Tsp@2023

LabVIEW FPGA @ CERN

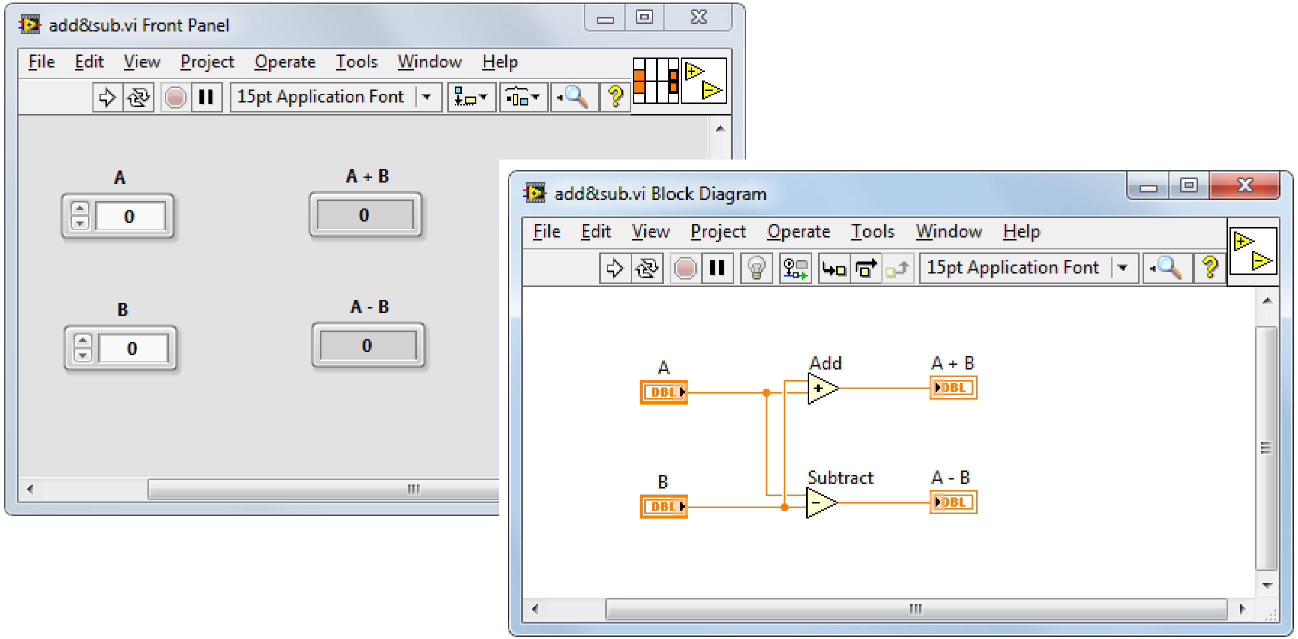
- Unofficial
- For fun
- Share knowledge

About the workshops

- Minimize theory
- Maximize practice
- Some fun examples

LabVIEW

- Intuitive
- Data driven
- Hardware integration



NI : part of Emerson

Leader in data acquisition technology with innovative modular instruments and LabVIEW graphical programming software

Ritu Favre / Head of Test and Measurement Business Unit

- Corporate headquarters in Austin, TX
- Offices in nearly 50 countries
- 35,000+ companies served annually
- More than 1,000 products
- Approx. 7,100 employees
- 600 Alliance Partners
- Part of Emerson Electric Co. (2023)

FORTUNE®
100 BEST
COMPANIES
TO WORK FOR

Diversity of applications

SEMICONDUCTOR

AUTOMOTIVE

AEROSPACE, DEFENSE, & GOVERNMENT

ELECTRONICS

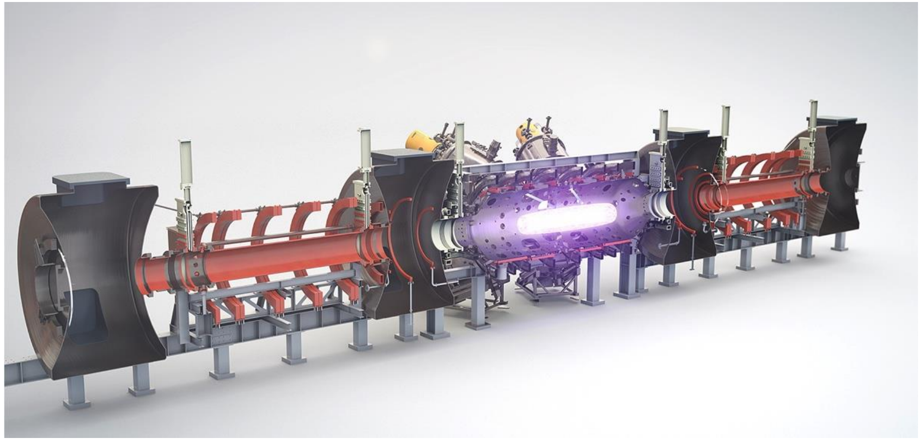
ENERGY

ACADEMIC & RESEARCH

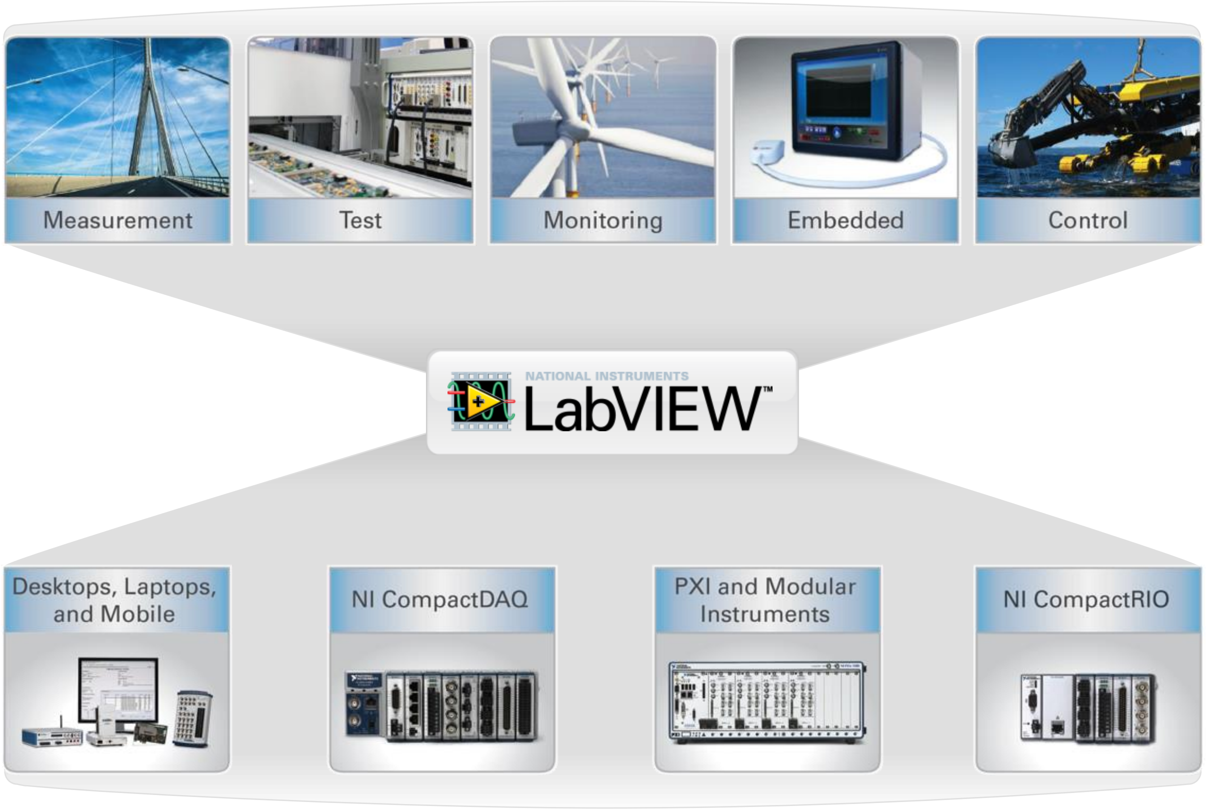
SPACEX

Falcon rocket launch pad software

Commercial nuclear fusion power



LabVIEW on different hardware

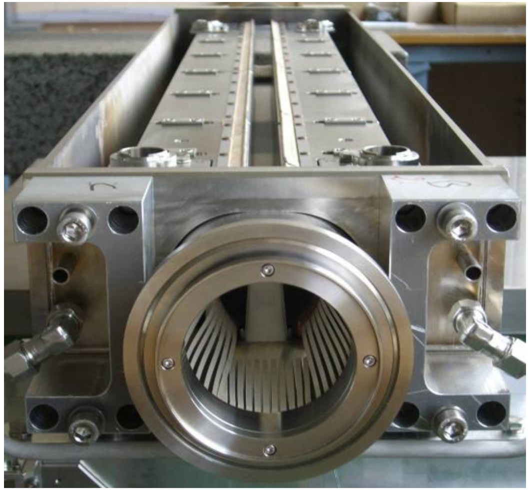


Applications

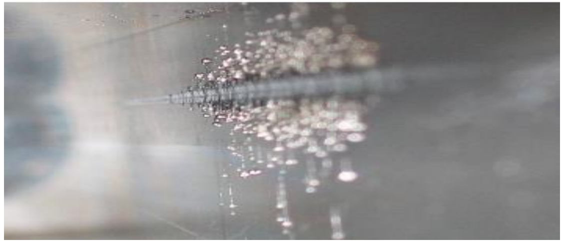
Hardware

Projects based on NI @ CERN

- LHC collimators real-time control system

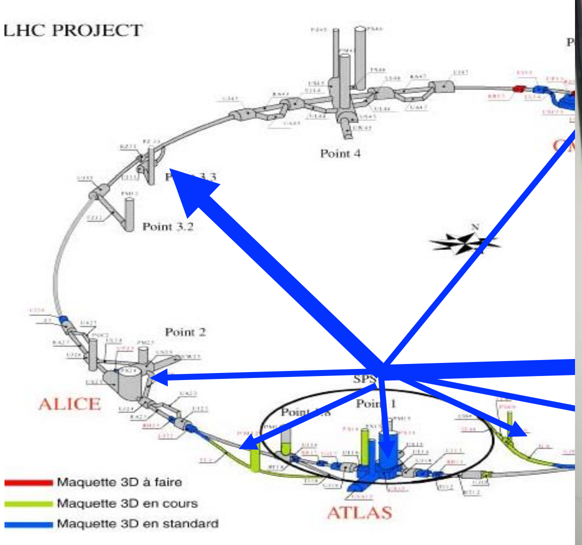


Control system requirements	
Axes positioning accuracy	few µm
Axes motion synchronization	below 1 ms
Response delay to a digital start trigger	100 µs
Position sensors RT survey frequency	100 Hz
Reliability	Very high



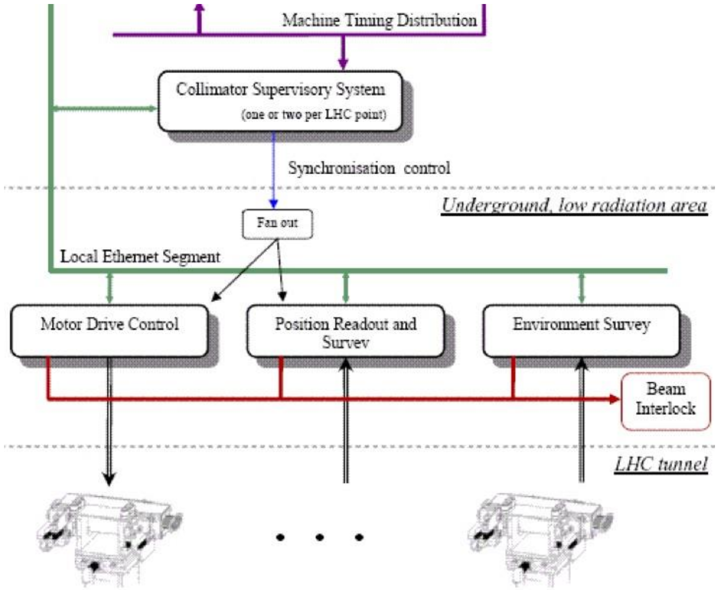
- LHC collimators real-time control system

Layout



120 systems

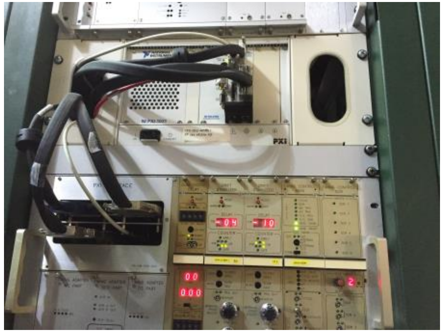
Architecture



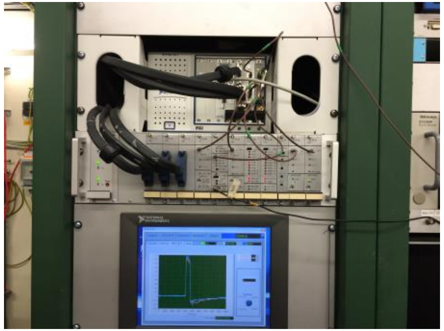
• Kicker magnet control systems

Kicker magnets steer the beam in and out of the different accelerators

Booster



Booster



LEIR

AD

LHC

CERN LabVIEW support

- Website: cern.ch/labview
- E-mail: labview.support@cern.ch or SNOW ticket with title: labview

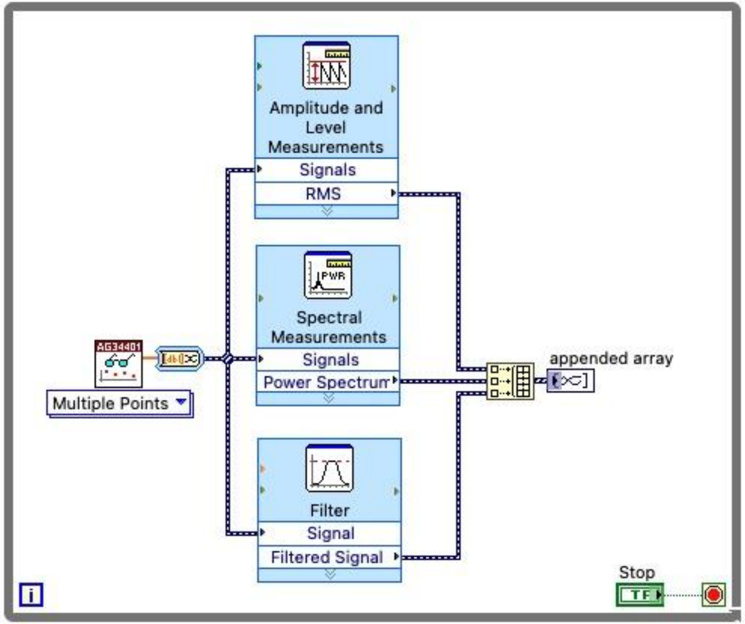
CERN LabVIEW support

- Website: cern.ch/labview
- E-mail: SNOW ticket with title: labview

Since 2017

Why LabVIEW?

- Same concepts as in traditional languages (data types, loops, event handling, recursion and OOP)
- Data flow (execution is data-driven, not determined by sequential lines of text)
  - Automatic parallelism
  - Automatic data synchronisation
- Intuitive
- Easy to debug
- NI hardware integration
- Combines with other languages



B. Project Explorer

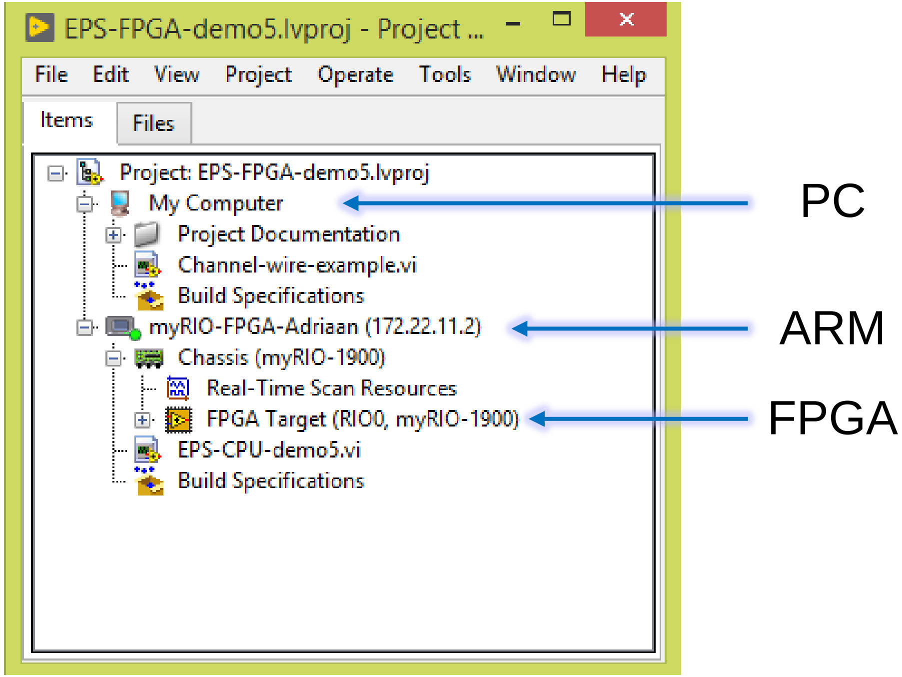
Project Explorer Window

Files Types

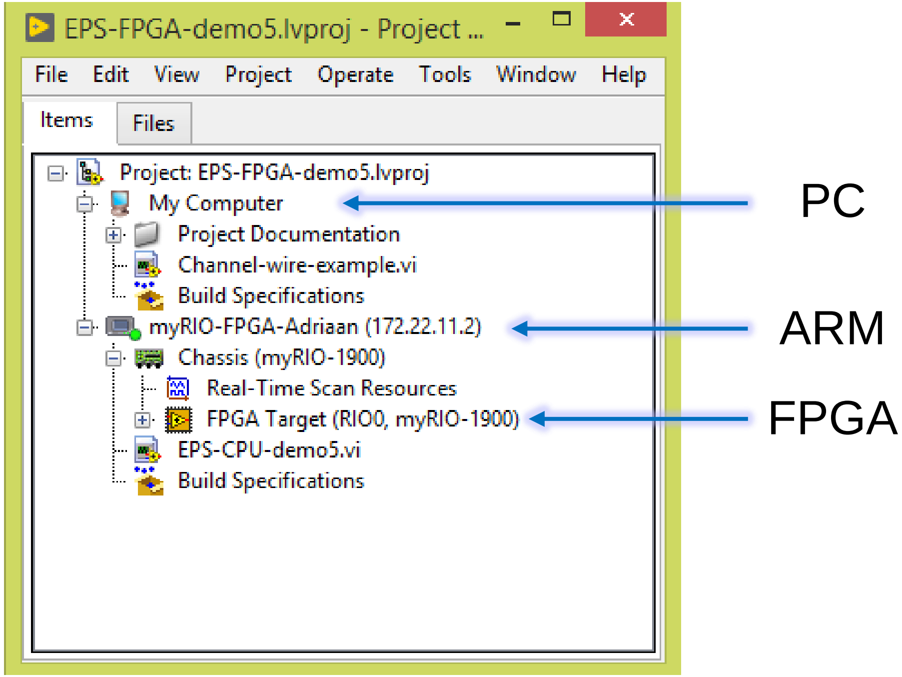
Project Folders

Project Explorer

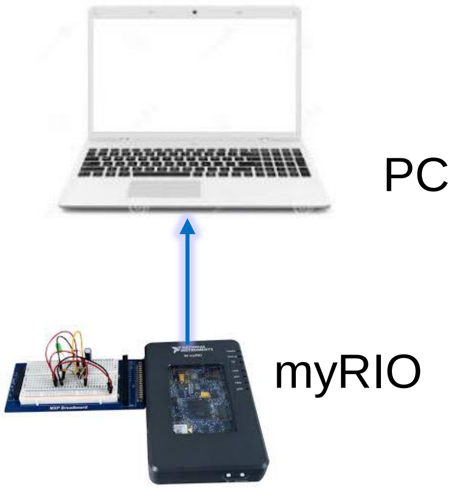
- See the hierarchy
- Organise project files
- Deploy files to targets
- Manage code for build options
  - Executables, installers, and zip files
- Integrate with source code control providers



