LabVIEW

ISOTDAQ 2023

Gary Boorman

19th June 2023

Agenda

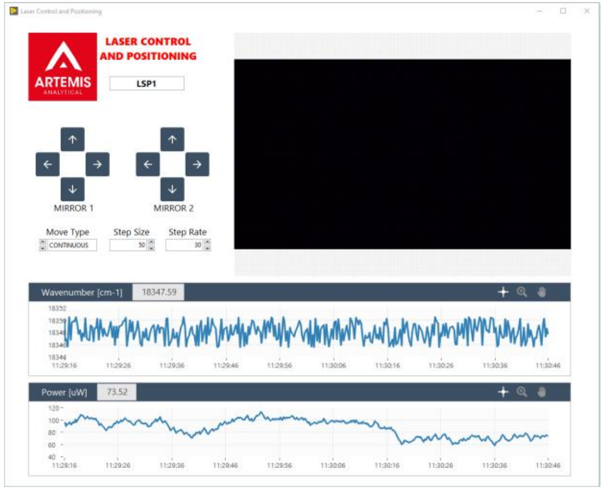
- Introduction to LabVIEW
- Application Development
- LabVIEW Integration for Hardware and Software
- LabVIEW for Accelerators and Detectors
- Summary

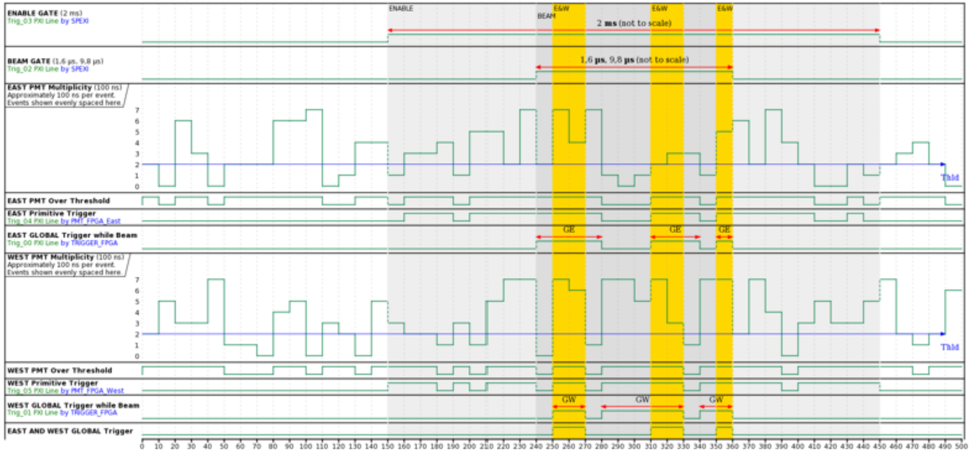
LabVIEW

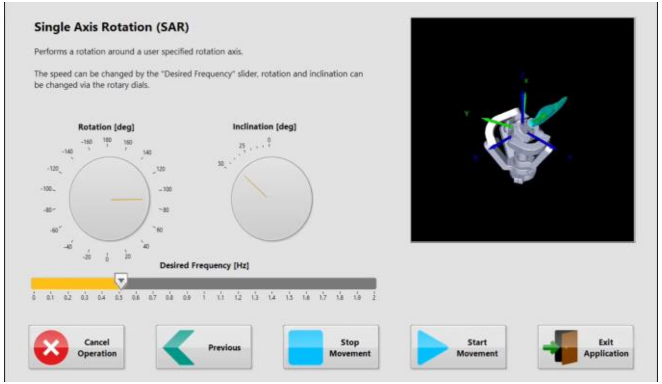
What is it?

What is LabVIEW?

LabVIEW is a graphical programming environment used by scientists and engineers to develop automated research, validation and test systems

