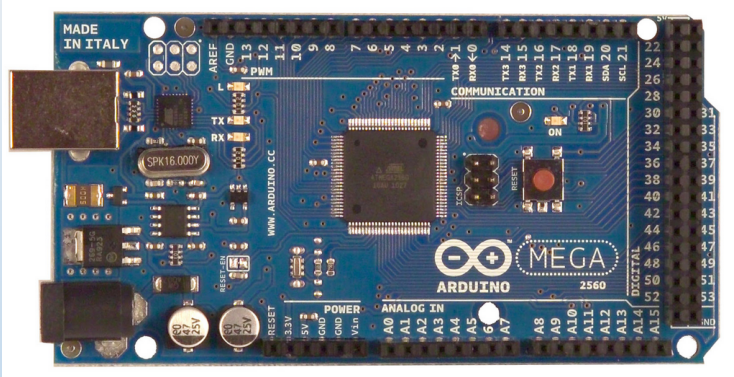
High current, high voltage signals must connect to the Arduino using an electronic interface circuit to provide the device with voltage and current level which can be safely handled by the Arduino.



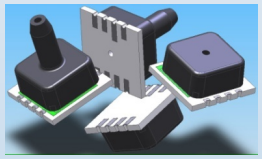
Conveyor

Hoist

Arduino Embedded Input Interface



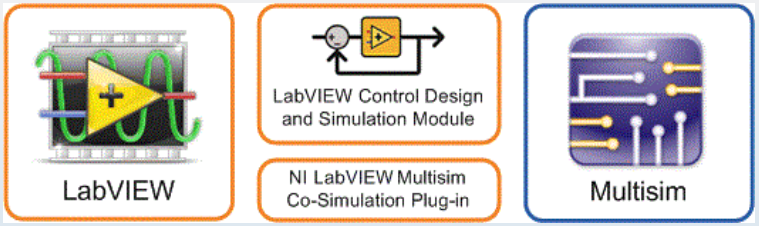
Interface



Pressure
Sensors

Limit
switches

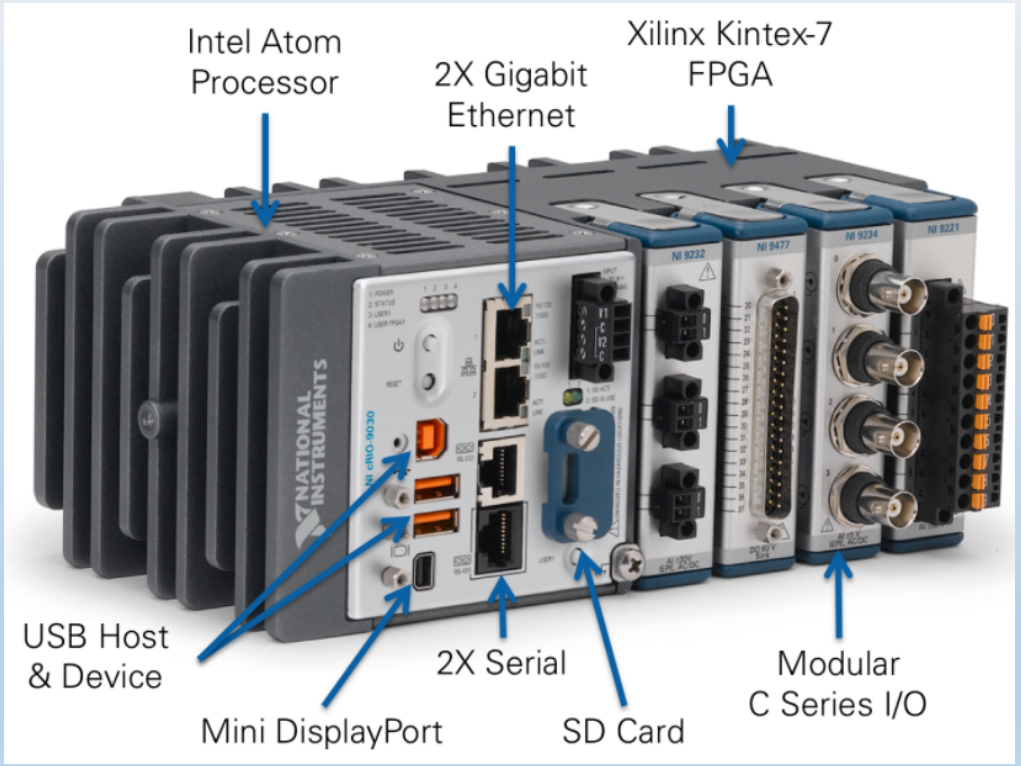
Temperature
Sensors



Overview

NI LabVIEW is a graphical programming language designed for engineers and scientists to develop test, control, and measurement applications. Created and optimized more than 20 years ago for engineers and scientists, the intuitive nature of LabVIEW graphical programming makes it easy for educators and researchers to incorporate the software in a range of courses and applications.

Graphical system design is a modern approach to designing, prototyping, and deploying embedded systems. It combines open graphical programming with hardware to dramatically simplify development.

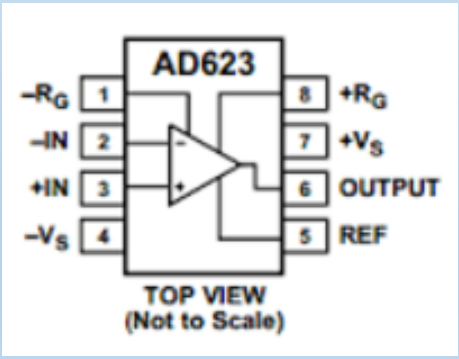


You will solve problems such as:

How do you measure the signal from a thermocouple that has an output of 42 uv/degree C.

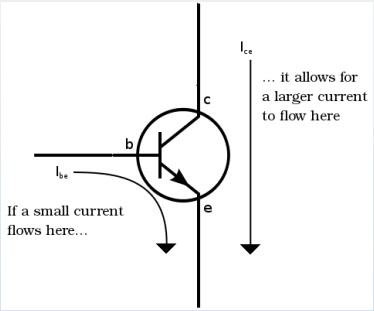
Using an A/D converter with a 10 bit 5.0 volt ADC with a resolution of 4.9 millivolts.

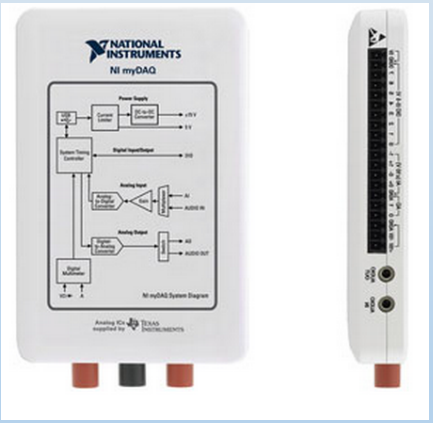
In this case an amplifier with a high gain could be used.



How do you measure the signal from an industrial discrete sensor with an output of 0 or 24 volts DC.

Using the myDAQ and LabVIEW or an Arduino(They operate at 5 volts and 3.3 volts.





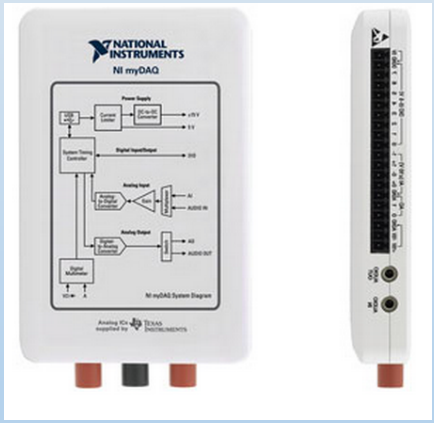
Digital 0 to 24 volts

Digital 0 to 5 volts

How do you measure the signal from an industrial discrete sensor with an output of 4 or 20 mA using the myDAQ and LabVIEW or an Arduino, they typically measure analog signals in the range of 0 to 5 or 0 to 10 volts?

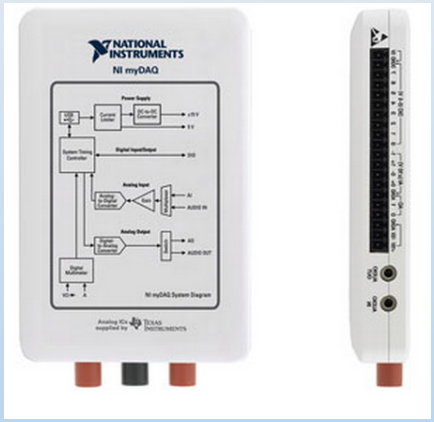
Ohms law, V=IR, V=volts, I=amps, R=ohms. So 4mA (0.004A)*250ohms = 1V
Likewise 20mA (0.020)*250 = 5V.

4 to 20 mA



Analog 0 to 10 volts.

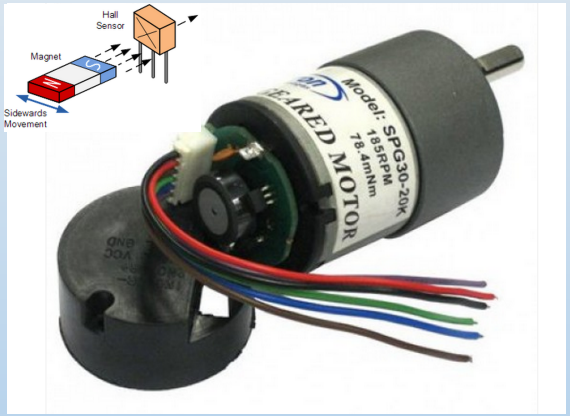
How do you control a 3 phase 30 amp motor using a myDAQ or Arduino?

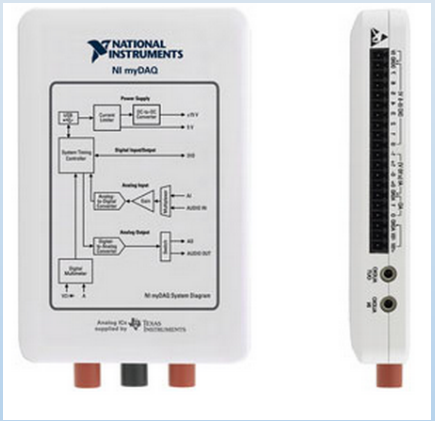


3.3 volts, 4 mA

3 PH 380 volts 30 amps

How do you measure the speed of a motor using hall effect quadrature encoder?

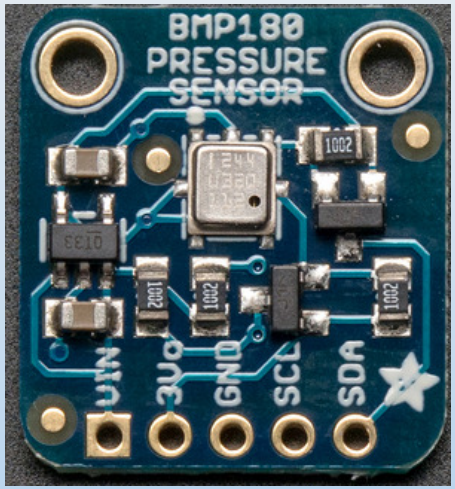




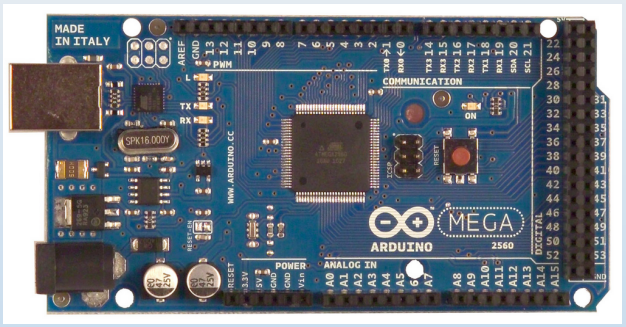
Pulses

High speed counter.

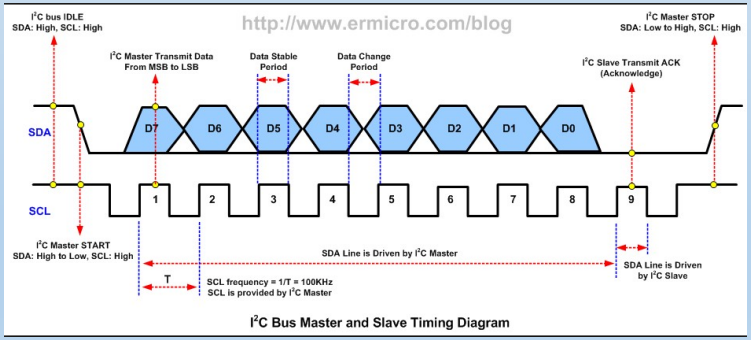
How do you measure altitude or barometric pressure using an Arduino controller?



2 wire I2C

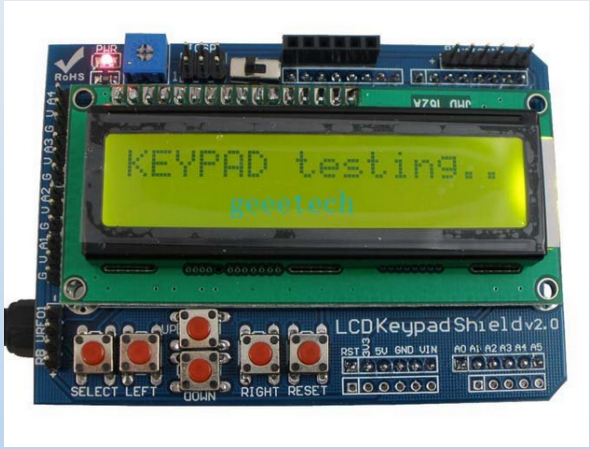


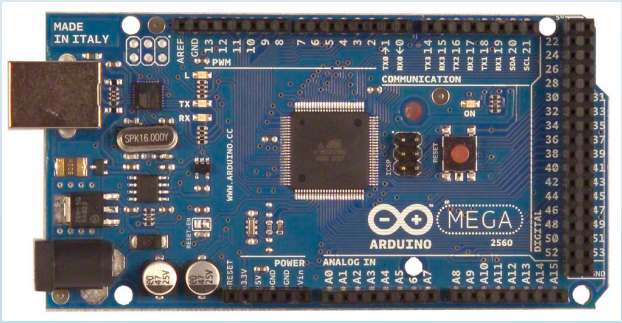
2 wire I2C



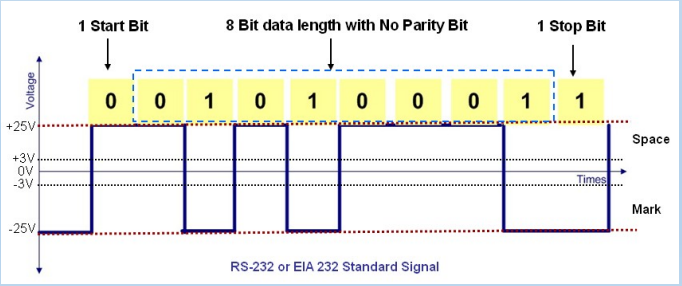
Inter IC communications

How do you display data in an embedded system using an LCD display?





Parallel or Serial Data?



Organizations that Use LabVIEW

National Research Council Canada | Conseil national de recherches Canada | Canada

Français | Contact Us | Help | Search | Canada Site
Aerodynamics | Flight Research | Structures and Materials | News | NRC Site
Gas Turbines | Manufacturing | | Links | About Us

Home > Aerodynamics Laboratory >

9 m x 9 m Low Speed Wind Tunnel

The 9 m x 9 m Low Speed Wind Tunnel facility is located on the National Research Council (NRC) campus adjacent to the Ottawa International Airport. In operation since 1970, the facility serves the aerodynamic testing requirements of government agencies, research institutes and private companies, in addition to supporting ongoing internal R&D activities at the NRC Institute for Aerospace Research (NRC Aerospace). Major renovations were recently carried out on its fan drive, balance weigh-beam controls and data acquisition systems, as well on important mechanical components such as the cooling system and main drive shaft components.

General areas of expertise include:

- wind tunnel testing techniques
- wind tunnel instrumentation
- model design and manufacture
- customized data processing
- computational fluid dynamics.

NRC 9 m x 9 m wind tunnel

National Research Council Canada | Conseil national de recherches Canada | Canada

Français | Contact Us | Help | Search | Canada Site
Aerodynamics | Flight Research | Structures and Materials | News | NRC Site
Gas Turbines | Manufacturing | | Links | About Us

Home > Aerodynamics Laboratory > Bluff Body Aerodynamics > Surface Vehicle Aerodynamics >

Racing Vehicles

The flow of air around a high-speed racing vehicle can have a dominant influence on the vehicle's top speed, acceleration, traction, stability, and engine performance. Optimization of both external and internal flows can significantly improve the vehicle under development.

NASCAR racing vehicle

NRC's major wind tunnels are among the largest and most versatile facilities available for vehicle aerodynamic research and development in North America, allowing a full range of testing from model-scale development in the high-speed 2 m x 3 m wind tunnel (maximum speed 300 mph, 500 km/h) to full-scale refinement in the large 9 m x 9 m wind tunnel (max. speed 120 mph, 200 km/h). In recent years, they have been used to test a wide range of racing vehicles, including Indy / Indy Lights cars, Group C cars, Top-Fuel dragsters, Stock Cars, and racing and record motorcycles.

Organizations that Use LabVIEW

Natural Resources Canada

Ressources naturelles Canada

Canada

Test Technician

Nanometrics - Kanata, ON

Skills in MS Excel, MS Access, LabView, and DXDesigner are an asset. Nanometrics is looking for contract Test Technicians to join our production team in Kanata....

Easily apply

15 days ago - save job - more...

indeed

Nanometrics Inc. is a world-class provider of precision instrumentation, network technology and software applications for seismological and environmental research.

Labview jobs in Ottawa, ON

USING AUTOMATION

C-FER conducts a wide range of large-scale tests on oilfield equipment. Each test requires custom built control and monitoring systems using LabVIEW.

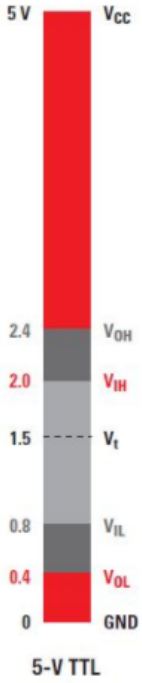
One testing system controls a multi-component, servo-hydraulic system with a total load capacity of 16 million pounds. This is used to determine the tensile capacity of large diameter pipe for gas pipelines.

Automation of the heating and cooling systems used for testing the performance of high-temperature well casing connections has reduced temperature cycle time by 40% while improving temperature variance from the target to less than 2°C. All of these systems are being continuously updated to improve the accuracy and efficiency of conducting the tests.

C-FER technicians utilizing automated instrumentation to remotely monitor loads, pressures, temperatures, displacements and strains on pipe specimen during testing.

myDAQ Specifications

myDAQ Digital I/O Specifications:

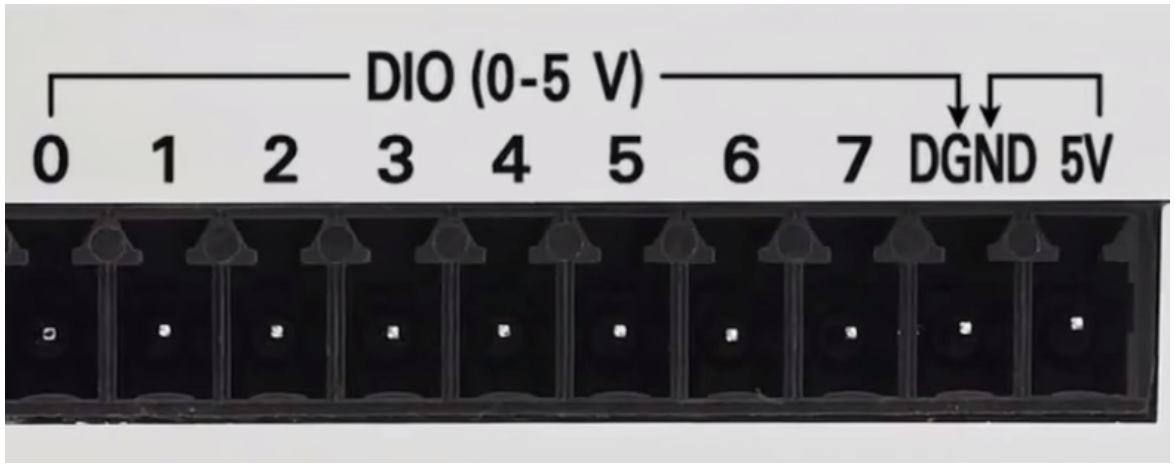


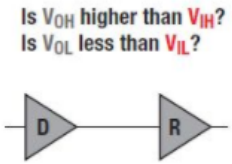
Digital In:

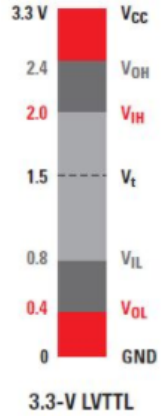
- 3.3 volts LVTTL inputs
- 5.0 TTL volt tolerant inputs

Digital Out:

- 3.3 volt LVTTL
- Max. 4 mA drive

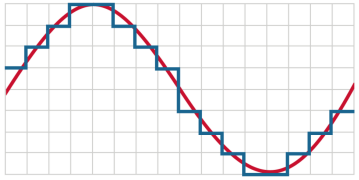






LVTTL – low voltage transistor-transistor logic.

myDAQ Analog I/O Specifications:

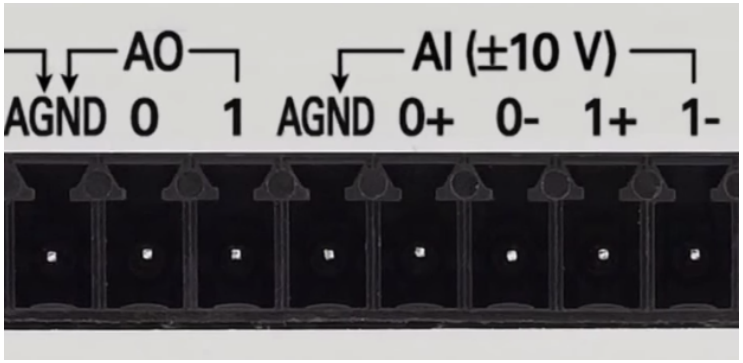


Analog Out

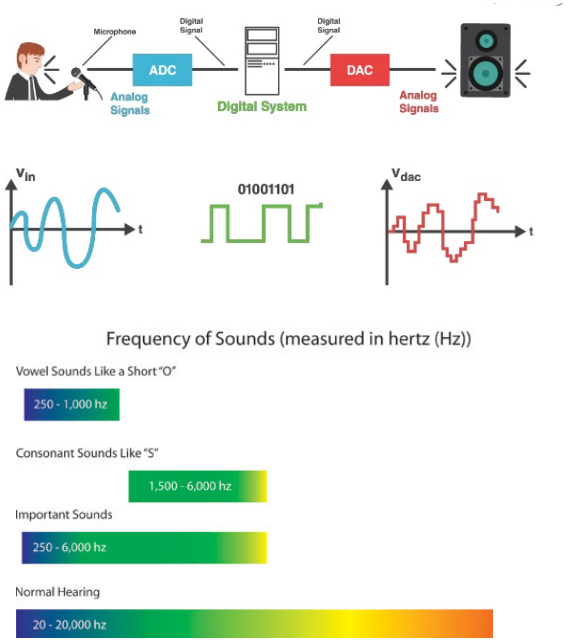
- +/- 10 volts
- 2 mA drive
- 16 bit resolution
- 200 kilo samples/second

Analog In

- +/- 10 volts
- 10 G ohms input impedance
- 16 bit resolution
- 200 kilo samples/second



For single ended inputs, using a ground reference, the 0- and 1- inputs must connect to ground.
Resolution = 20v/65,535 = about 0.3 mV.