Project Explorer

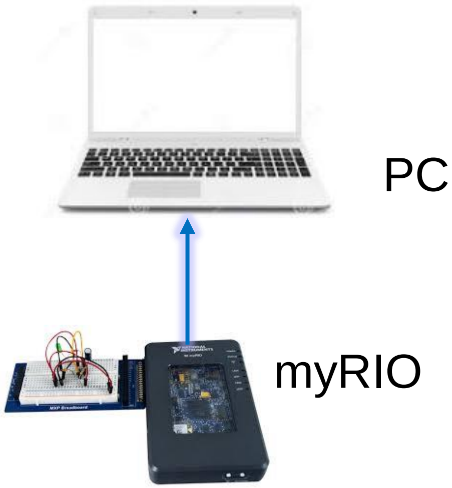


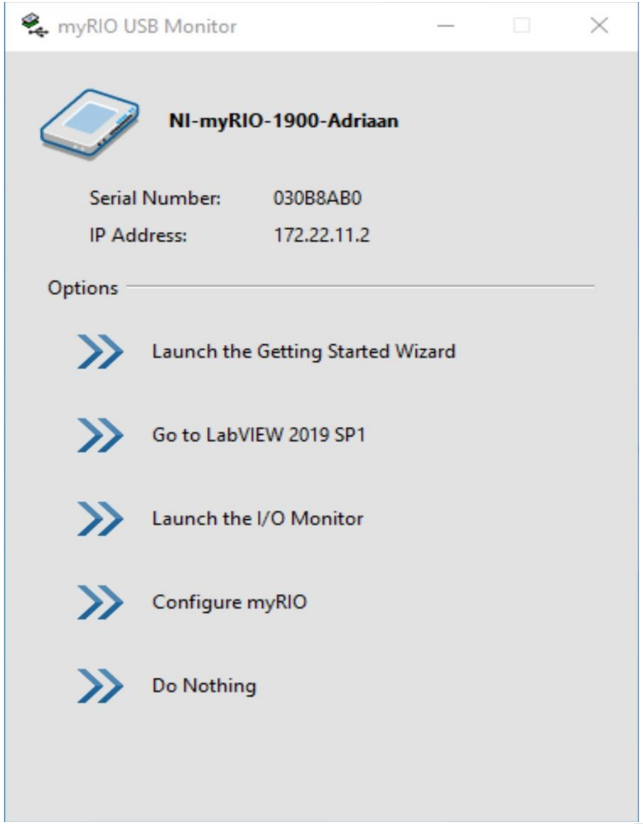
Connect to myRIO



1. Don’t have the myRIO connected yet
2. Power up the myRIO
3. Wait until the Status LED is off
4. Connect the myRIO to your PC

Start LabVIEW





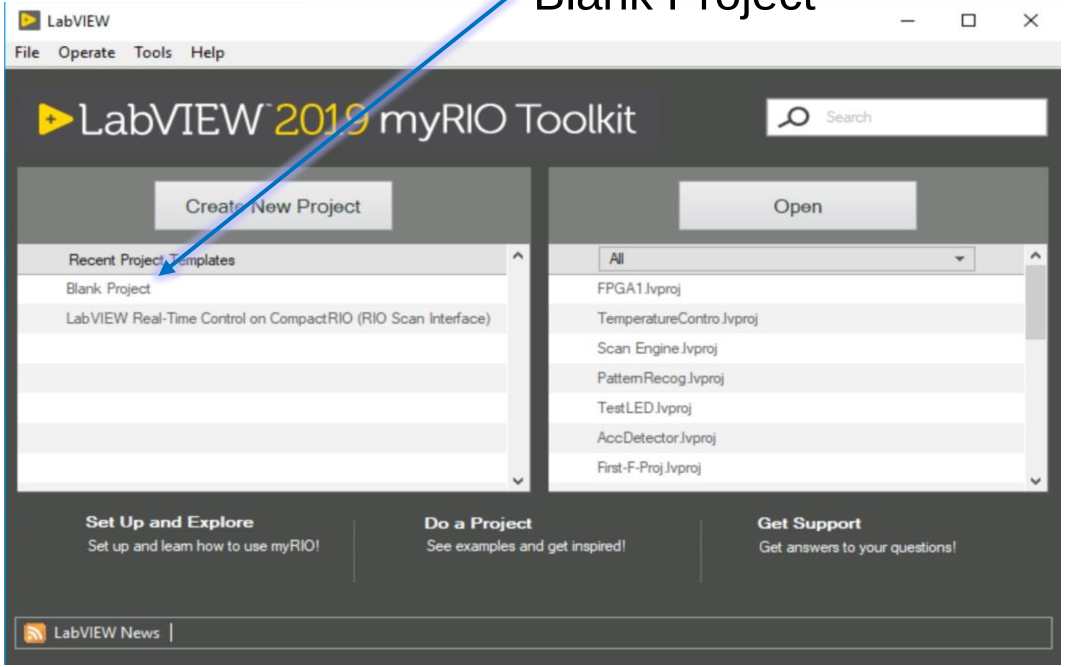
Go to LabVIEW

Project

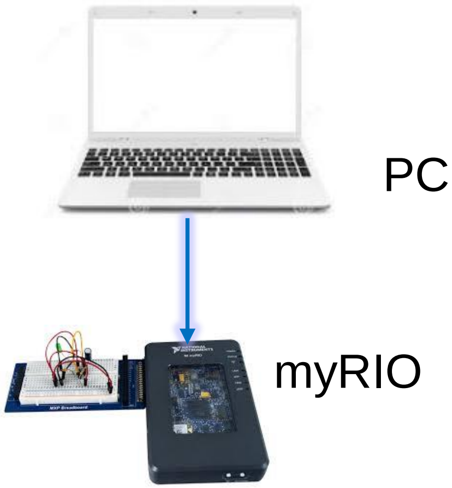
PC

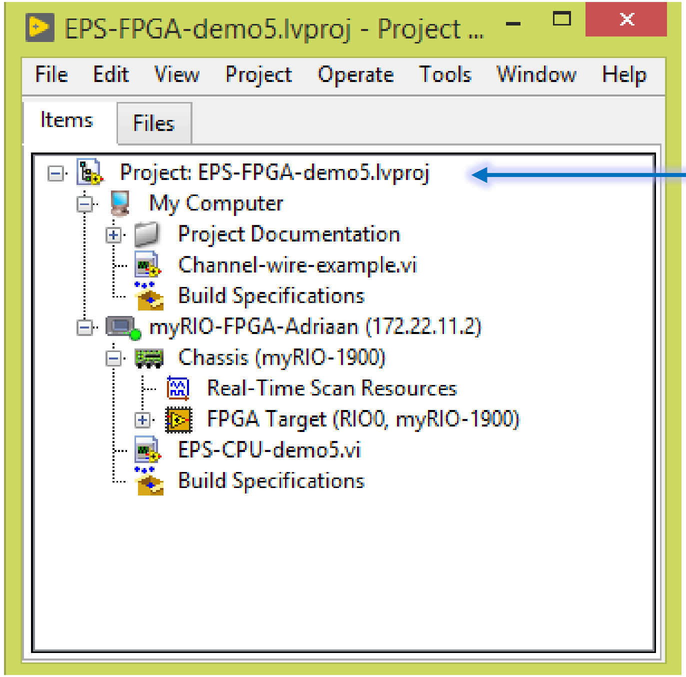
myRIO

Blank Project



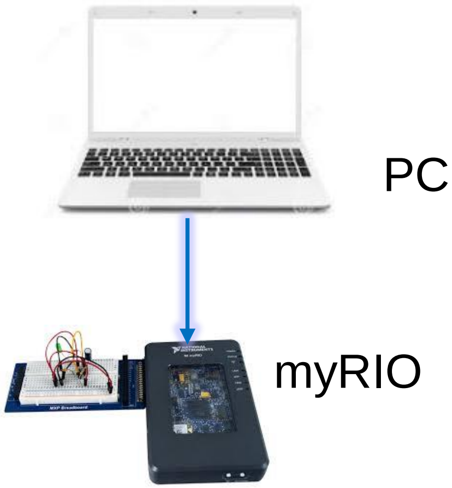
New target

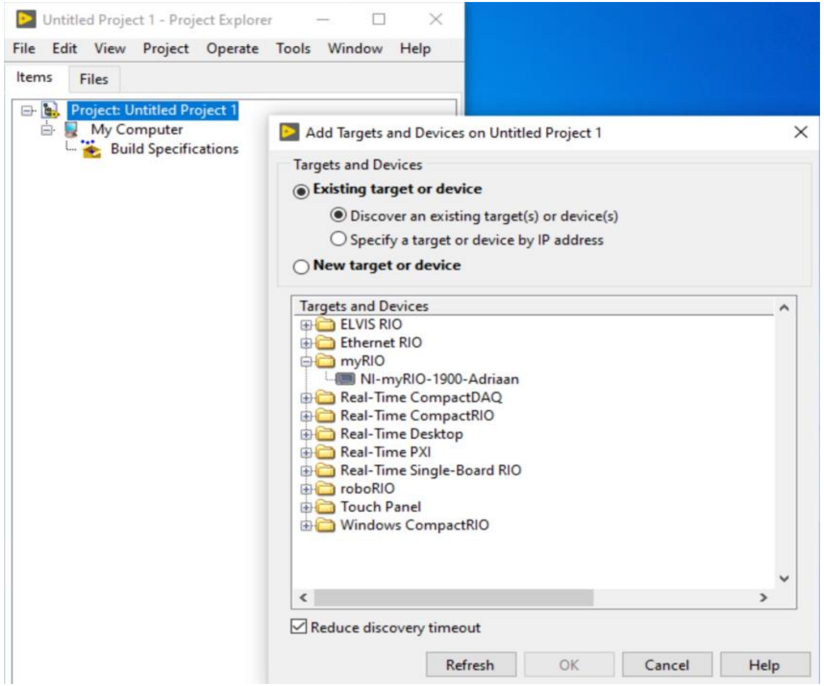




Right click Project
Choose: New -> targets and devices

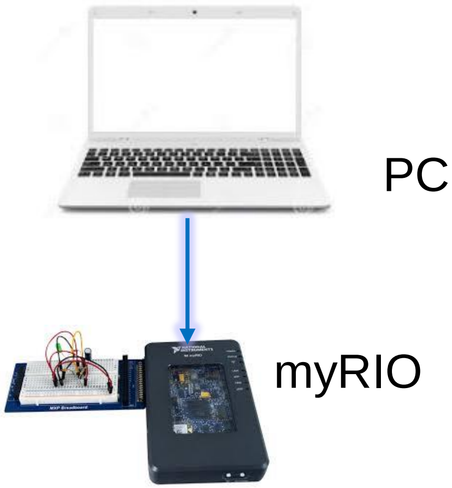
Select myRIO

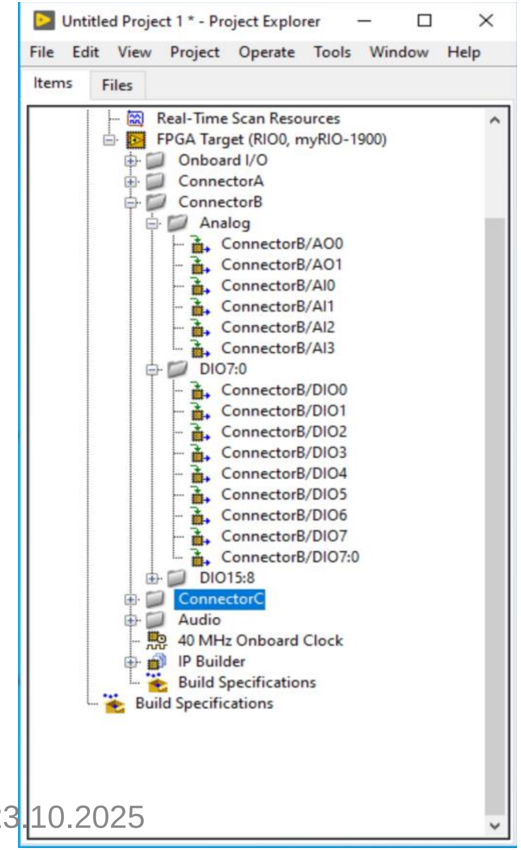




Open myRIO with +
Select NI myRIO
OK

Prepare myRIO





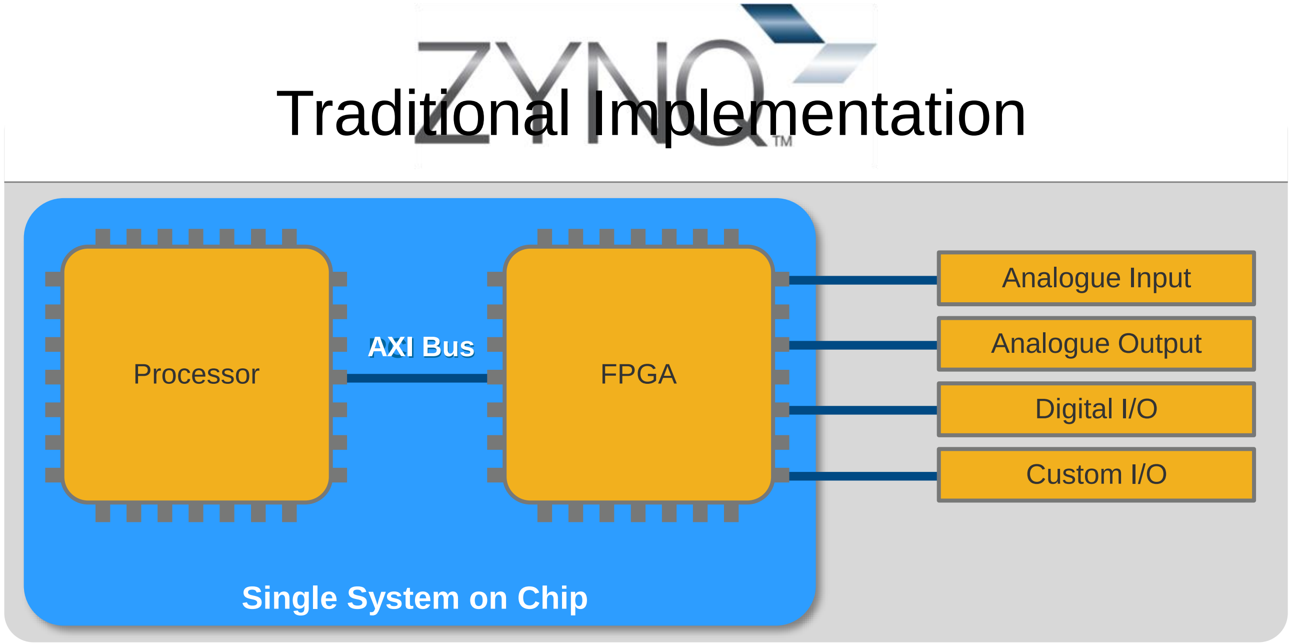
Close tabs:

- Onboard I/O
- Connector A
- Connector B DIO15:8
- Connector C
- Audio

NI myRIO Product Overview: Front View

XILINX Zynq SoC

What is Zynq?



ZYNQ
Traditional Implementation
Processor
AXI Bus
FPGA
Analogue Input
Analogue Output
Digital I/O
Custom I/O
Single System on Chip

Why Zynq Matters in Education

- Smaller Size, Lower Power
- 667 MHz Dual-Core ARM Cortex-A9 Processor
- Artix-7 FPGA, 28k logic cells
- 16 DMA Channels
- 92 Billion calculations per second

Why Zynq **Really** Matters in Education

Leading Industry Grade Technology

The same technology is used in the modular I/O Compact RIO systems

C. Parts of a VI

Front Panel

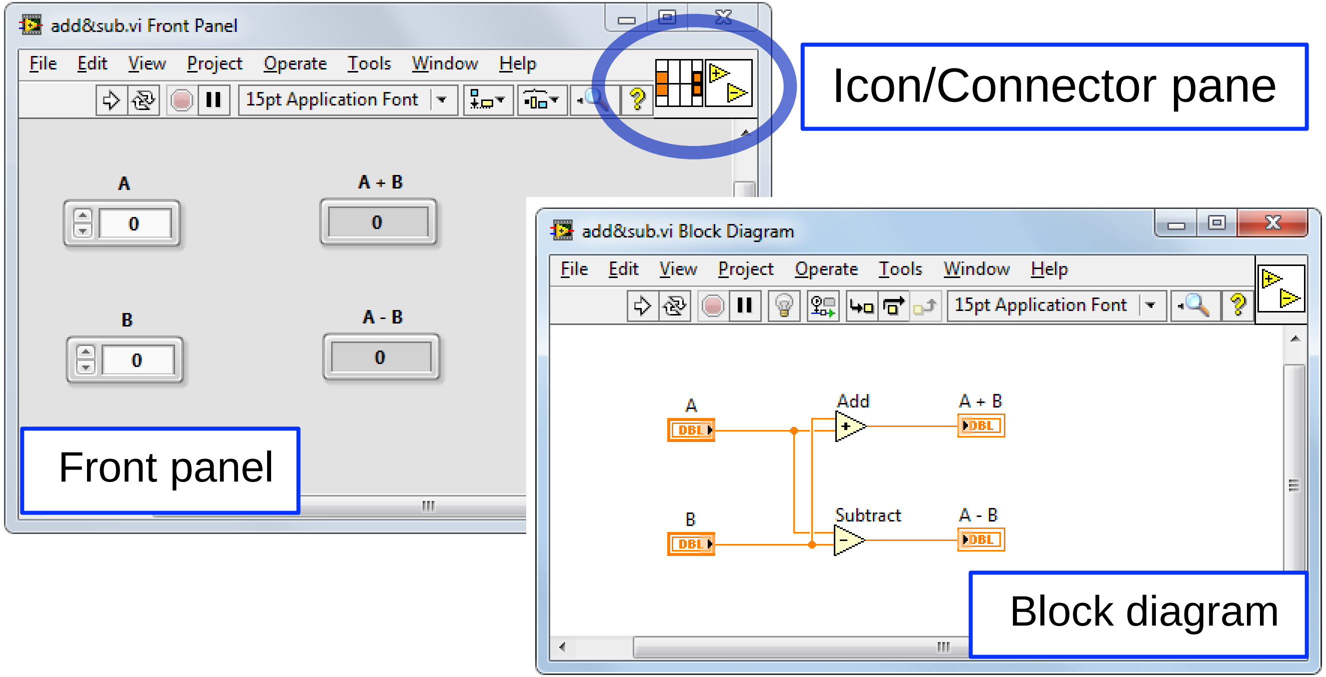
Block Diagram

Icon

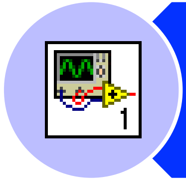
Connector Pane

Parts of a VI

VIs have 3 main components:



