Introduction to NI LabVIEW and Computer-Based Measurements

National Instruments

Today, We'll Explore:

The Challenges of Making Measurements	Characteristics of Mixed-Measurement Systems
	The National Instruments Approach
	Architecture of a Measurement System
Break	Enjoy Coffee and Networking With Industry Peers
Introduction to LabVIEW	History and Philosophy of LabVIEW
	Navigating the LabVIEW Environment
	Gaining LabVIEW Proficiency
Break	Enjoy Coffee and Networking With Industry Peers
Fundamentals of Data Acquisition	Essential Data Acquisition Concepts
	The Basics of Signal Conditioning
	The Value of National Instruments Hardware Platforms
Uniting Software and Hardware	Architecture of the NI-DAQmx Driver
	Measurement Services and Utilities
	Exploring and Using the NI-DAQmx API

The Challenges of Making Measurements

Exploring the Traditional Approach to Measurements

The Origin of Automated Measurements

- Traditional pen-and-paper approach
- Redundant circuitry between instruments (e.g., displays)
- Manual data recording and analysis
- Error-prone processes
- Difficult to reproduce or redo

Thermoelectric Voltage in mV											
°C	0	1	2	3	4	5	6	7	8	9	10
0	0.000	0.050	0.101	0.151	0.202	0.253	0.303	0.354	0.405	0.456	0.507
10	0.507	0.558	0.609	0.660	0.711	0.762	0.814	0.865	0.916	0.968	1.019
20	1.019	1.071	1.122	1.174	1.226	1.277	1.329	1.381	1.433	1.485	1.537
30	1.537	1.589	1.641	1.693	1.745	1.797	1.849	1.902	1.954	2.006	2.059
40	2.059	2.111	2.164	2.216	2.269	2.322	2.374	2.427	2.480	2.532	2.585
50	2.585	2.638	2.691	2.744	2.797	2.850	2.903	2.956	3.009	3.062	3.116
60	3.116	3.169	3.222	3.275	3.329	3.382	3.436	3.489	3.543	3.596	3.650
	.650	3.703	3.757	3.810	3.864	3.918	3.971	4.025	4.079	4.133	4.187
		4.240	4.294	4.348	4.402	4.456	4.510	4.564	4.618	4.672	4.726
			4.835	4.889	4.943	4.997	5.052	5.106	5.160	5.215	5.269

Measurement Challenges Are Compounded By:

- Compressed Timelines
- Fixed Software and Hardware
- Conflicting Programming Approaches
- Inadequate Hardware Performance
- Disparate Driver APIs
- Varying Sensors and Connectivity
- Custom Signal Conditioning
- Advanced Visualization
- Changing Application Requirements
- Complex Analysis Algorithms
- Evolving Technology Trends
- Confusing Data Storage
- Differing Sampling Rates

NATIONAL INSTRUMENTS

Mixed-Measurement Applications Are Diverse

Vibration

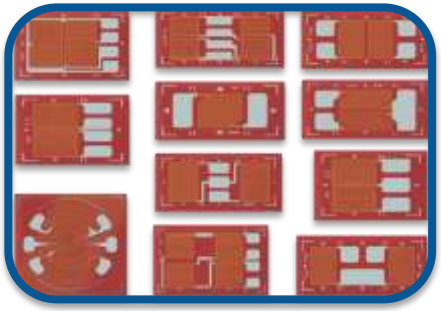
Torque

Displacement

Pressure

Temperature

Force



Strain

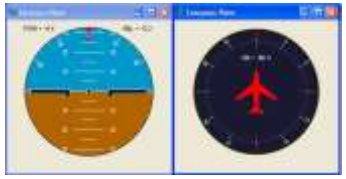
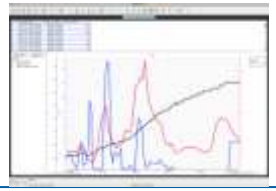
Example Application: Air Quality Measurements

- Potential Sensors Needed:
  - Context
    - GPS
      - Timestamp
      - Position
    - Attitude
    - Altitude
    - Range Finder
  - Environmental
    - Temperature
    - Oxygen
    - Carbon Dioxide
    - Ozone
    - Nitrogen