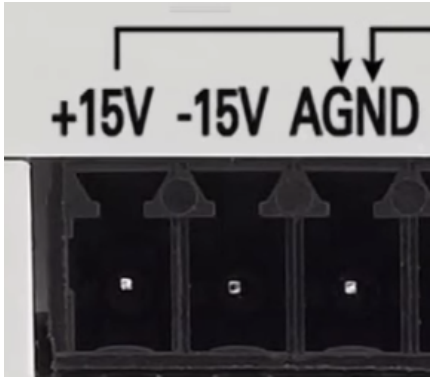


myDAQ Power Supply Specifications:

- +/- 15 volts
- 32 mA drive / supply
- +/- 14 volts at max. load

- + 5.0 volts
- 100 mA max. drive
- 4.0 volts(min) at maximum load

Note the current limitations of the supplies. Going over the current limits will cause the supply voltage level to decrease. Powered by USB 5v, 500 mA. Most laptops have an automatic resettable fuse if overcurrent occurs.



Caution:

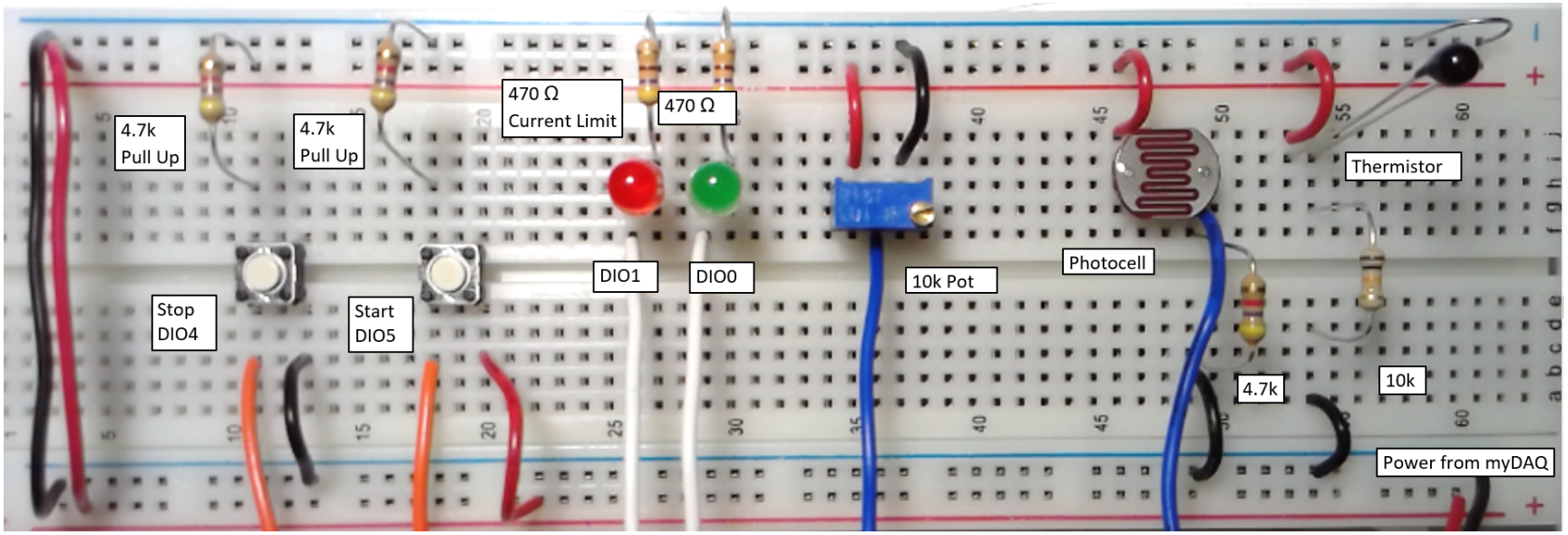
Do not connect another power supply to the myDAQ.

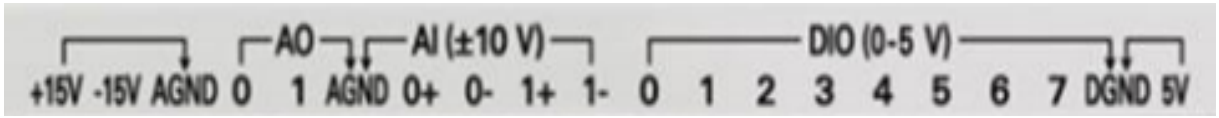
There is a 1.25 A internal fuse.

Protecting your myDAQ.

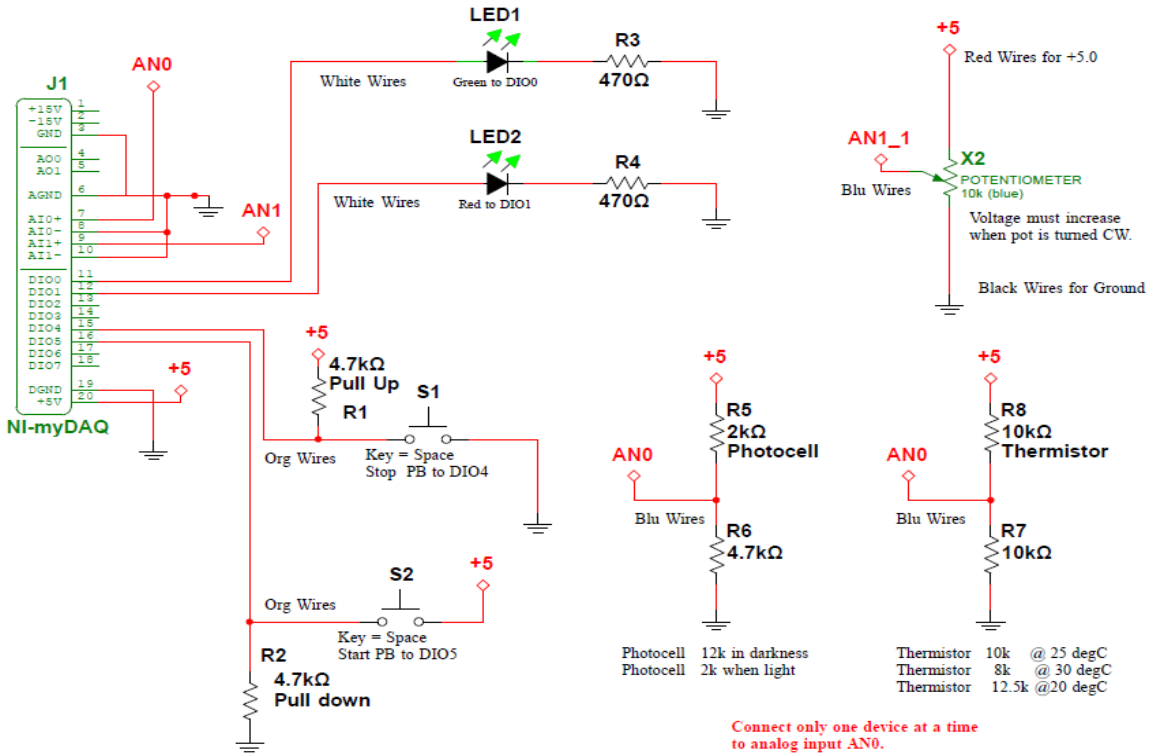
- Do not change your circuit wiring while connected to the USB supply.
- Use the myDAQ voltmeter to test circuit voltages.
- After making a change to your circuit test it with a basic program (VI).
- Do not leave your circuit connected to the USB if it does not function as expected. Something may be drawing too much current and may damage your circuit or the myDAQ.
- Always have a circuit diagram and a layout diagram available when troubleshooting your circuit.

myDAQ Basic I/O Circuits And Sensors



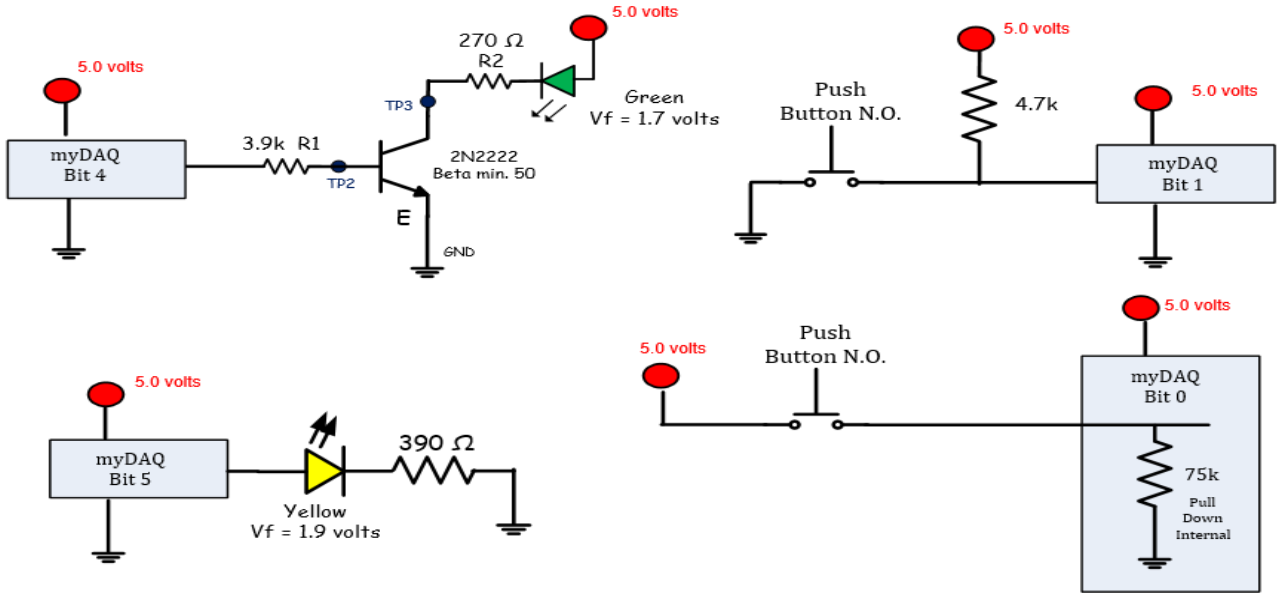


myDAQ Basic I/O Circuit (2018):

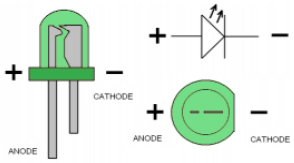


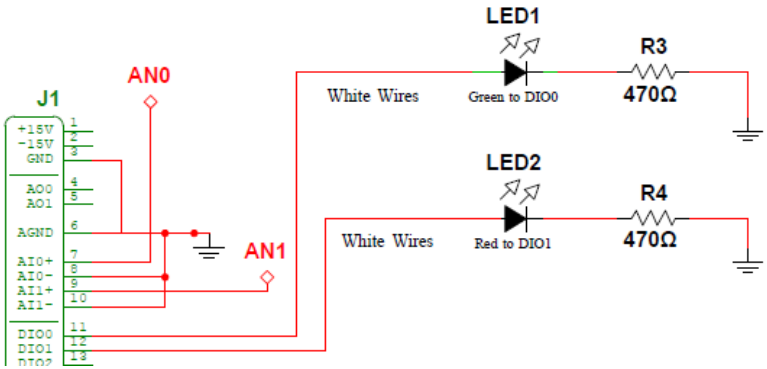
This is a typical schematic using Multisim 14. The schematic shows all the components, component values and their connection to ground, the supply, other parts of the circuit or to the myDAQ interface device.

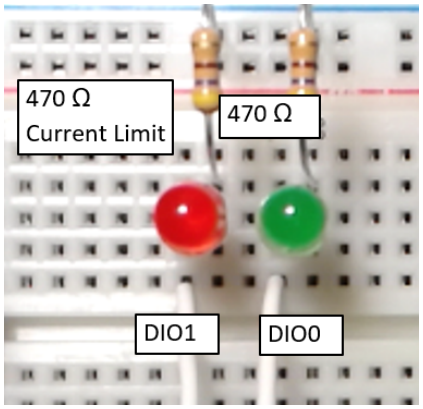
myDAQ Basic I/O Circuits And Sensors



myDAQ Basic I/O Circuits And Sensors

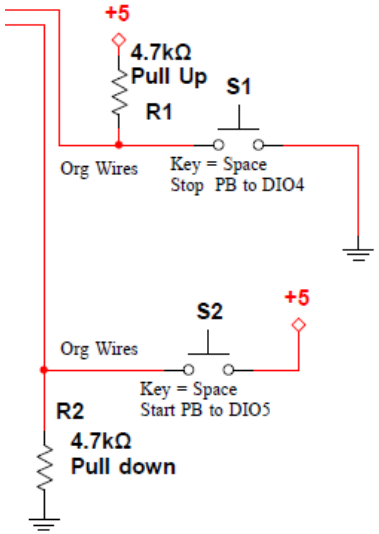


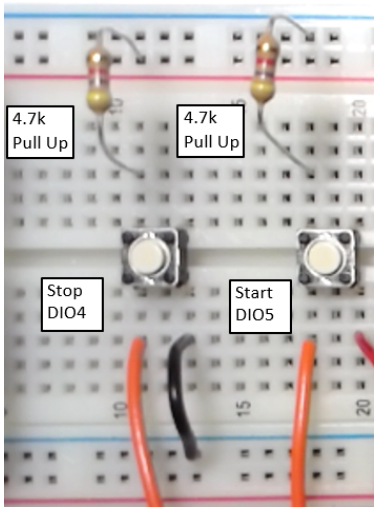


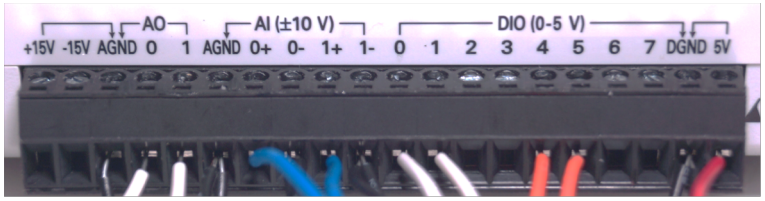


Wire neatly on the prototype board, use red wires to connect to the supply and black wires to connect to ground. The circuit uses 470 ohm resistors for current limiting and 4.7 k pull up resistors to prevent a floating input. An open switch is pulled to 5 volts, a closed switch connects to ground. The 470 ohm led resistor limits the myDAQ current to about 3.4 mA, it has a max of 4.0 mA.

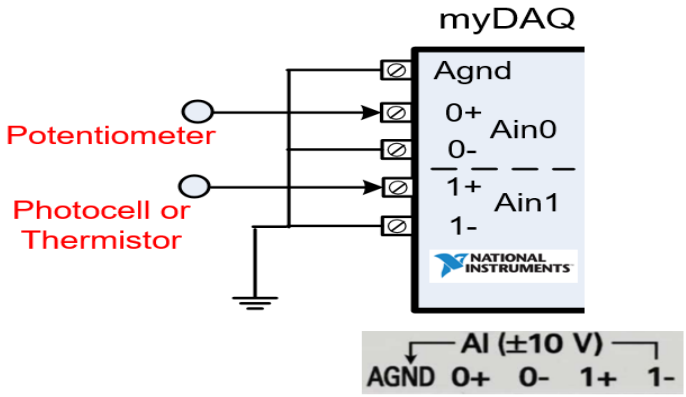
myDAQ Basic I/O Circuits And Sensors

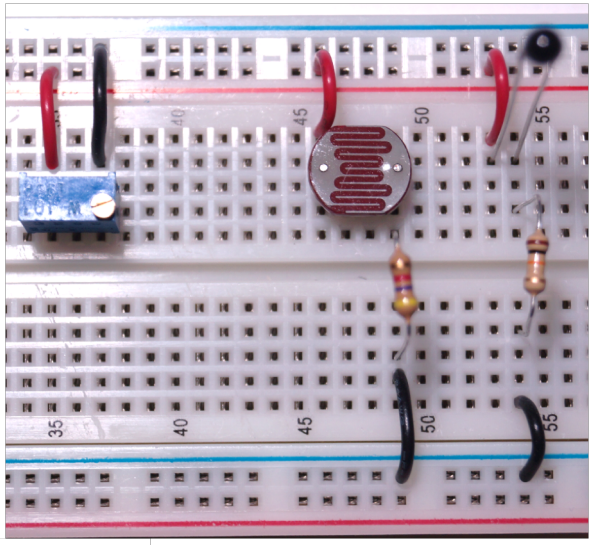


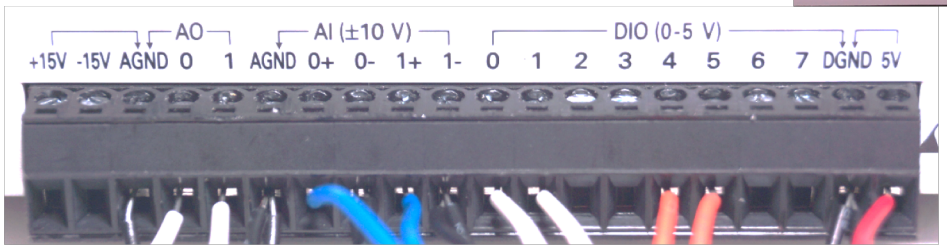




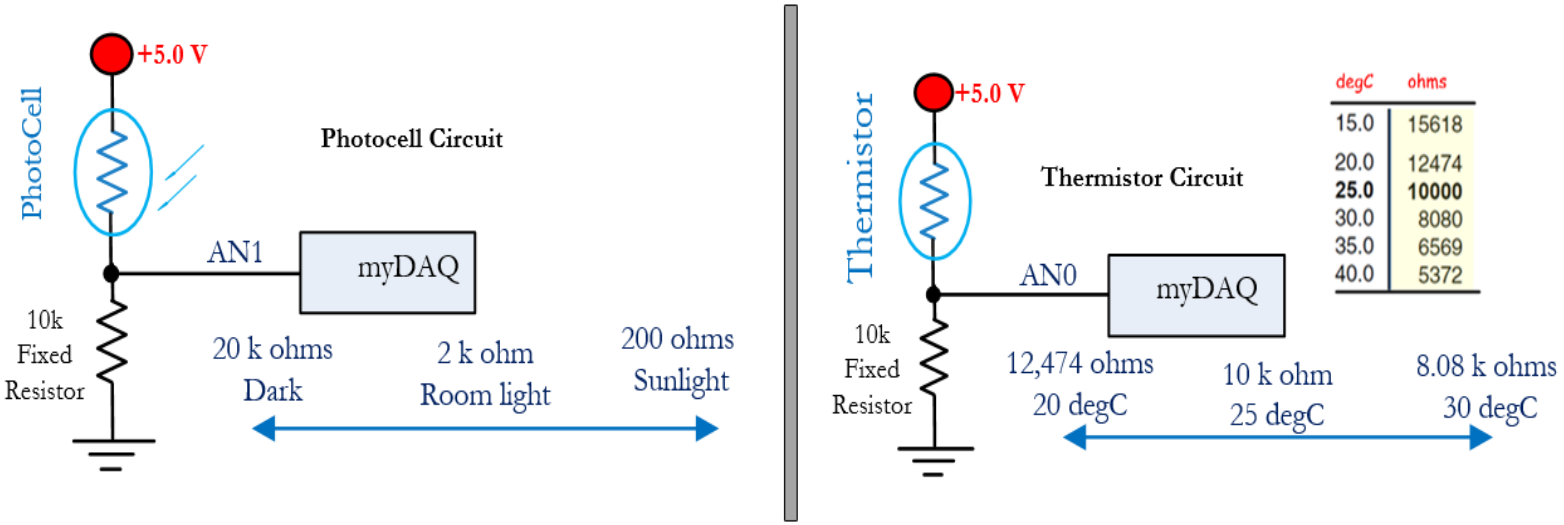
myDAQ Basic I/O Circuits And Sensors



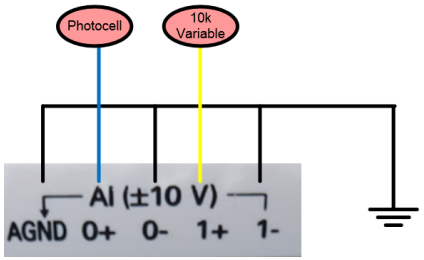




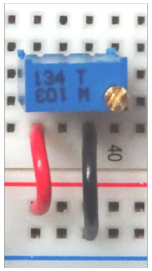
myDAQ Basic I/O Circuits And Sensors



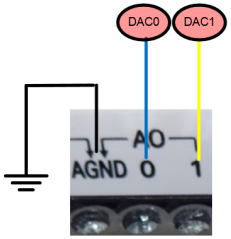
myDAQ Analog I/O Wiring



The AGND, 0- and 1- must connect to the ground of your circuit when measuring signals that are referenced to ground.