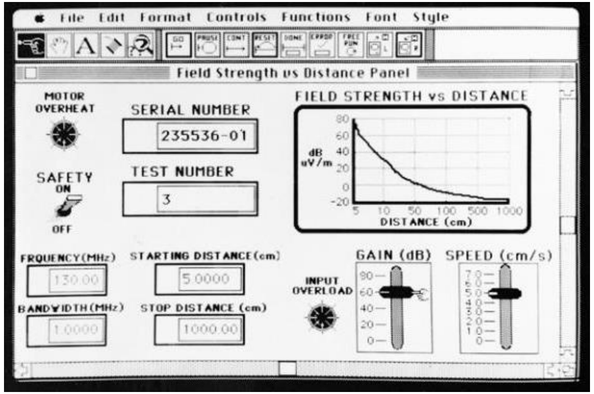


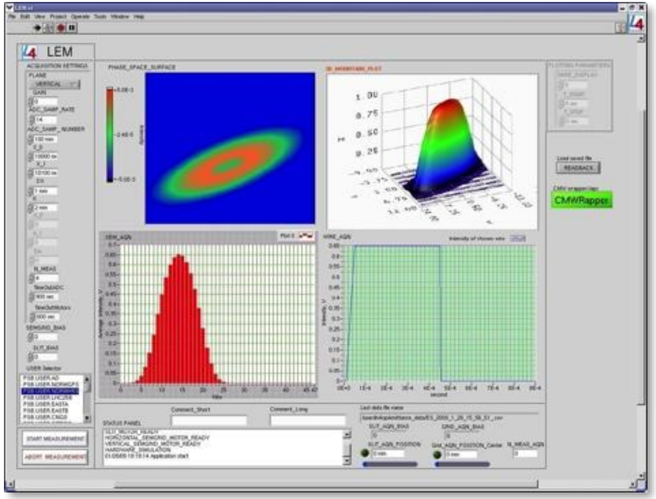




Background

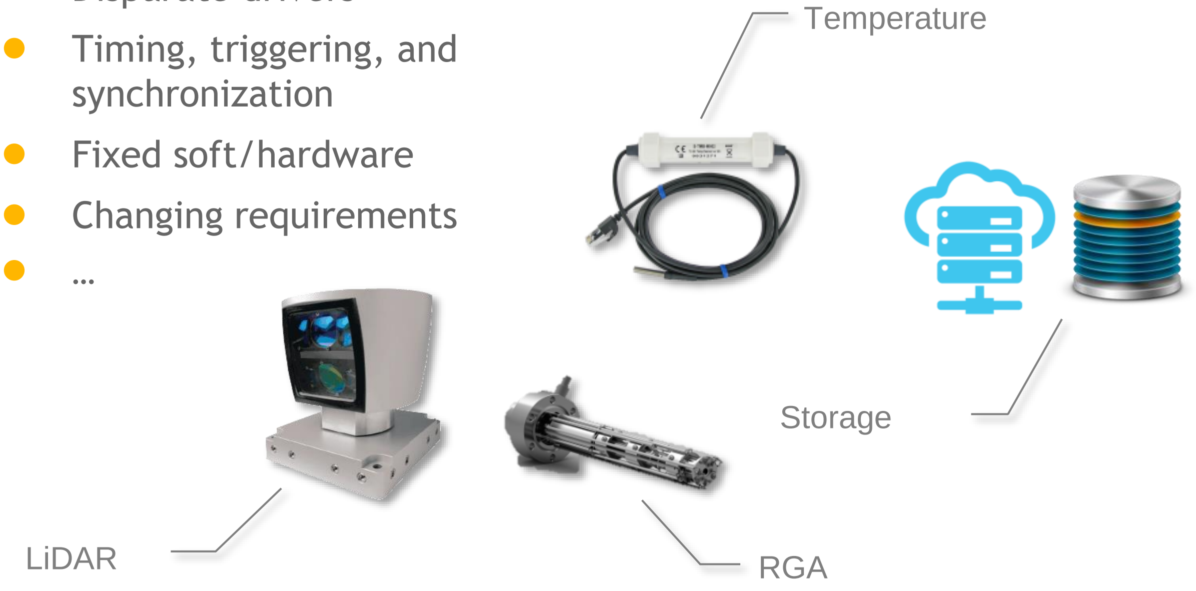
NI was started by a couple of physicists….
NI created hardware in 1970s, then added an easy way to program it in 1986 – LabVIEW





Measurement challenges

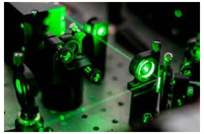
- Conflicting programming approaches
- Disparate drivers
- Timing, triggering, and synchronization
- Fixed soft/hardware
- Changing requirements
- …



Heterogeneous systems

HW and SW Integration

- Integration is the key to efficiently creating a control and measurement system.
- LabVIEW effortlessly joins together hardware and software.




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

NI Modular Instruments

Compact DAQ

Compact RIO

PXI

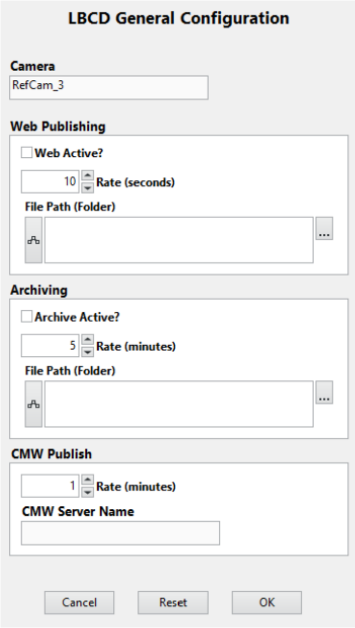
PXI/PXIe modules

chassis

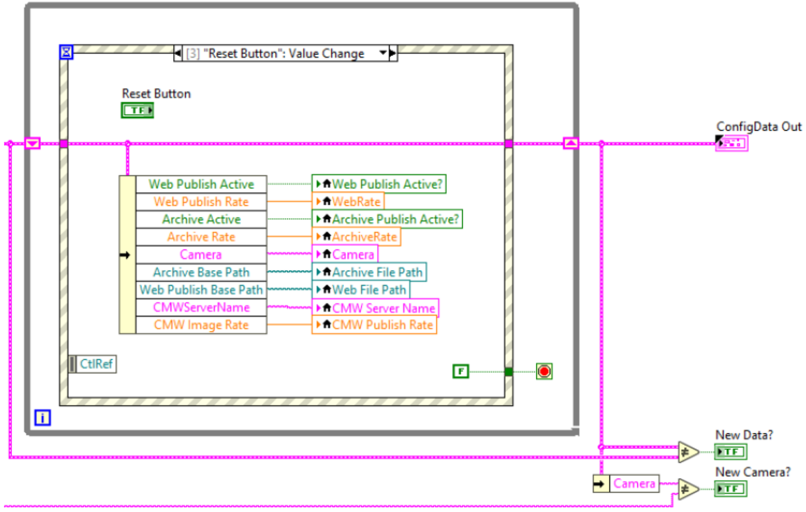
Application

Creating Code

LabVIEW Environment



LabVIEW Front Panel
(the user interface)