Sensors, Interfaces, and Signal Conditioning

Sensor		Interface	Conditioning?
GPS		RS232	No
Attitude, Altitude		RS232	No
LiDAR		Ethernet	No
Temperature		Analog Voltage	Required
O_2, CO_2, O_3, NH_3		Analog Voltage	Required

Software Provided With Sensors

Sensor		Software
GPS		
Attitude, Altitude		
LiDAR		
Temperature		
O_2, CO_2, O_3, NH_3		<No Software Provided>

With a System Like This, How Do You Accommodate…

- …changes in requirements?
- …mixed measurements in a single system?
- …varying connectivity?
- …signal conditioning for sensors?
- …adding or replacing measurements or sensors?
- …incorporating timing, triggering, or synchronization?
- …leveraging emerging technology trends?
- …multiple disparate software environments and APIs?

National Instruments' Strategy: Graphical System Design

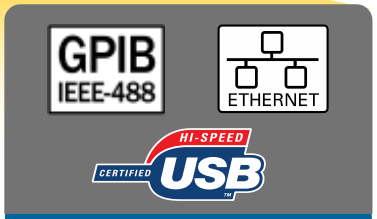
Your Investment in a Platform-Based Approach to Measurements Scales Across…

NATIONAL INSTRUMENTS LabVIEW™

Desktops and PC-Based DAQ

PXI and Modular Instruments

NI CompactRIO



Open Connectivity With Third-Party I/O

Top Benefits of an Integrated Measurement *Platform*

1. Accelerated Productivity
2. Proven Performance and Accuracy
3. Scalability, Adaptability, and Flexibility

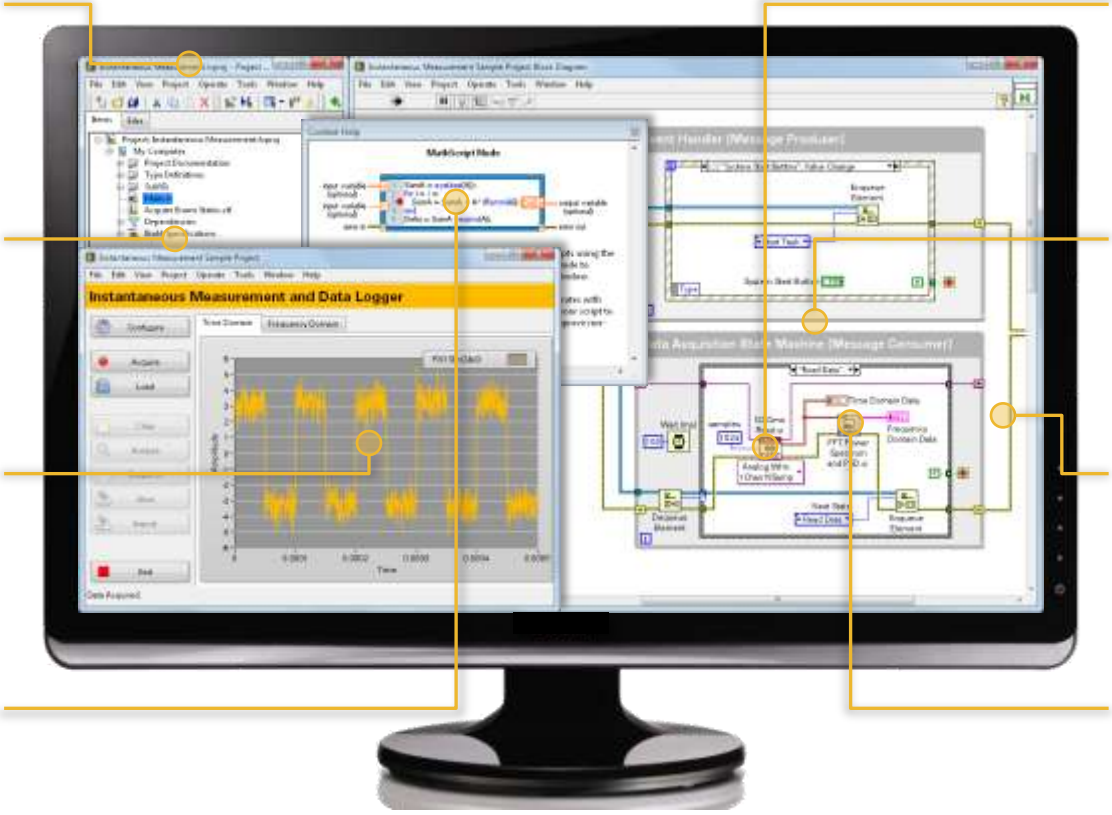
Accelerate Your Productivity With LabVIEW

Unified Software Solution
Manage and organize all system resources in a single software environment.