Parts of a VI – Front Panel

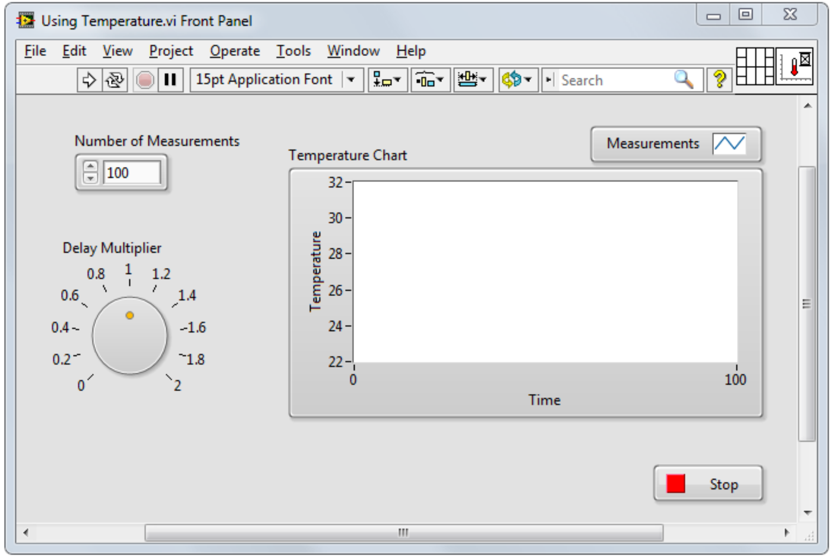


Front Panel – User interface for the VI

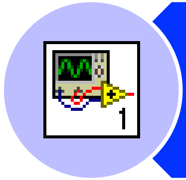
You build the front panel with:

controls (inputs) and

indicators (outputs)

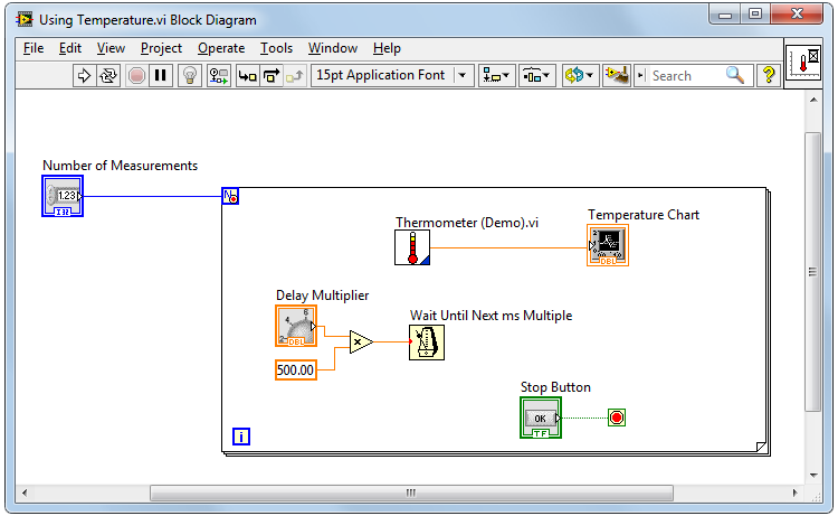


Parts of a VI – Block Diagram

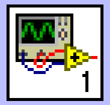


Block Diagram – Contains the graphical source code

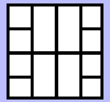
Front panel objects appear as terminals on the block diagram



Parts of a VI – Icon/Connector Pane



Icon – Graphical representation of a VI



Connector Pane – Map of the inputs and outputs of a VI

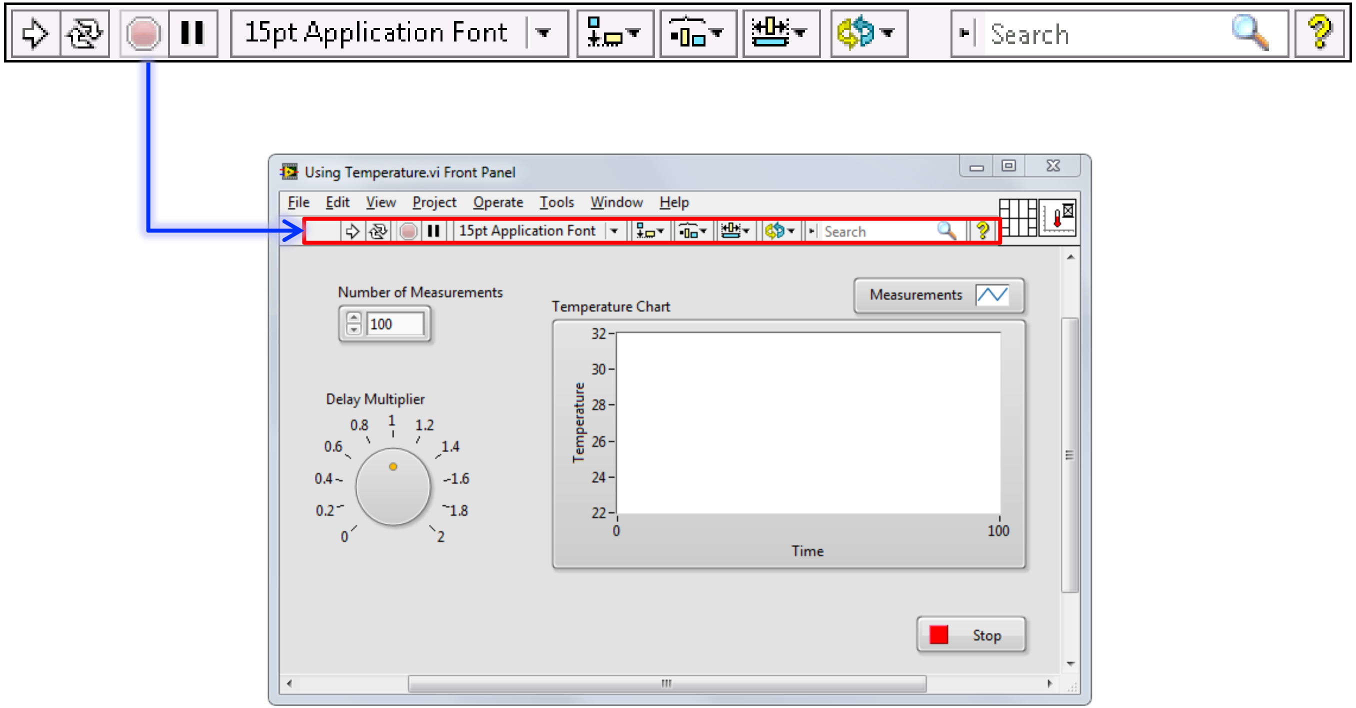
Icons and connector panes are necessary to use a VI as a subVI.

- A subVI is a VI that appears on the block diagram of another VI.
- A subVI is similar to a subroutine or function in a text-based programming language.

Show – off (2)

Figures

Front Panel



15pt Application Font
Search
Using Temperature.vi Front Panel
File
Edit
View
Project
Operate
Tools
Window
Help
15pt Application Font
Search
Number of Measurements
100
Delay Multiplier
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Temperature Chart
Measurements
32
30
28
26
24
22
Temperature
0
100
Time
Stop

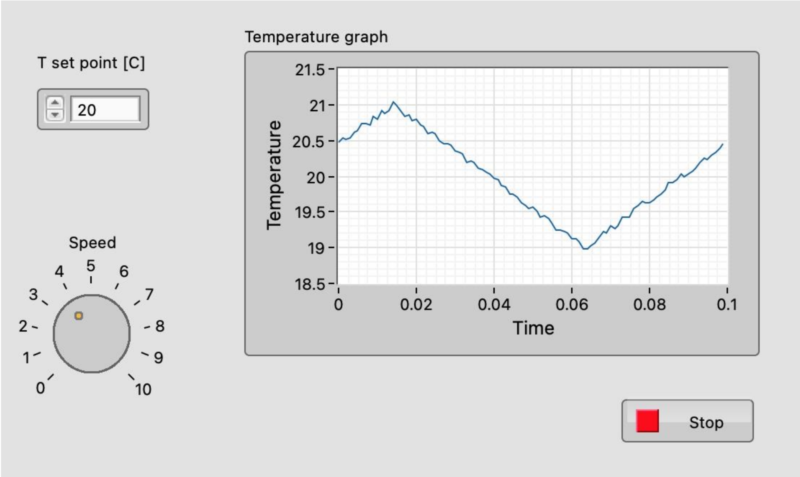
Controls and Indicators

Controls

- Input devices
- Knobs, buttons, slides
- Supply data to the block diagram

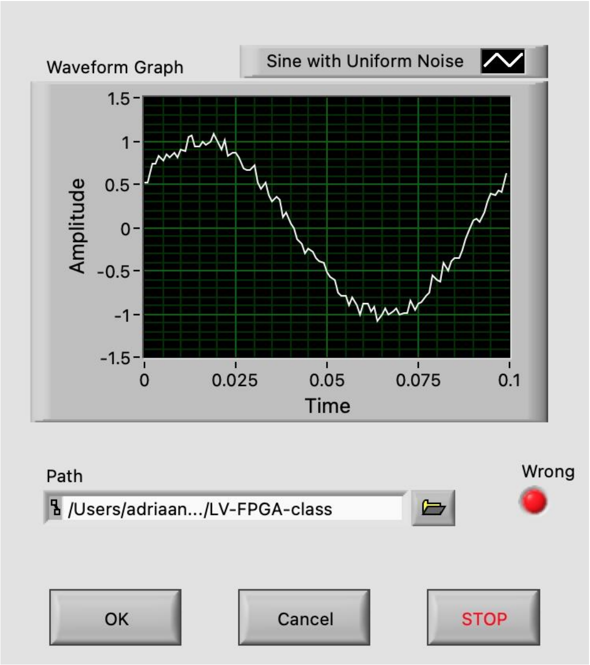
Indicators

- Output devices
- Graphs, LEDs
- Display data the block diagram acquires or generates

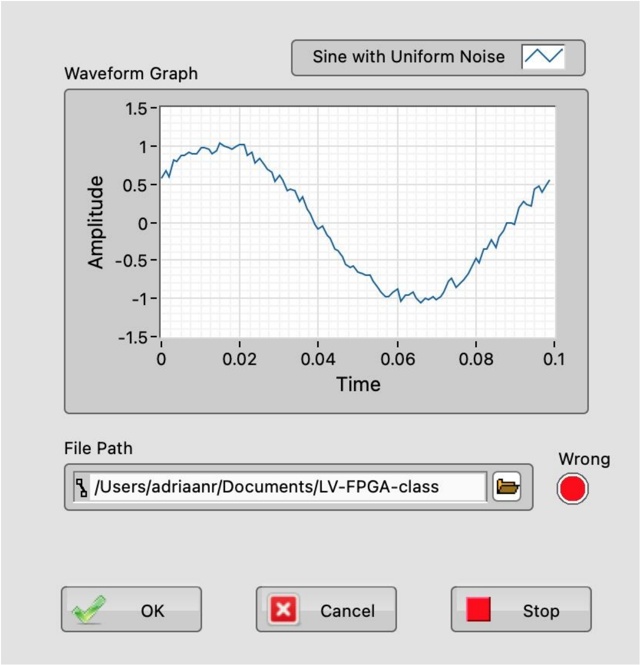


Front Panel Object Styles

Modern

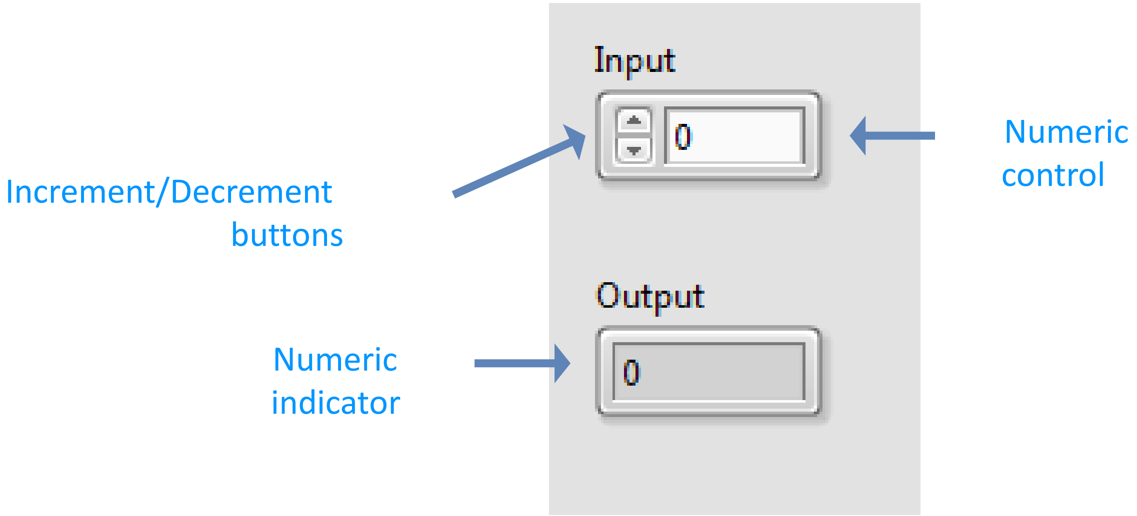


Silver



Numeric Controls and Indicators

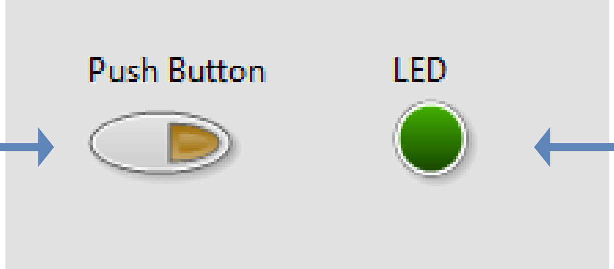
The numeric data in a control or indicator can represent numbers of various types, such as integer or floating-point.



Boolean Controls and Indicators

- The Boolean data type represents data that has only two options, such as True/False or On/Off.
- Use Boolean controls and indicators to enter and display Boolean (TRUE/FALSE) values.
- Boolean objects simulate switches, push buttons and LEDs.

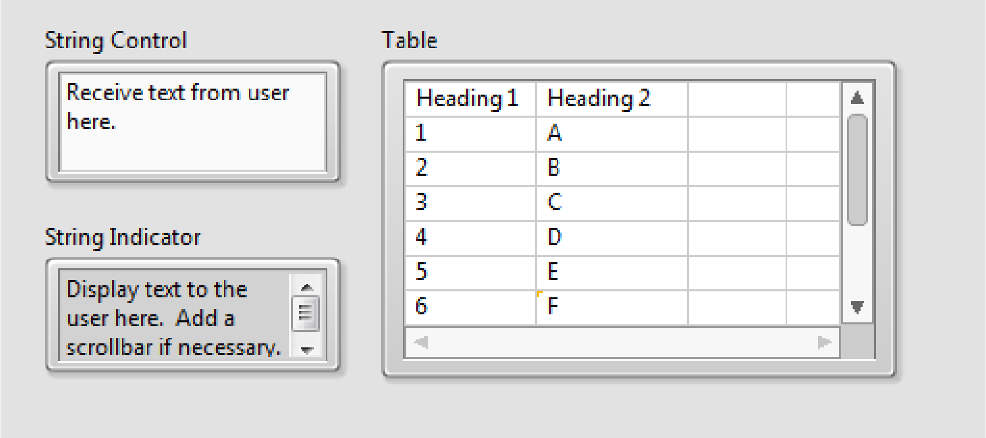
Boolean control



Boolean indicator

Strings

- The string data type is a sequence of ASCII characters.
- Use string controls to receive text from the user.
- Use string indicators to display text to the user.



