



# LabVIEW

# ISOTDAQ 2022

Gary Boorman  
20<sup>th</sup> - June - 2022

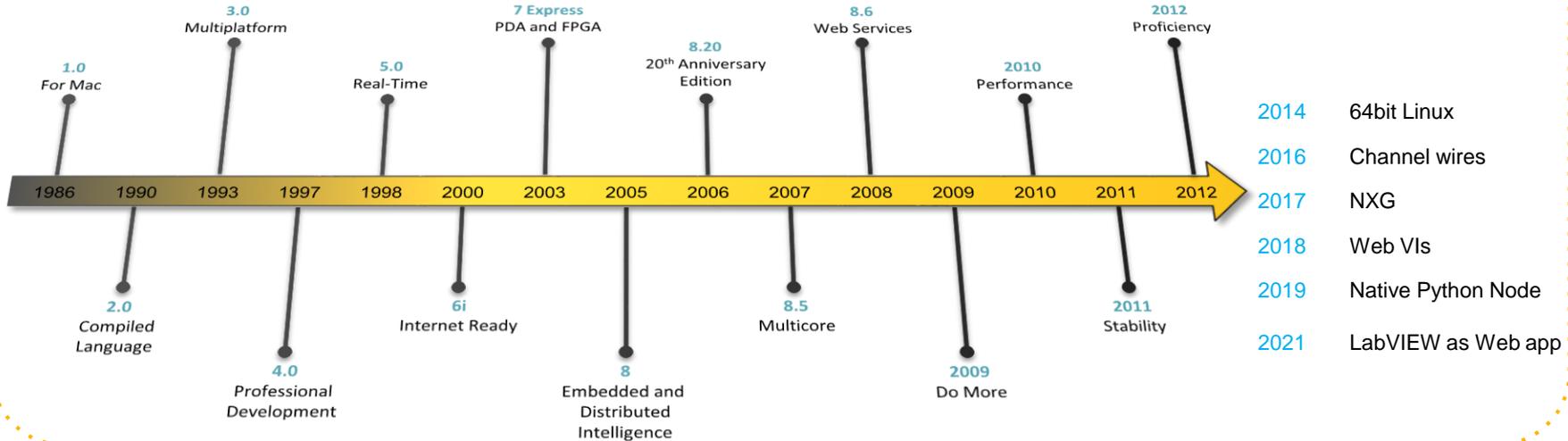
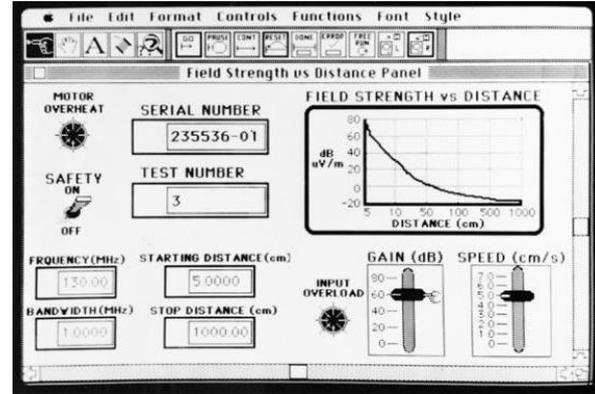


# Agenda

- Introduction to LabVIEW
- Instrumentation and Data Acquisition
- Application Development
- LabVIEW for Accelerators and Detectors
- Other Research Applications