Deployment Targets
Deploy LabVIEW code to the leading desktop, real-time, and FPGA hardware targets.

Convey With a Clear UI
Create modern user interfaces to display measurements and results.

Integrate Existing Code
Combine and reuse .m files, C code, and HDL with graphical code.



Hardware Connectivity
Bring real-world signals into LabVIEW from any I/O on any instrument.

Parallel Programming
Easily create independent loops that automatically execute in parallel.

Measure in Minutes
Reduce development time with abundant sample projects and templates.

Analysis Libraries
Use built-in high-performance analysis libraries designed for measurement applications.

LabVIEW abstracts low-level complexity and integrates all of the tools engineers and scientists need to build **any** measurement or control system.

Performance and Accuracy You Can Trust

Superior Absolute Accuracy
NI DAQ hardware eliminates errors from temperature drift, offset, gain, and nonlinearity to guarantee measurement accuracy.

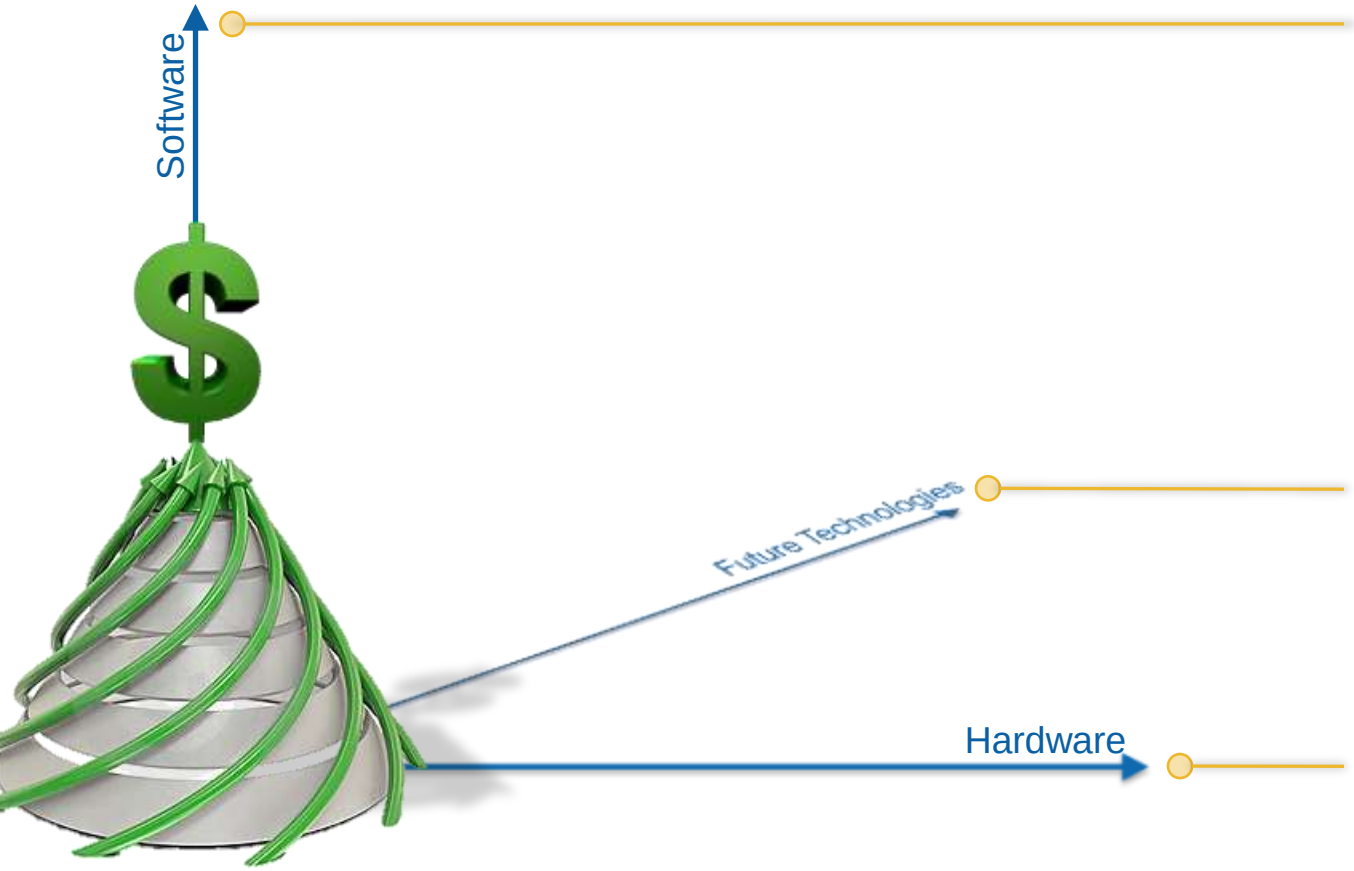
Calibrated to a Standard
NI hardware is calibrated to a traceable standard of known accuracy and NI offers services to support future calibration needs.