E. Block Diagram

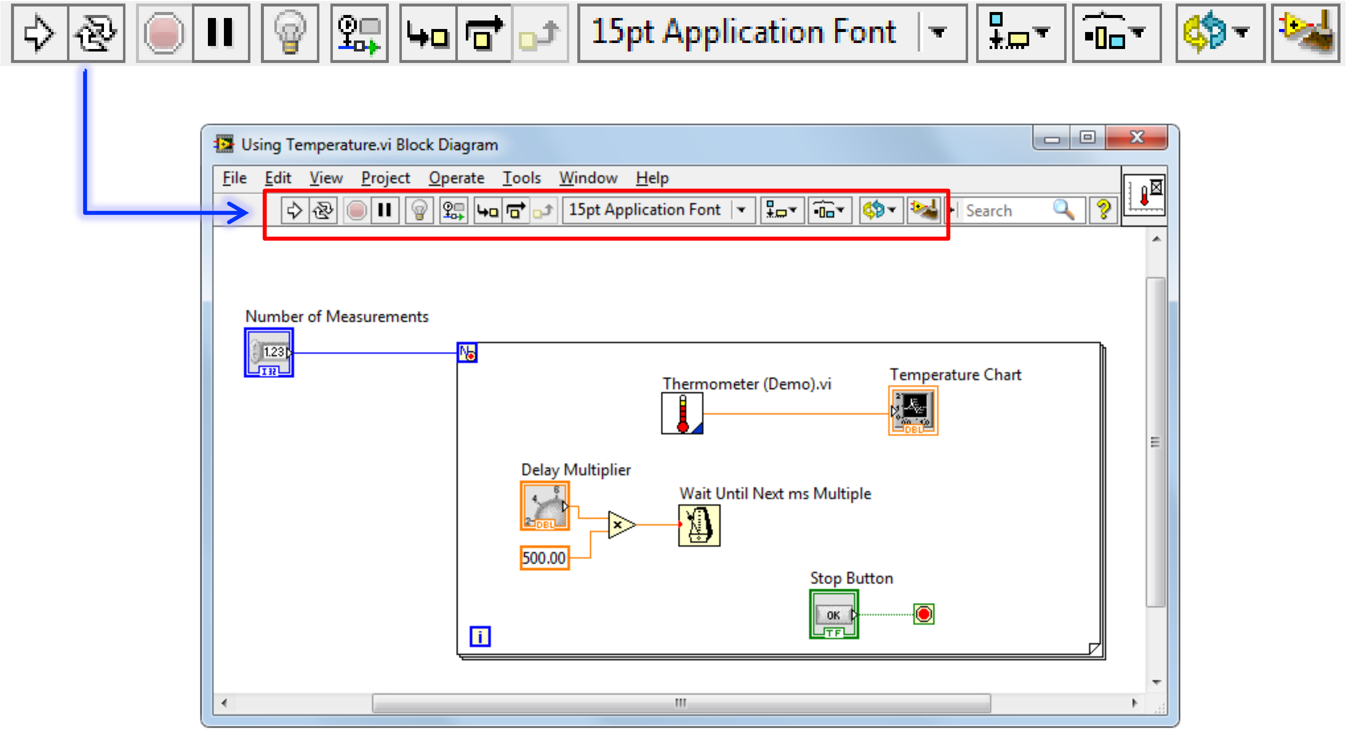
Terminals

Nodes

Wires

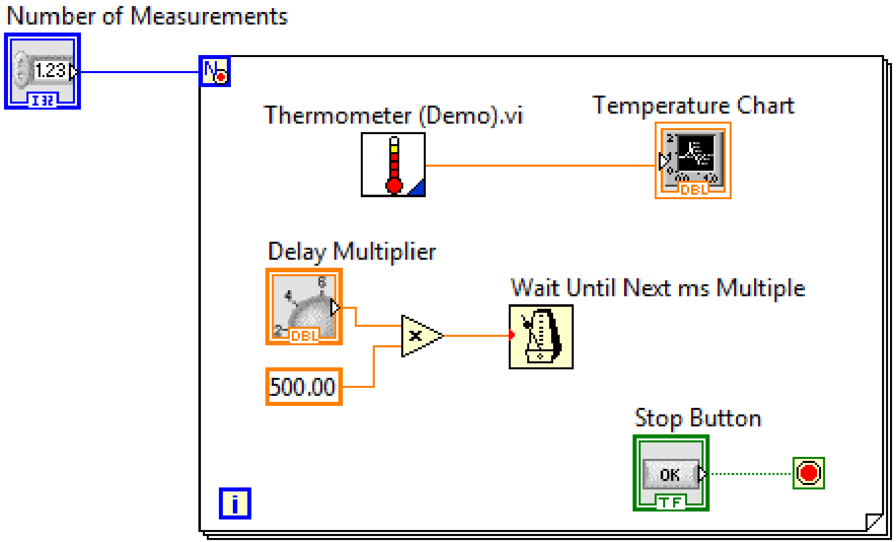
Help

Block Diagram

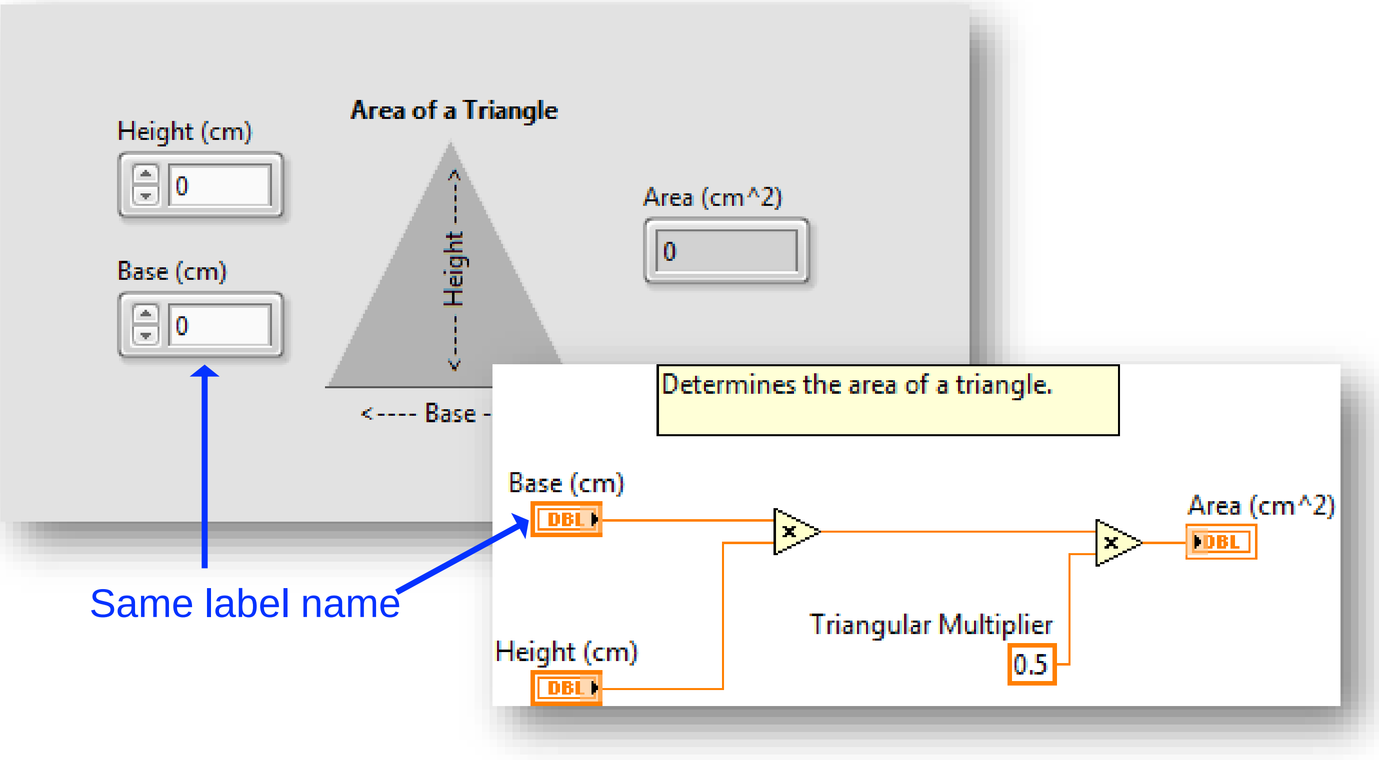


Block Diagram

- Block diagram items:
  - Terminals
  - Constants
  - Nodes
    - Functions
    - SubVIs
    - Structures
  - Wires
  - Free labels



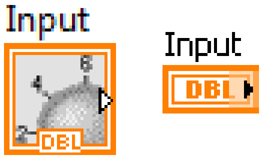
Terminals


Area of a Triangle
Height (cm)
0
Base (cm)
0
Height
<---- Base -
Area (cm^2)
0
Determines the area of a triangle.
Base (cm)
DBL
Height (cm)
DBL
Triangular Multiplier
0.5
Area (cm^2)
DBL
Same label name

Terminals for Front Panel Objects

- Terminals are:
  - Entry and exit ports that exchange information between the front panel and block diagram
  - Analogous to parameters in text-based programming languages

- Double-click a terminal to locate the corresponding front panel object

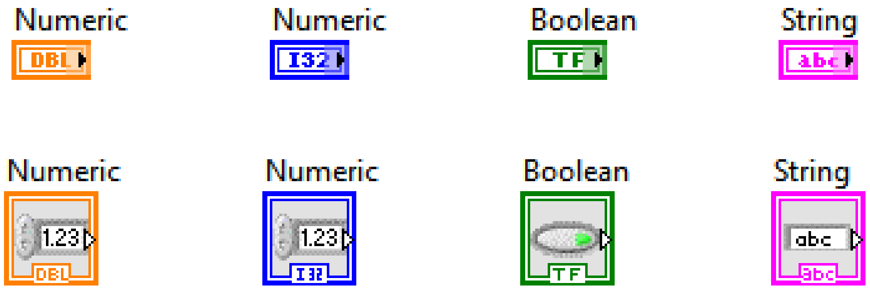


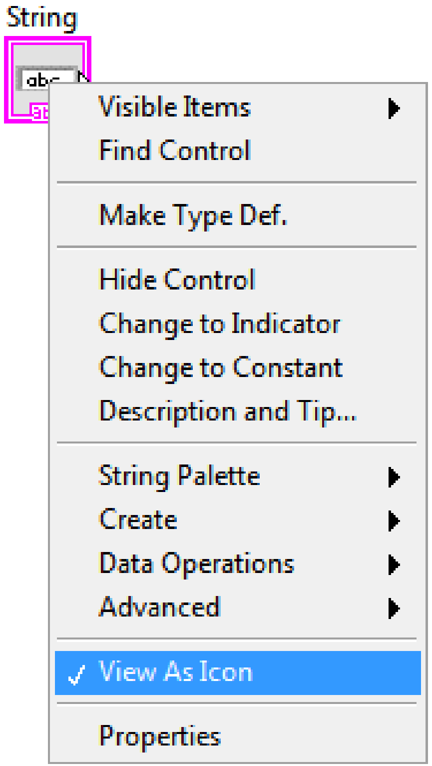
View Terminals as Icons

- By default, View as Icon option enabled.
- Deselect View as Icon for a more compact view.

Compact

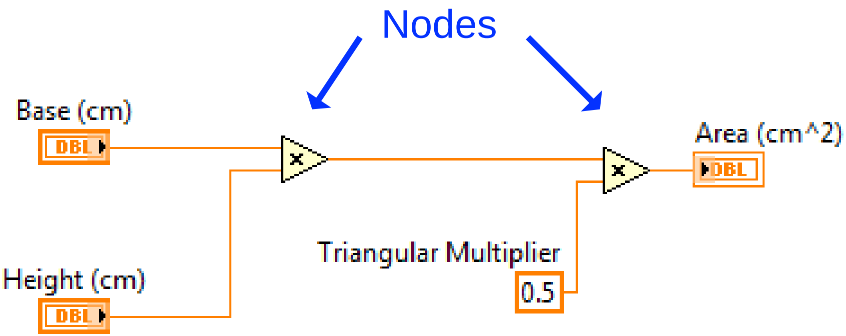
Icon



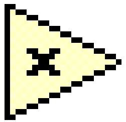


Nodes

Nodes are objects on the block diagram that have inputs and/or outputs and perform operations when a VI runs.



Function Nodes



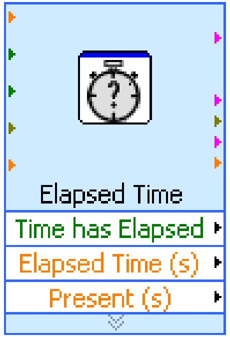
- Functions are:
  - Fundamental operating elements of LabVIEW.
  - Do not have front panels or block diagrams, but do have connector panes.
  - Have a pale yellow background on their icon.
- Functions do not open like VIs and subVIs.

SubVI Nodes

- SubVIs :
  - Are VIs that you use on the block diagram of another VI.
  - Have front panels and block diagrams.
  - Use the icon from the upper-right corner of the front panel as the icon that appears when you place the subVI on a block diagram.
- When you double-click a subVI, the front panel and block diagram open.
- Any VI has the potential to be used as a subVI.

Express VIs

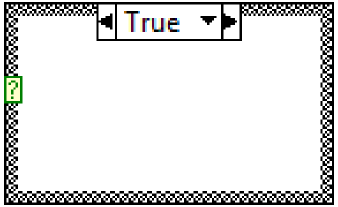
- Express VIs:
  - Are a special type of subVI.
  - Require minimal wiring because you configure them with GUI dialog boxes.
  - Save each configuration as a subVI.
- Icons for Express VIs appear on the block diagram as icons surrounded by a blue field.



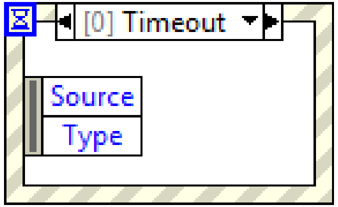
Structures

- Structures in LabVIEW have the form of frames

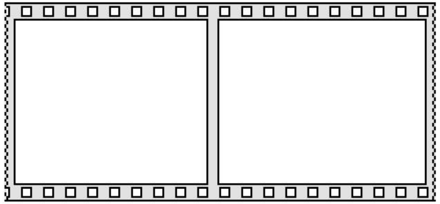
While loop



Case (if .. then)



Timed loop

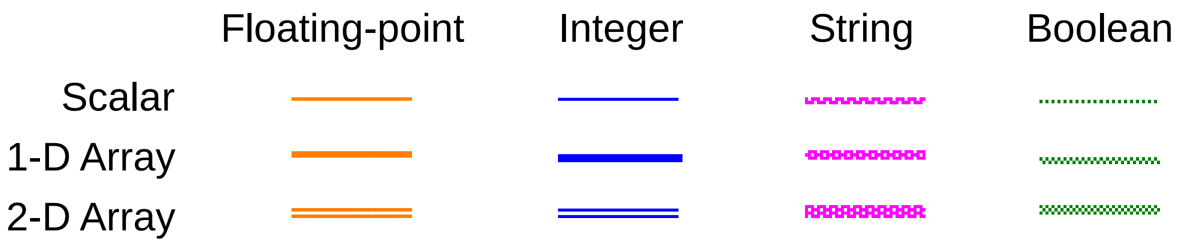


Sequence

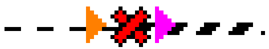
- Other nodes (functions, subVIs, more structures) can be inserted into the frames

Wires

- Wires transfer data between block diagram objects
- Wires are different colors, styles, and thicknesses, depending on their data types



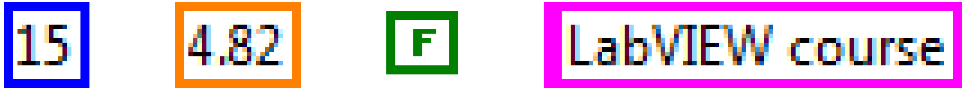
- A broken wire appears as a dashed black line with a red X in the middle



Constants

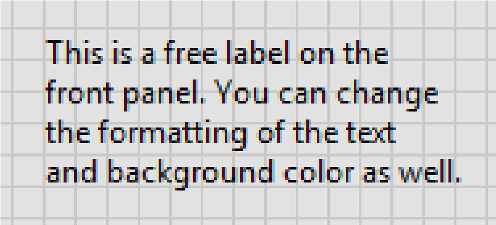
- Constants are the source of values just as control terminals, but their value is fixed in the code

- You can create a constant of each data type



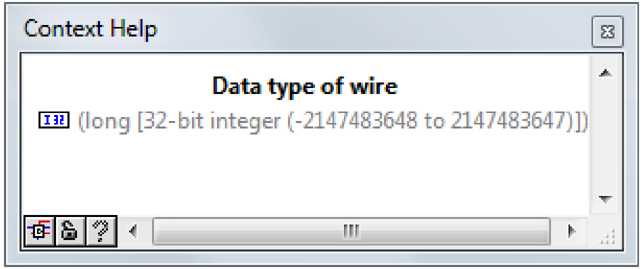
Free labels

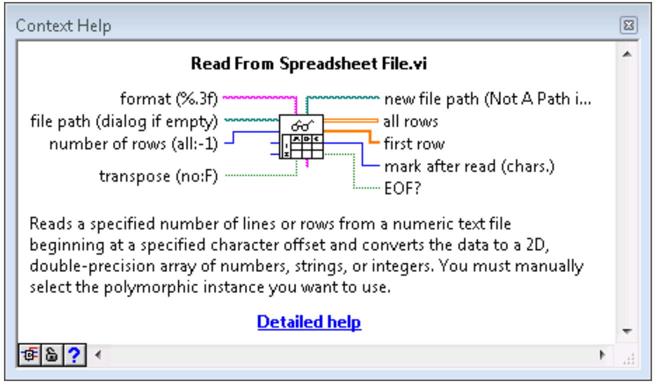
- A free label is a label (a text box) not attached to any object.
- Free labels can be put on the front panel or block diagram. They are created by double-clicking on empty space in the window
- They can serve as comments or instructions to the user of the application



Context Help

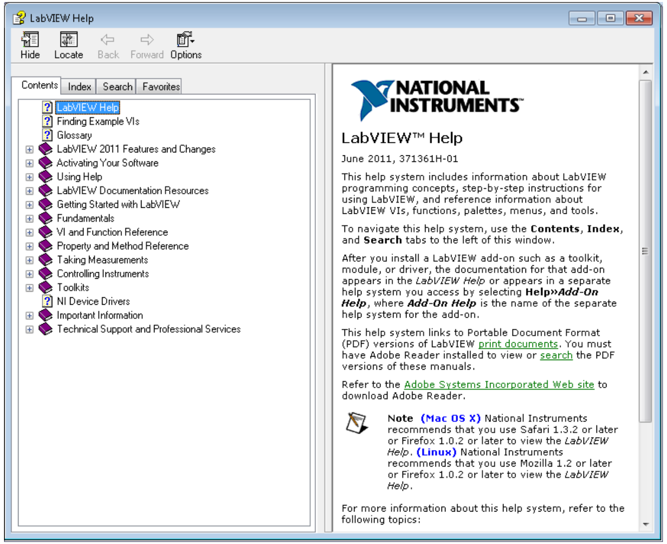
- Displays basic information about wires and nodes when you move the cursor over an object
- Can be shown or hidden in the following ways:
    - Select **Help»Show Context Help** from the LabVIEW menu
    - Press <Ctrl-H>
    - Click the following button on the toolbar:





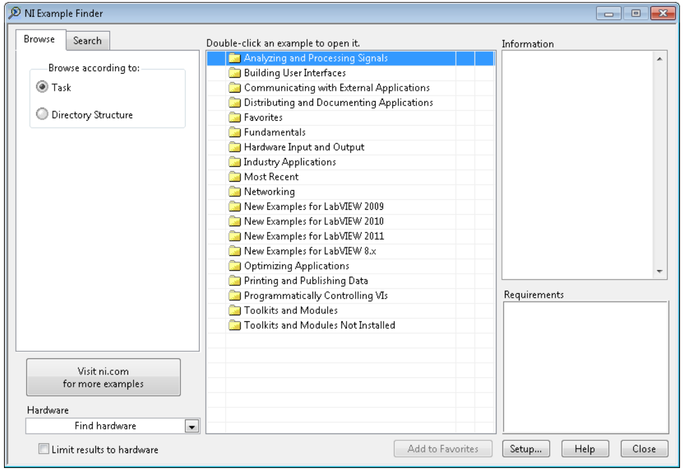
LabVIEW Help

- Contains detailed descriptions and instructions for most palettes, menus, tools, VIs, and functions.
- Can be accessed by:
  - Selecting Help» LabVIEW Help from the menu.
  - Clicking the Detailed help link in the Context Help window.
  - Right-clicking an object and selecting Help from the shortcut menu



Examples

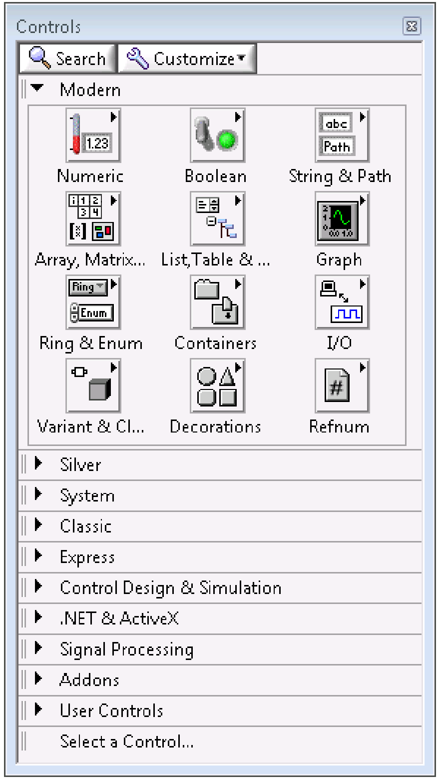
- LabVIEW includes hundreds of example VIs.
- Use NI Example Finder to browse and search installed examples
  - Select **Help»Find Examples** in the menu.



- Click the example buttons in *LabVIEW Help* topics

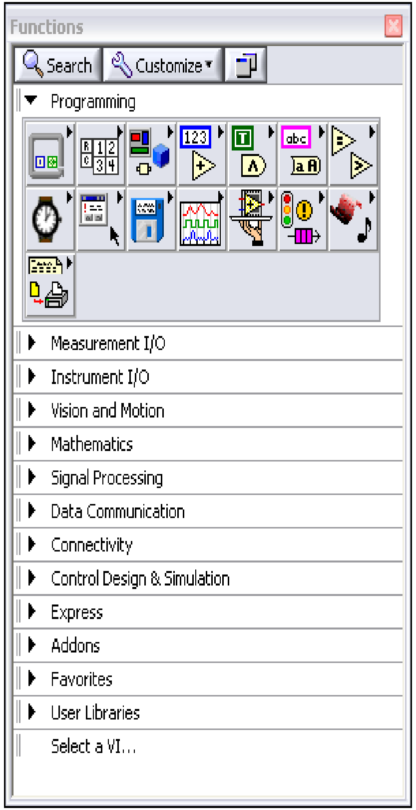
Controls Palette

- Contains the controls and indicators you use to create the front panel.
- Navigate the subpalettes or use the **Search** button to search the Controls palette.