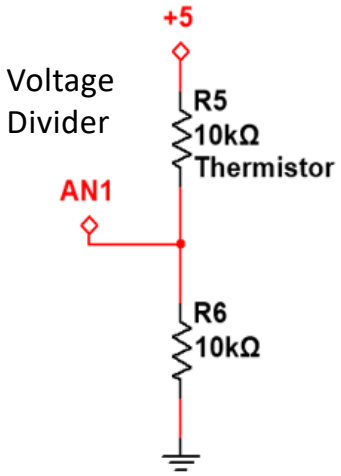


The 10k variable is used to provide a variable DC signal from 0 – 5 volts to the analog inputs. Connect the device so that turning it CW increases the voltage on the middle terminal.



Photocell

A photocell increases resistance in darkness and decreases when exposed to daylight. They typically range from about 200 ohms to 100,000 ohms and are non linear. When the circuit is wired as above the AIN0 voltage decreases in darkness.



A thermistor decreases resistance as it is heated and increases resistance when cooled. The thermistor used in the lab is:

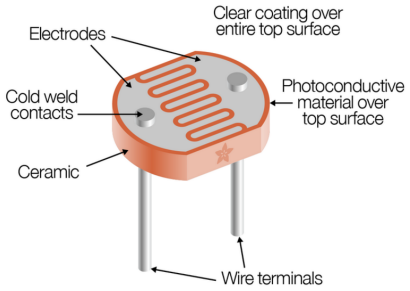
10k at 25 degC.
8k at 30 degC
12.4k at 20 degC.

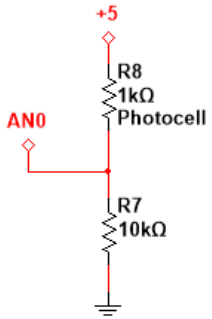
As the temperature increases the voltage to AN1 increases.

myDAQ/Arduino Sensors – Photocell

https://learn.adafruit.com/photocells/measuring-light

Illuminance	Example
0.002 lux	Moonless clear night sky
0.2 lux	Design minimum for emergency lighting (AS2293).
0.27 - 1 lux	Full moon on a clear night
3.4 lux	Dark limit of civil twilight under a clear sky
50 lux	Family living room
80 lux	Hallway/toilet
100 lux	Very dark overcast day
300 - 500 lux	Sunrise or sunset on a clear day. Well-lit office area.
1,000 lux	Overcast day; typical TV studio lighting
10,000 - 25,000 lux	Full daylight (not direct sun)
32,000 - 130,000 lux	Direct sunlight





The photocell resistance decreases with higher visible light and increases in darkness.

Placed in a voltage divider circuit, as shown, the output voltage will increase when exposed to brighter light levels.

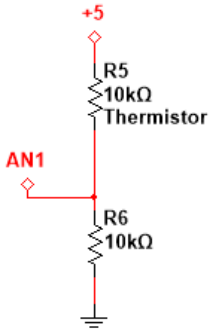
(Wikipedia) Illuminance is a measure of how much luminous flux is spread over a given area. One can think of luminous flux (measured in lumens) as a measure of the total "amount" of visible light present, and the illuminance as a measure of the intensity of illumination on a surface. A given amount of light will illuminate a surface more dimly if it is spread over a larger area, so illuminance (lux) is inversely proportional to area when the luminous flux (lumens) is held constant.

200 k ohms dark room

2.0 k ohms office lighting

200 ohms - daylight

myDAQ/Arduino Sensors – Thermistor



In this voltage divider the voltage increases as the temperature increases. For example at 30 degC the voltage out of the divider equals 2.7 volt. At 20 degC the voltage out is about 2.2 volts. The value in the Arduino register will range from 0 to 1023 (10 bits) At 20 degC the register equals 454.

https://learn.adafruit.com/thermistor/using-a-thermistor

You can also determine resistance given the A/D register value.

For example the register value (AN1) equals 566.

AN1 = 10k / (10k + Rtherm) * 1023

or

Rtherm = ((10k * 1023)/AN1) – 10k

Answer = 8080 ohms. (about 30 degC)

	B57891S0103F008 Thermistor Data
R/T No.	4901
T (°C)	$B_{25/100}$ = 3950 K, R_{25} = 10000 Ω, T_R = 25 °
	R_{nom}[Ω]
0.0	32014
5.0	25011
10.0	19691
15.0	15618
20.0	12474
25.0	**10000**
30.0	8080
35.0	6569
40.0	5372
45.0	4424
50.0	3661
55.0	3039
60.0	2536
65.0	2128
70.0	1794
75.0	1518
80.0	1290
85.0	1100
90.0	941.8
95.0	809.0
100.0	697.2

Data for the thermistor used in the lab.

Resistance decreases as the temperature increases.

The change is not linear.

The change in ohms/deg at low temperature is greater than at high temperatures.

At 25 degC the resistance = 10k

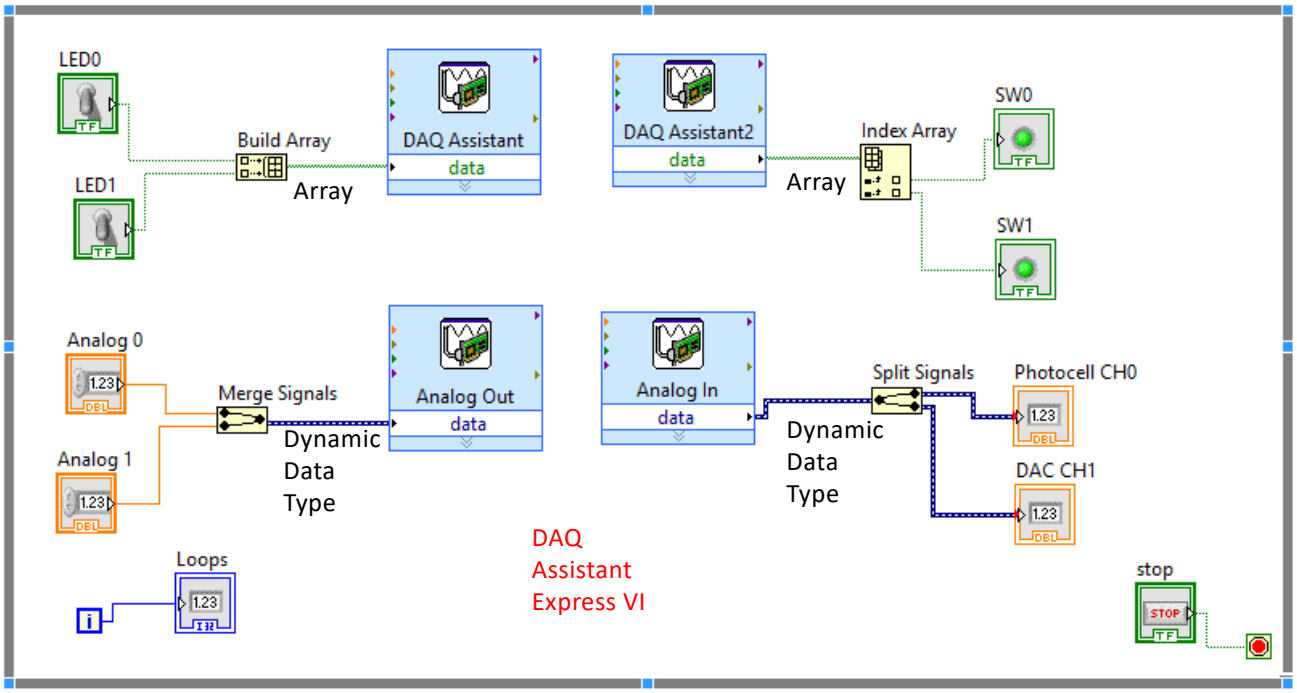
LabVIEW block diagram used to interface to the myDAQ.

The DAQ Assistant is a configurable function that is used to allow signals from the myDAQ interface to be controlled or monitored. There are 4 possibilities.

1) Digital In to read the digital signals. 2) Digital Out to control the logic levels on an output pin.

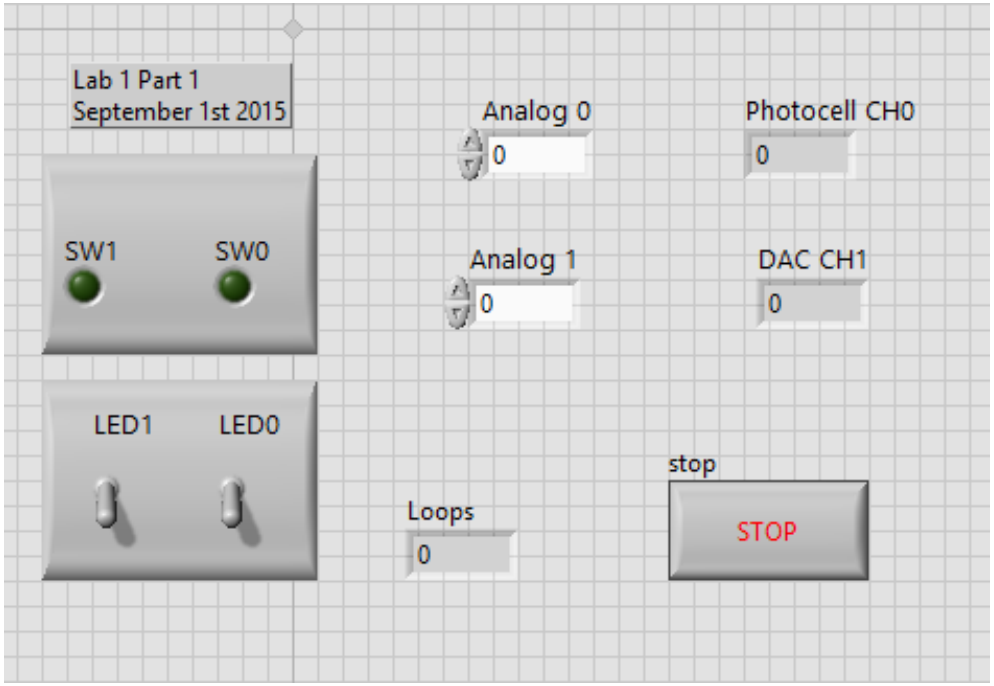
3) Analog In to read the voltage on an analog input pin.

4) Analog out to control the voltage on a pin.

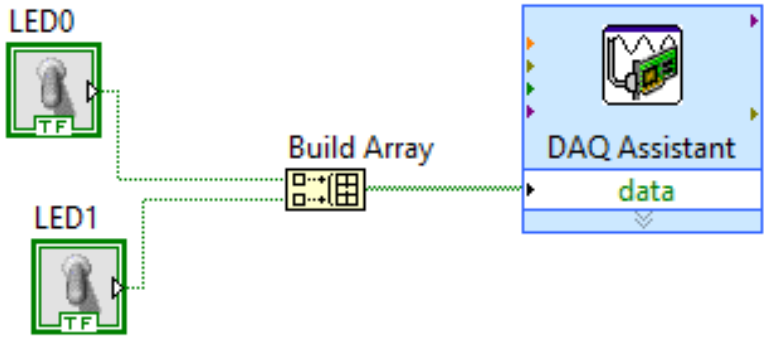


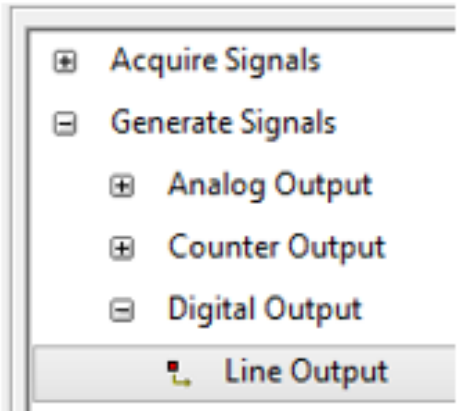
LabVIEW Front Panel (User Interface)

The front panel acts as the user interface. Controls and indicators are placed on the front panel. The controls allow the user to change a Boolean or numeric value. The indicators display the results.

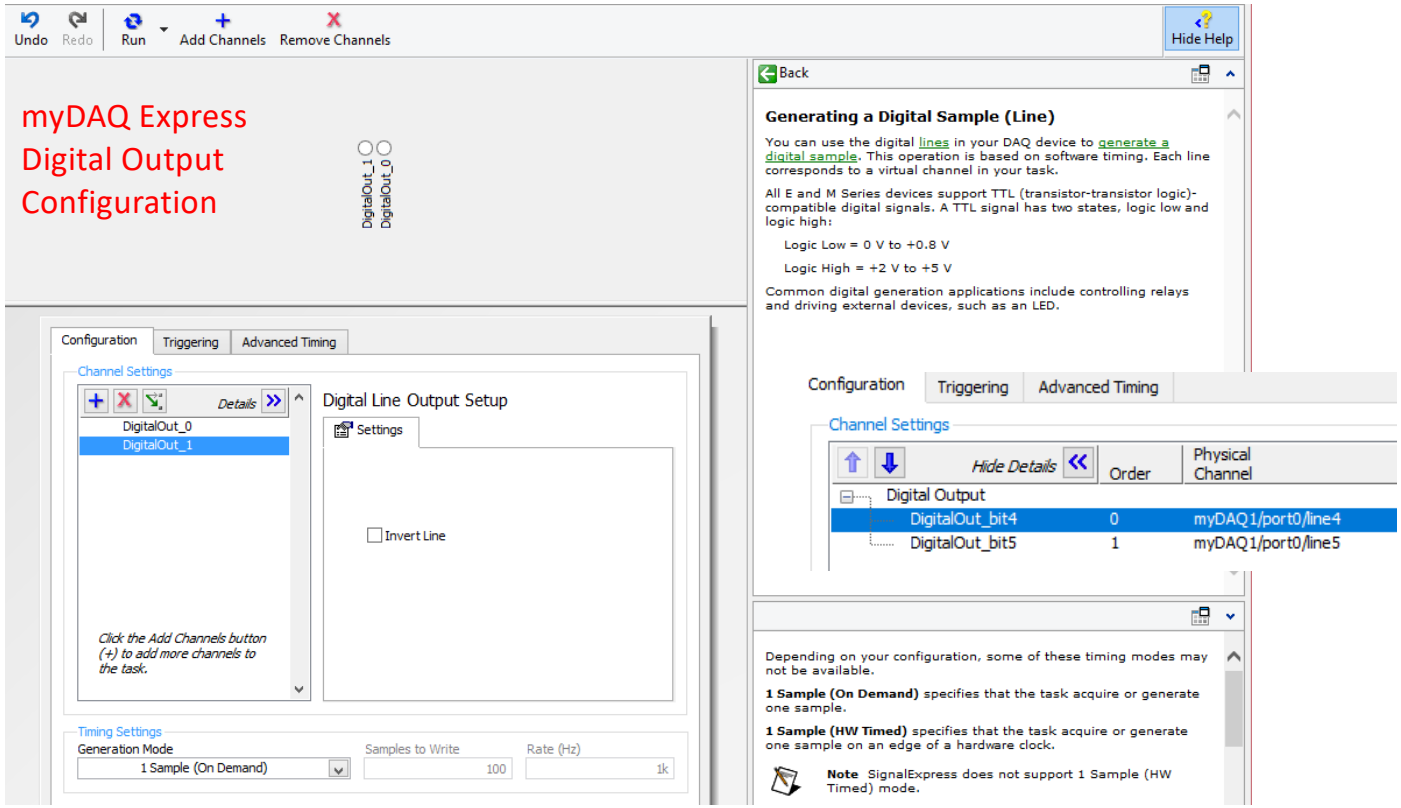


myDAQ Express Digital Output (generate)



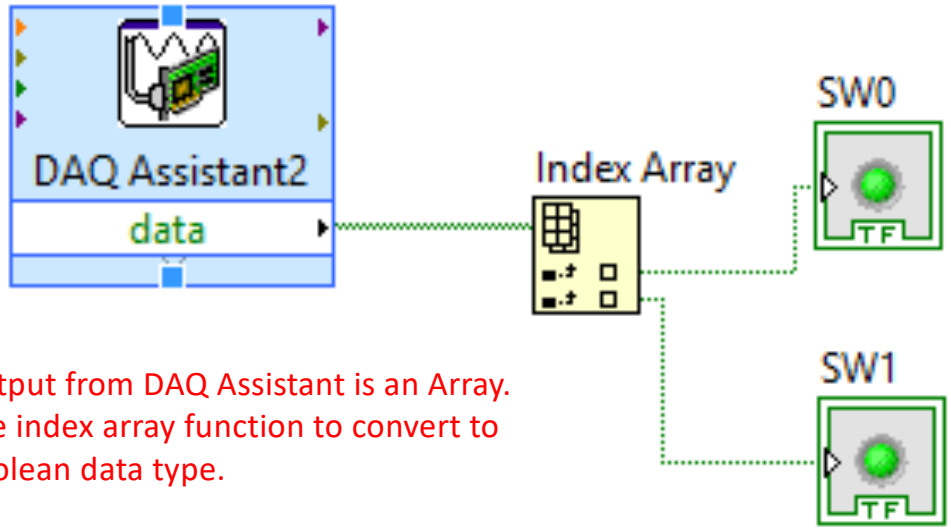


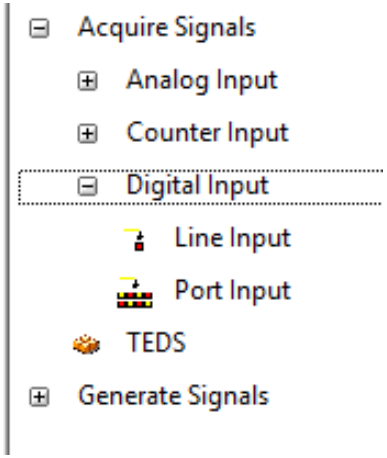
The following is the VI (virtual instrument) or block diagram used to control a digital output on the myDAQ interface. The input to the Express VI is a Boolean array. The build array function combines the array elements into the array. The upper element is element 0.



The express VI, is used to select the channel (pin) to which the signal passes through. The signal can also be tested at this point.