Integrated Signal Conditioning
Avoid the headache of custom signal conditioning with hardware that includes circuitry to minimize ground loops and noise.

Maximized Under-the-Hood Performance
Powerful driver technology incorporates data streaming, logging, timing, and synchronization.

NI DAQ is the most trusted computer-based measurement hardware due to innovative design and rigorous testing that yield superior performance and accuracy.

Scalable, Adaptable, and Flexible Solutions Are a Greater Return on Investment



Scalable Software Investment
Avoid fixed-functionality software and reuse code across applications with minimal changes necessary to accommodate hardware upgrades or modifications.

Adaptable Technology Platform
Future-proof your software and hardware investment with a platform that is adaptable to advancing technologies including:

- Bus or connectivity
- Operating systems
- Emerging trends (e.g., mobile, cloud)

Flexible, Configurable Hardware
Choose from a variety of programmable hardware options to create a solution custom to the needs of today's application that is flexible enough for tomorrow.

An integrated measurement platform enables scalability and flexibility across three vectors: software investment, technology future-proofing, and hardware reuse.

Scalability Example: Utilizing Technology Trends With an Integrated Platform

NATIONAL INSTRUMENTS LabVIEW™

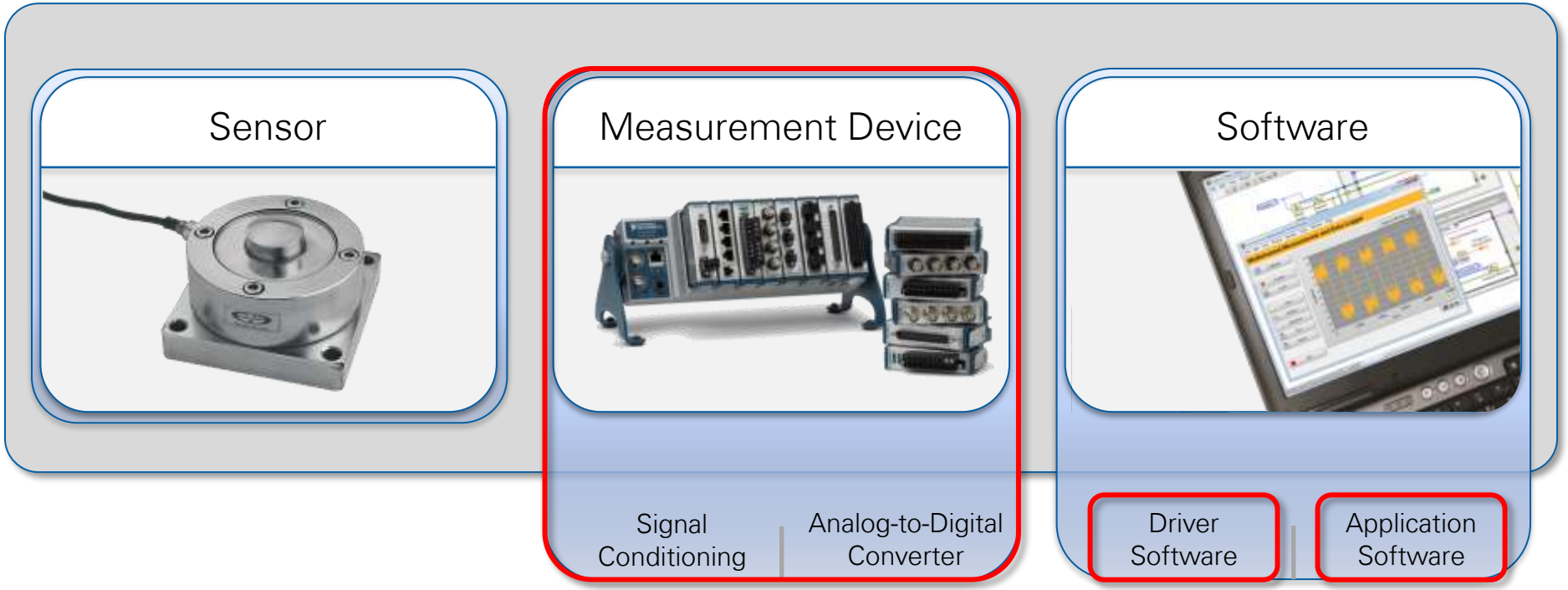
Data Dashboard for LabVIEW

Available on the App Store

Control and visualize data from LabVIEW systems on an iPad

Architecture of an Integrated Measurement System

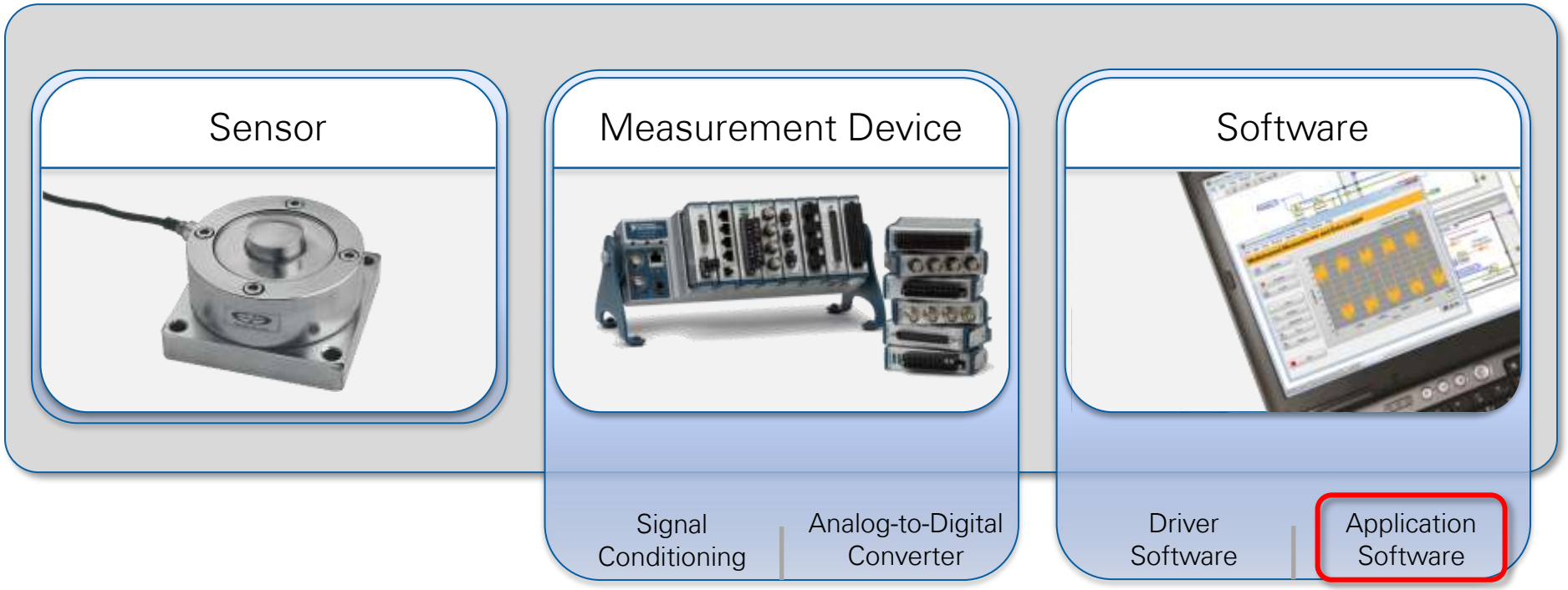
Today, we'll learn about three key differentiating components of a National Instruments data acquisition system:



Architecture of an Integrated Measurement System

NATIONAL INSTRUMENTS LabVIEW™

LabVIEW is system design software that provides engineers and scientists with the tools needed to create and deploy measurement and control systems through unprecedented hardware integration.