# Background

National Instruments





# DAQ Comparison

Software Used for Data Acquisition and Instrument Control

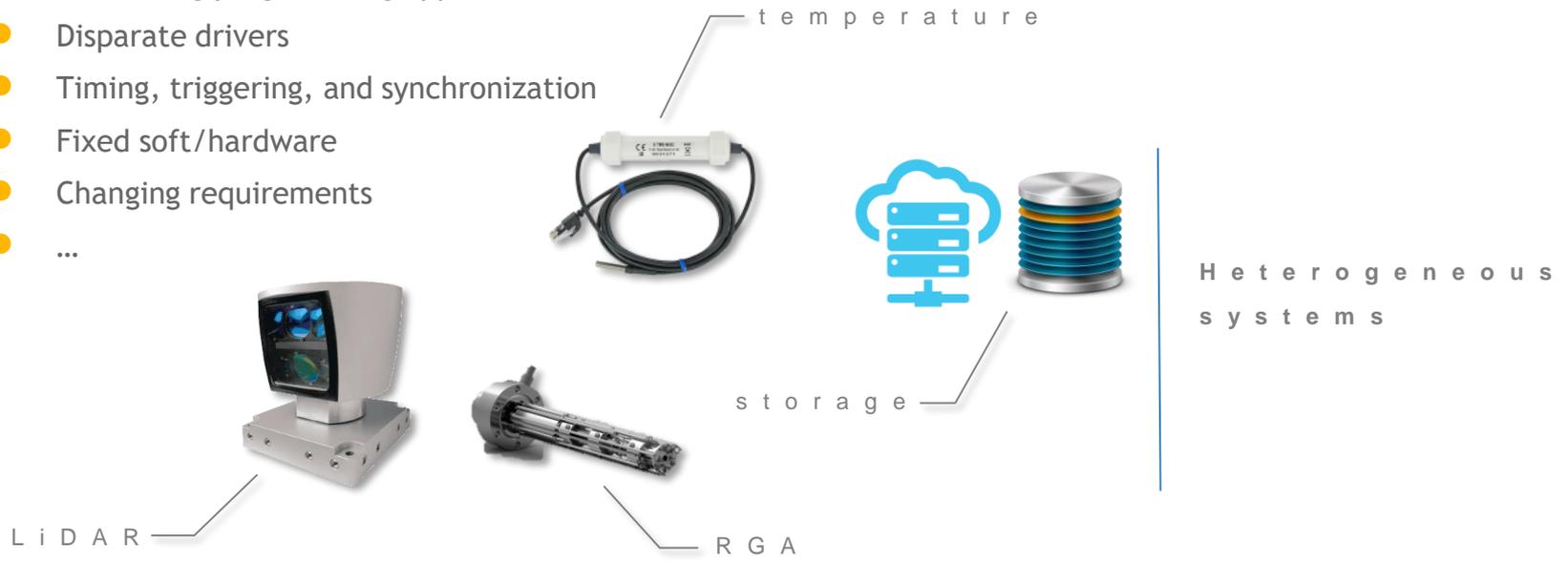
OPTIONS	C++/C#/JS/VB	LabVIEW	MATLAB	DASYLab
Ease of programming (novice)	Difficult	Easy	Medium	Easy
Programming Community size	Very large	Large	Large	Medium
Complex Applications	Yes	Yes	No	No
Built-in DAQ Support	No	Yes	Some	Yes
Built-in Analysis	No	Yes	Yes	Yes

# DAQ & Instrumentation



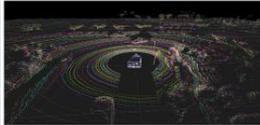
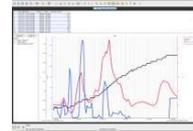
# Measurement challenges

- Conflicting programming approaches
- Disparate drivers
- Timing, triggering, and synchronization
- Fixed soft/hardware
- Changing requirements
- ...

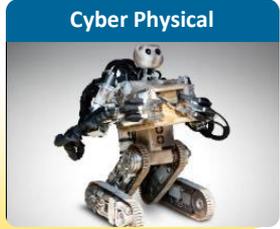




# Measurement challenges

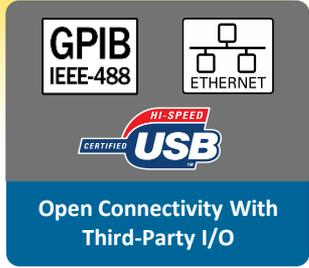
Sensor	Interface	Conditioning?	Software
		n o	
		y e s	
		y e s	-
		n o	