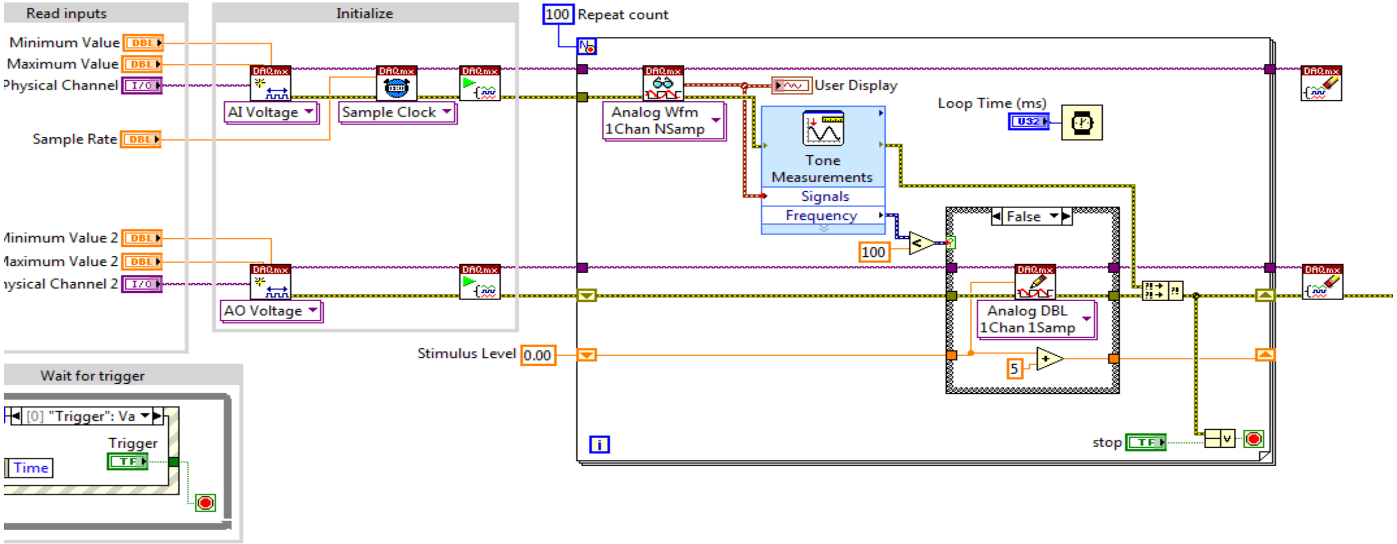


LabVIEW Block Diagram
(the source code)

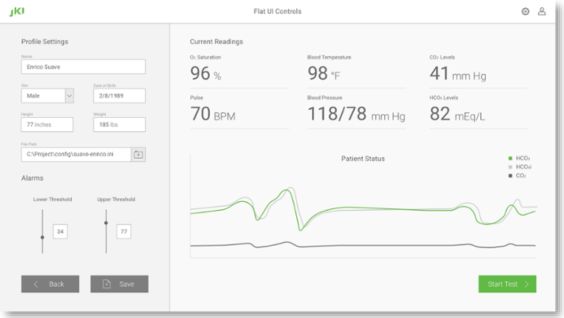
Together the Front Panel and the Block Diagram create a **VI** – a virtual instrument