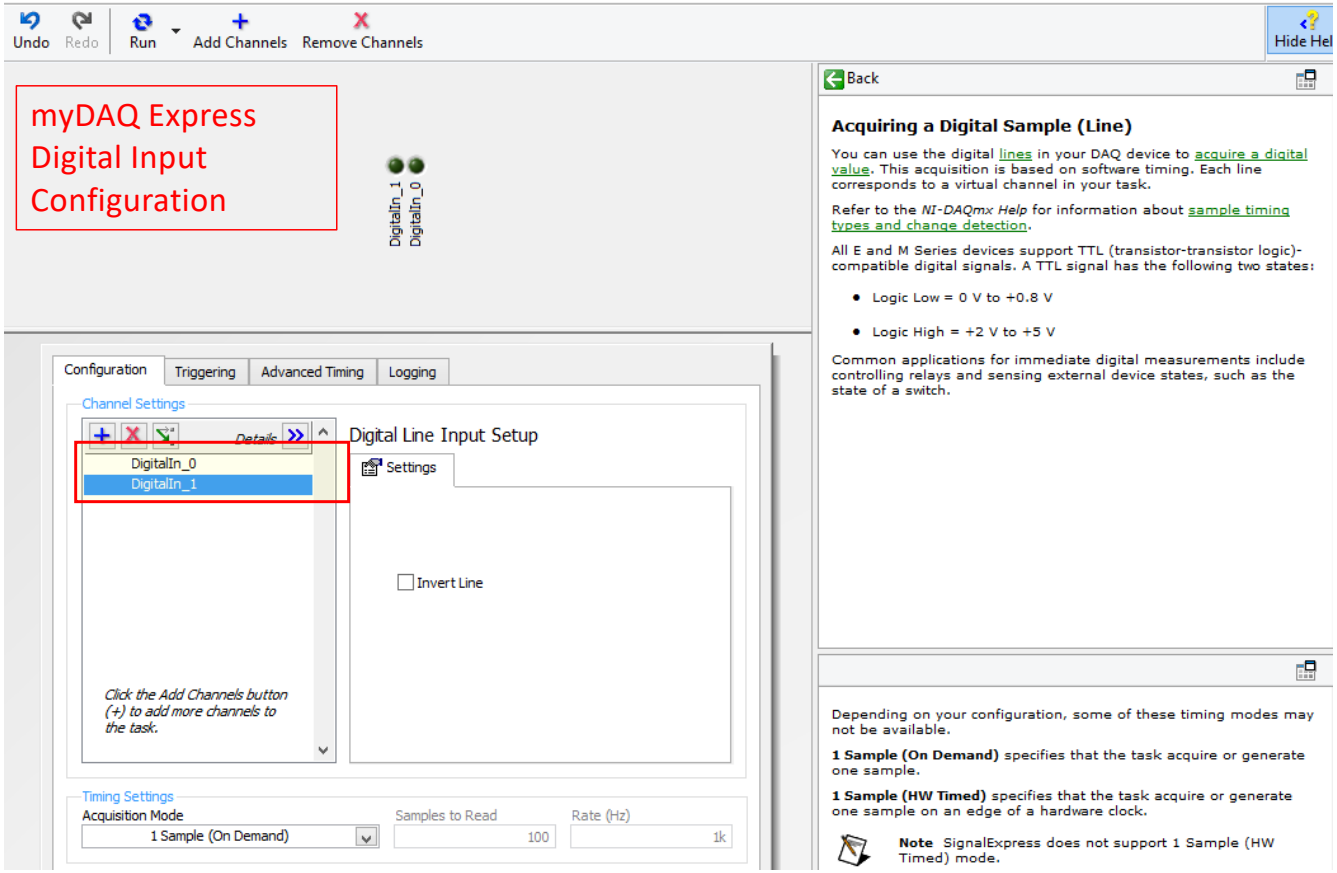
myDAQ Express Digital Input (acquire)



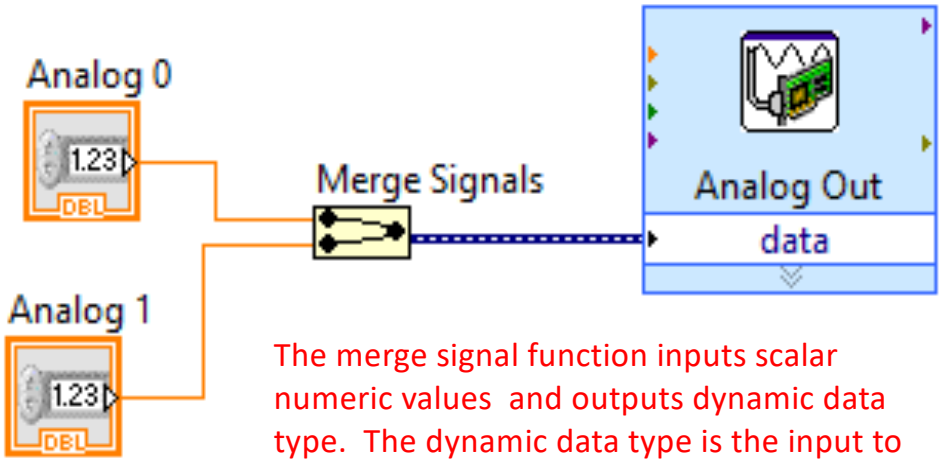


Output from DAQ Assistant is an Array. Use index array function to convert to Boolean data type.

The following is the VI (virtual instrument) or block diagram used to display the data from digital inputs of the myDAQ interface. The DAQ assistant outputs the data as an array. The index array breaks down the data to individual bits. The top element is bit 0 of the array.

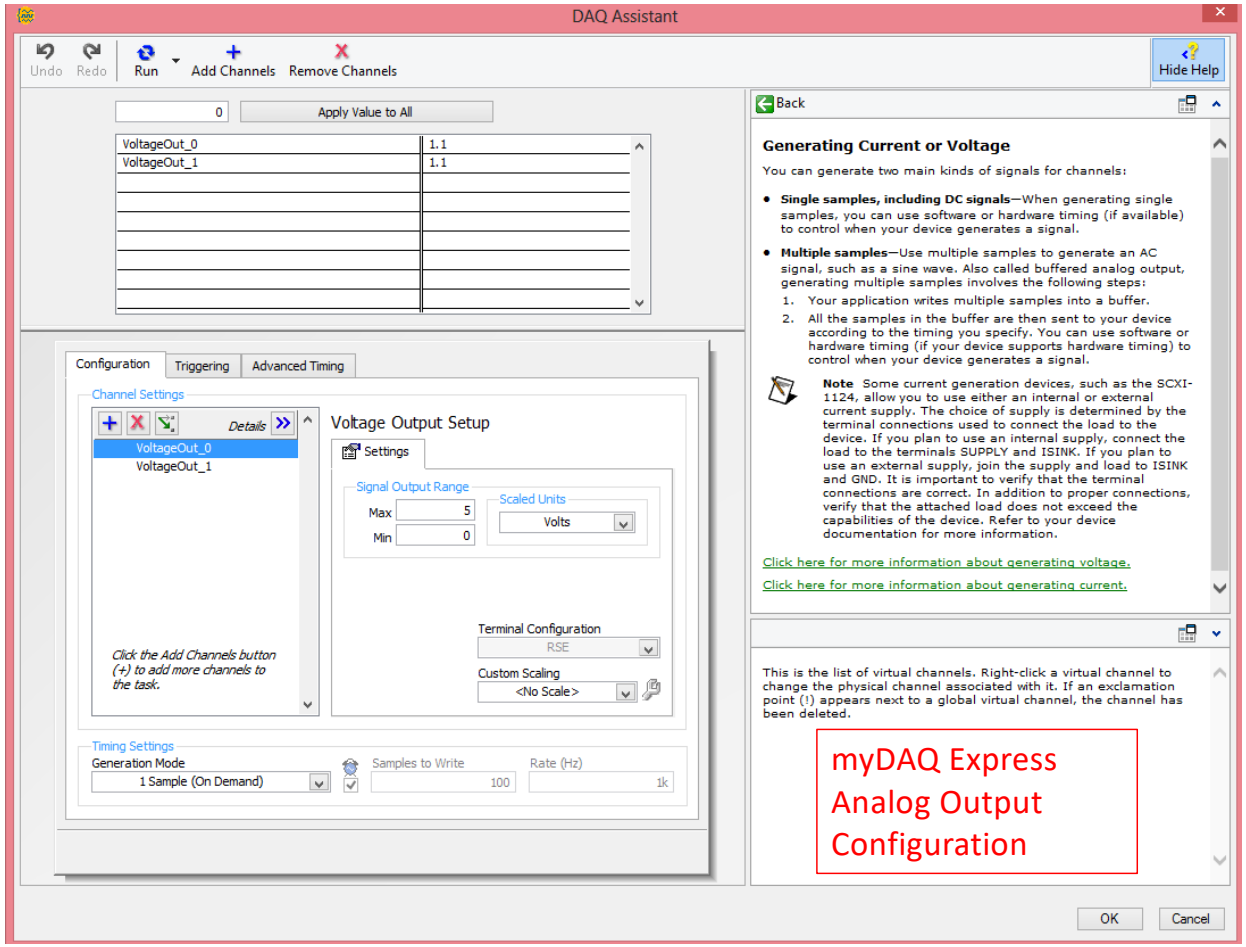

Undo
Redo
Run
Add Channels
Remove Channels
Hide Hel
myDAQ Express
Digital Input
Configuration
DigitalIn_1
DigitalIn_0
Back
Acquiring a Digital Sample (Line)
You can use the digital lines in your DAQ device to acquire a digital value. This acquisition is based on software timing. Each line corresponds to a virtual channel in your task.
Refer to the NI-DAQmx Help for information about sample timing types and change detection.
All E and M Series devices support TTL (transistor-transistor logic)-compatible digital signals. A TTL signal has the following two states:
Logic Low = 0 V to +0.8 V
Logic High = +2 V to +5 V
Common applications for immediate digital measurements include controlling relays and sensing external device states, such as the state of a switch.
Configuration
Triggering
Advanced Timing
Logging
Channel Settings
Details
DigitalIn_0
DigitalIn_1
Digital Line Input Setup
Settings
Invert Line
Click the Add Channels button (+) to add more channels to the task.
Timing Settings
Acquisition Mode
1 Sample (On Demand)
Samples to Read
100
Rate (Hz)
1k
Depending on your configuration, some of these timing modes may not be available.
1 Sample (On Demand) specifies that the task acquire or generate one sample.
1 Sample (HW Timed) specifies that the task acquire or generate one sample on an edge of a hardware clock.
Note SignalExpress does not support 1 Sample (HW Timed) mode.

myDAQ Express Analog Output

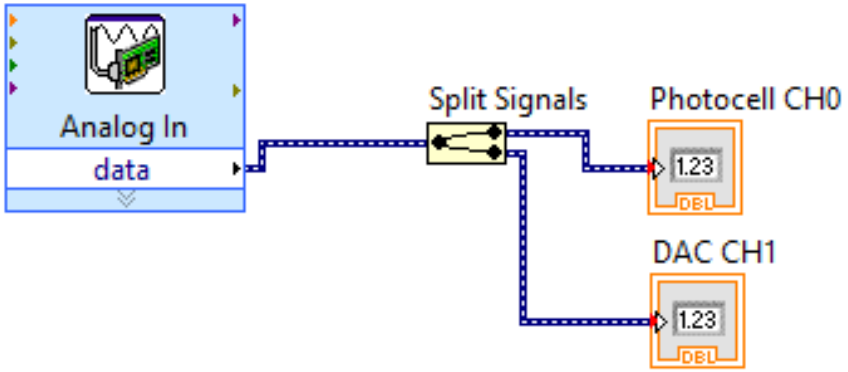


The merge signal function inputs scalar numeric values and outputs dynamic data type. The dynamic data type is the input to the express VI.

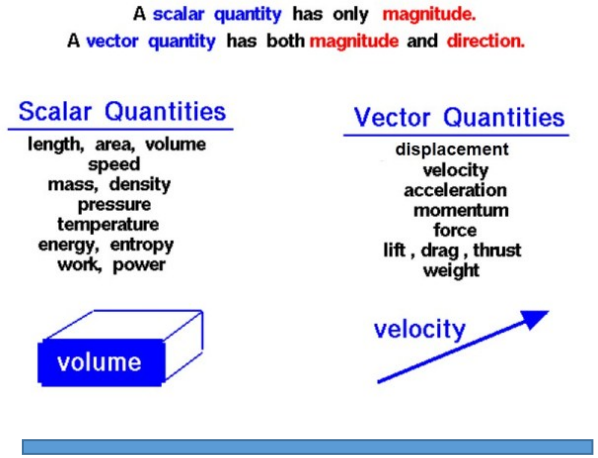
The following is the VI (virtual instrument) or block diagram used to control the analog voltage to the myDAQ analog out terminals. The data into and out from the DAQ assistant for analog data is a Dynamic data type. The merge signals function can combine many scalar data values into one dynamic data type signal. The express VI reads the signals in sequence. The upper value will be applied to the first output signal.


DAQ Assistant
Undo
Redo
Run
Add Channels
Remove Channels
Hide Help
0
Apply Value to All
VoltageOut_0
VoltageOut_1
1.1
1.1
Configuration
Triggering
Advanced Timing
Channel Settings
Details
VoltageOut_0
VoltageOut_1
Click the Add Channels button (+) to add more channels to the task.
Voltage Output Setup
Settings
Signal Output Range
Max
5
Min
0
Scaled Units
Volts
Terminal Configuration
RSE
Custom Scaling
<No Scale>
Timing Settings
Generation Mode
1 Sample (On Demand)
Samples to Write
100
Rate (Hz)
1k
Back
Generating Current or Voltage
You can generate two main kinds of signals for channels:
Single samples, including DC signals—When generating single samples, you can use software or hardware timing (if available) to control when your device generates a signal.
Multiple samples—Use multiple samples to generate an AC signal, such as a sine wave. Also called buffered analog output, generating multiple samples involves the following steps:
1. Your application writes multiple samples into a buffer.
2. All the samples in the buffer are then sent to your device according to the timing you specify. You can use software or hardware timing (if your device supports hardware timing) to control when your device generates a signal.
Note Some current generation devices, such as the SCXI-1124, allow you to use either an internal or external current supply. The choice of supply is determined by the terminal connections used to connect the load to the device. If you plan to use an internal supply, connect the load to the terminals SUPPLY and ISINK. If you plan to use an external supply, join the supply and load to ISINK and GND. It is important to verify that the terminal connections are correct. In addition to proper connections, verify that the attached load does not exceed the capabilities of the device. Refer to your device documentation for more information.
Click here for more information about generating voltage.
Click here for more information about generating current.
This is the list of virtual channels. Right-click a virtual channel to change the physical channel associated with it. If an exclamation point (!) appears next to a global virtual channel, the channel has been deleted.
myDAQ Express Analog Output Configuration
OK
Cancel

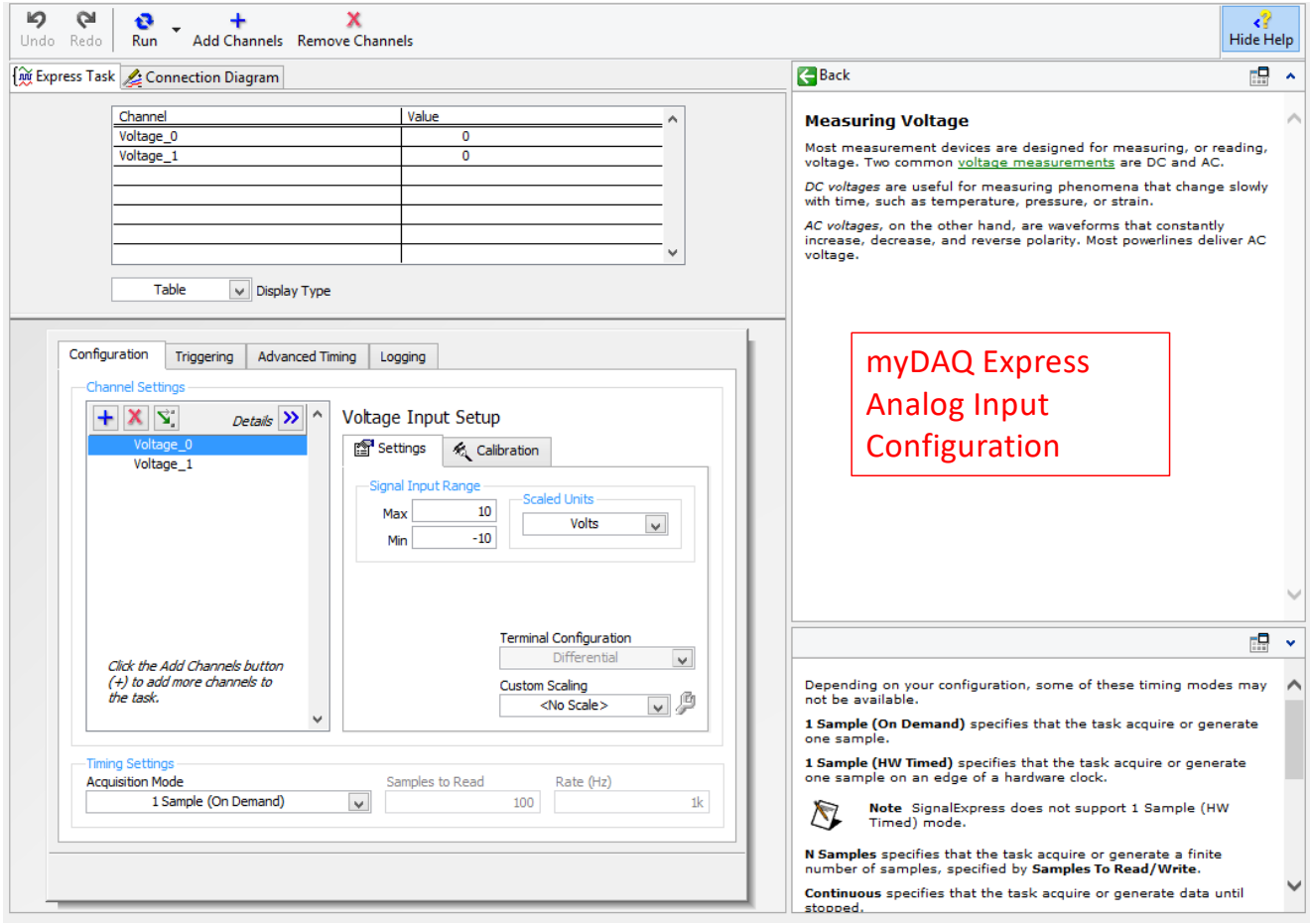
myDAQ Express Analog Input



The following is the VI (virtual instrument) or block diagram used to monitor the analog voltage to the myDAQ analog input terminals. The data into and out from the DAQ assistant for analog data is a Dynamic data type. The split signals function can breakdown multiple signals to individual scalar data types. The express VI outputs the signals in sequence. The upper value is from the first input(CH0) the lower value from (CH1).



The dynamic data type includes the data associated with a signal, as well as attributes that provide information about that signal, such as the name of the signal or the date and time the data was acquired.


Undo
Redo
Run
Add Channels
Remove Channels
Hide Help
Express Task
Connection Diagram
Channel
Value
Voltage_0
0
Voltage_1
0
Table
Display Type
Back
Measuring Voltage
Most measurement devices are designed for measuring, or reading, voltage. Two common voltage measurements are DC and AC.
DC voltages are useful for measuring phenomena that change slowly with time, such as temperature, pressure, or strain.
AC voltages, on the other hand, are waveforms that constantly increase, decrease, and reverse polarity. Most powerlines deliver AC voltage.
myDAQ Express Analog Input Configuration
Configuration
Triggering
Advanced Timing
Logging
Channel Settings
Details
Voltage_0
Voltage_1
Click the Add Channels button (+) to add more channels to the task.
Voltage Input Setup
Settings
Calibration
Signal Input Range
Max
10
Min
-10
Scaled Units
Volts
Terminal Configuration
Differential
Custom Scaling
<No Scale>
Timing Settings
Acquisition Mode
1 Sample (On Demand)
Samples to Read
100
Rate (Hz)
1k
Depending on your configuration, some of these timing modes may not be available.
1 Sample (On Demand) specifies that the task acquire or generate one sample.
1 Sample (HW Timed) specifies that the task acquire or generate one sample on an edge of a hardware clock.
Note SignalExpress does not support 1 Sample (HW Timed) mode.
N Samples specifies that the task acquire or generate a finite number of samples, specified by Samples To Read/Write.
Continuous specifies that the task acquire or generate data until stopped.

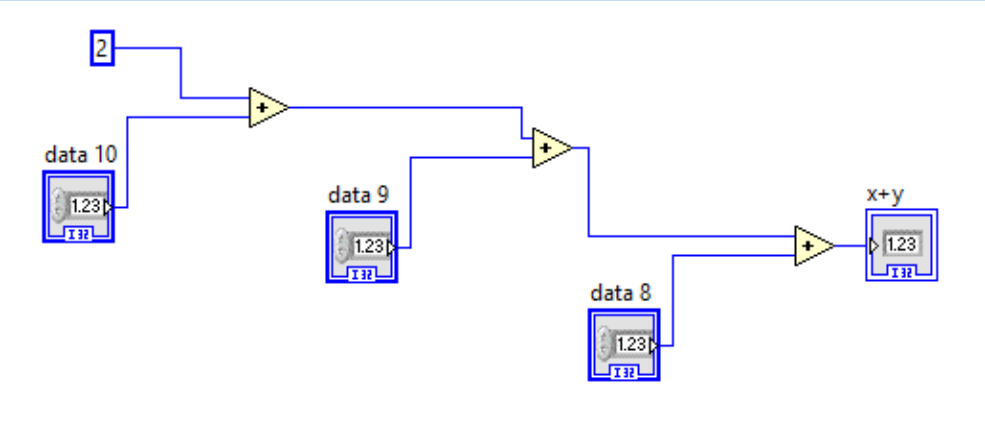
LabVIEW Block Diagrams and Front Panel Indicators and Controls

LabVIEW

- A graphical programming language used for instruments and control applications.
- LAB Virtual Instrument Electronic Workbench.
- LabVIEW easily allows developers to create graphical interfaces that control and monitor hardware devices and sensors.

LabVIEW Data Flow Execution

- LabVIEW uses data flow execution, when all inputs to a function are available, the function executes, and produces an output.



LabVIEW Data Flow Programming

LabVIEW follows a dataflow model for running VIs. A block diagram node executes when it receives all required inputs. When a node executes, it produces output data and passes the data to the next node in the dataflow path. The movement of data through the nodes determines the execution order of the VIs and functions on the block diagram.

Visual Basic, C++, JAVA, and most other text-based programming languages follow a control flow model of program execution. In control flow, the sequential order of program elements determines the execution order of a program.