Introduction to LabVIEW

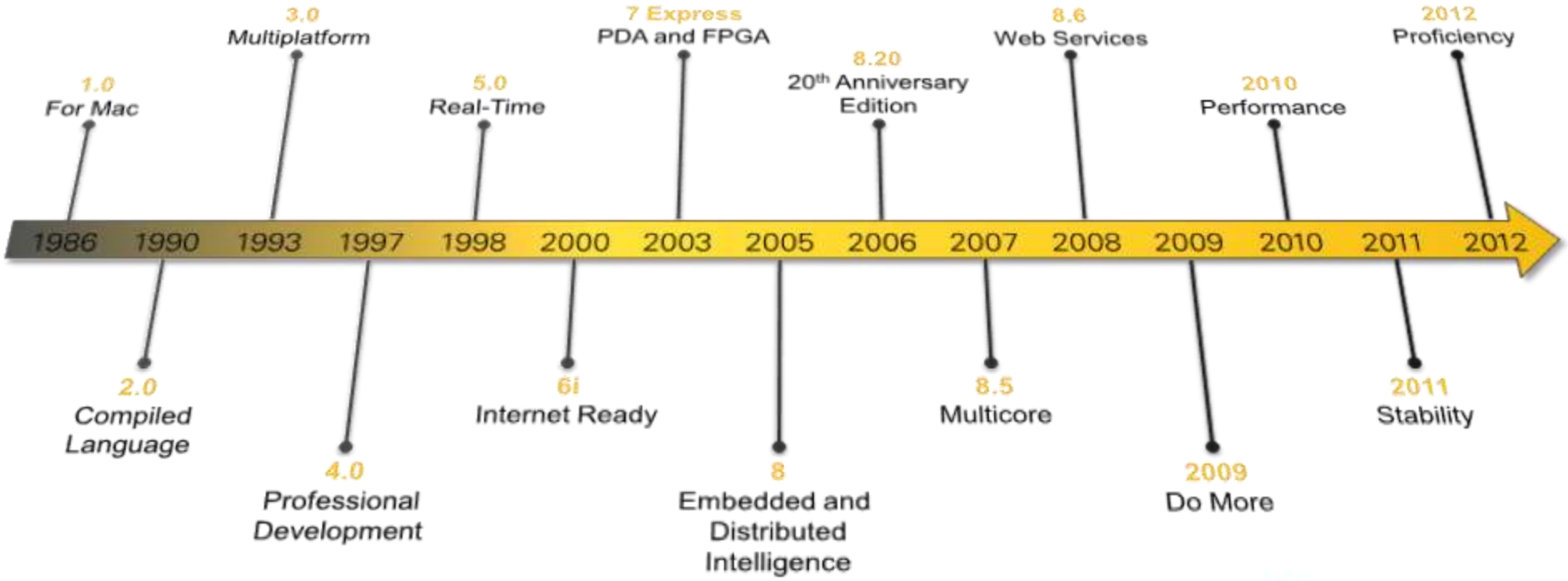
System Design Software for Any Measurement Application