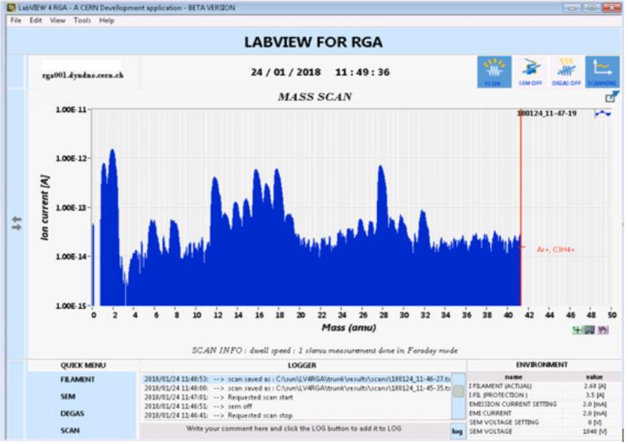
Application development

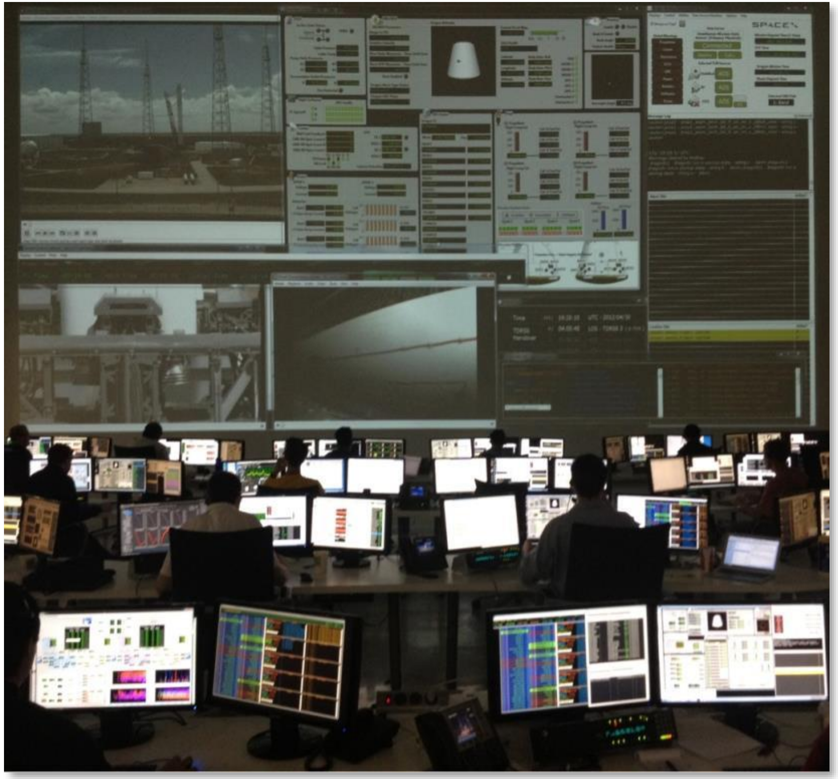
- Program as you think



Graphical interface

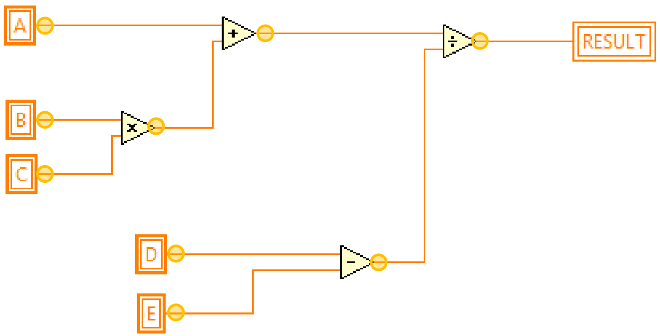






Dataflow I

- Data driven execution

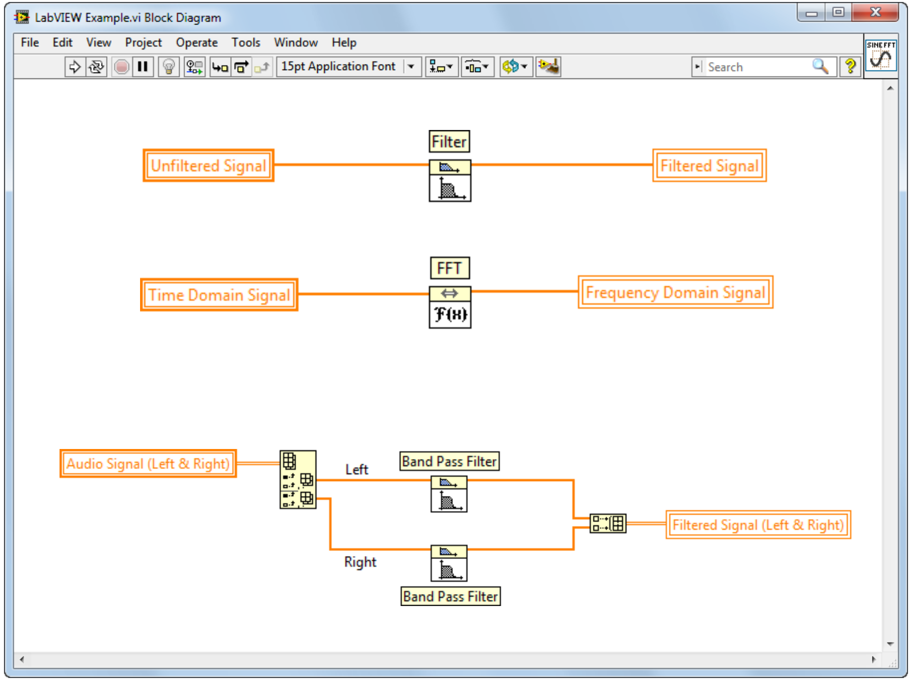


Intrinsic **Parallelism**

Dataflow II

- Data driven execution
- Multi-threaded (no explicit commands required, just careful code design)

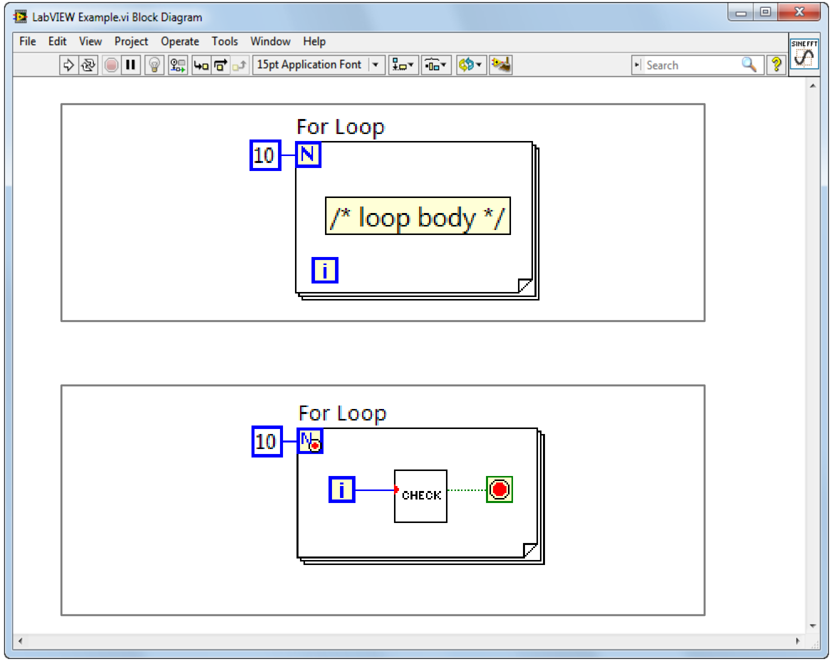
Intrinsic Parallelism



Comparison with text

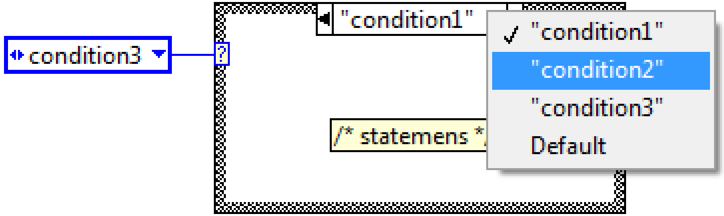
```
for (i = 0; i < 10; i++)
{
    /* loop body */
}
```