For a dataflow programming example, consider a block diagram that adds two numbers and then subtracts 50.00 from the result of the addition, as shown in **Figure 1**. In this case, the block diagram executes from left to right, not because the objects are placed in that order, but because the Subtract function cannot execute until the Add function finishes executing and passes the data to the Subtract function. Remember that a node executes only when data are available at all of its input terminals and supplies data to the output terminals only when the node finishes execution.

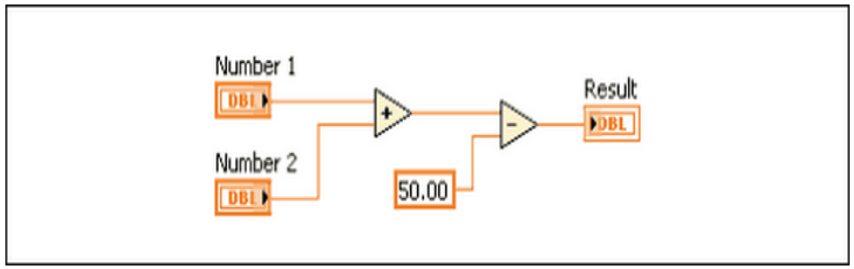
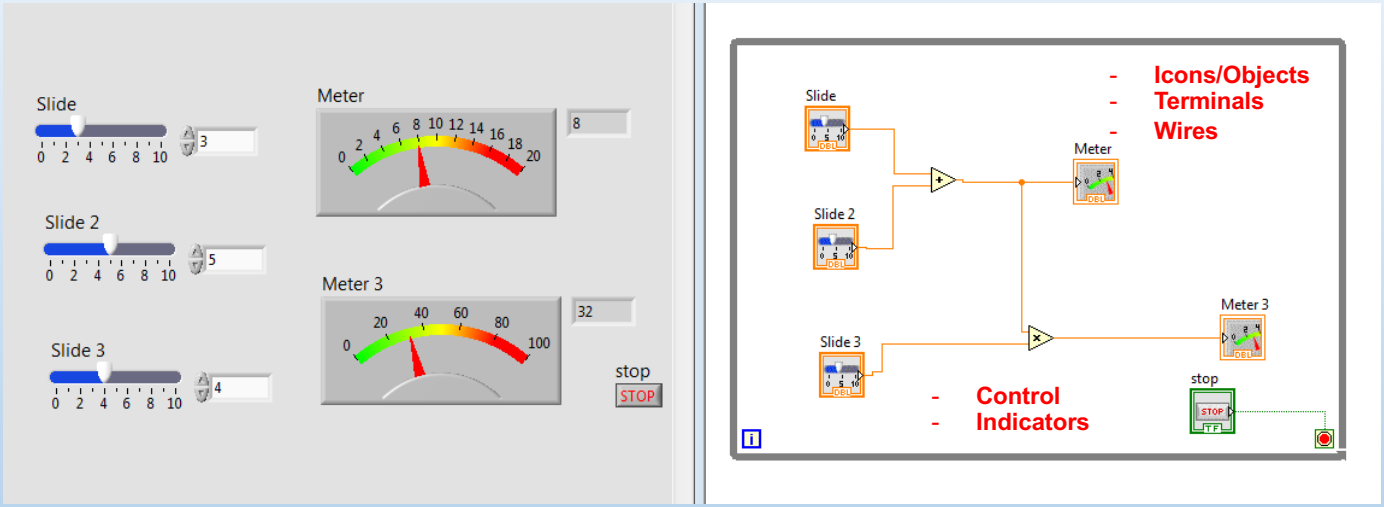


Figure 1. Dataflow Programming Example

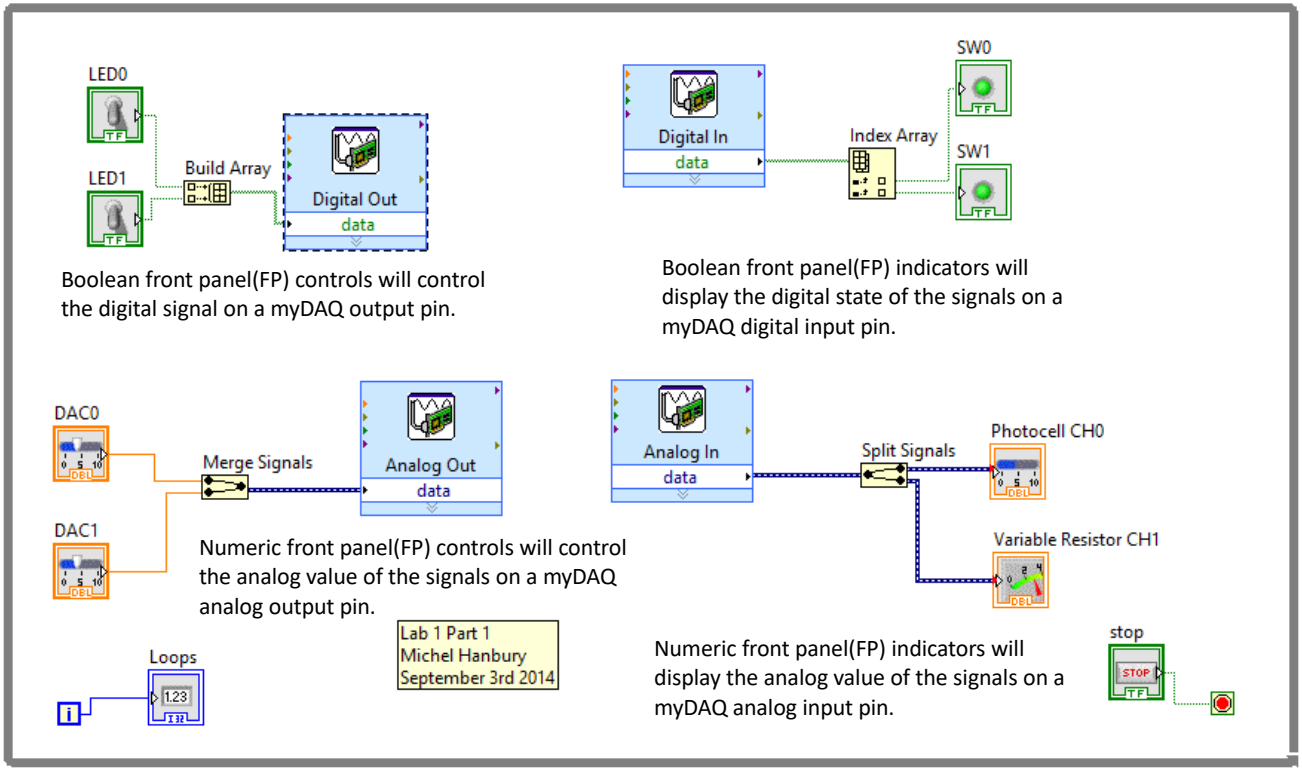
LabVIEW Front Panel & Block Diagram



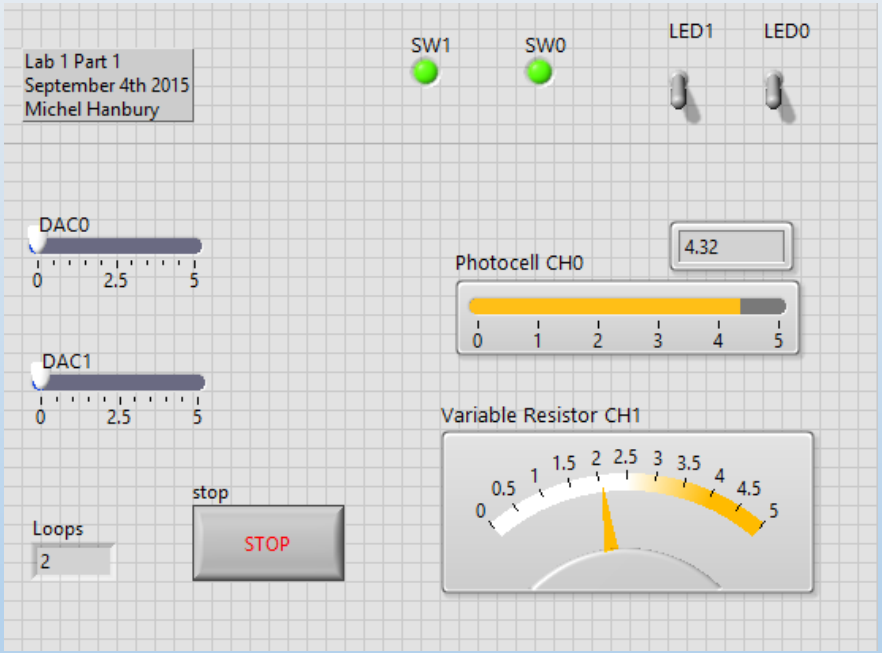
Front Panel **Block Diagram**

Place Graphical Objects on the Front Page – objects are automatically generated on the block diagram.

LabVIEW Block Diagram (Program)

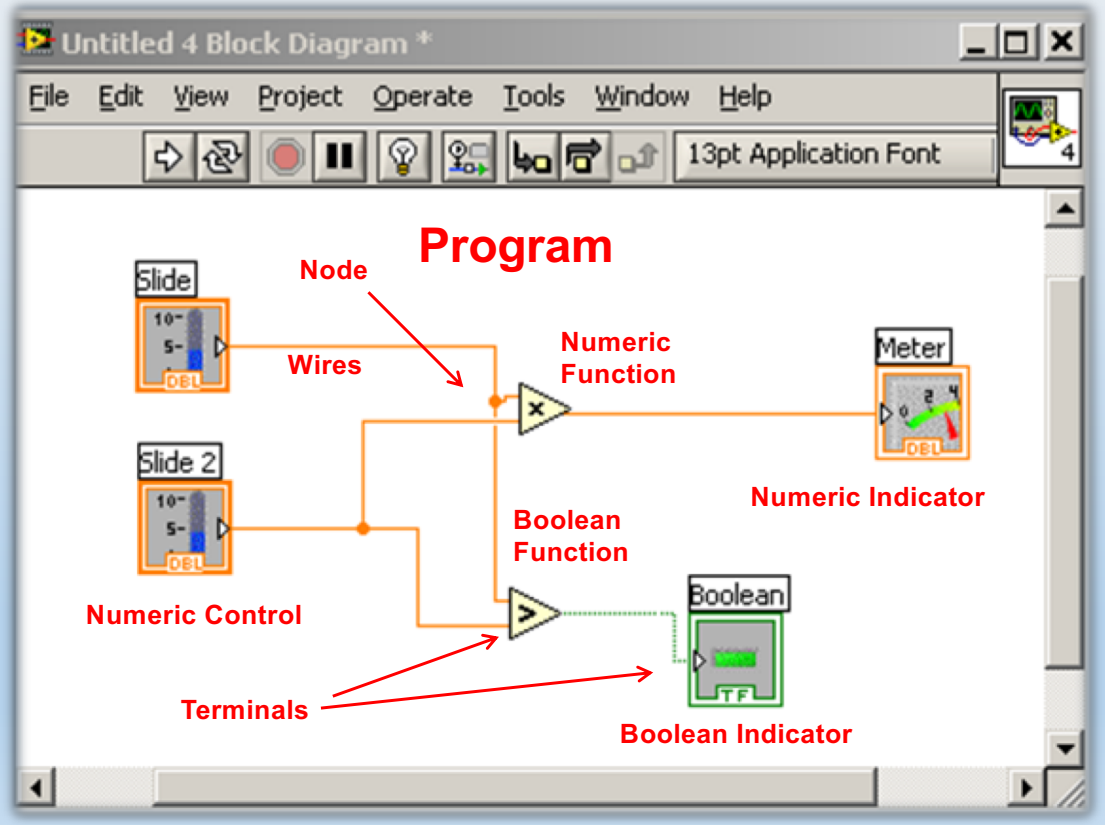


LabVIEW Front Panel (User Interface)

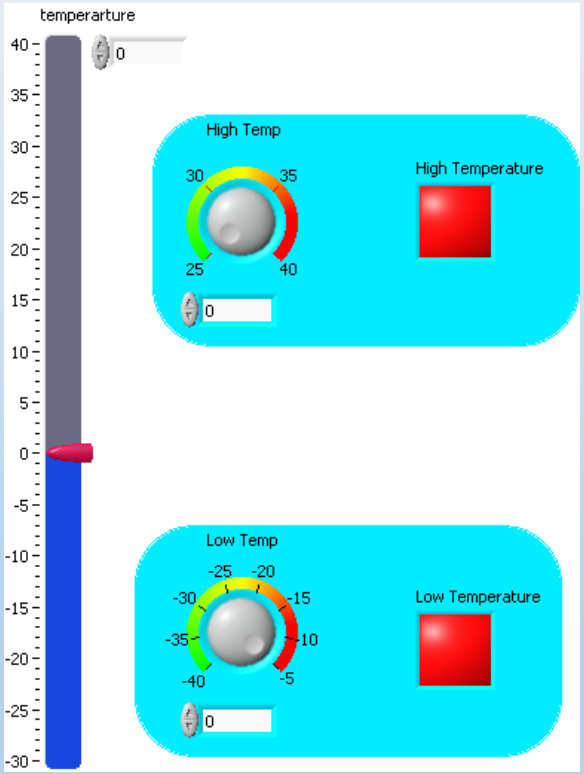


This is the front panel used to monitor and control digital and analog signals from and to the myDAQ interface.

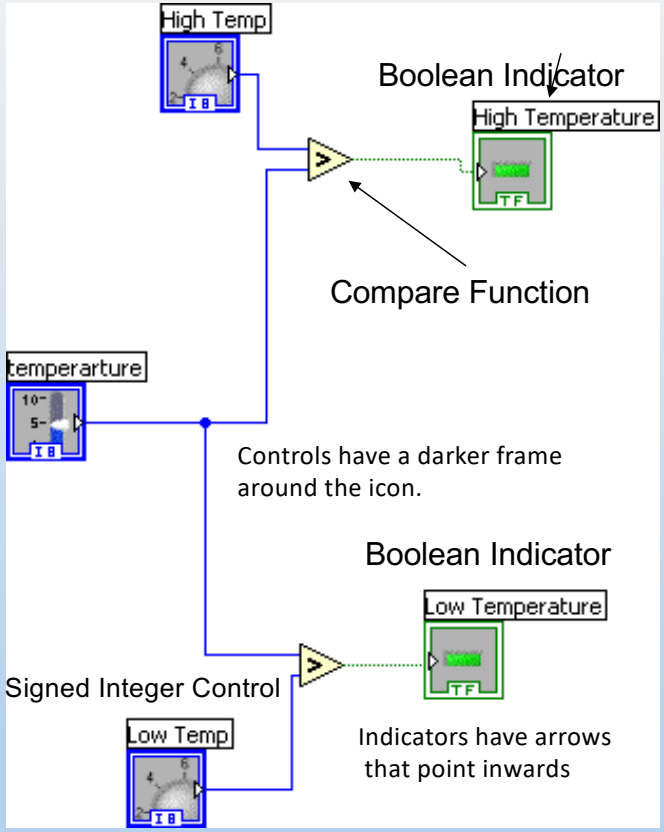
LabVIEW Block Diagram (program)



Properties such as scale, colours and representation can be modified with a right-click of the mouse.



Front Panel



Block Diagram

LabVIEW Menu Bar –

(Running and Stopping a Program)

- Run Once
- Run Continuously
- Stop
- Pause
- Step (step through program)

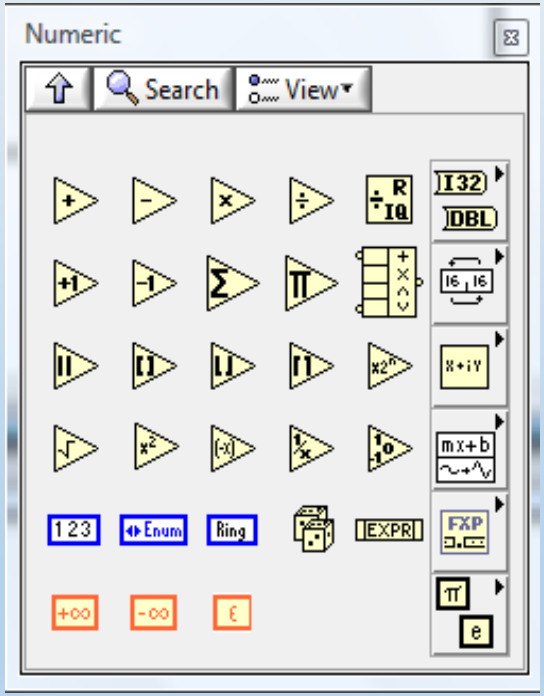
Note:

It is quite common to use a while loop in all programs. The while loop executes until a condition is met.

LabVIEW Front Panel Controls Palette

- **Controls**
  - Dials
  - Tanks
  - Thermometers
  - Switches
- **Indicators**
  - Leds
  - Gauges

LabVIEW Numeric Functions Palette

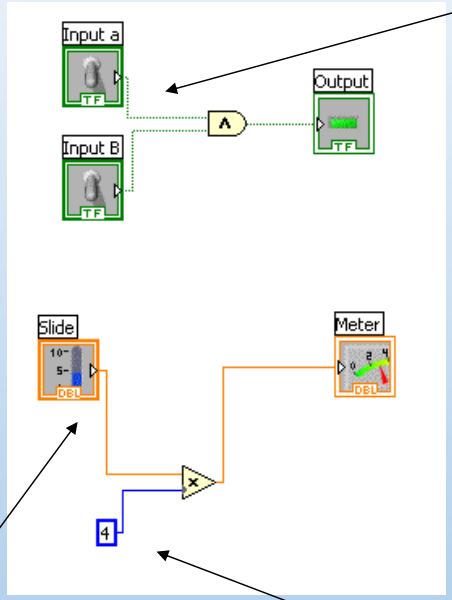


- Block Diagram Functions:
  - Add
  - Subtract
  - Multiply
  - Sum
  - Square Root

Functions that require numeric inputs and produce numeric outputs.

LabVIEW Numeric Functions Palette

Floating point numbers can be Double Precision (8 bytes) or single precision (4 bytes)



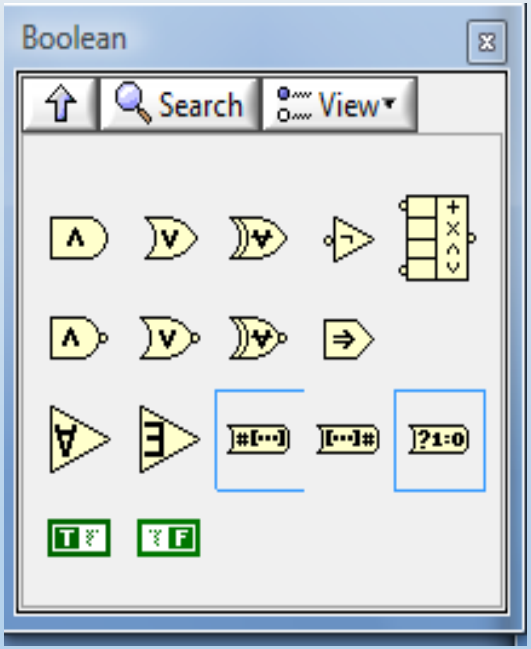
Boolean variables have only one of two possibilities zero or one or (true / false), (on/off).

Double Precision numbers are floating point numbers (Ex. 2.3456), they can be used to represent fractional amounts.

LabVIEW uses the colour orange to indicate floating-point numbers.

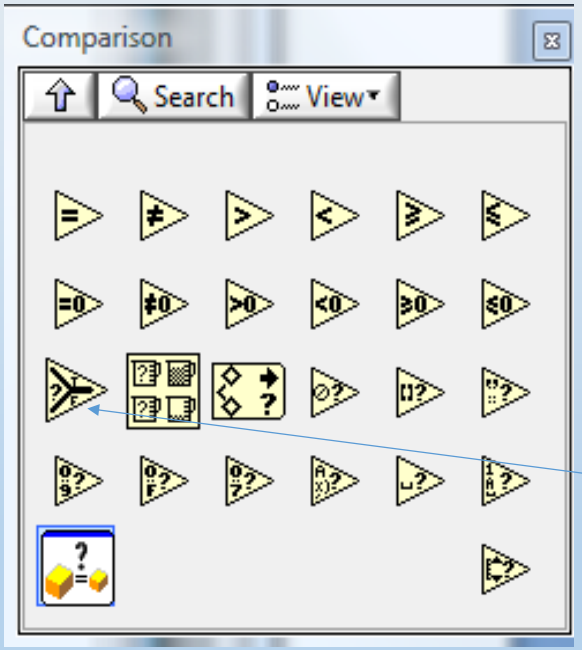
The variable shown in blue is an integer variable. Integers can be signed or un-signed. Signed numbers use the most significant bit to indicate sign. Integers can be 8,16,32 and 64 bits.

LabVIEW Boolean Functions Palette



- Block Diagram Functions:
  - Logical And
  - Logical OR
  - Logical Invert
  - The two green T/F boxes near the bottom are Boolean constants.
  - The function with the question mark is used to convert from Boolean to numeric.