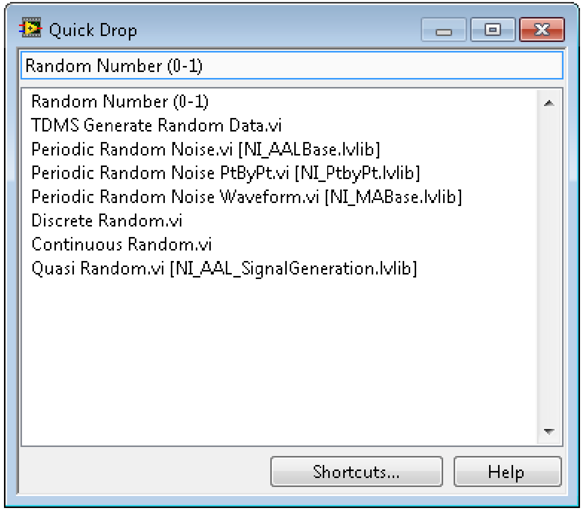
Functions Palette

- Contains the VIs, functions, and constants you use to create the block diagram.
- Navigate the subpalettes or use the **Search** button to search the Functions palette.



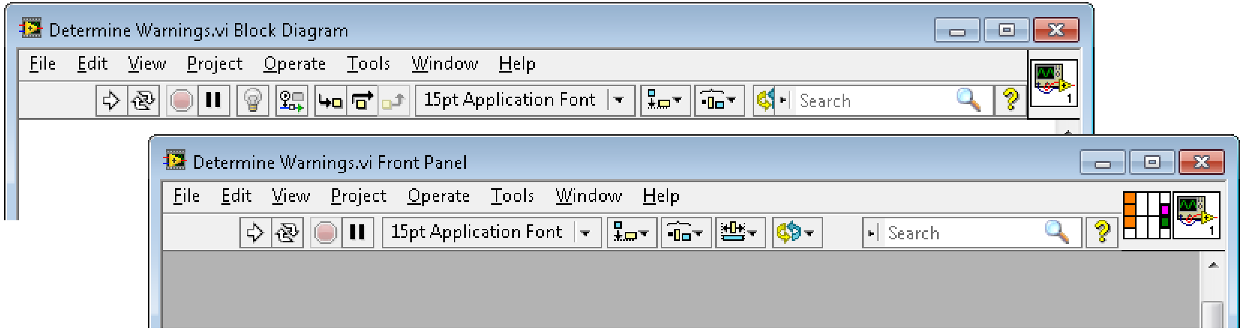
Searching with Quick Drop

- Lets you quickly find controls, functions, VIs, and other items by name.
- Press the <Ctrl-Space> keys to display the Quick Drop dialog box.



Global Search

Use the Search bar in the top right of the front panel and block diagram windows to search palettes, *LabVIEW Help*, and `ni.com`



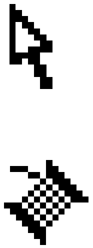
G. Selecting a Tool

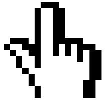
Selecting a Tool

Block Diagram Clean-Up

Selecting a Tool

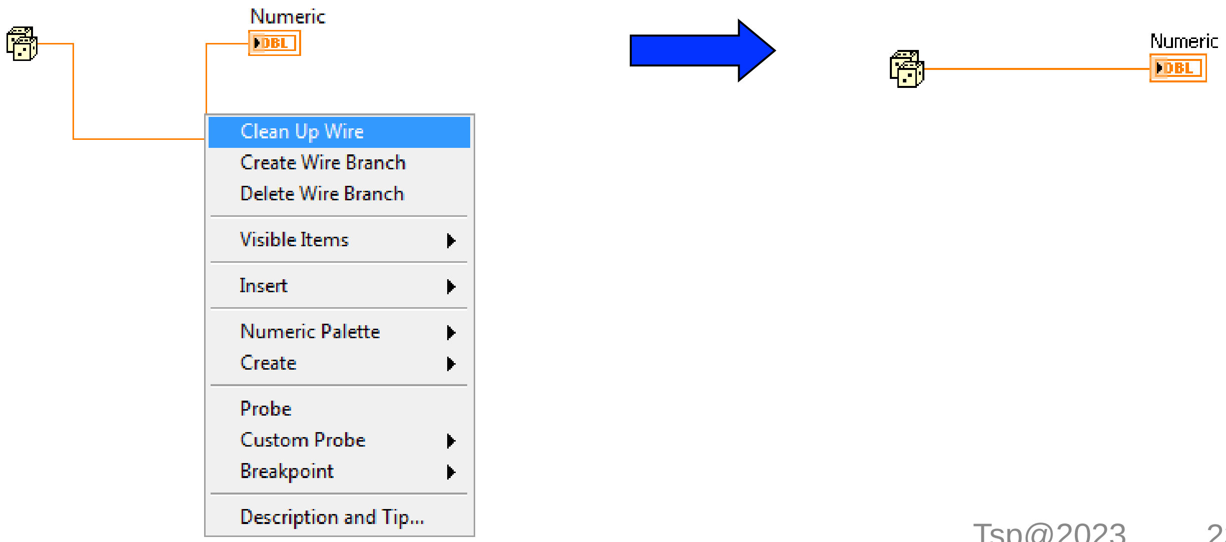
- A tool is a special operating mode of the mouse cursor.
- Create, modify, and debug VIs using the tools provided by LabVIEW.
- By default, LabVIEW automatically selects tools based on the context of the cursor.
- If you need more control, use the **Tools** palette to select a specific tool.
  - Select **View»Tools Palette** to open the **Tools** palette.



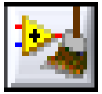


Wiring Tips

- Press <Ctrl-B> to delete broken wires.
- Right-click and select **Clean Up Wire** to reroute the wire.

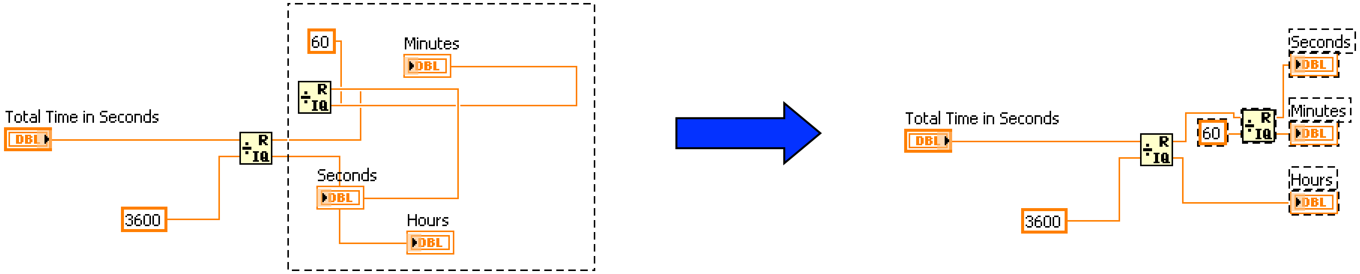


Wiring Tips – Clean Up Diagram



Use the Clean Up Diagram tool to reroute multiple wires and objects and to improve readability.

1. Select a section of your block diagram.
2. Click the Clean Up Diagram button on the block diagram toolbar (or press <Ctrl-U>).



Cloning Items

Clone an object in Windows using the following steps:

1. Select the Positioning tool
2. Press the <Ctrl> key while clicking an object
3. Drag the copy to new location

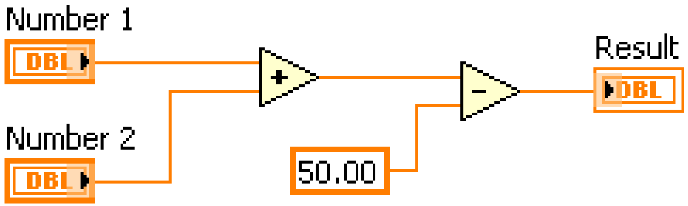
Note: Avoid cutting and pasting objects as this can impact related items. For example, cutting and pasting a block diagram terminal also moves the front panel object.

Setting options for the environment

- In **Tools»Options…** dialog box you can customize settings for the LabVIEW environment.
- Suggested changes:
  - Front Panel page
    - Set Control Style for New VIs to **Silver style**
  - Block Diagram page
    - Uncheck **Place front panel terminals as icons**
    - Configure **Block Diagram Cleanup** to customize your block diagram

H. Dataflow

Dataflow



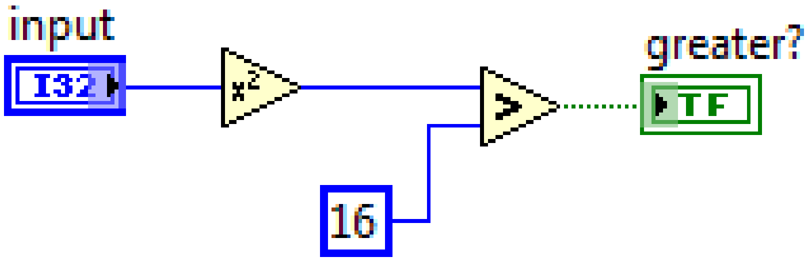
LabVIEW follows a dataflow model for running VIs.

- A node executes only when data are available at all of its required input terminals
- A node supplies data to the output terminals only when the node finishes execution

Dataflow – Quiz

What are the nodes in this fragment of code?

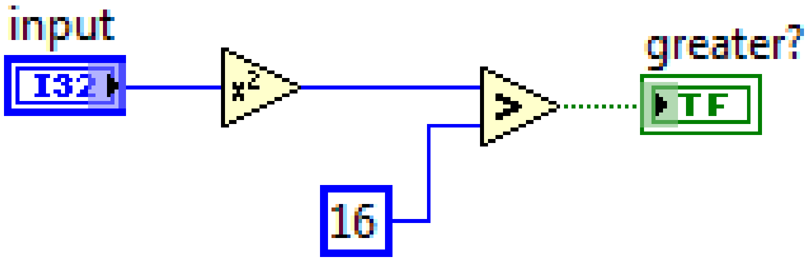
Which node executes first?



Dataflow – Quiz Answer

There are two nodes: „square” and „greater than?” functions

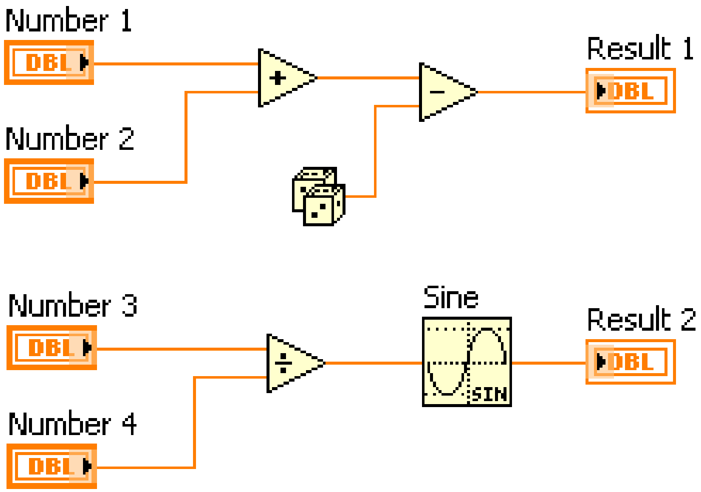
Square executes first



Dataflow – Quiz

Which node executes first?

a) Add
b) Subtract
c) Random Number
d) Divide
e) Sine



Dataflow – Quiz Answer

No single correct answer.

Which node executes first?

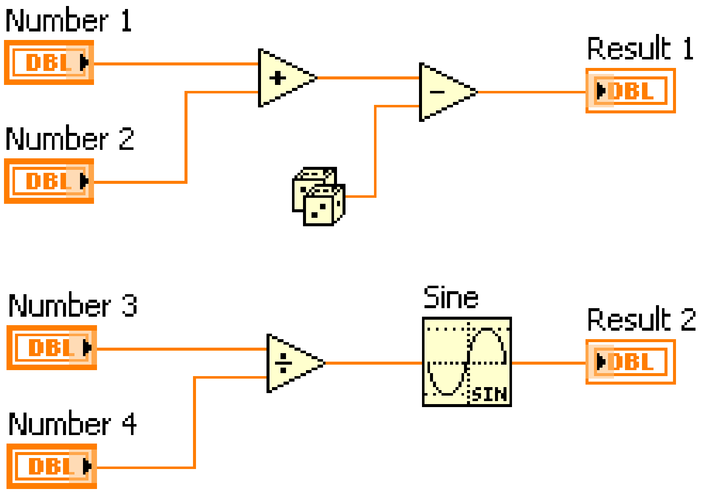
a) Add – **Possibly**

b) Subtract – **Definitely not**

c) Random Number – **Possibly**

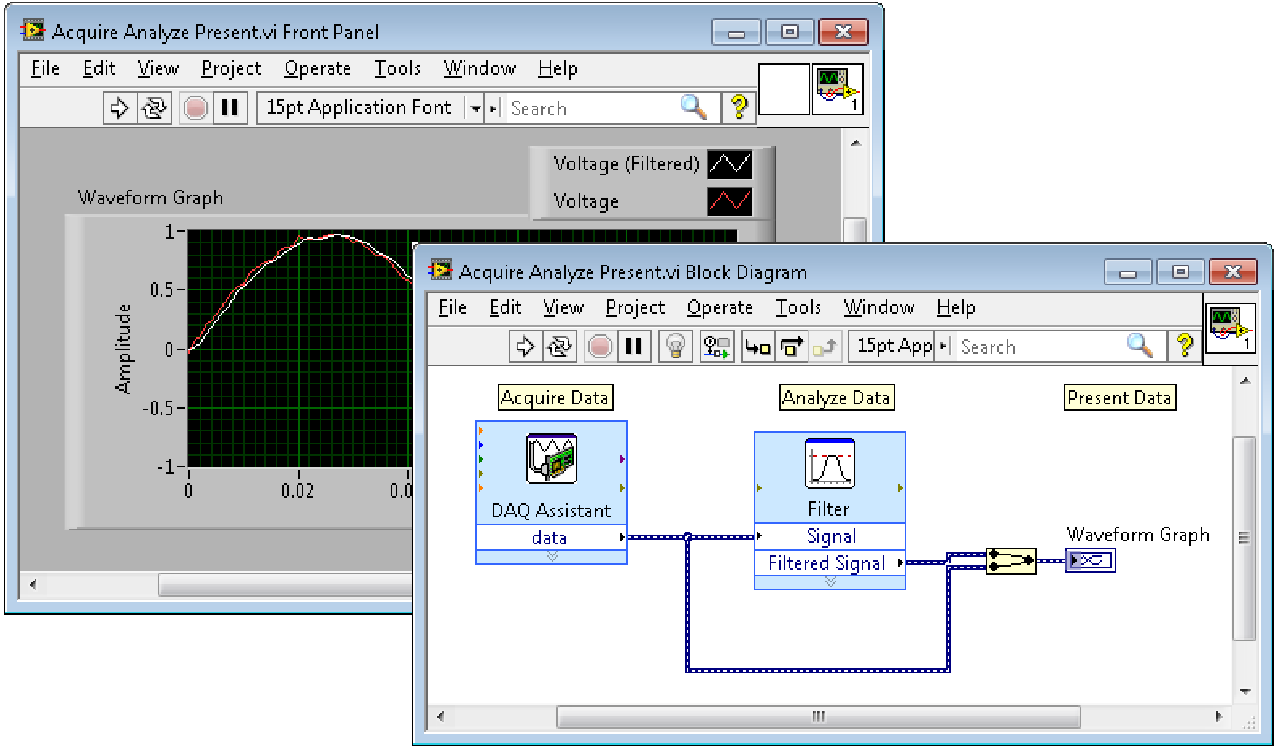
d) Divide – **Possibly**

e) Sine – **Definitely not**



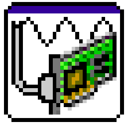
I. Building a Simple VI

Building a Simple VI

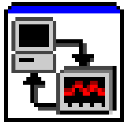


Acquire Express VIs

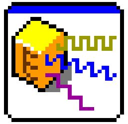
- DAQ Assistant Express VI



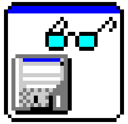
- Instrument I/O Assistant Express VI



- Simulate Signal Express VI

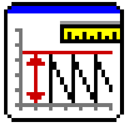


- Read from Measurement File Express VI

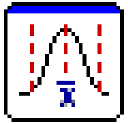


Analyze Express VIs

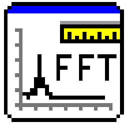
- Amplitude and Level Measurements Express VI



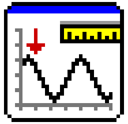
- Statistics Express VI



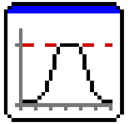
- Spectral Measurements Express VI



- Tone Measurements Express VI

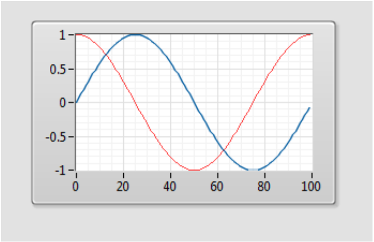


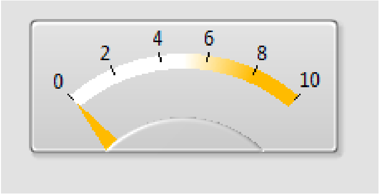
- Filter Express VI



Present Express VIs and Indicators

- Display Message Express VI
- Play Waveform Express VI
- Report Express VI
- Write to Measurement File Express VI
- DIAdem Report Express VI

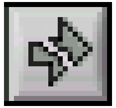


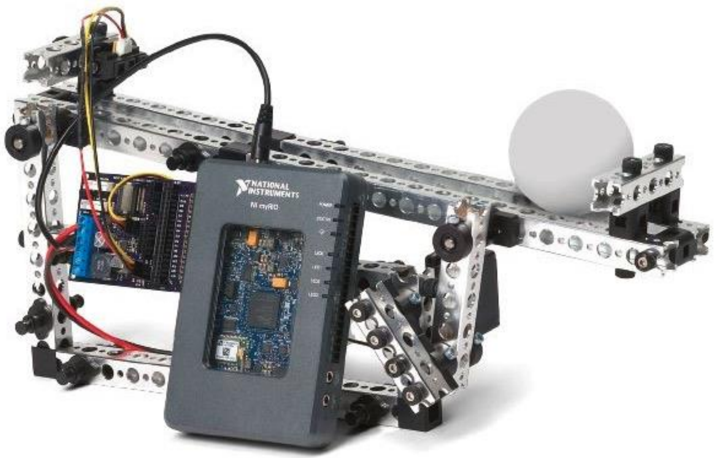


Building and Running a VI

1. Place Express VI on the block diagram
2. Configure the dialog box that opens
3. Wire Express VIs together
4. Save and run the VI

The **Run** button appears broken when the VI you are creating or editing contains errors