NATIONAL INSTRUMENTS

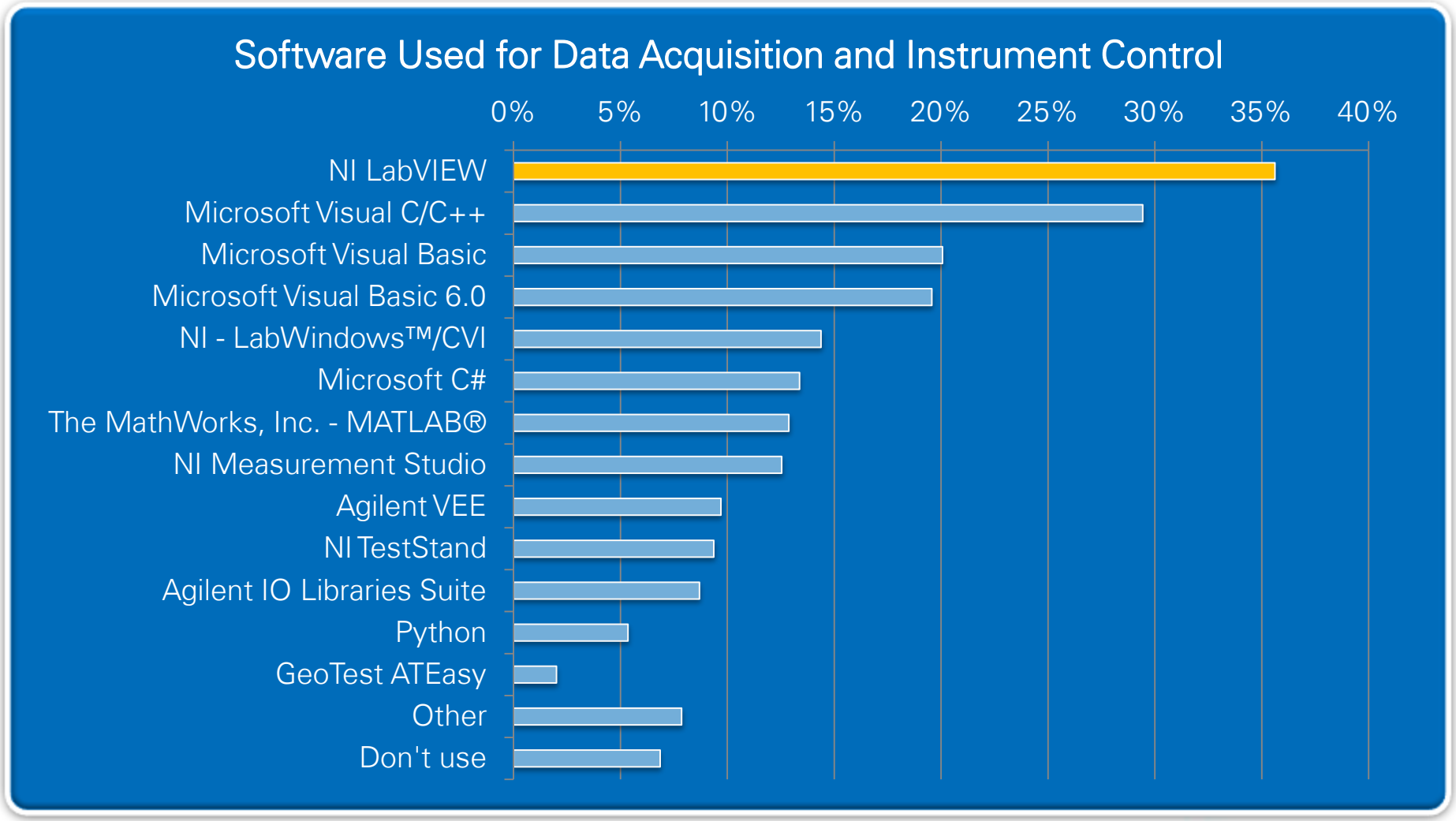
Because It Has Been Proven Over Nearly 30 Years…

Withstanding the test of time across operating systems, buses, technologies, and more

NATIONAL INSTRUMENTS LabVIEW™



…LabVIEW Is the Standard for Making Measurements



Unrivaled Hardware Integration in a Single Environment

- NI hardware
  - 200+ data acquisition devices
  - 450+ modular instruments
  - Cameras
  - Motion control
- Third-party hardware
  - Instrument Driver Network
    - 10,000+ instrument drivers
    - 350+ instrument vendors
    - 100+ instrument types
  - Communicate over any bus

The Foundation of LabVIEW: Virtual Instrumentation

Automation through software led to a realization about fixed-functionality instrumentation…

Redundancy: Power Supplies
Each separate instrument requires its own power supply to run measurement circuitry that captures the real-world signal.

Redundancy: Displays
Instrument vendors provide a limited-quality display per instrument, even though monitor technology is far more advanced.

Redundancy: Processors
Chip manufacturers rapidly enhance processors according to Moore's law, but instruments have fixed processing power.

Redundancy: Memory
PCs can quickly capitalize on a performance boost from a memory upgrade from readily available RAM.