Heterogeneous systems



+

# LabVIEW™





# Modular Instruments



Compact DAQ

PXI



Compact RIO

PXI/PXIe modules



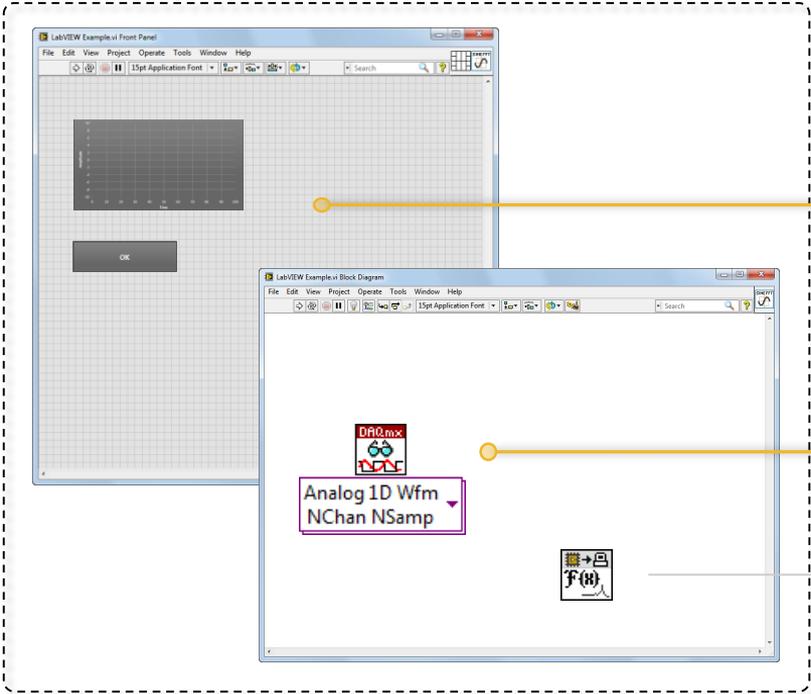
chassis

# Application

Creating Code



# Application development



**LabVIEW Front Panel**  
The user interface of a VI

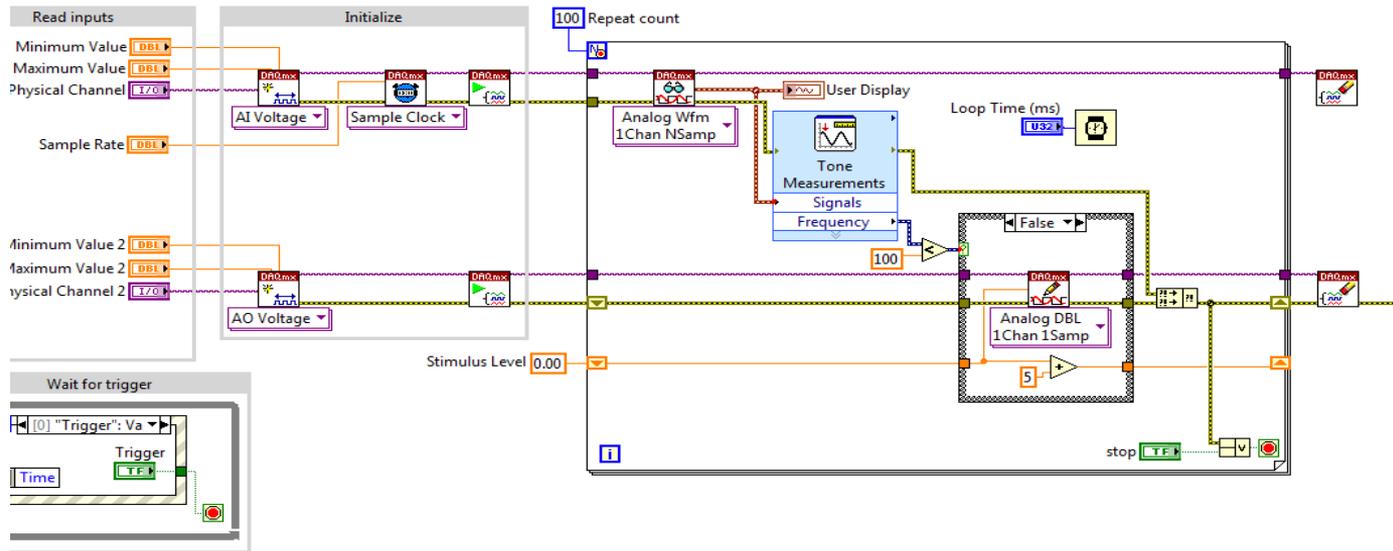
**LabVIEW Block Diagram**  
The source code of a VI