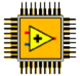
LabVIEW FPGA hands-on

part 2

Adriaan Rijllart
Odd Øyvind Andreassen
CERN

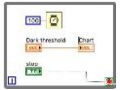
Content of LabVIEW FPGA hands-on 2

- A few more LabVIEW basics



- Introduction to LabVIEW FPGA

- Overview of NI myRIO



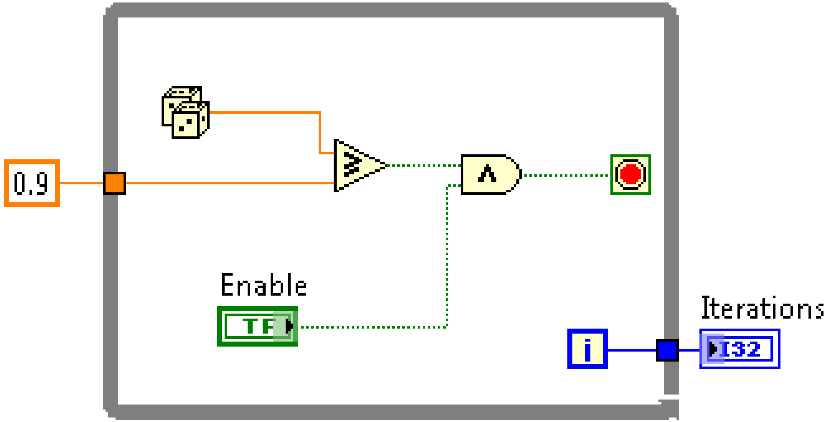
- Exercises

- Resources and Next Steps

A few more LabVIEW basics

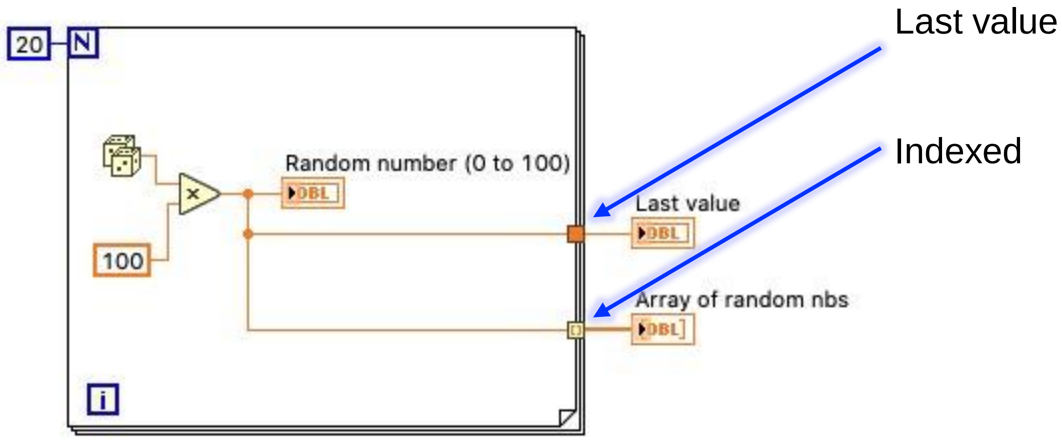
While loop tunnel

- Tunnels transfer data into and out of structures.
- When a tunnel passes data into a loop, the loop
executes only after data arrive at the tunnel (at all tunnels, if there is more than one)
- Data pass out of a loop after the loop terminates



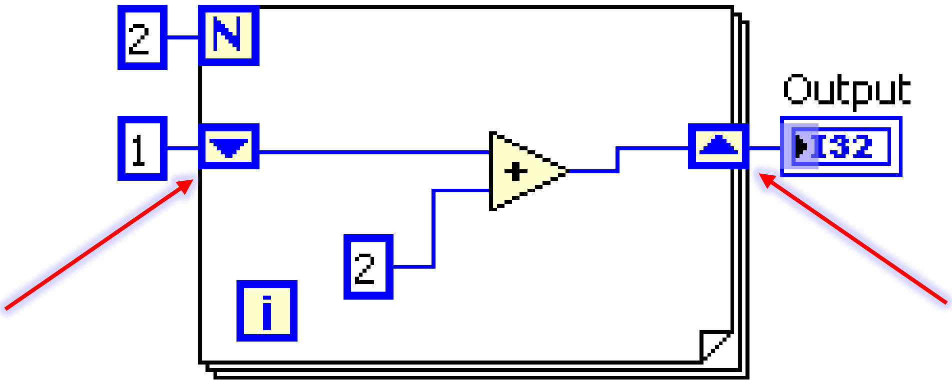
For loop

- The value in the count terminal (an input terminal) indicates how many times to repeat the subdiagram



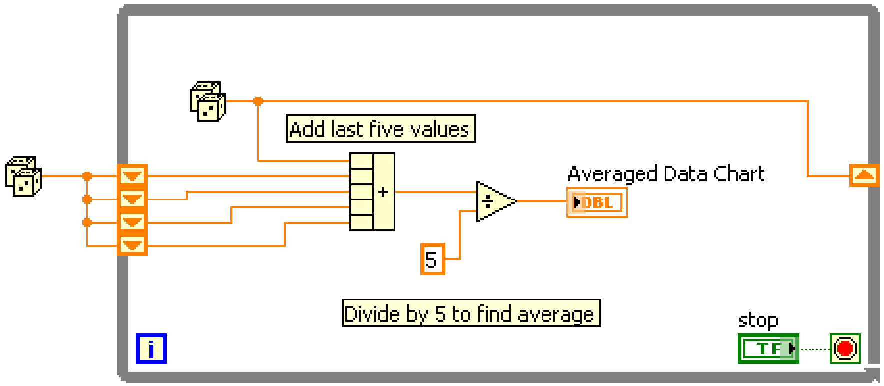
Shift register

- When programming with loops, you often need to know the values of data from previous iterations of the loop
- Shift registers **transfer values** from one loop iteration to the next



Shift register, multiple values

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



Introduction to FPGA

Why Are FPGAs Useful?

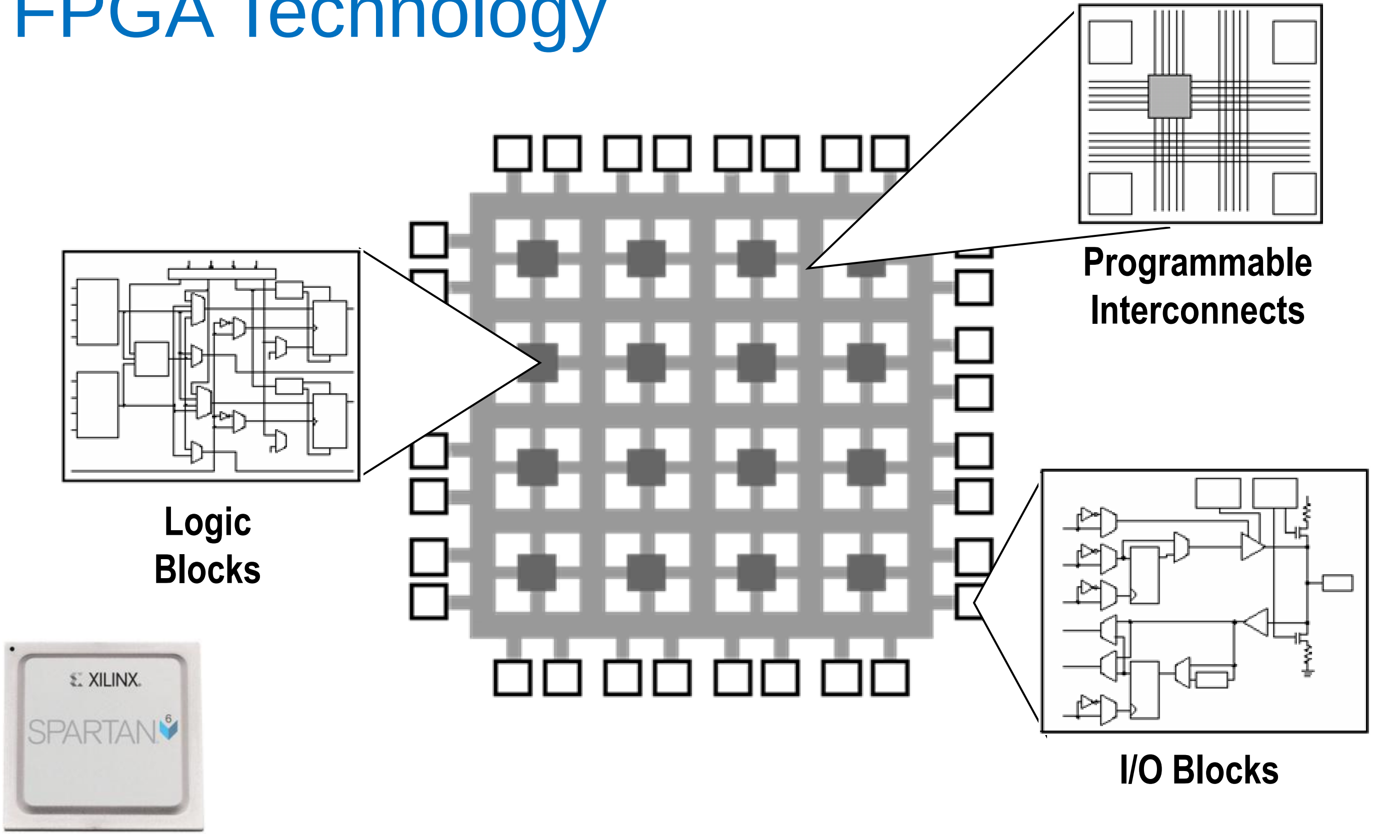
- **True Parallelism**
  Provides parallel tasks and pipelining

- **High Reliability**
  Designs become a custom circuit

- **High Determinism**
  Runs algorithms at deterministic rates down to 25 ns (faster in many cases)

- **Reconfigurable**
  Create new and alter existing tasks easily

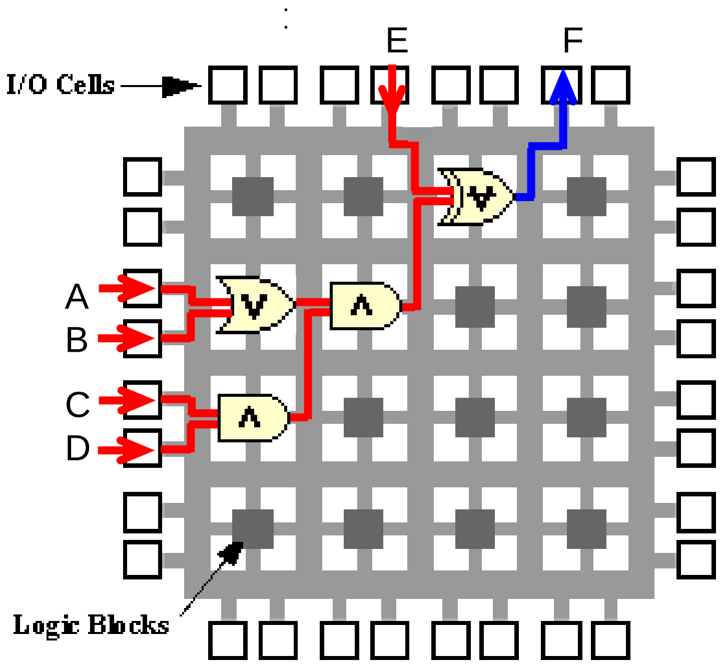
FPGA Technology



FPGAs are Dataflow Systems

Implementing Logic on FPGA:

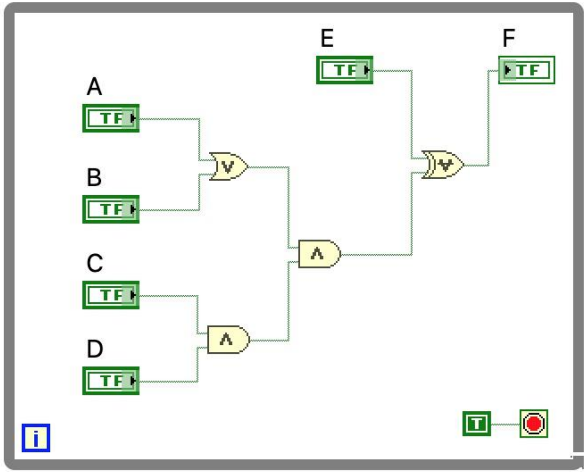
$$F = \{(A+B)CD\} \oplus E$$

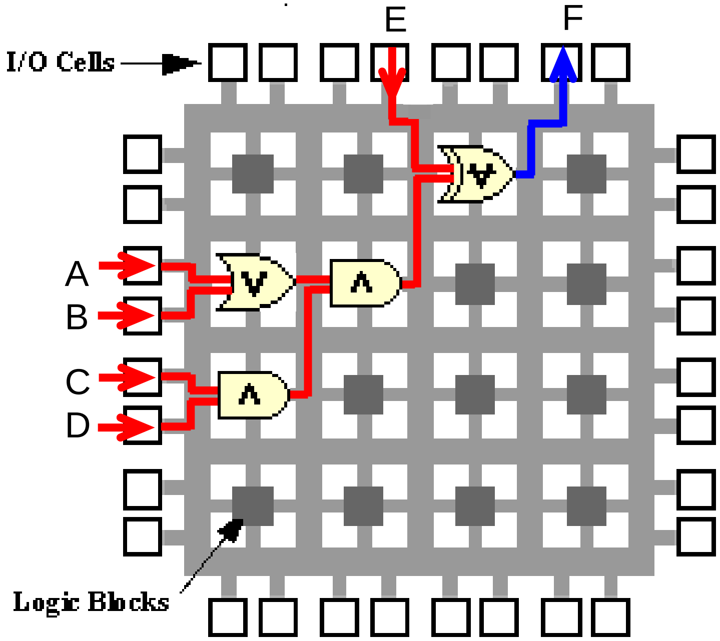


FPGAs are Dataflow Systems

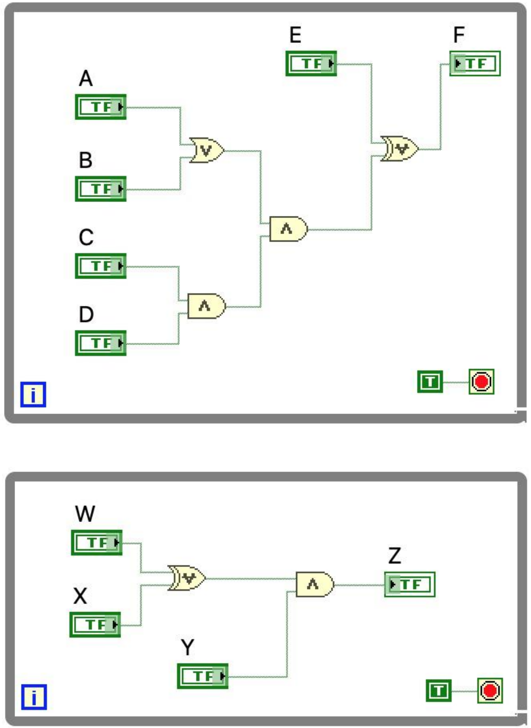
Implementing Logic on FPGA: F = {(A+B)CD} ⊕ E

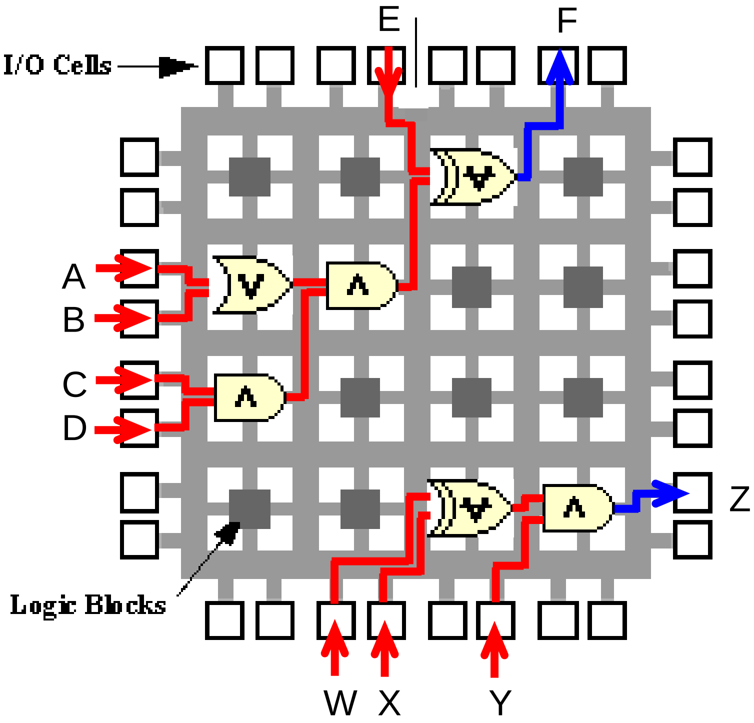
LabVIEW FPGA Code



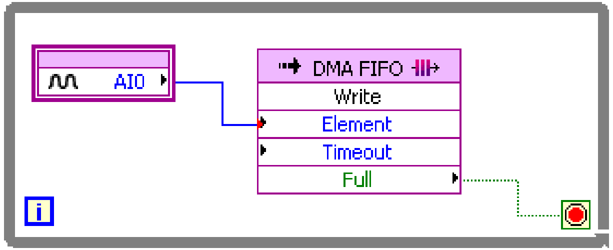


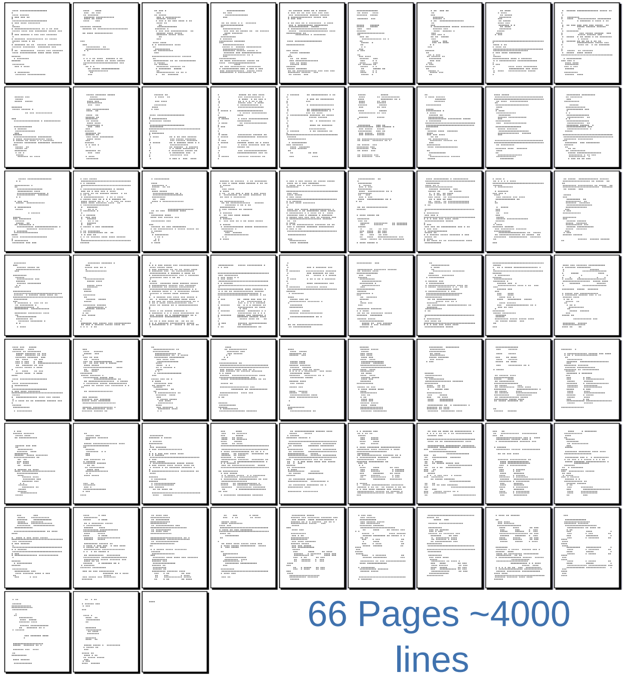
FPGAs are Parallel Dataflow Systems





LabVIEW FPGA vs. VHDL

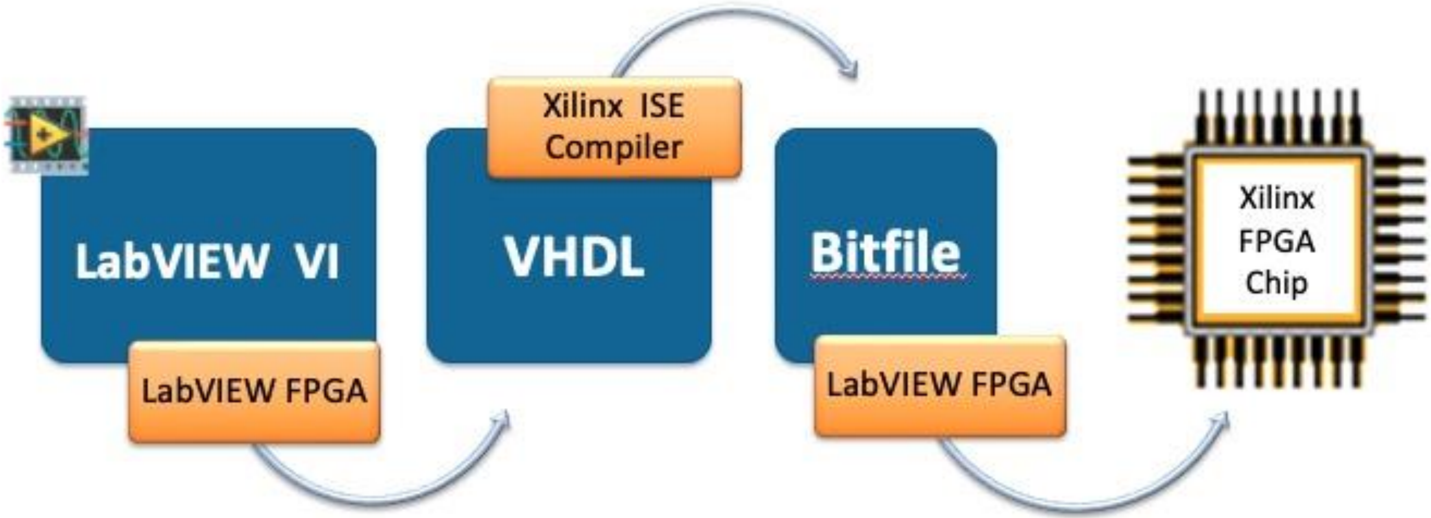




66 Pages ~4000 lines

I/O with DMA

LabVIEW FPGA: How does it work?