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

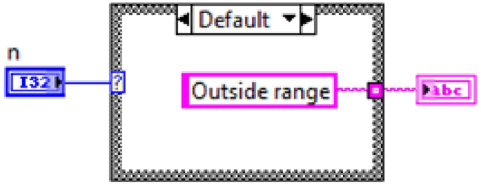


Comparison with text

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

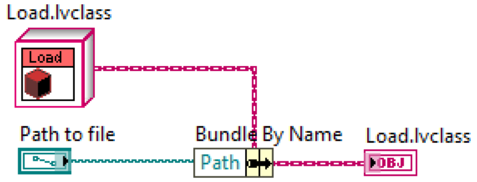


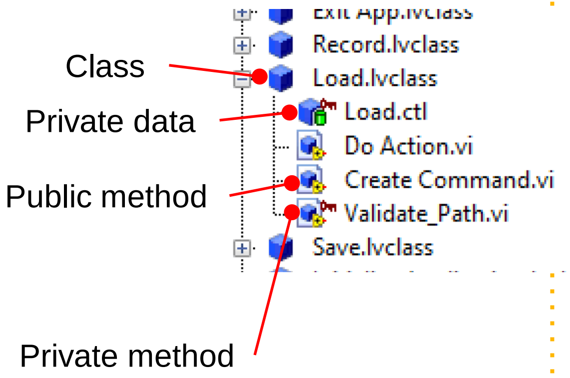

```
switch (n) {
  case 5:
    printf("Small number.");
    break;
  case 100:
    printf("Large number.");
    break;
default:
    printf("Outside range");
    break;
}
```



LabVIEW OOP

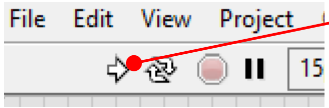
- LabVIEW has object-oriented capabilities – encapsulation and inheritance
- But **BEWARE**
  - LabVIEW is a by-value language, including its objects
    - Most other OO environments use by-reference objects
  - All data is private
    - Explicit accessor methods must be used to access the data
- Methods 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





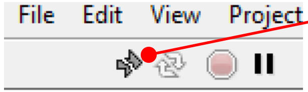
The LabVIEW Compiler

- The LabVIEW environment continually parses the block diagram
  - Valid code ->



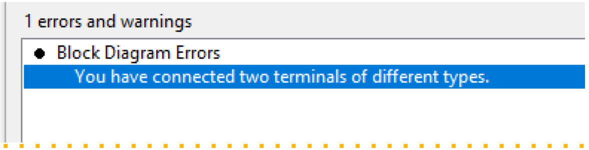
Solid RUN button

  - Invalid/incomplete code ->



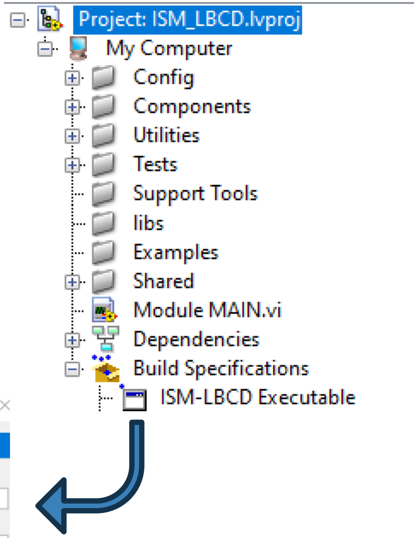
Broken RUN button

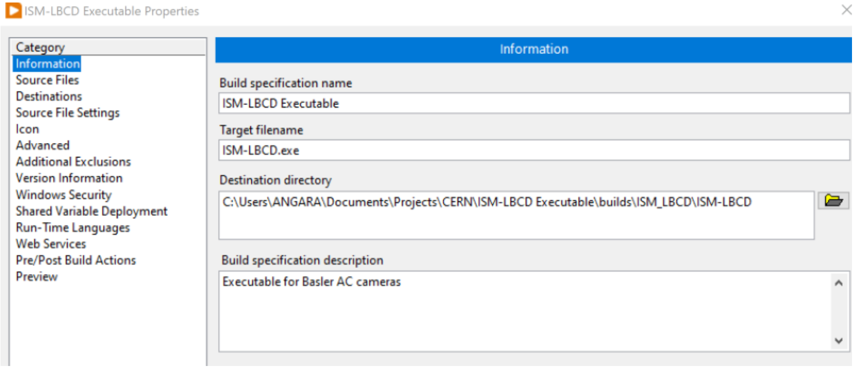
- If code is valid, clicking on the RUN button causes LabVIEW to compile the code and then execute it
- Click on a broken RUN button to get detailed information on the error



Creating Executables

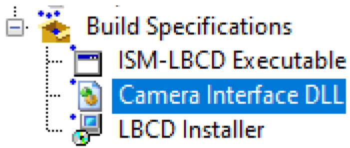
- When developing/debugging LabVIEW code it can be run and tested within the LabVIEW environment
- Once the code is working as desired it can be compiled into an executable (.exe etc), then launched like any other program
  - LabVIEW supports both 32 and 64-bit OS: Windows, Linux and IOS