Functions:  
Virtual  
Instruments

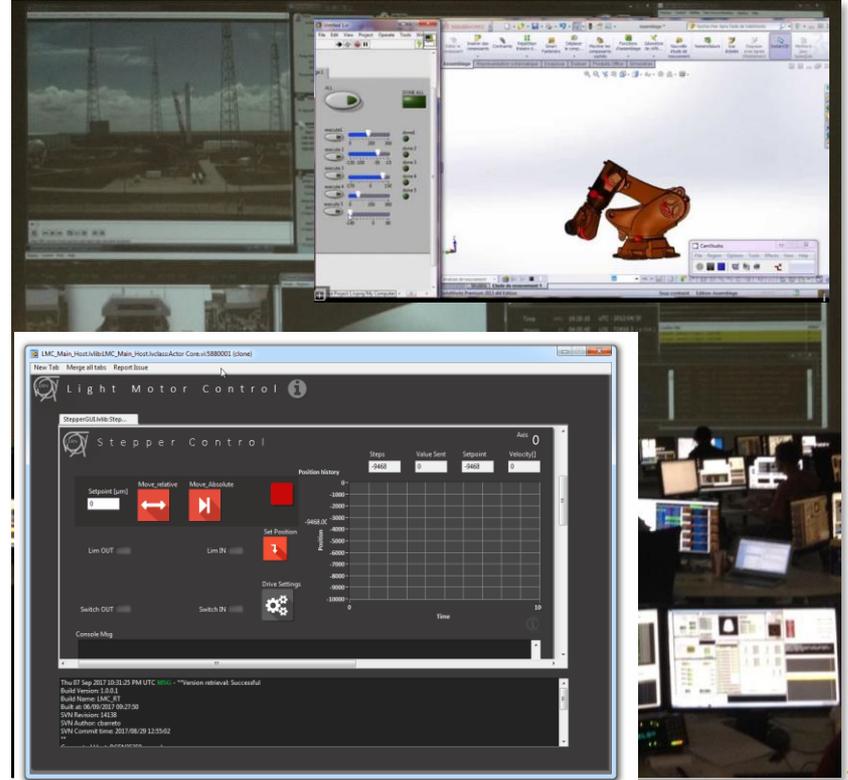
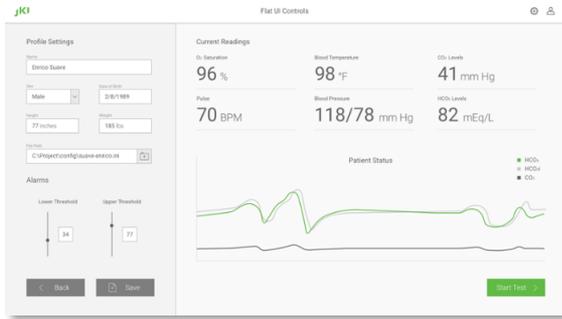