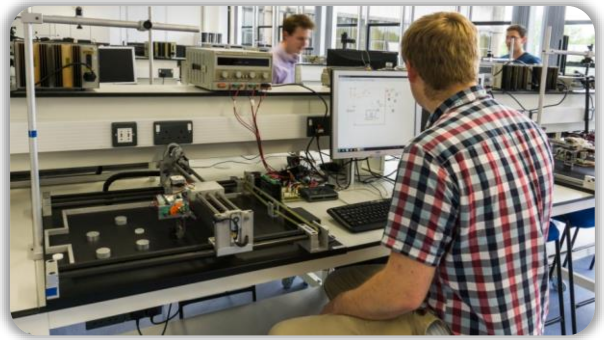


Robotic Table Football

Revolutionising Mechatronics Education

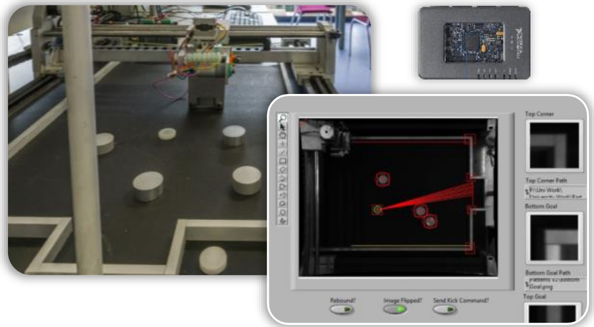
Loughborough University

The Challenge



Students struggled to realise their innovations using textual programming, due to unintuitive syntax and complex hardware integration. Following many research successes, Loughborough wanted to incorporate LabVIEW into their refined Mechatronic module

The Solution

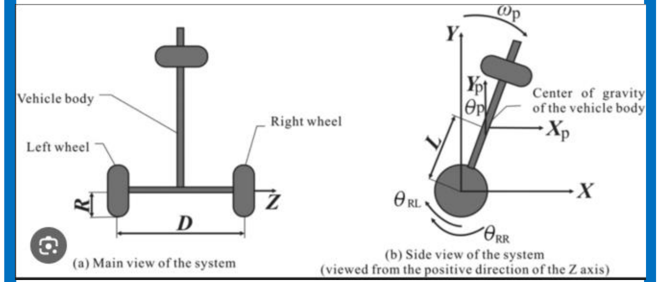


Using **LabVIEW** and **myRIO** to develop the Robotic Table Football challenge. This practical approach to teaching *mechatronic systems integration* resulted in a marked increase in student engagement, improved grades and the best system implementations to date.

Robotic Table Football student projects

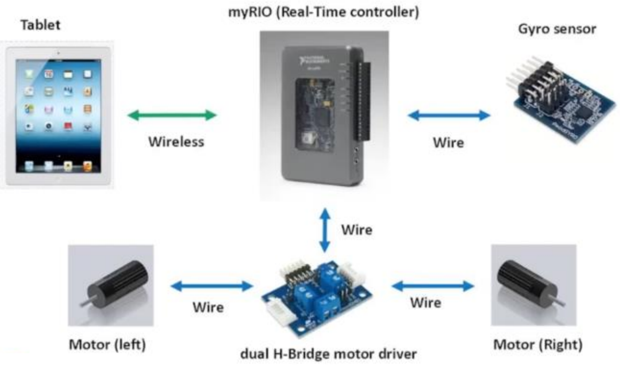
Self-balancing robot

The Challenge



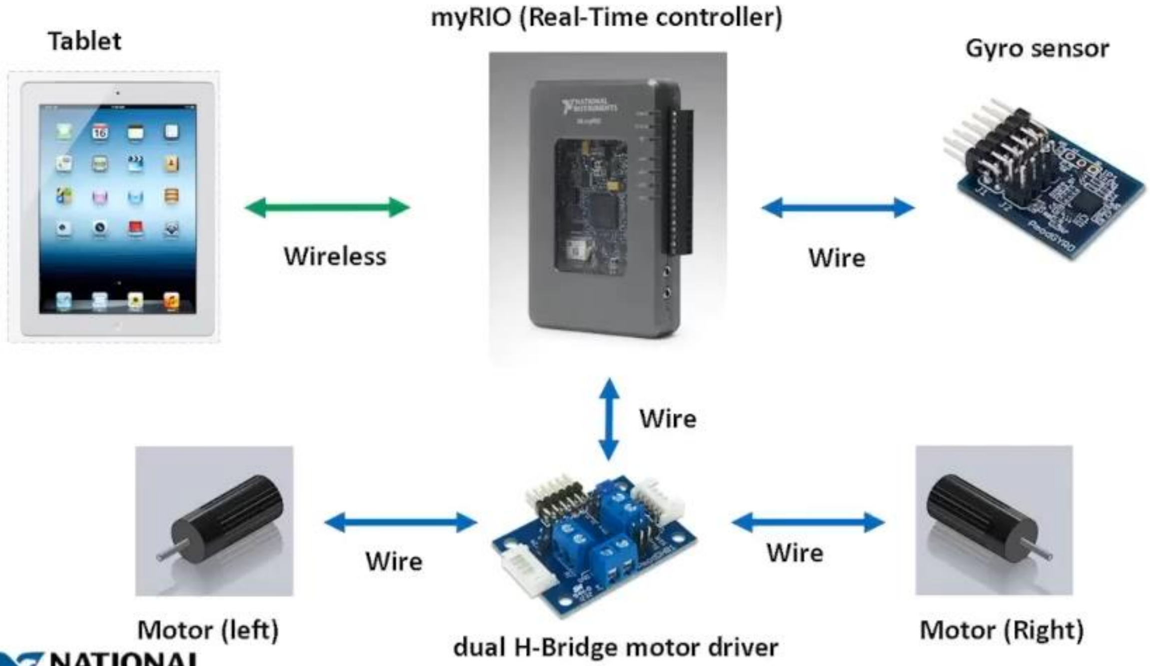
Make a self-balancing robot that can follow a track and stay up also when meeting obstacles

The Solution



Using **LabVIEW** and **myRIO** it's fun to implement

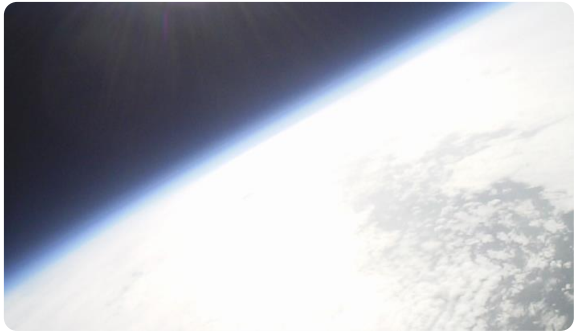
Self-balancing robot



Orseus

myRIO in Space

The Challenge



Developing an embedded system which operates under low pressure and temperature conditions - **space**. The system must carry out various experiments, including the study of solar radiation and atmospheric pollution

The Solution

Using **myRIO** to control all on-board sensors and experimental equipment in a high altitude balloon, from the launch to the landing with real time monitoring and post processing.

Student Design Contest Winner 2014

Sepios, the Omnidirectional Cuttlefish Robot

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

The Challenge

Creating a nautical robot driven by cuttlefish inspired fins to study this unique propulsion mechanism and its advantages

The Solution

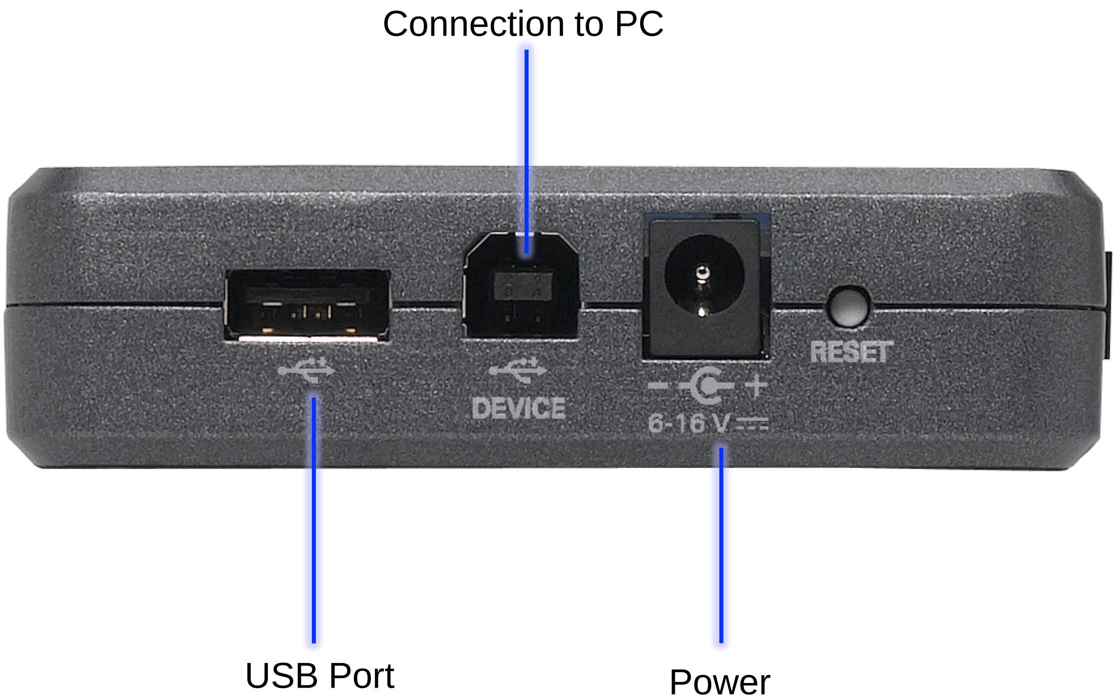
A four-finned robot, each fin equipped with nine servo motors to generate waves of various shapes and perform any conceivable manoeuver. All this is coordinated by a single NI myRIO at the heart of the drone.

NI myRIO Product Overview: Front View

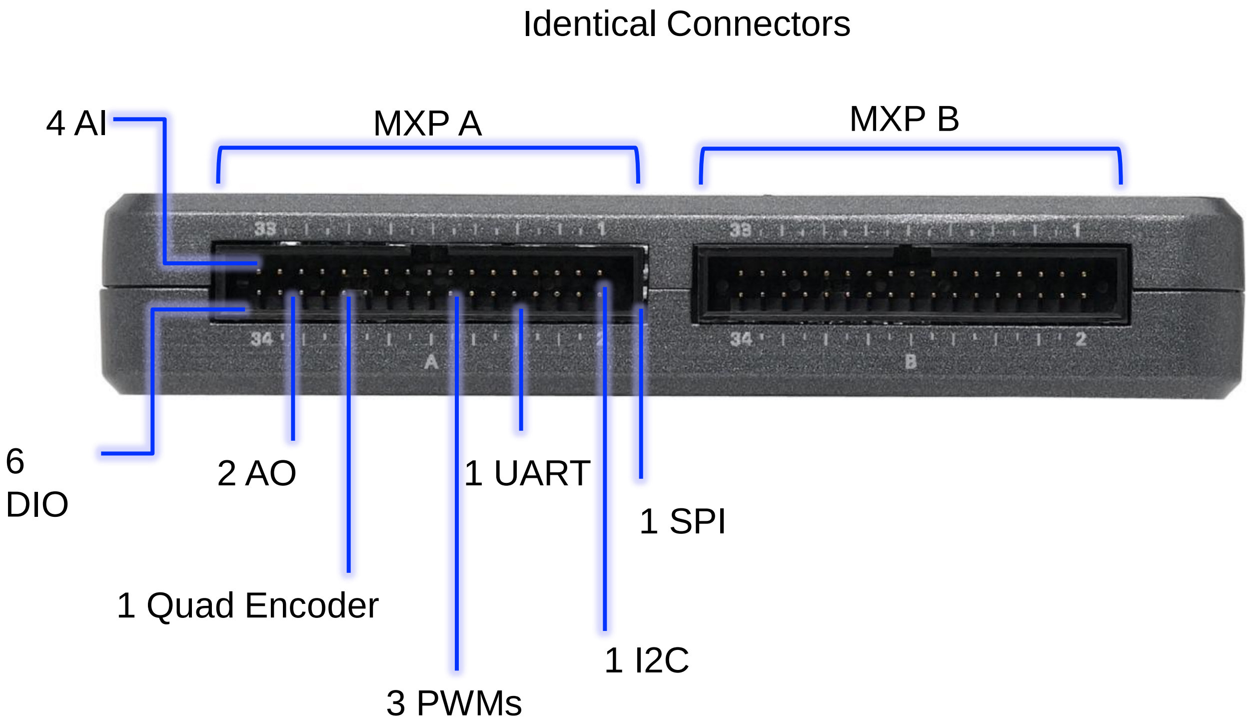
User Defined LEDs

Built-in Accelerometer

Top View



NI myRIO Expansion Port (MXP)

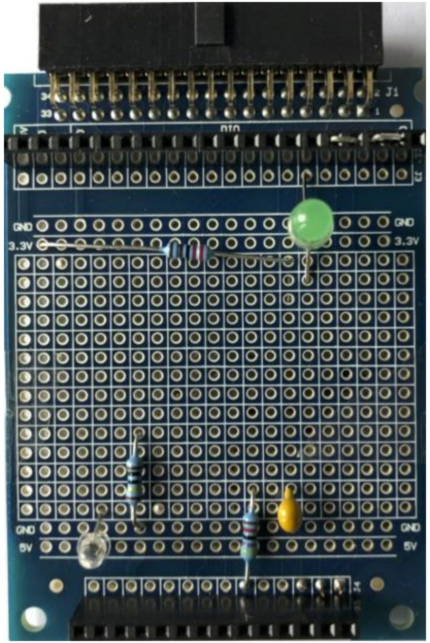


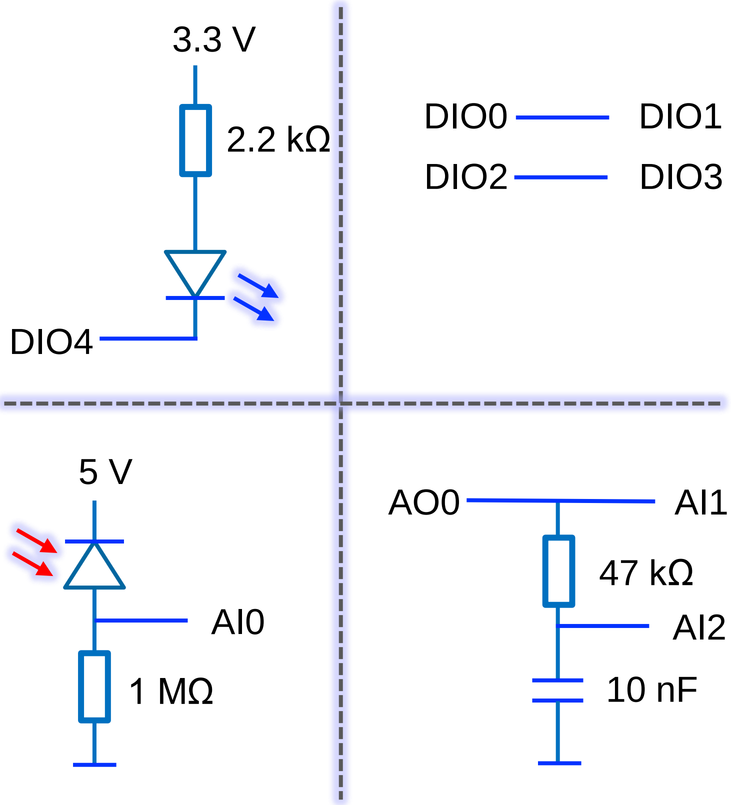
NI miniSystems Port (MSP)

Audio in/out more analog and digital I/O

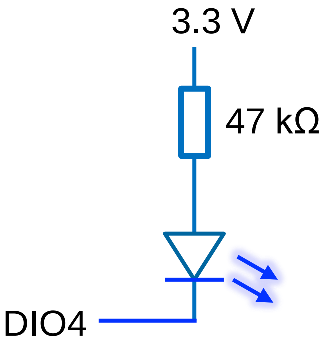
myRIO exercise board

Exercise board

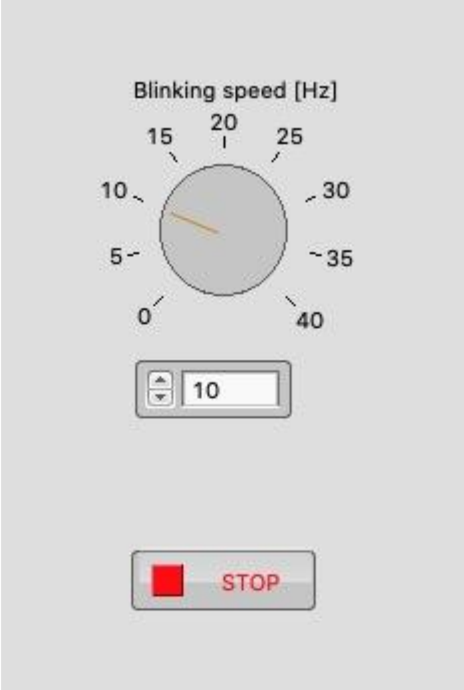




Exercise 1 Blinking LED

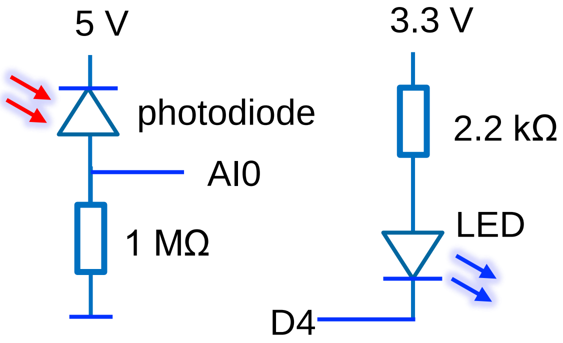


Make the LED blink with a controllable speed from 1 to 40 Hz



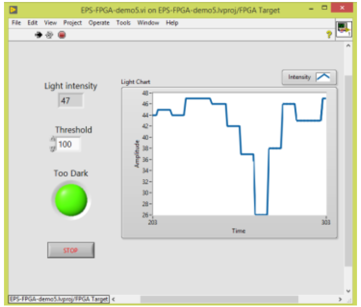
Question:
At what frequency you don’t see the blinking anymore?