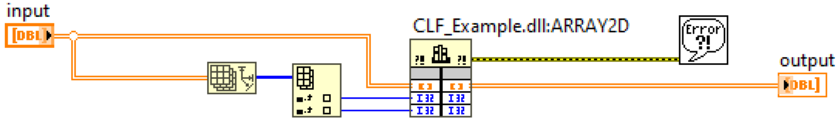


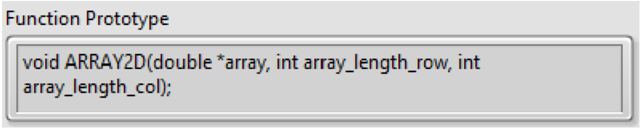
Creating/Calling DLLs & SOs

- 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



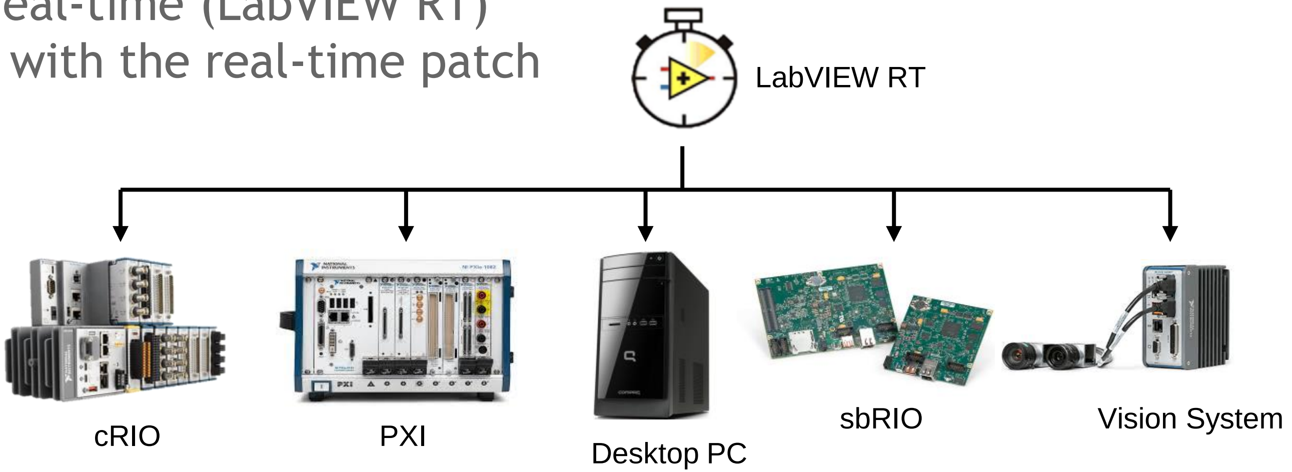


Integration

Bridging hardware and software

Real-time Systems

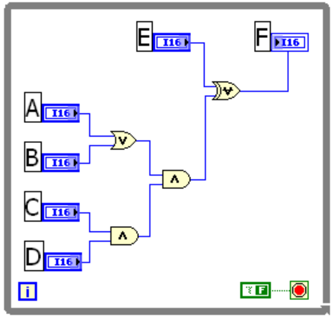
- Deterministic code operation
- Create distributed control/test/acquisition systems
- LabVIEW real-time (LabVIEW RT)
  - Linux with the real-time patch

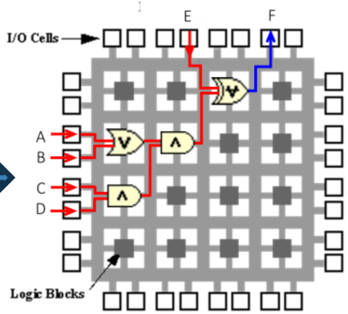


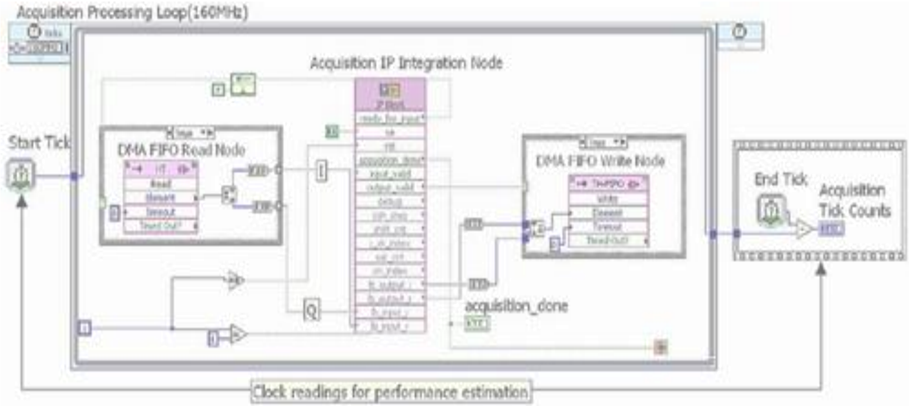
Compiling LabVIEW for RT Systems

- 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 FPGA







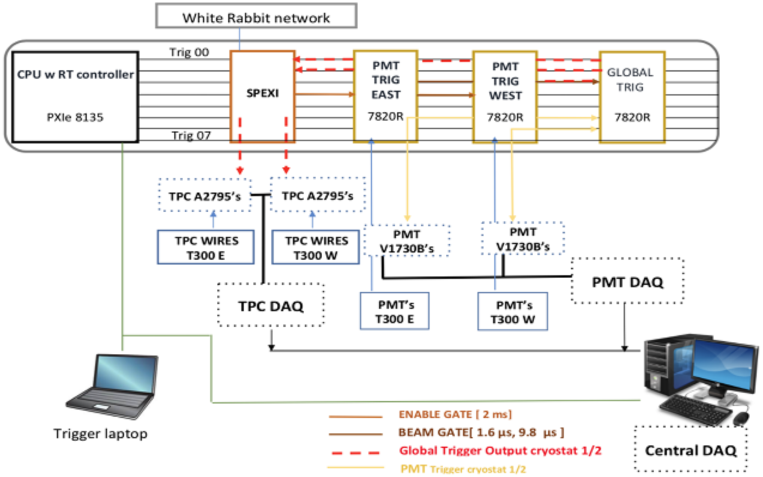
- LabVIEW first generates VHDL then uses the VIVADO compiler to make a bitfile