# Application development

- Program as you think



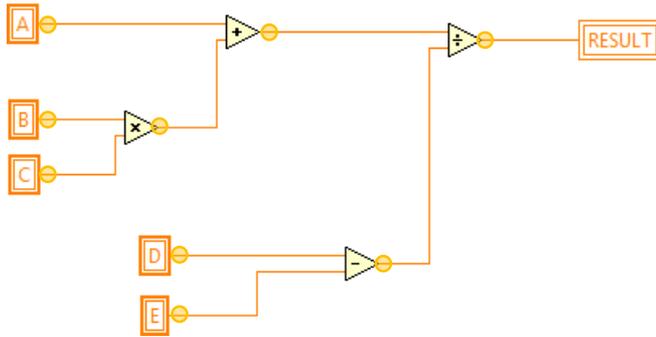
# Graphical interface





# Dataflow

- Data driven execution

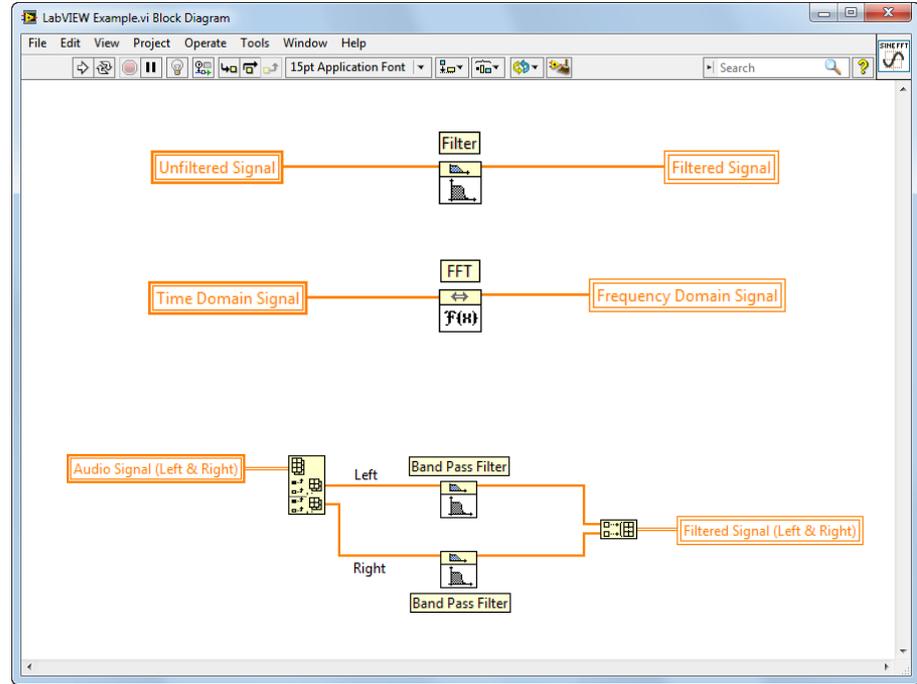


Intrinsic Parallelism

# Dataflow

- Data driven execution

Intrinsic Parallelism

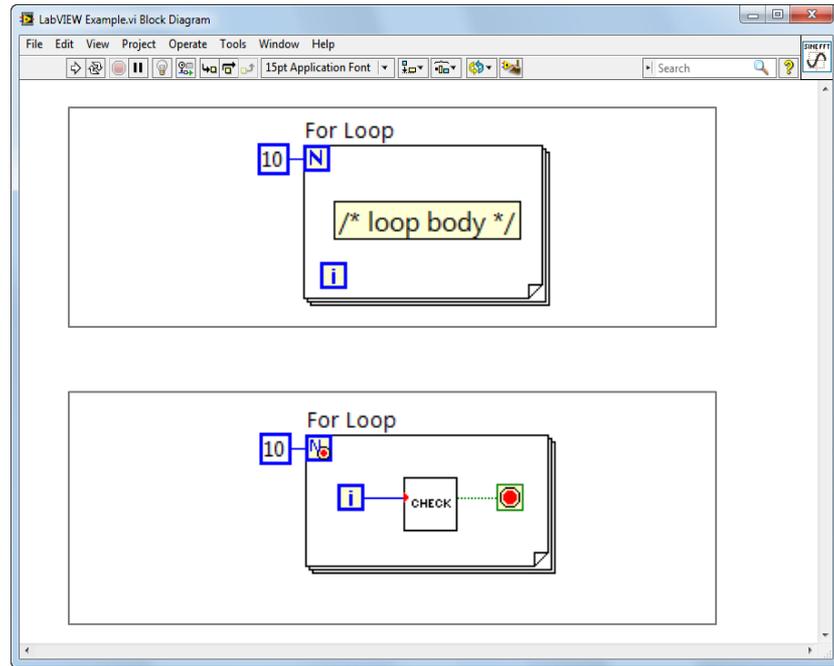


# Comparison with text



```
for (i = 0; i < 10; i++)  
{  
    /* loop body */  
}
```

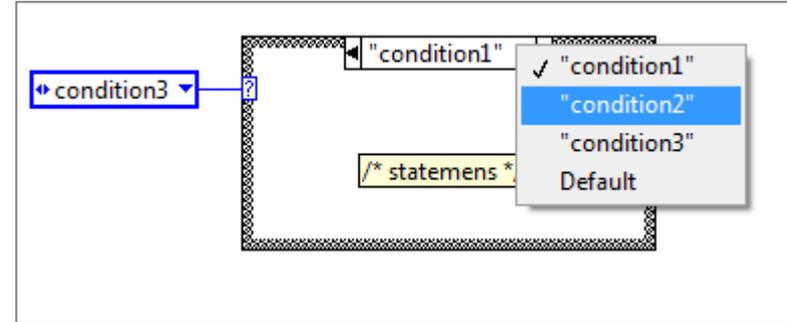
```
for (i = 0; i < 10; i++)  
{  
    if(check(i)) break;  
}
```