LabVIEW Comparison Functions Palette

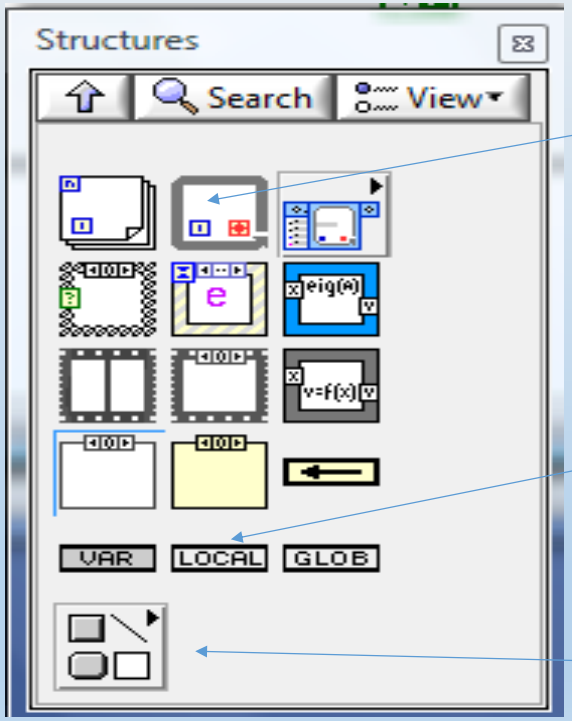


- Block Diagram Functions:

- Comparison Functions
  - Greater than
  - Less than
  - Not equal to zero
  - Select
  - Min and Max

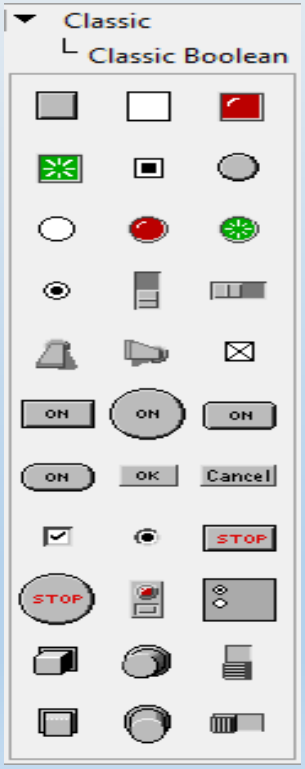
  - The Select function requires a Boolean input and outputs one of two values based on the condition of the Boolean input.

LabVIEW Structures Palette:



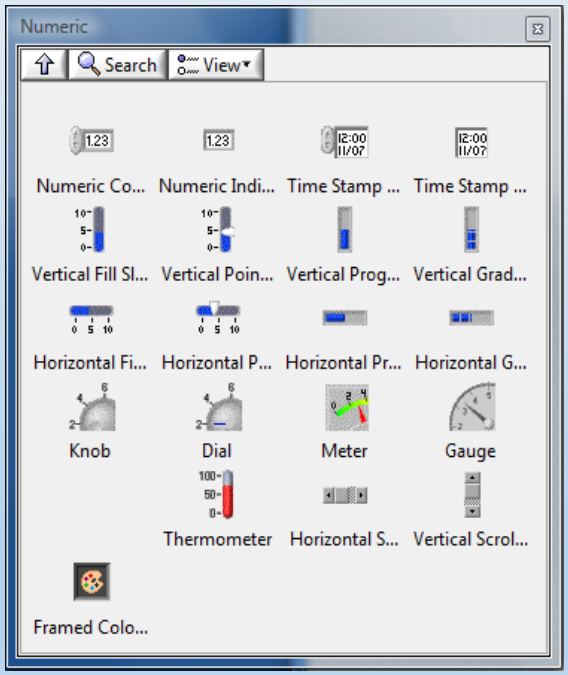
- Structures:
  - For Loop Structure
    - Executes the loop a specific number of times.
  - While Loop Structure
    - Continues until a condition is true.
  - Case Statement
    - Executes a statement bases on the input value of the case statement.
  - Local Variables
    - Use local variable instead of wires to connect one point to another.
  - Sequence Structure
    - Used to execute code sequence.
  - Decorations
    - Used to draw boxes, test and graphics.

LabVIEW Front Panel Boolean Controls and Indicators



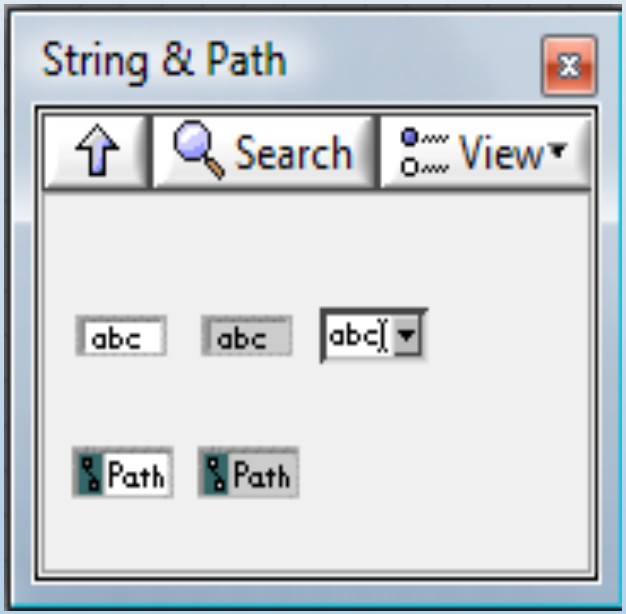
- Switches
- Push Buttons
- LEDs
- True/False
- Check Boxes

LabVIEW Front Panel Numeric Controls and Indicators

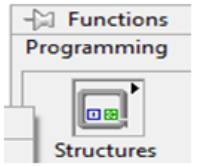


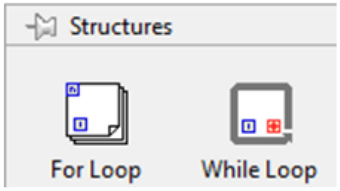
- Numeric Controls
- Numeric Indicators
- Slides
- Knobs
- Meters
- Gauges

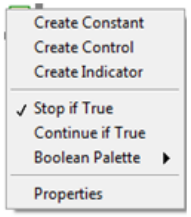
LabVIEW Front Panel Strings:



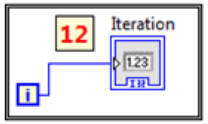
- String / Characters /Text
  - Controls
  - Indicators
  - Allows text to be entered and displayed on the Front Panel.

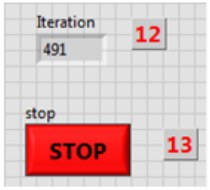




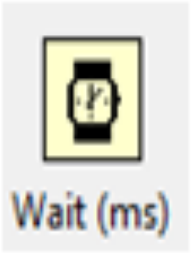


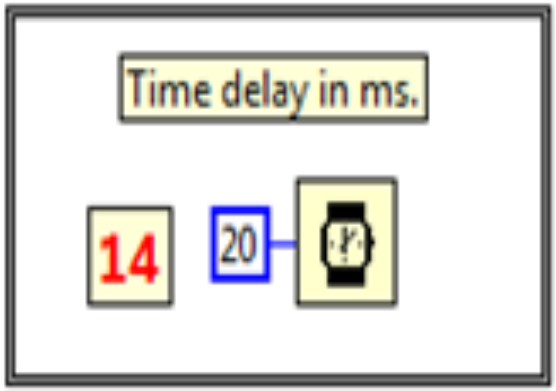
To repeat code – use a While Loop. The code will continue to execute until stop is true. The loop also has an iteration counter that increments by one for each loop count.





LabVIEW Structures – For Loops and While Loops.

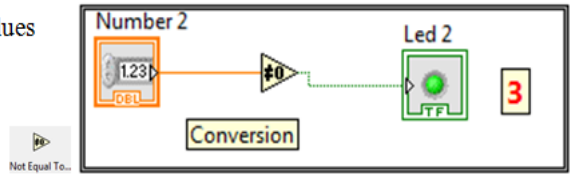




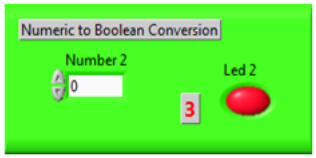
Programs often require time delays. LabVIEW has a function to create a delay in increments of milliseconds.

LabVIEW Functions – Time Delay

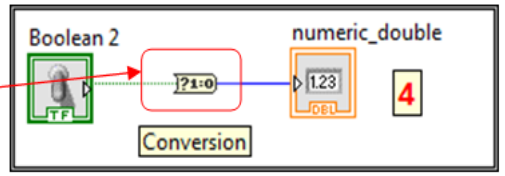
17. At times you will want to convert Numeric values to Boolean. The Not equal to 0 icon will output a true when the number is not equal to zero. Found in the comparisons palette.

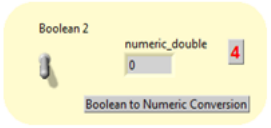


18. Change the colour of an led (or other objects) by right clicking on them and changing the Colors properties.



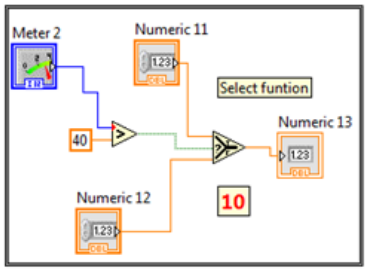
19. At other time it may be required to convert Boolean to numeric. This can be done with the following function found in the Boolean function palette.





LabVIEW Select Function

27. The select function allows you to send one of two data values (Boolean, Text, or Numeric) through the select function dependent on the value of the Boolean input. In this example the value of Meter 2 must be greater than 40 for Numeric 11 to pass through otherwise Numeric value 12 is passed through. The top input is passed to the output when the Boolean input is true.



Select Function

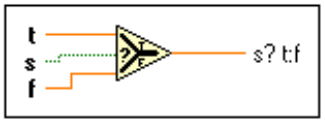
Owning Palette: Comparison Functions

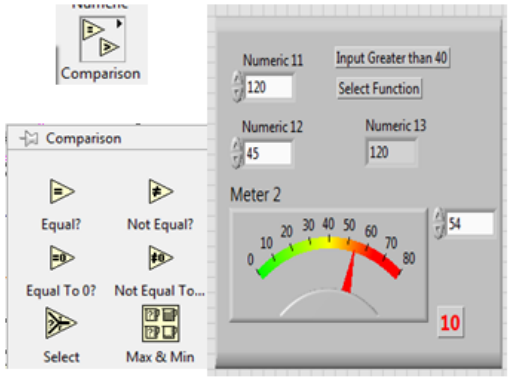
Requires: Base Package

Returns the value wired to the **t** input or **f** input, d

The connector pane displays the default data type:

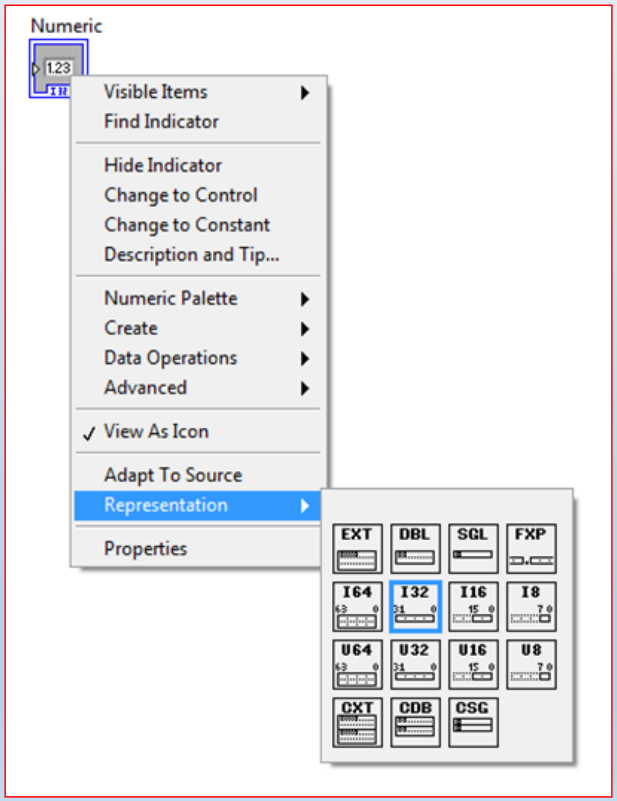
Example





LabVIEW Data Types

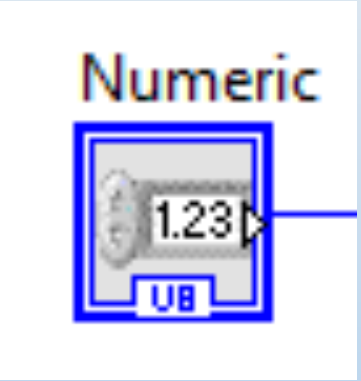
LabVIEW Data Types



- LabVIEW contains various numeric types.
- Numeric types can be changed with a right-click of the mouse and then select representation.

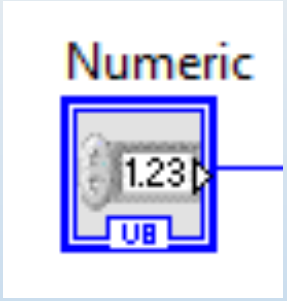
- I – Signed Integer
- U – Unsigned Integer
- SGL – single precision real data

LabVIEW Variable Types



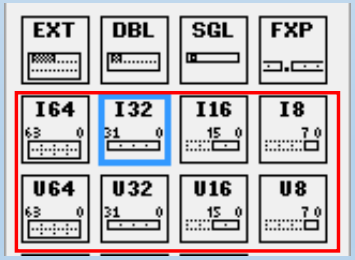
- Controls or indicators in blue represent integer data type.
- Integers can be 8,16,32 and 64 bit values.
- Integers may also be signed (negative and positive) or unsigned (positive only).

LabVIEW Integer Data Type



- Eight bit unsigned integers (U8) range from 0 to 255 in decimal.
- Signed eight bit integers (I8) range from -128 to positive 127.

Bit 7 of the 8 bit number is called the sign bit. (a 1 in bit 7 indicates a negative value)



Integers do not have a fractional part in the number. Besides 8 bit signed and un-signed there are also 16,32 and 64 bit signed and unsigned integers.

Integers are used to represent: - a count, a value in memory, an address location or any whole number.

Single/Double Precision

- Single precision numbers require less storage space. Single precision numbers can store data to 6.000000 decimal places. This is more than enough for what is typically done in the lab.

- LabVIEW uses double precision by default.

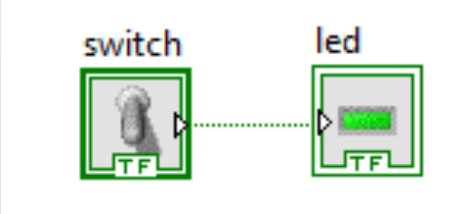
- Double precis 

ow more significant digits (approx. 15).

Real Number:
Single/Double Precision Floating-Point

- Represents analog values – real numbers have an integer and a fractional part. Examples are shown below.
  - Temperature – 34.56 degrees C
  - Level – 2.658 metres
  - Voltage – 12.678 volts
  - Pressure – 22.4 kPa

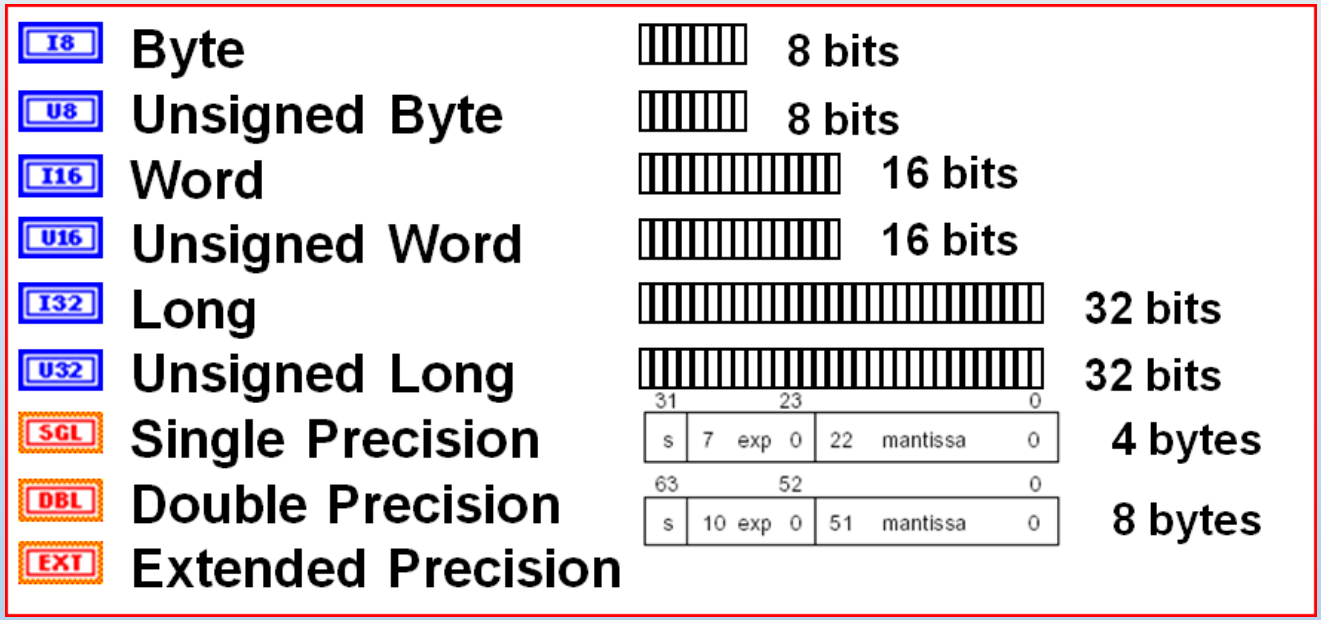
Boolean Data Type:



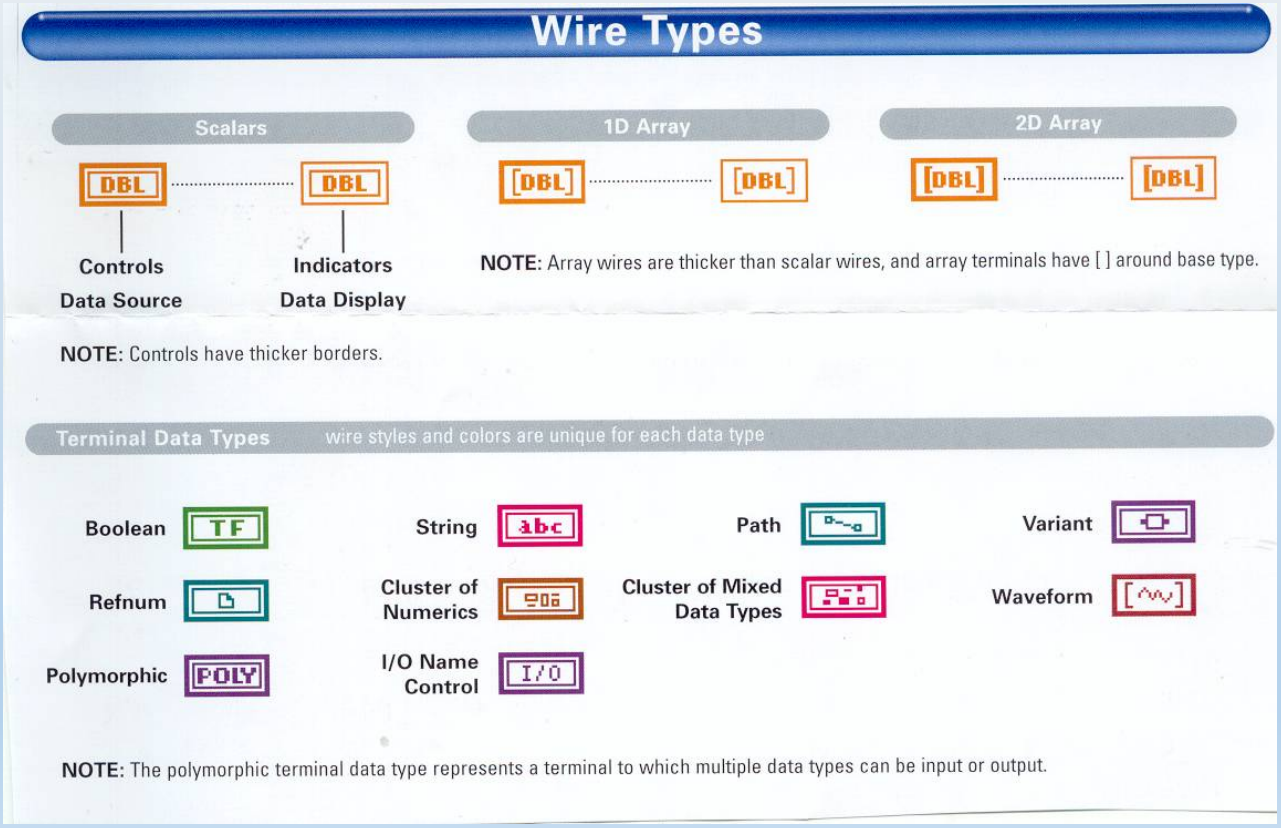
- True or False
- 1 or 0
- Only two possibilities

- A false is stored as an 8 bit value of all zeroes.
- Boolean values are used to read the state of a switch or a two state sensor. As outputs Boolean data can be used to control the state of motors, relays, lamps and heaters.