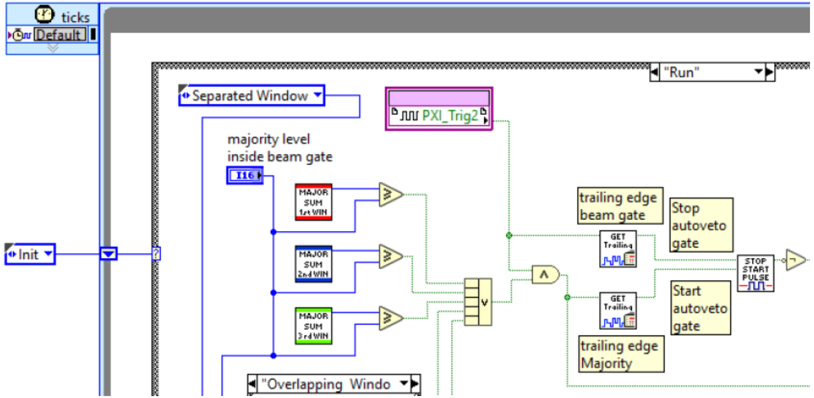
Compiling LabVIEW for FPGA

- Many LabVIEW functions are available for FPGA
    - ***Some exceptions:***
        - Unbound arrays, queues, strings
        - Double precision numbers (Single is permitted)
        - Non-homogeneous arrays of objects
- Can **add pre-existing HDL** to LabVIEW FPGA code
- 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 µs)

FPGA Example

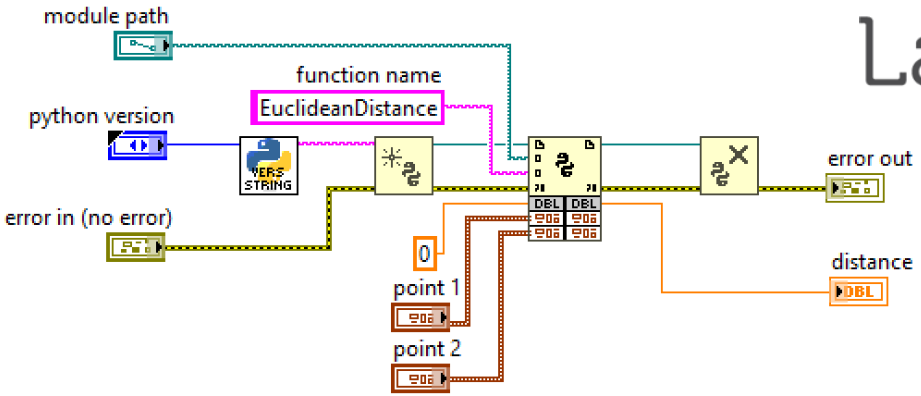
- **ICARUS Detector at Fermilab**
- Based on PXIe chassis, all programmed with LabVIEW
- Trigger system uses three FPGA cards acquiring data at 40 MS/s
- Triggers are time-stamped using White Rabbit; global trigger sent to readout electronics