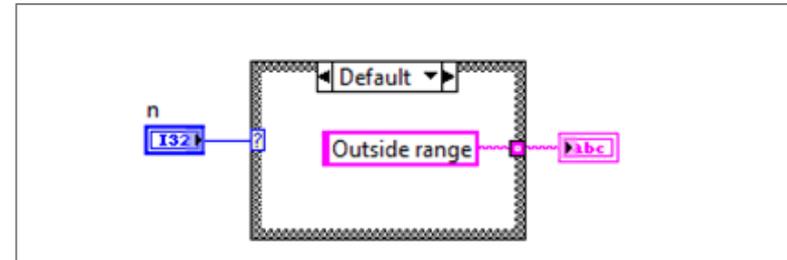


# Comparison with text

```
if condition1 then
  -- statements;
elseif condition2 then
  -- more statements
elseif condition3 then
  -- more statements;
else
  -- other statements;
end if
```



```
switch (n) {
  case 5:
    printf("Small number.");
    break;
  case 100:
    printf("Large number.");
    break;
  default:
    printf("Outside range");
    break;
}
```





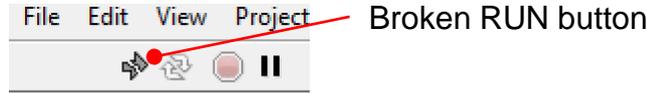
# The LabVIEW Compiler I

- The LabVIEW environment continually parses the block diagram

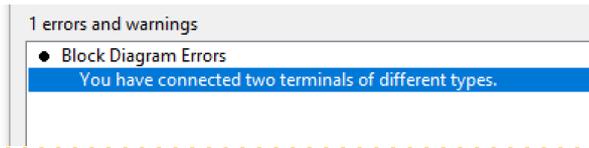
- Valid code ->



- Invalid/incomplete code ->



- If code is valid, clicking on the RUN button causes LabVIEW to compile the code and then execute it
- Click on a broken RUN button to get detailed information on the error

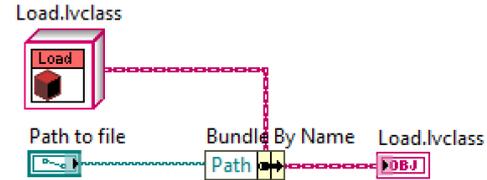




# The LabVIEW Compiler II

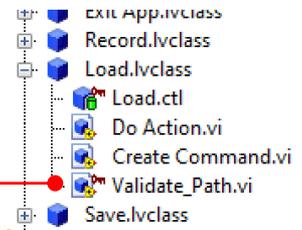
- When developing/debugging LabVIEW code it can be run and tested within the LabVIEW environment
- Once the code is working as desired it can be compiled into an executable (.exe etc), then launched like any other program
  - LabVIEW supports both 32 and 64-bit OS: Windows, Linux and IOS
- Code can also compile into a windows library (.DLL) or Linux library (.SO)
  - Calls to DLL or SO require knowledge of the function prototypes - LabVIEW will generate the appropriate documentation
- LabVIEW can call functions within other DLL and SO libraries

# LabVIEW OOP



- LabVIEW has object-oriented capabilities - encapsulation & inheritance
- But **BEWARE**
  - LabVIEW is a by-value language, including its objects
    - Most other OO environments use by-reference objects
  - All data is private
    - Explicit accessor methods must be used to access the data
- Methods are public by default but can be made private (called by class's methods only) or protected (called by child classes too)
- LabVIEW objects are supported on Desktop, RT and FPGA
- Objects can be by-reference if needed