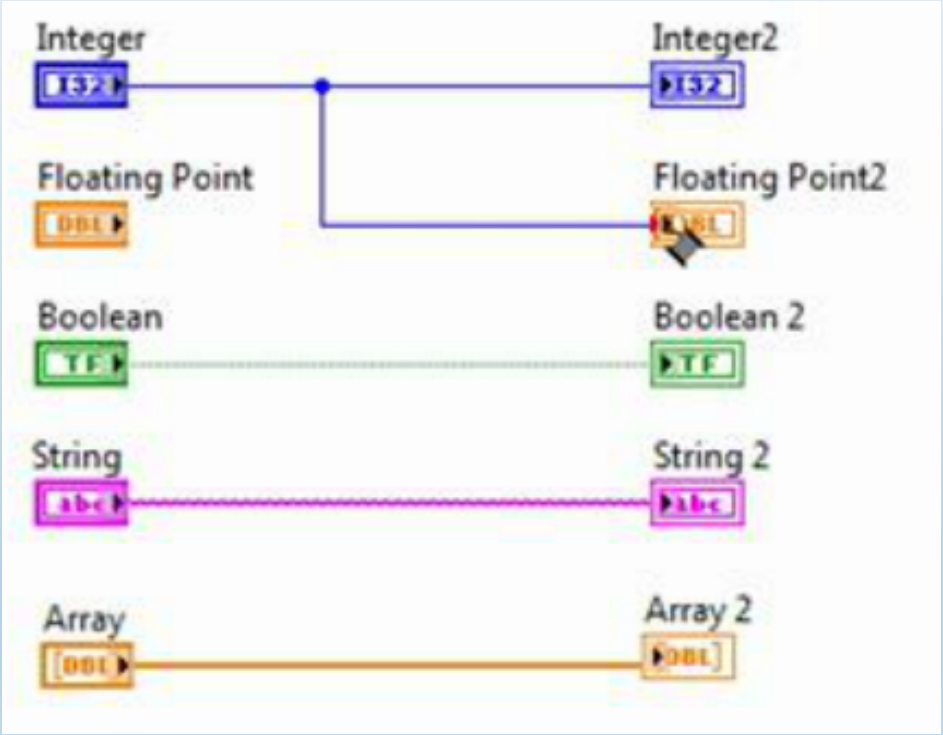
LabVIEW Numeric Types

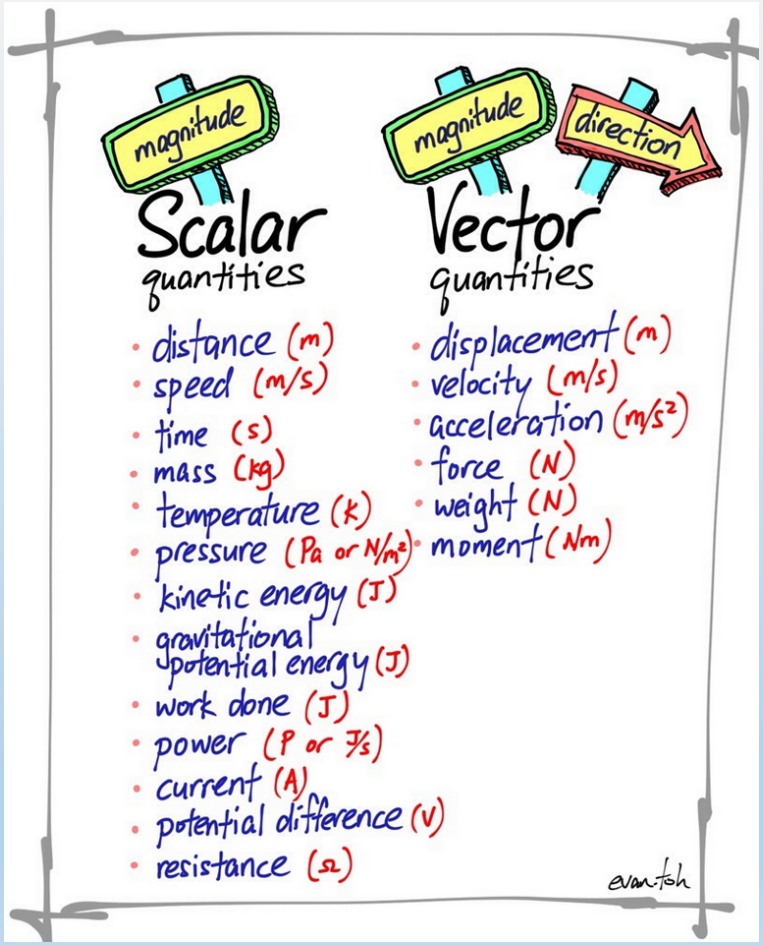


LabVIEW Wire Colours and Type:



LabVIEW Wire Colours:





Most values used in LabVIEW are numeric scalars.

Scalars have magnitude only.

A scalar example would be a distance such as 8 km.

Vectors have magnitude and direction.

Example: Displacement is a vector – move 8 metres north and then 4 metres north-east.

Vectors have magnitude and direction.

Introduction

Definitions of Vectors and Scalars

Physical quantities can be classified under two main headings -- Vectors and Scalars.

A **vector quantity** is any quantity that has **both magnitude (size) and direction.**
E.g., velocity, acceleration, force, momentum.

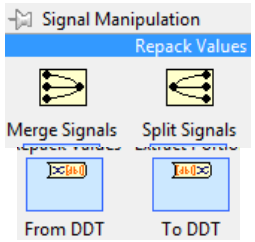
A **scalar quantity** is any quantity that has **magnitude only**, while direction is not taken into account.
E.g., speed, pressure, temperature, energy.

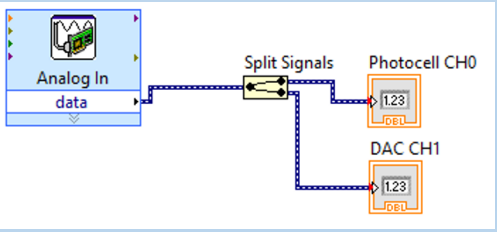
Using the Dynamic Data Type

Most Express VIs accept and/or return the dynamic data type. The dynamic data type appears as a dark blue terminal, shown as follows.

The dynamic data type accepts data from and sends data to the following data types, where the scalar data type is a floating-point number or a Boolean value:

- 1D array of waveforms
- 1D array of scalars
- 1D array of scalars—most recent value
- 1D array of scalars—single channel
- 2D array of scalars—columns are channels
- 2D array of scalars—rows are channels
- Single scalar
- Single waveform



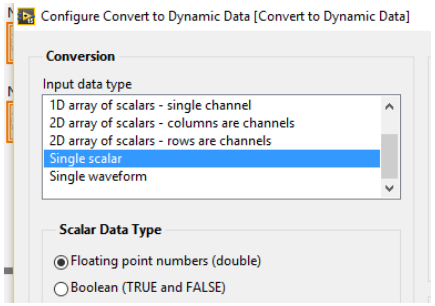


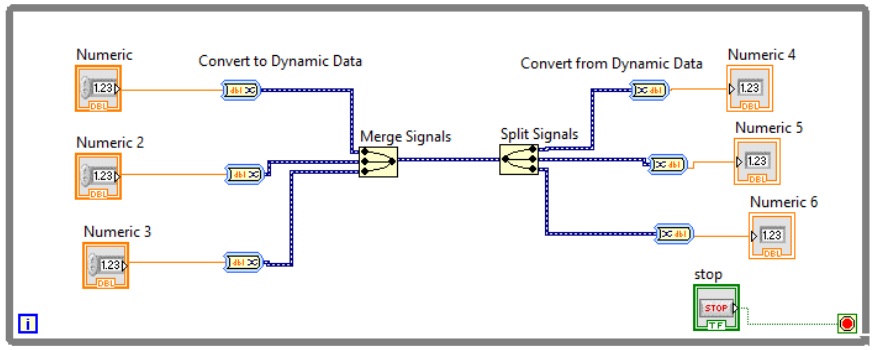
Express Vis in LabVIEW use dynamic data types. Dynamic data types may include multiple signals, arrays, different data types and time stamps. Split Signal, Combine signal, convert to DDT and convert from DDT are functions that may be required when using dynamic data types. These functions are found under Express -- Signal manipulation.

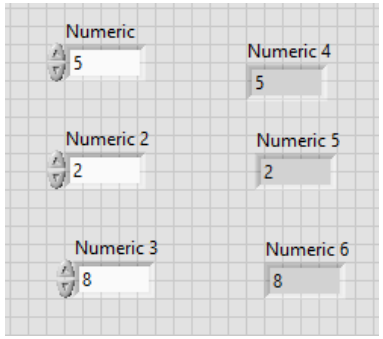
Working with Dynamic Data Types:

The example below shows the use of convert, split and merge dynamic signals. The order in which the signals are merged will be the same in which they are split.

When converting to or from dynamic data type you must double click on the function to select the data type, in this example the data is a single numeric scalar. A scalar is a value with magnitude only.





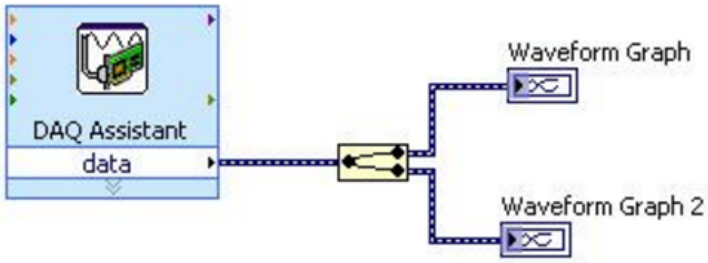


Split Signals

Signals are extracted from the Split Signals Express VI in the same order that they were merged together. This means that if you create a combined signal from 3 individual signals (using the Merge Signals Express VI) and you later want to retrieve only the third signal, you must resize the Split Signals Express VI to have three outputs.Then, use the output from the third (bottom) terminal. If you want to retrieve the second signal, only two outputs are needed.

Note: LabVIEW does allow you to expand the Split Signals Express VI to have more outputs than the number of individual signals present. However, the extra outputs do not return any data.

Split Signals Express VI is especially useful when you are using a DAQ Assistant that acquires data from multiple channels. When you have your DAQ Assistant configured to sample multiple channels, you can wire the Dynamic Data Type from the output of the DAQ Assistant to the Split Signals Express VI. You must resize the Split Signals Express VI to the number of channels you are sampling. This is done by grabbing the bottom of the VI and dragging down until you have the desired number of outputs. The signals go in order, so the first channel configured in the DAQ Assistant is the top most in the Split Signals Express VI. You can now wire graph indicators to each of the outputs to be able to view each signal individually. The picture below shows what the code might look like.



A string is a sequence of displayable or nondisplayable ASCII characters. Strings provide a platform-independent format for information and data. Some of the more common applications of strings include the following:

- Creating simple text messages.
- Controlling instruments by sending text commands to the instrument and returning data values in the form of either ASCII or binary strings, which you then convert to numeric values.
- Storing numeric data to disk. To store numeric data in an ASCII file, you must first convert numeric data to strings before writing the data to a disk file.
- Instructing or prompting the user with dialog boxes.

On the front panel, strings appear as tables, text entry boxes, and labels. LabVIEW includes built-in VIs and functions you can use to manipulate strings, including formatting strings, parsing strings, and other editing. LabVIEW represents string data with the color pink.

Numeric Data Type

Back to top

LabVIEW represents numeric data as floating-point numbers, fixed-point numbers, integers, unsigned integers, and complex numbers. Double and Single precision as well as Complex numeric data is represented with the color orange in LabVIEW. All Integer numeric data is represented with the color blue.

Note: The difference among the numeric data types is the number of bits they use to store data and the data values they represent.

Certain data types also provide extended configuration options. For example, you can associate physical units of measure with floating-point data, including complex numbers, and you can configure the encoding and range for fixed-point data.

Find more information on numeric data

View a table of numeric data types

Boolean Data Type

Back to top

LabVIEW stores Boolean data as 8-bit values. You can use a Boolean in LabVIEW to represent a 0 or 1, or a TRUE or FALSE. If the 8-bit value is zero, the Boolean value is FALSE. Any nonzero value represents TRUE. Common applications for Boolean data include representing digital data and serving as a front panel control that acts as a switch that has a mechanical action often used to control an execution structure such as a Case structure. A Boolean control is typically used as the conditional statement to exit a While Loop. In LabVIEW, the color green represents Boolean data.

Find more information about Boolean control of mechanical actions

Dynamic Data Type

Back to top

Most Express VIs accept and/or return the dynamic data type, which appears as a dark blue terminal.

Using the Convert to Dynamic Data and Convert from Dynamic Data VIs, you can convert floating-point numeric or Boolean data of the following data types:

- 1D array of waveforms
- 1D array of scalars
- 1D array of scalars—most recent value
- 1D array of scalars—single channel
- 2D array of scalars—columns are channels
- 2D array of scalars—rows are channels
- Single scalar
- Single waveform

Wire the dynamic data type to an indicator that can best present the data. Indicators include a graph, chart, or numeric, or Boolean indicator. However, because dynamic data undergoes an automatic conversion to match the indicator to which it is wired, Express VIs can slow down the block diagram execution speed.

The dynamic data type is for use with Express VIs. Most other VIs and functions that are shipped with LabVIEW do not accept this data type. To use a built-in VI or function to analyze or process the data the dynamic data type includes, you must convert the dynamic data type.

Arrays

Back to top

Sometimes it is beneficial to group related data. Use arrays and clusters to group related data in LabVIEW. Arrays combine data points of the same data type into one data structure, and clusters combine data points of multiple data types into one data structure.

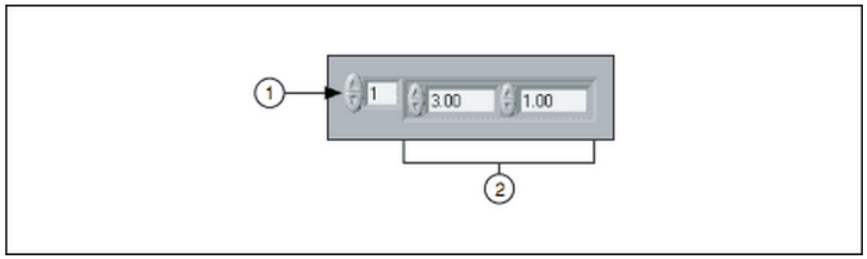
An array consists of elements and dimensions. Elements are the data points that make up the array. A dimension is the length, height, or depth of an array. An array can have one or more dimensions and as many as (2^{31})—1 elements per dimension, memory permitting.

You can build arrays of numeric, Boolean, path, string, waveform, and cluster data types. Consider using arrays when you work with a collection of similar data points and when you perform repetitive computations. Arrays are ideal for storing data you collect from waveforms or data generated in loops, where each iteration of a loop produces one element of the array.

Note: Array indexes in LabVIEW are zero-based. The index of the first element in the array, regardless of its dimension, is zero.

Array elements are ordered. An array uses an index so you can readily access any particular element. The index is zero-based, which means it is in the range of 0 to n-1, where n is the number of elements in the array. For example, n-12 represents the 12 months of the year, so the index ranges from 0 to 11. March is the third month, so it has an index of 2.

Figure 1 shows an example of an array of numerics. The first element shown in the array (3.00) is at index 1, and the second element (1.00) is at index 2. The element at index 0 is not shown in this image because element 1 is selected in the index display. The element selected in the index display always refers to the element shown in the upper left corner of the element display.



(1) Index Display | (2) Element Display

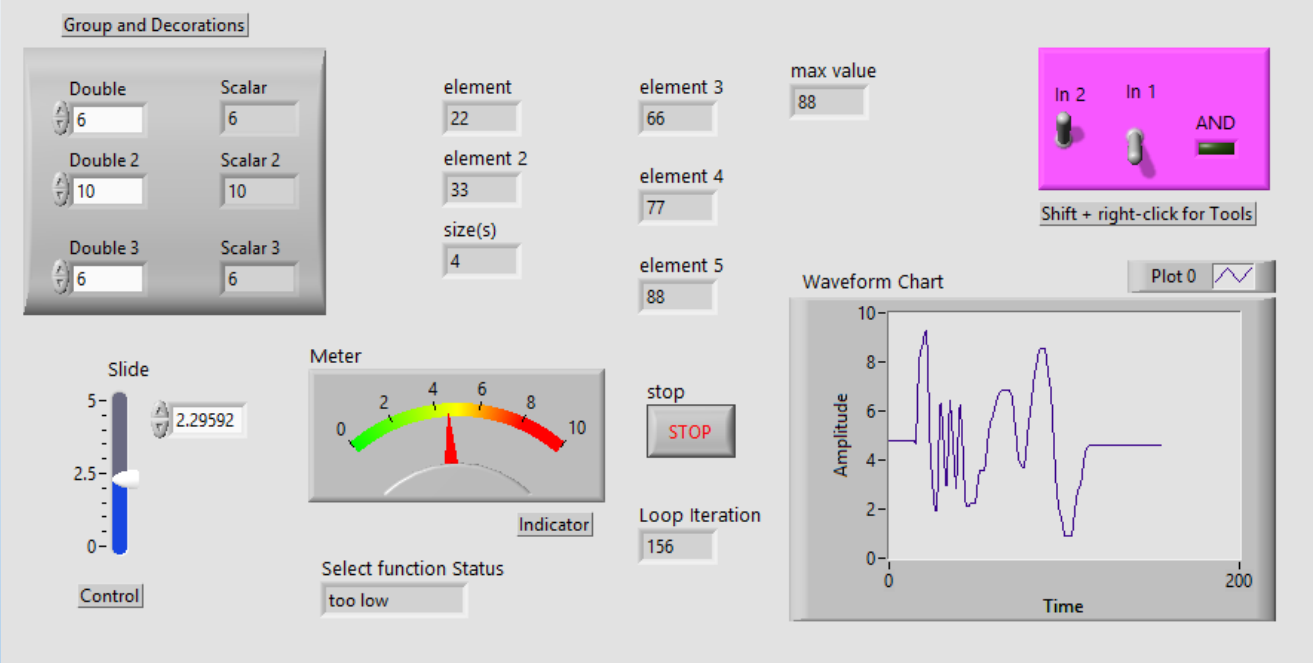
Figure 1. Array Control of Numerics

LabVIEW Tools:

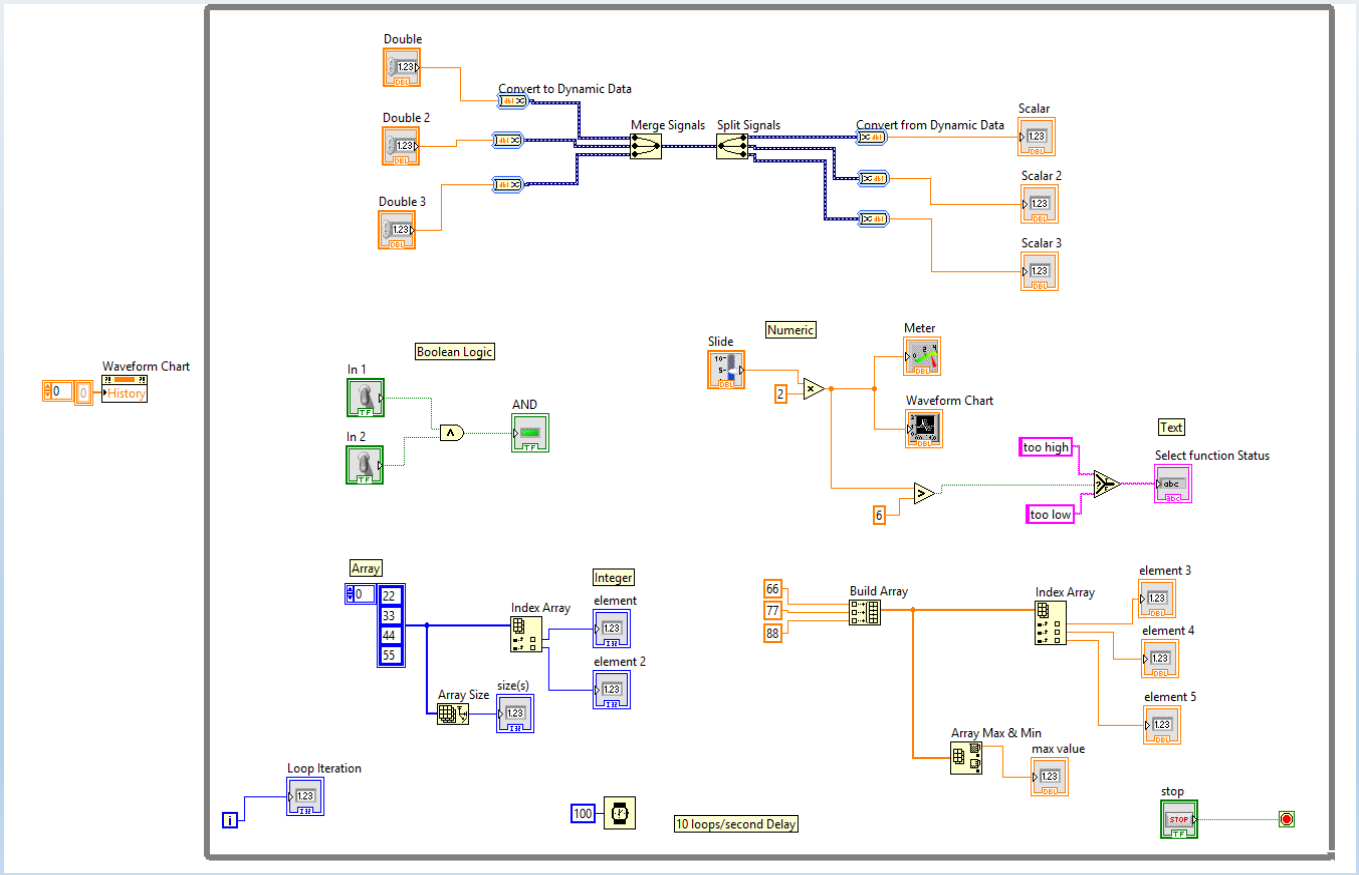
Icon	Tool
	Automatic Tool Selection
	Operating tool
	Positioning tool
	Labeling tool
	Wiring tool
	Probe tool

	Automatic Tool Selection - Automatically selects between the Operate Value, Position/Size/Select, Edit Text and Connect Wire Tools. It is the most commonly Selected item on the Tools Palette.
	Operate Value - Manipulates the values of Front Panel Controls and Indicators. Generally selected while running a VI.
	Position/Size/Select - Selects objects in a VI, moves them and resizes them.
	Edit Text - Creates Free Labels on the Front Panel and edits their text. Also edits text in Owned Labels.
	Connect Wire - Wires Block Diagram nodes and objects together.
	Object Shortcut Menu - Displays an object's pop-up menu with the left mouse button.
	Scroll Window - Pans the content on a VI window without using the window's scroll bars.
	Set/Clear Breakpoint - Create breakpoints in a VI Block Diagram for debugging purposes. Breakpoints cause the VI to suspend execution at a particular point.
	Probe Data - Places Probes on wires in the Block Diagram for debugging purposes. A Probe displays the data on a wire where it is placed.
	Get Color - Copies colors from VI objects to pasting with the Set Color Tool.
	Set Color - Colors objects in a VI and displays the foreground and background colors of the object.