LabVIEW and Python

- **The Python node**
- Call a function from a Python module



LabVIEW in the Physics Lab

LabVIEW at CERN

550 LabVIEW Users

30+ Project clients

CERN LabVIEW

Support

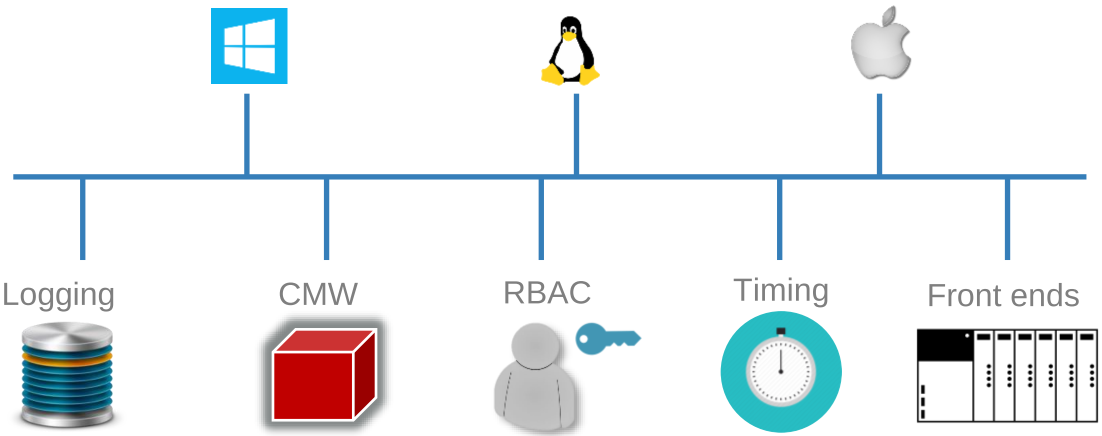
Training

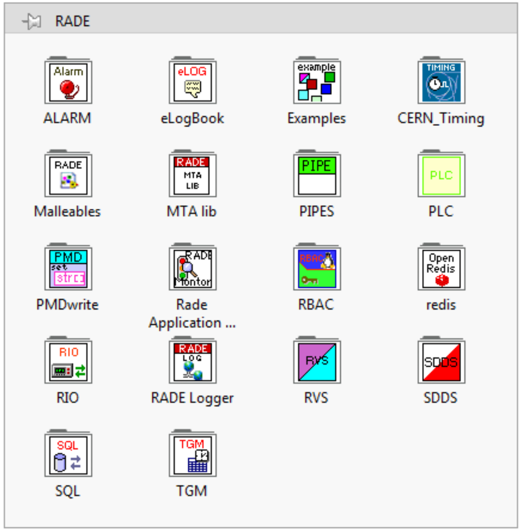
LabVIEW™

Center of Excellence

RADE – Rapid Application Development Environment

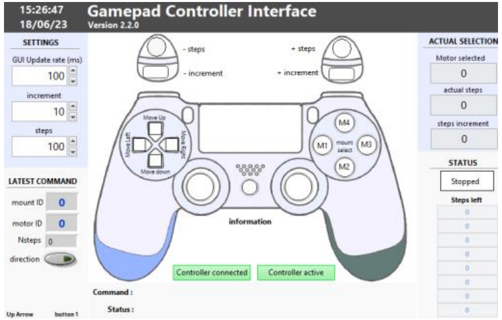
The LabVIEW solution at CERN to develop expert tools, machine development analysis and test facilities integrated with the CERN control infrastructure.
Contains a *middleware* interface for accessing instruments/processes on CERN network

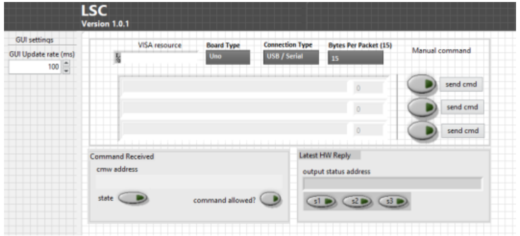


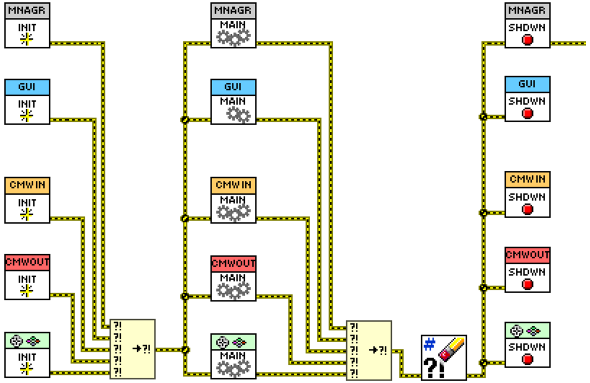


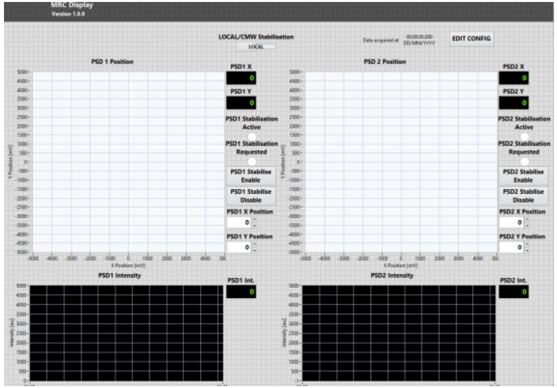
ISOLDE Suite of Tools

- Consistent code structure for all modules (20+) based on custom template
- *Average* LabVIEW developers can modify code
- Uses RADE palette for logging, CMW



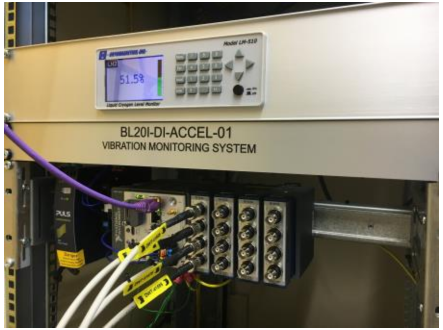


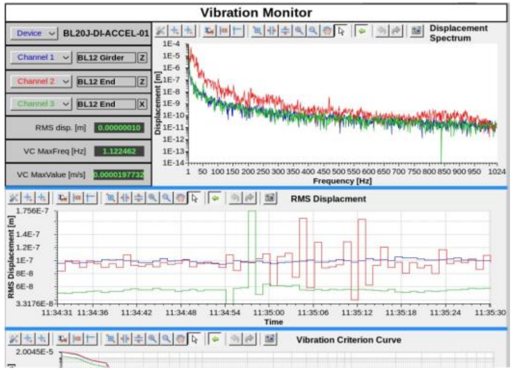


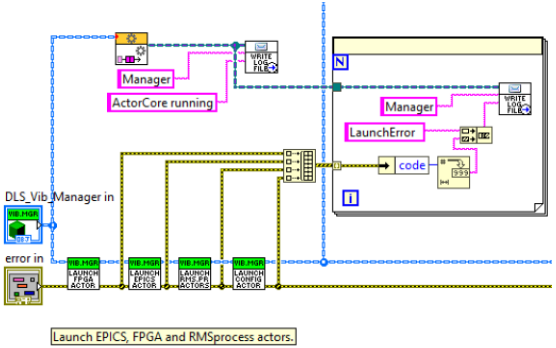


LabVIEW and Middleware

- 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)
- Example
  - **Vibration monitor at Diamond** – cRIO system with EPICS interface







Other Applications

LabVIEW Web Module

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

- Try *www.webvi.io*

Custom hardware

PXI

CTRP-PMC
(CERN)

PMC carrier
(Kontron)

Fine delay-FMC
(CERN)

FMC carrier
(INCAA)

cRIO

White rabbit timing (CERN)

Fibre-based triggering
(ANGARA Technology)

Summary

- One development tool – multiple platforms: Desktop, RT, FPGA, GPU, Web
- Full integration between hardware and software

The power of LabVIEW Integration

Thank you

Contact me: gary.boorman@angaratech.ch