Private method



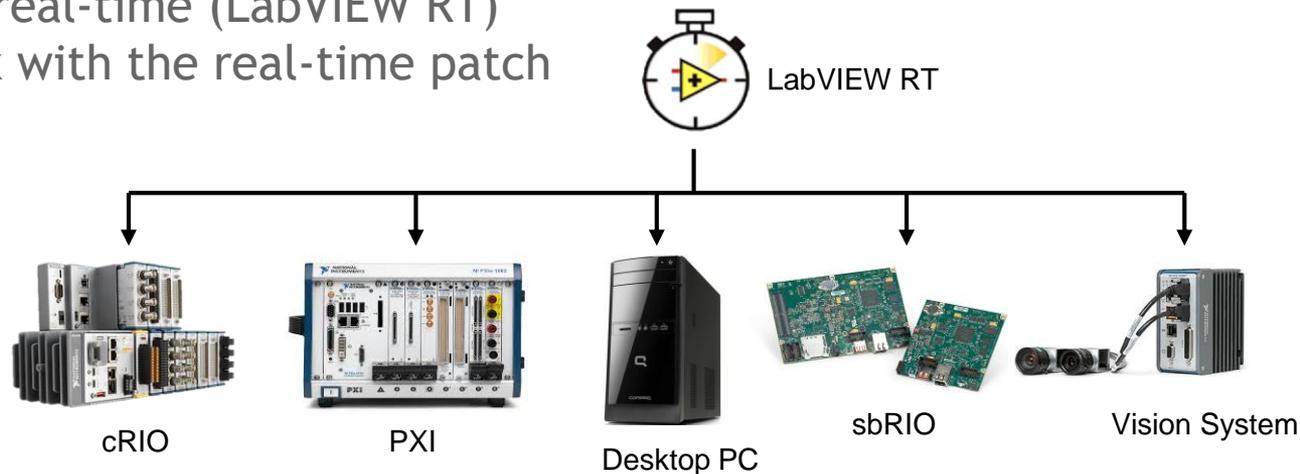
# Embedded Systems

Extending the LabVIEW environment



# Real-time Systems

- Deterministic code operation
- Create distributed control/test/acquisition systems
- LabVIEW real-time (LabVIEW RT)
  - Linux with the real-time patch





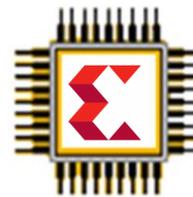
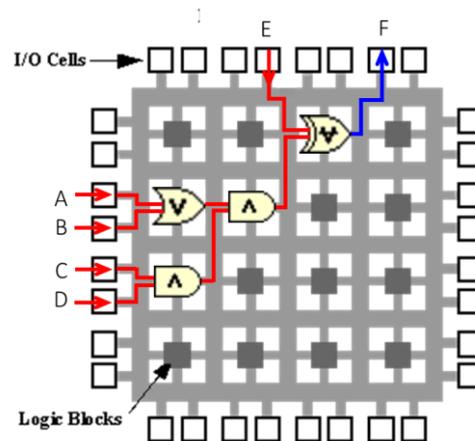
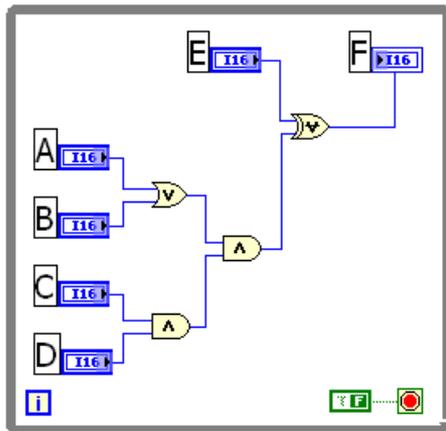
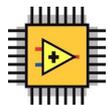
# Compiling LabVIEW for RT Systems

- LabVIEW can run RT code within the development environment
  - Code is executed on the RT system
  - User interface is on the desktop/development system
- Code can usually be run on different RT targets with only minimal changes (file paths, hardware interfaces etc)
- Once the code is running as expected, compile the code into an RT executable
  - Executable can be deployed on RT system
  - Executable starts running once the RT has powered up and loaded its operating system
  - Code is usually designed for running 24/7



# LabVIEW to the pin

- LabVIEW FPGA





# Compiling LabVIEW for FPGA

- Many LabVIEW functions are available for FPGA
  - *Some exceptions:*
    - Unbound arrays, queues, strings
    - Double precision numbers (Single is permitted)
    - Non-homogeneous arrays of objects
- LabVIEW FPGA code needs to be compiled - automatically launches and uses the Xilinx Vivado environment. Can add existing VHDL IP
- The RT system accesses the FPGA using:
  - Front panel controls and indicators (fairly slow)
  - Direct memory access, DMA (very fast, up to GB/s depending on backplane)
  - Interrupts (latency in order of  $\mu\text{s}$ )

# LabVIEW for Accelerators and Detectors



# LabVIEW at CERN

550 LabVIEW Users



30+ Project clients



CERN LabVIEW  
Support



LabVIEW™

Center of Excellence



# The access challenge



GPN



TN



Logging



CMW



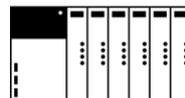
RBAC



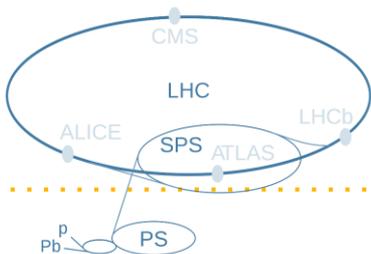
timing



Front ends



FESA





# Custom hardware

P X I



CTRIP-PMC  
(CERN)