Exercise 2 Switch on when it’s dark



Switch on the LED when the photodiode signal is below the threshold 100 (arbitrary units)

- Plot the photodiode signal in a chart
- The threshold value should be set using a control
- Remember the LED is on when D4 is False

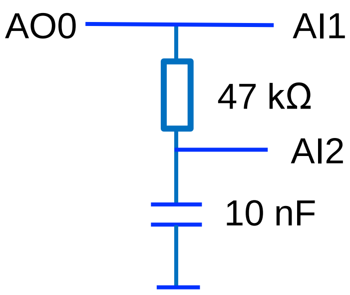


To test, block the light to the photodiode or increase light using your mobile phone

Question:

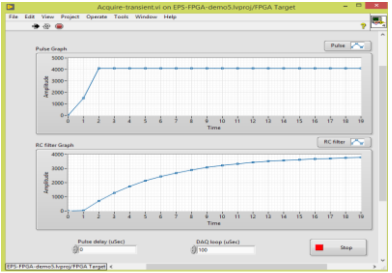
- What would happen when the photodiode would pick up the LED light?

Exercise 3 Acquire transient



Generate a step function from 0 – 5 V (int. value 4095)
Acquire step function signal and response of RC circuit

- Once per second
- Generate output voltage from 0 to 5 V (and reverse)
- Acquire both AI1 and AI2 signals using 20 points
- Show both in a graph
- RC value is 470 µs
- Set DAQ loop time (with a control) to 100 µs



Questions:
- Is the step function (AI1) really a step?
- What do you see when changing the DAQ loop time (both AI1 and 2)?

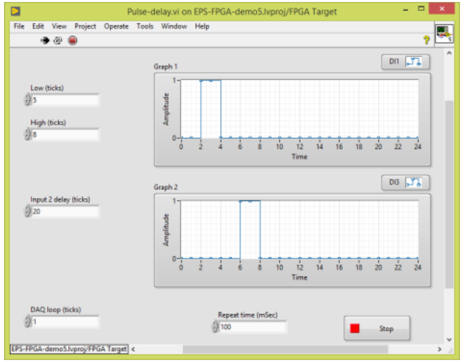
Exercise 4 Pulse delay

DIO0 → DIO1

DIO2 → DIO3

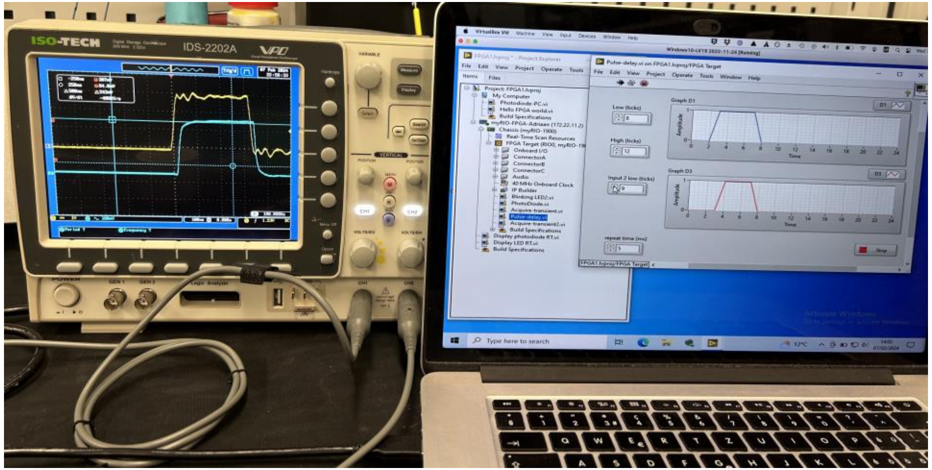
Generate short pulse on D0 and D2 (low – high – low)
Make a separate control for low D0 and D2 (using ticks)
Acquire 20 points on D1 and D3 with 1 tick loop delay
Repeat at 100 ms (10 Hz)

- Control for low time of pulse (4 ticks)
- Control for high time of pulse (8 ticks)
- Control for DAQ loop (1 tick)
- Graph D1
- Graph D3

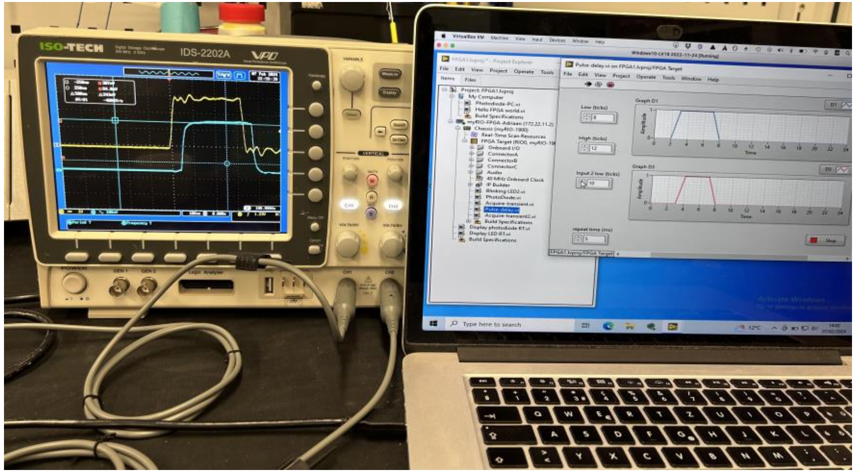


Questions:
- What do you see when changing the high and low values?
- Can you explain?

Exercise 4 Pulse delay, measured with scope

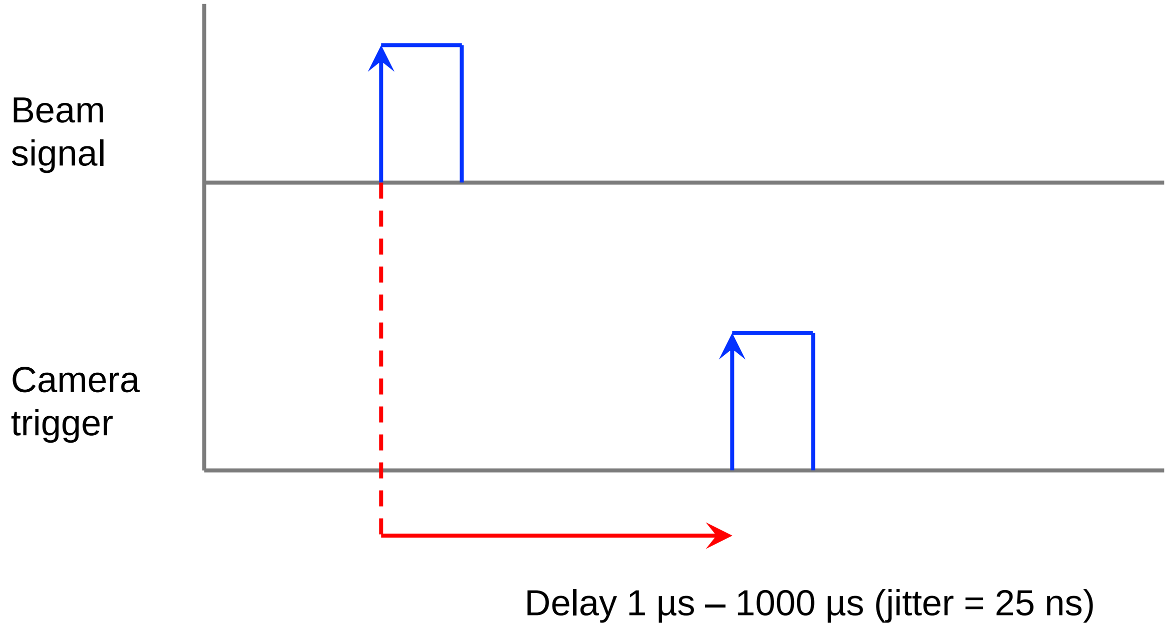


Lower pulse delayed by 25 ns

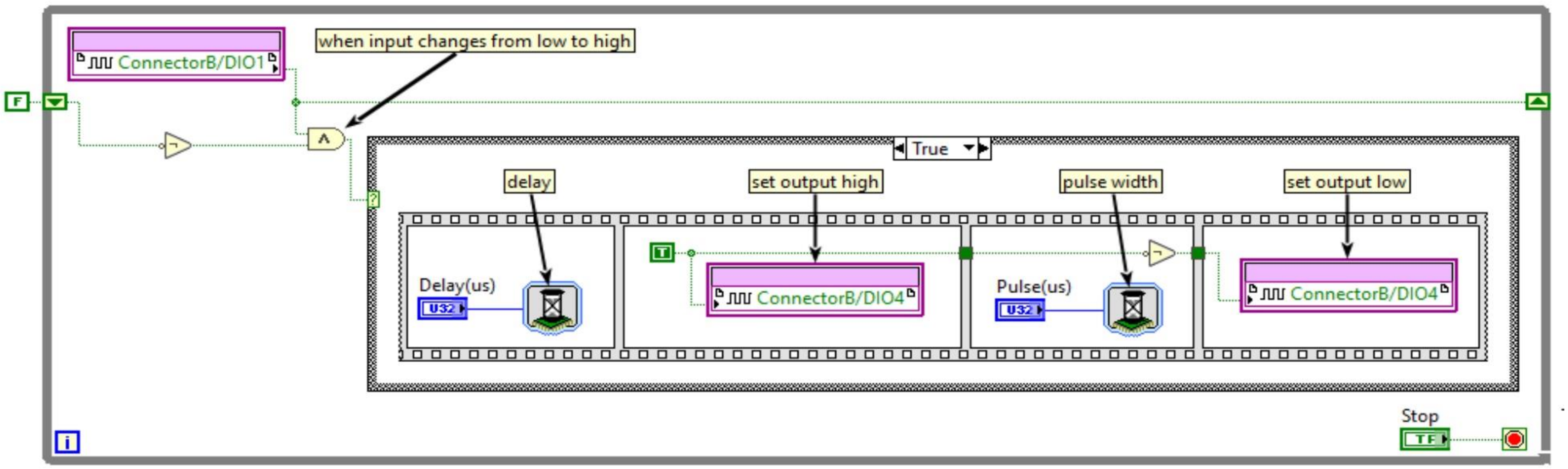


Lower pulse delayed by 50 ns

Task: Generate a trigger with programmable delay

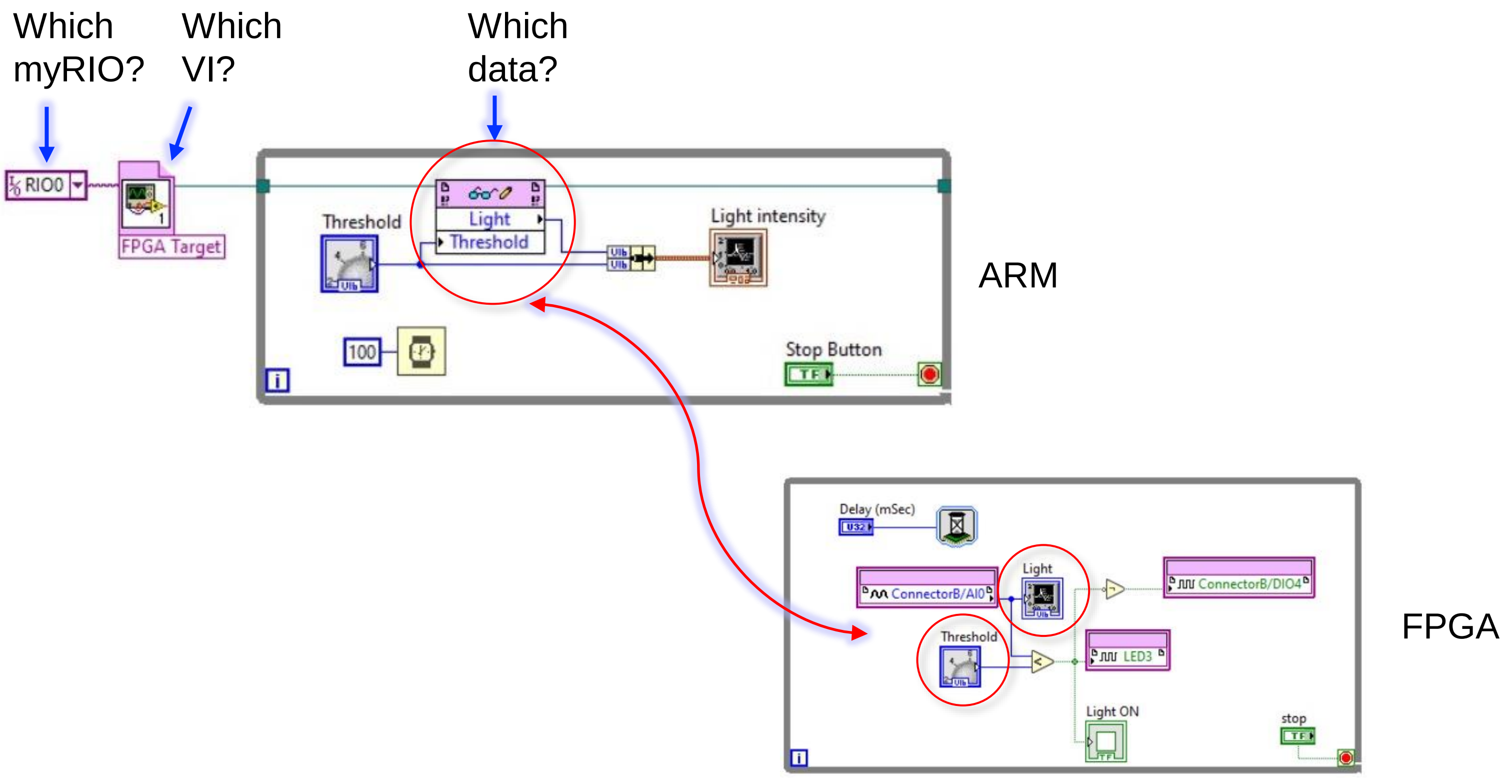


Trigger with programmable delay diagram

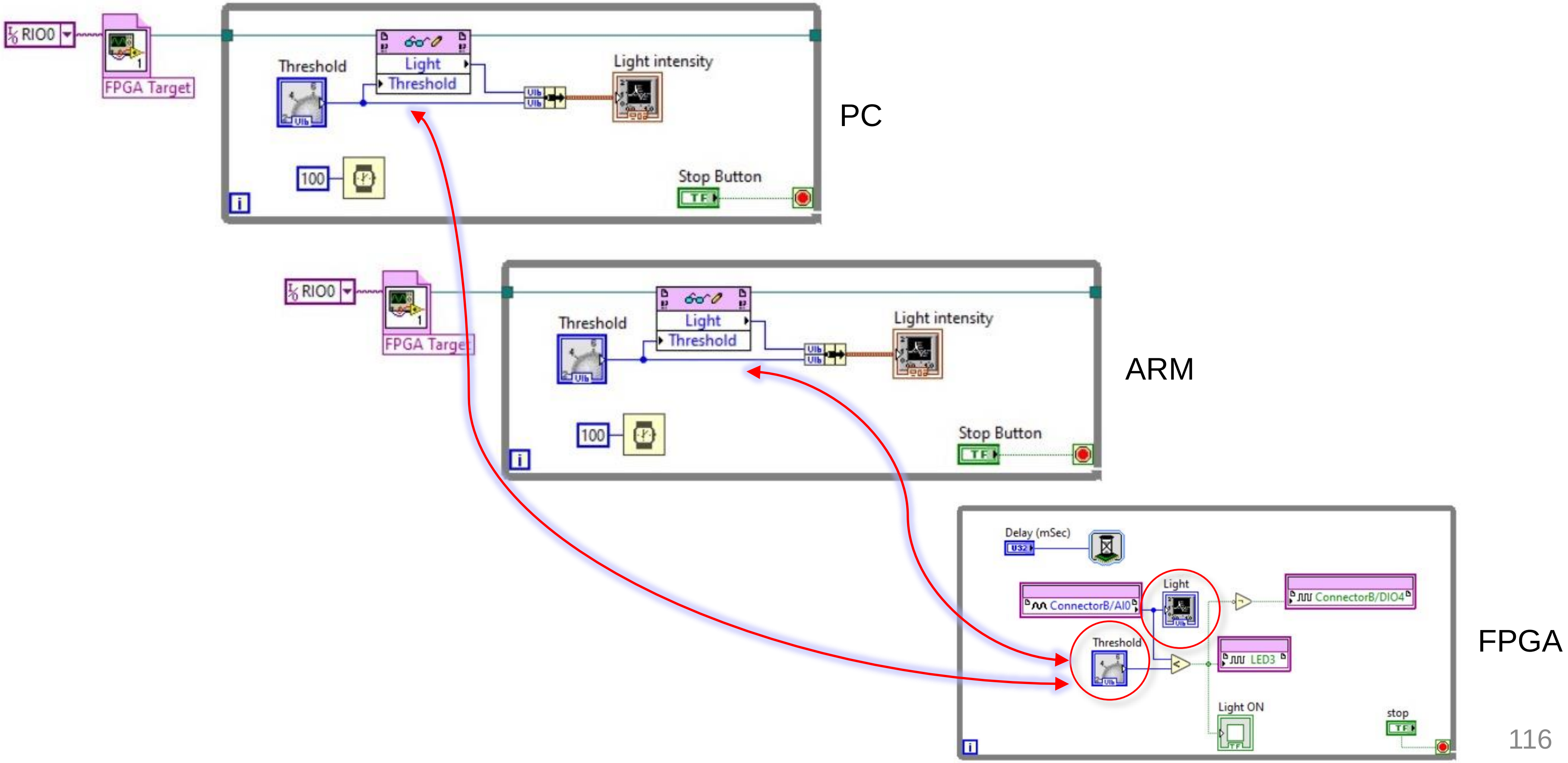


How to communicate data between FPGA – ARM – PC ?

FPGA to ARM communication



FPGA to ARM to PC communication



From small to big FPGA’s

1. myRIO

2. sbRIO

3. cRIO

4. PXIe R-series boards

5. PXIe FlexRIO boards

Big FlexRIO system for X-band cavity test

PXIe FlexRIO boards

Resources and Next Steps

NI myRIO Kits | ni.com/myrio

Starter

LEDs & switches
7-segment display
Potentiometer
Thermistor
Photo resistor
Hall effect
Microphone/Speaker
Battery holder
DC motor

Mechatronics

DC gear motors/encoders
H-bridge driver
Accelerometer
Triple-axis gyro
Infrared proximity sensor
Ambient light sensor
Ultrasonic range finder
Compass
Hobby servo motors

Embedded

RFID reader kit
Numeric keypad
LED matrix
Digital potentiometer
Character LCD
Digital temp sensor
EEPROM

Learn More About Programming NI myRIO

ni.com/learn-myRIO
ni.com/community/myrio

Thank you !!!