Front Panel Example:



Block Diagram Example:

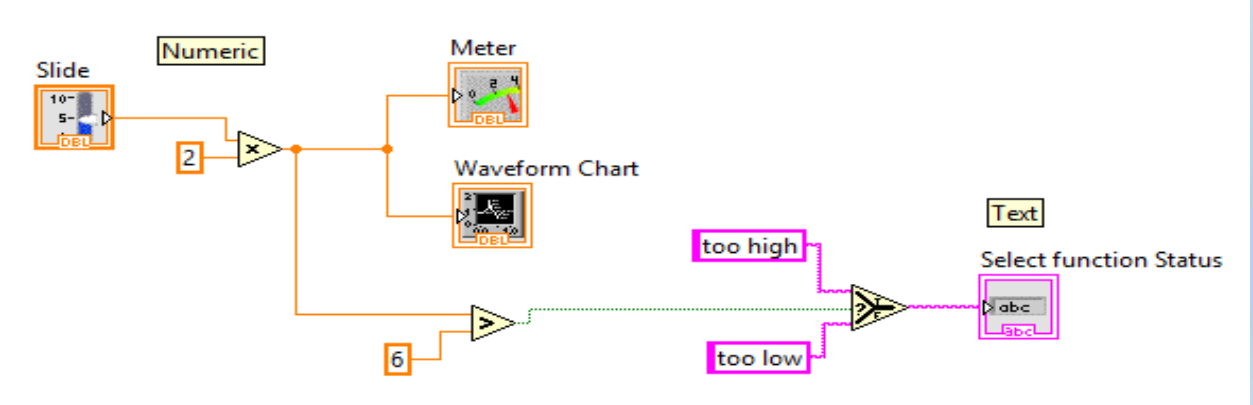


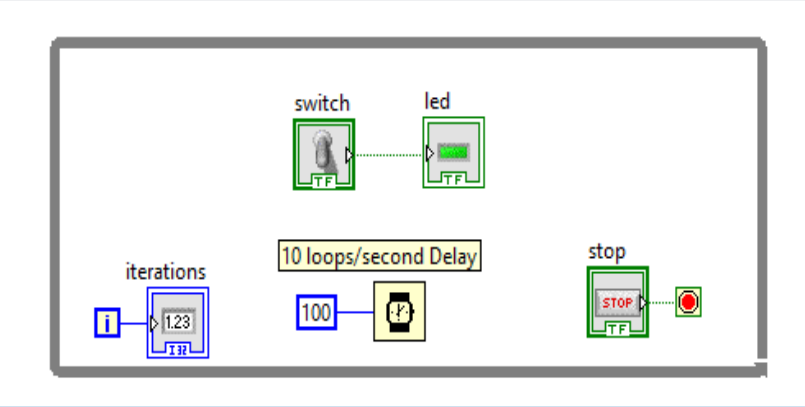
Working with Numeric Data Types:

The example below uses a Multiply Function to multiply a numeric input from the Front Panel and multiply it by 2. The result is displayed on a Front Panel (FP) meter and Waveform Chart.

A Compare function is used to compare the multiplication result with 6. If the result is greater than six the Boolean output is True.

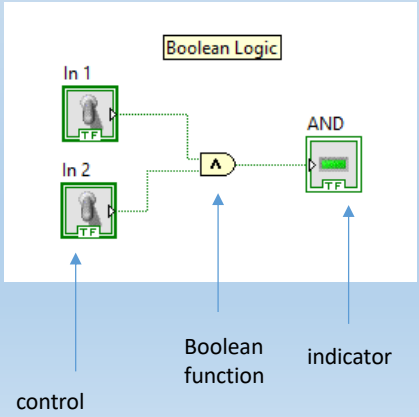
The Select Function then uses the Boolean result to display “too high” if true and “too low” if the Boolean input is false. The Select input can accept many different data types that will be sent to the output. The centre input must be Boolean.





While Loop – the loop continues until stop is true.

The number of time the loop has executed is shown by the iteration indicator.



Boolean indicator and control.

Controls have a darker boarder and an arrow pointing outwards.

Controls are a source of data.

Indicators have arrows pointing inwards, they receive data.

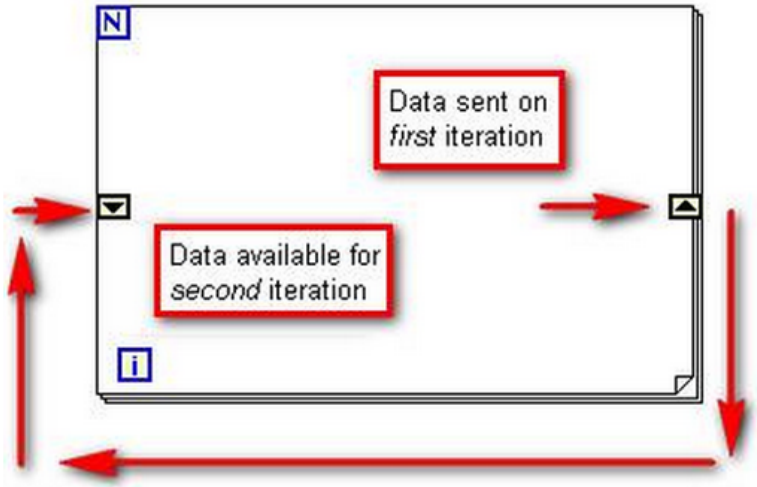
LabVIEW

Shift Registers
Mechanical Action
Formula Node

Shift Register –

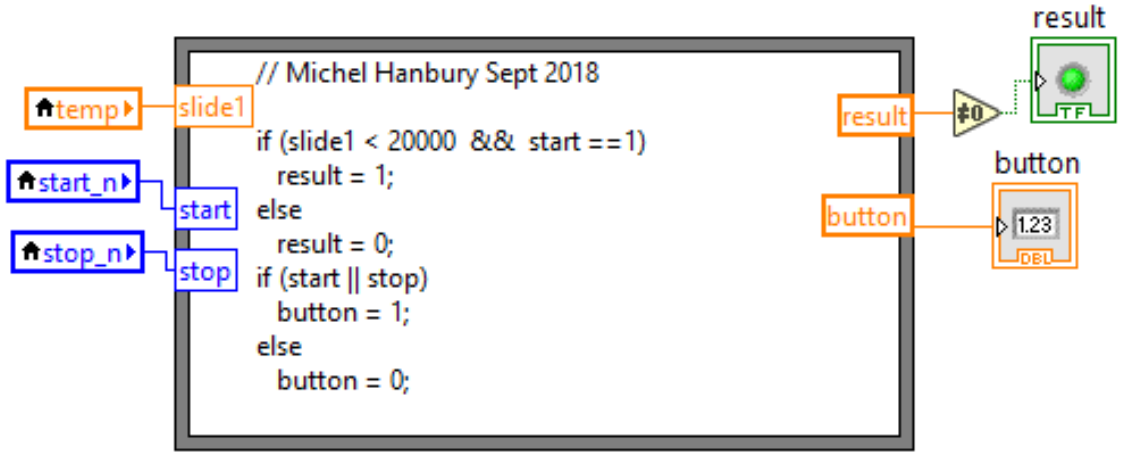
On each iteration of the loop the data from the right side of the loop is sent to the left. The constant on the left is only active on the first iteration.

Data enters the right shift register and is passed to the left shift register on the next iteration of the loop.



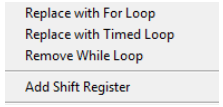
Expression Node –

"C" like code used within LabVIEW. Inputs are created by right clicking on the left side of the frame. Outputs are on the right.

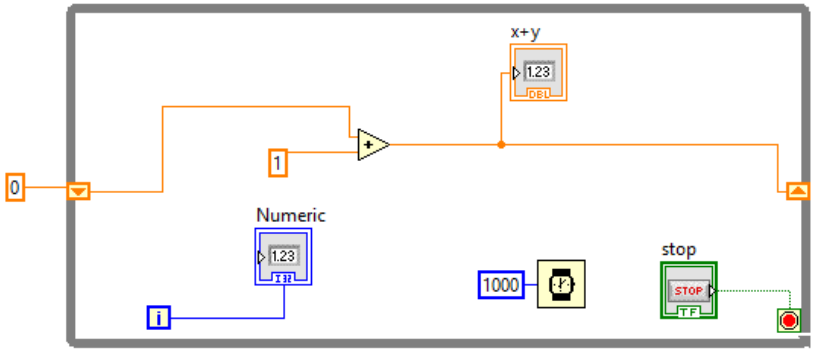


LabVIEW Shift Register Example

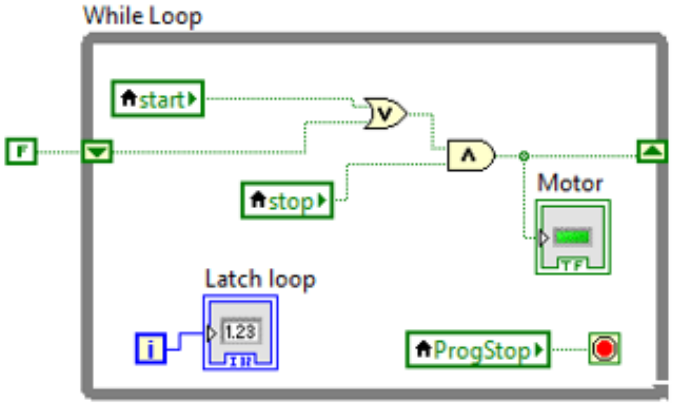
To create the shift register right click on the border the While Loop.
Select Add Shift Register.

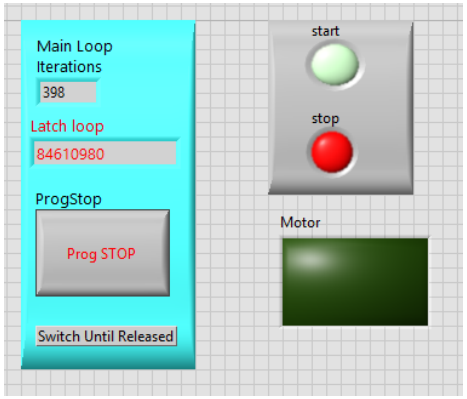


On each iteration of the loop the data from the right side of the loop is sent to the left. The constant on the left is only active on the first iteration.



LabVIEW Shift Register Example





Shift Register 2nd Example:

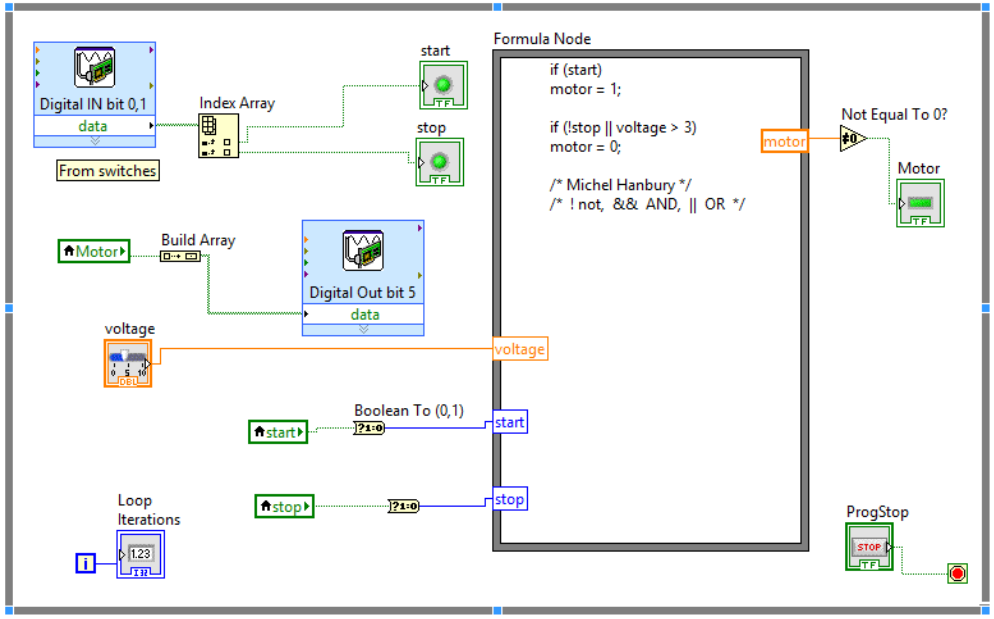
To create the shift register right click on the border of the while loop.
Select Add Shift Register.

The motor signal will go true when start is true and remain that way because of the shift register until stop is true. Take some time to understand the shift register, try it out in the lab.

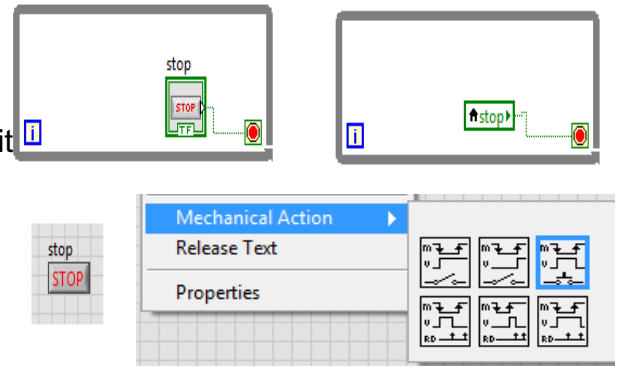
The Boolean constant “F” is true only on the first iteration of the while loop.

Formula Node:

The Formula node allows a programmer to write “C” code within LabVIEW. The program uses functions to convert Boolean to numeric and numeric to Boolean. The program also uses local variables to reduce the amount of wiring. The block diagram (BD) includes an express VI for myDAQ digital input and output.



Mechanical Action:



LabVIEW push buttons have a feature called mechanical action which can be changed to suit the type of switch the application requires. The are 6 choices.

** Use Switch until release when you use a local variable.

Here is a glossary of the different mechanical actions.

1. Switch When Pressed
   - When the button is pressed, it will stay pressed until you press it again.
   - The value of the button will be true when the button is pressed.
2. Switch When Released
   - When the button is pressed, it will stay pressed until you press it again.
   - The value of button will change, when you release it after pressing it. For example, you click it, but until you release it, the value will still be what it was before you clicked it.
3. Switch Until Released
   - When the button is pressed, it will stay pressed until you release it.
   - The value of the button will change when the button is pressed, then when it is released it will change back.
4. Latch When Pressed
   - When the button is pressed, it will stay pressed until LabVIEW reads the value of the control. Then it will revert to its default state.
   - The value of the button will change when the button is pressed, then when the value is read by LabVIEW, it will change back to its default state.
5. Latch When Released
   - When the button is pressed, it will stay pressed until LabVIEW reads the value of the control. Then it will revert to its default state.
   - The value of the button will change when the button is released, then when the value is read by LabVIEW, it will change back to its default state.
6. Latch Until Released
   - When the button is pressed, it will stay pressed until the button is released and LabVIEW has read the value.
   - The value of button will change when the button is pressed, then change back once the button is released and LabVIEW has read the value.

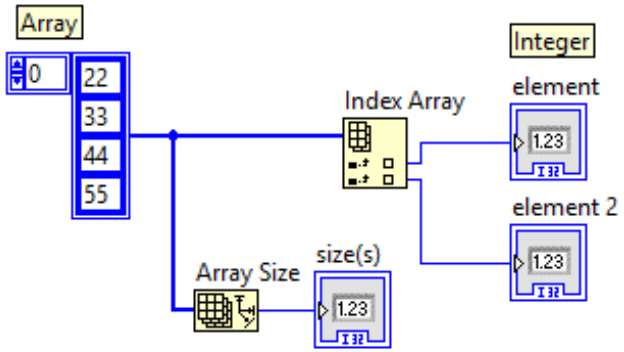
LabVIEW Arrays Basics

LabVIEW Arrays

An array is a group of elements of the same data type. Arrays are often used in LabVIEW to store data from analog readings. The data can then be used in a spreadsheet file. Arrays are accessed by their indices; each element's index is in the range of 0 to *N*-1 where *N* is the total elements in the array.

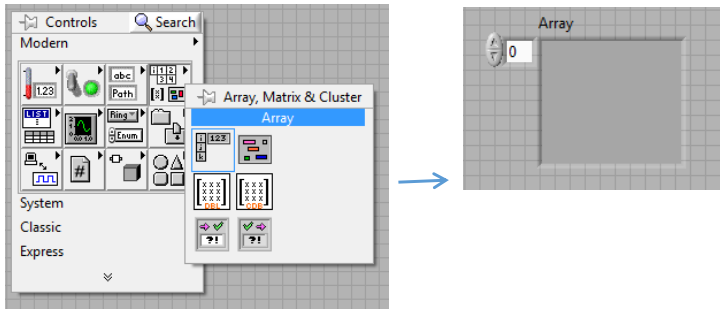
The lines connecting arrays are thicker than integer or numeric values. The first element in an array is 0.

An Array Index is used to point to a specific element within the array.
The Array size function returns the number of elements in an array.

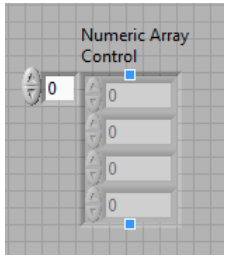


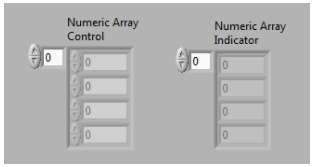
Creating Arrays in LabVIEW

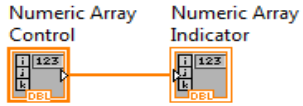
1. Place an empty array shell onto the block diagram.

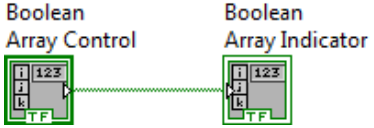


2. Drag a data object(such as a numeric indicator or control) into the empty element window. Resize the shell by dragging the frame handles. Arrays can be an any data type including character and Boolean. The arrays can be indicators or controls.



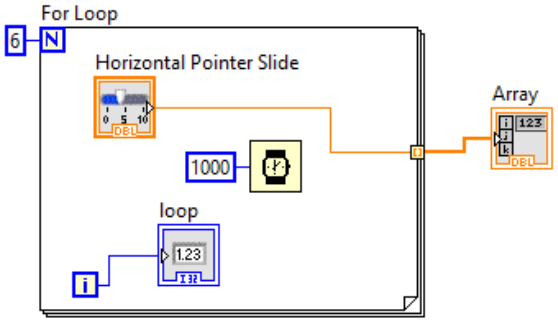


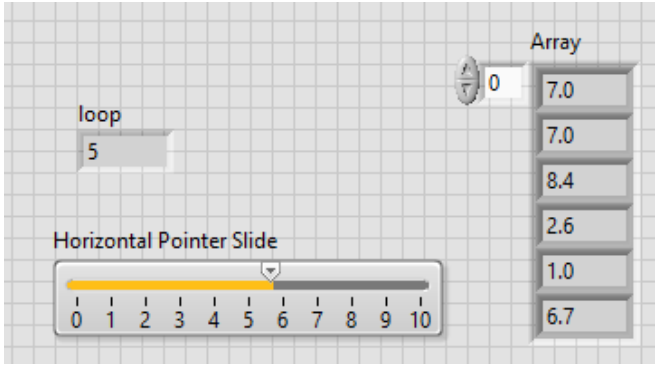




LabVIEW Arrays for Data Acquisition

An array is often used in data acquisition systems to read or sample data and store the sample in an array. The data can later be displayed in a spreadsheet file once it has been captured. The example below uses a "For Loop" that executes 6 times (once/second) and captures one sample of data on each loop and stores the data in an array.





Arrays Functions:

There are array functions to:

- Find the size of an array
- Find the sum of an array.
- Find the min and max value of an array.
- Find the index of the min and max value.
- Find the average of the array values.