PMC carrier  
(Kontron)



Fine delay-FMC  
(CERN)



FMC carrier  
(INCAA)

c R I O



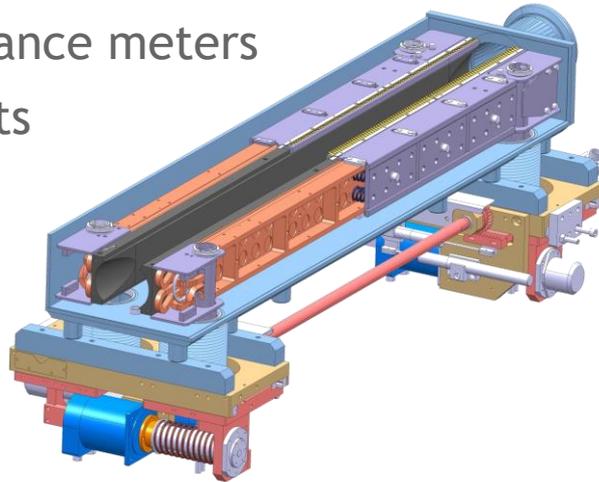
White rabbit timing (CERN)



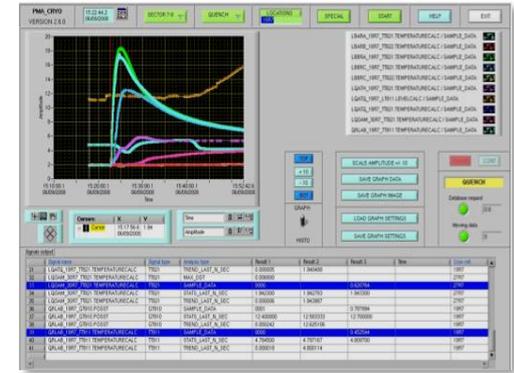
Fibre-based triggering  
(ANGARA Technology)

# Example applications

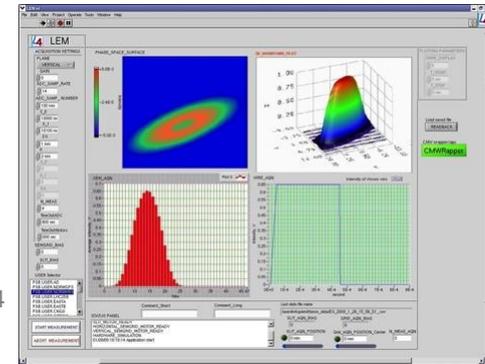
- LHC collimators
- LINAC4 emittance meters
- Kicker Magnets
- AWAKE
- CLIC
- ...



Linac 4



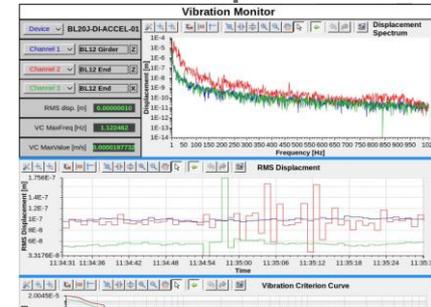
Post-Mortem analysis





# LabVIEW and Middleware

- EPICS support built-in
  - Create EPICS IOCs to run (usually) on Embedded systems
  - Create EPICS Clients on both Embedded and Desktop systems
  - Several third-party solutions that improve performance or the scope of data-types (LNLS, ANL etc)
- CMW (Controls Middleware) at CERN
  - The MTA group has created RADE
    - Embedded systems running LabVIEW can read/write to the standard CERN tools/databases
- TANGO
  - Third-party support from some European and US labs



Other Applications



# LabVIEW Web Module

- Compile LabVIEW and run within web-page (Javascript)
- View compiled code on any device



- Try [www.webvi.io](http://www.webvi.io)



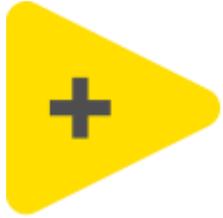
# Support for emerging technology

- Extensive HW and SW support of RF
  - Vector Signal Transceiver (VST) with accessible FPGA
  - 5G research and metrology
- Autonomous vehicles
- Industrial Internet of Things (IIoT)





# Thank you



Contact me:  
[gary.boorman@angaratech.ch](mailto:gary.boorman@angaratech.ch)





# Credits

- National Instruments



- CERN EN-SMM group