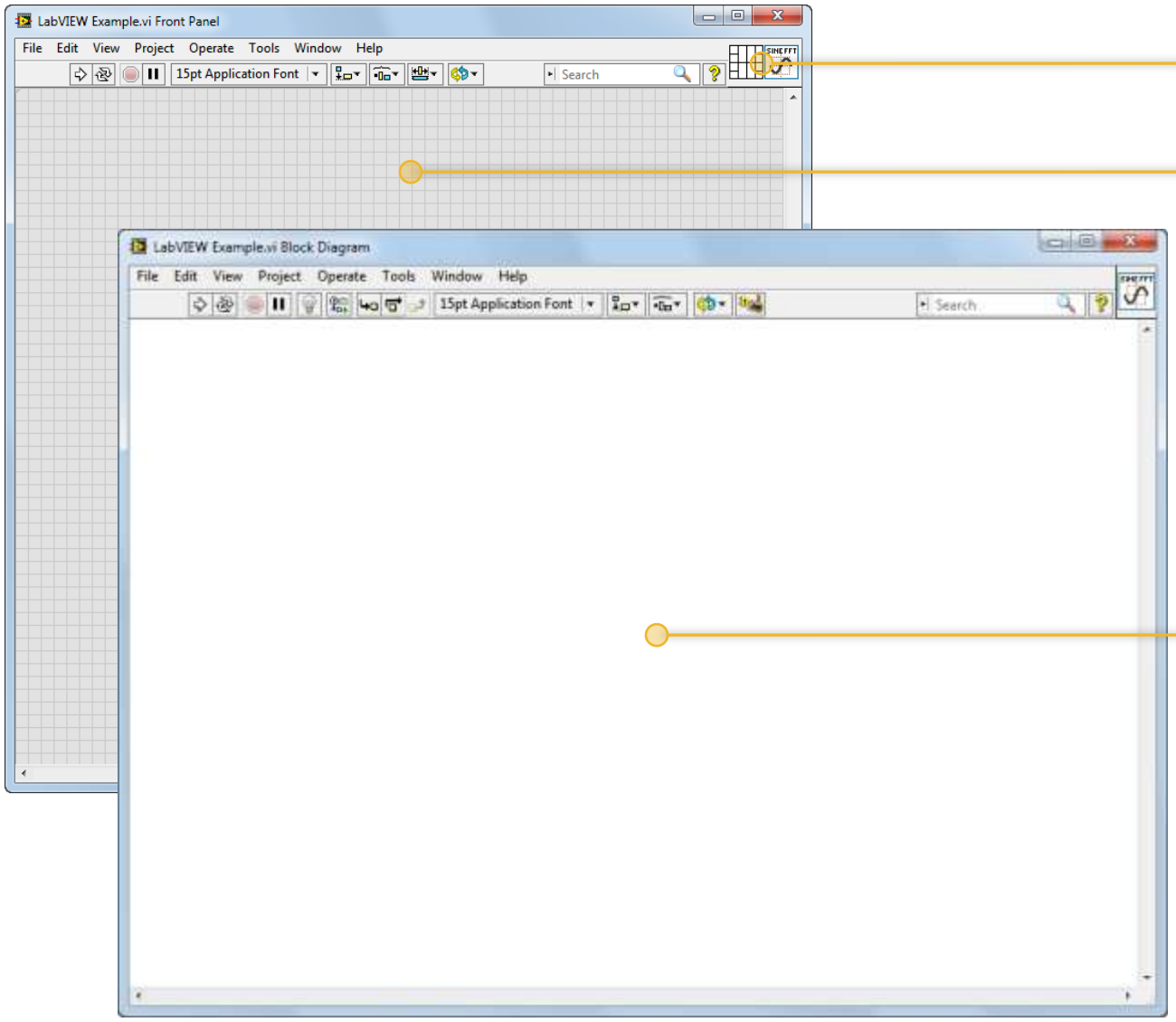
Redundancy: Storage
Each instrument duplicates onboard storage even though PC hard drives are plentiful and cost-effective.

The Foundation of LabVIEW: Virtual Instrumentation

By leveraging COTS PC components, the software becomes the instrument

LabVIEW unlocks the power of instrument and data acquisition hardware by capitalizing on the PC industry and abstracting redundant circuitry.

Therefore, LabVIEW Building Blocks Are Called Virtual Instruments (*.VI)



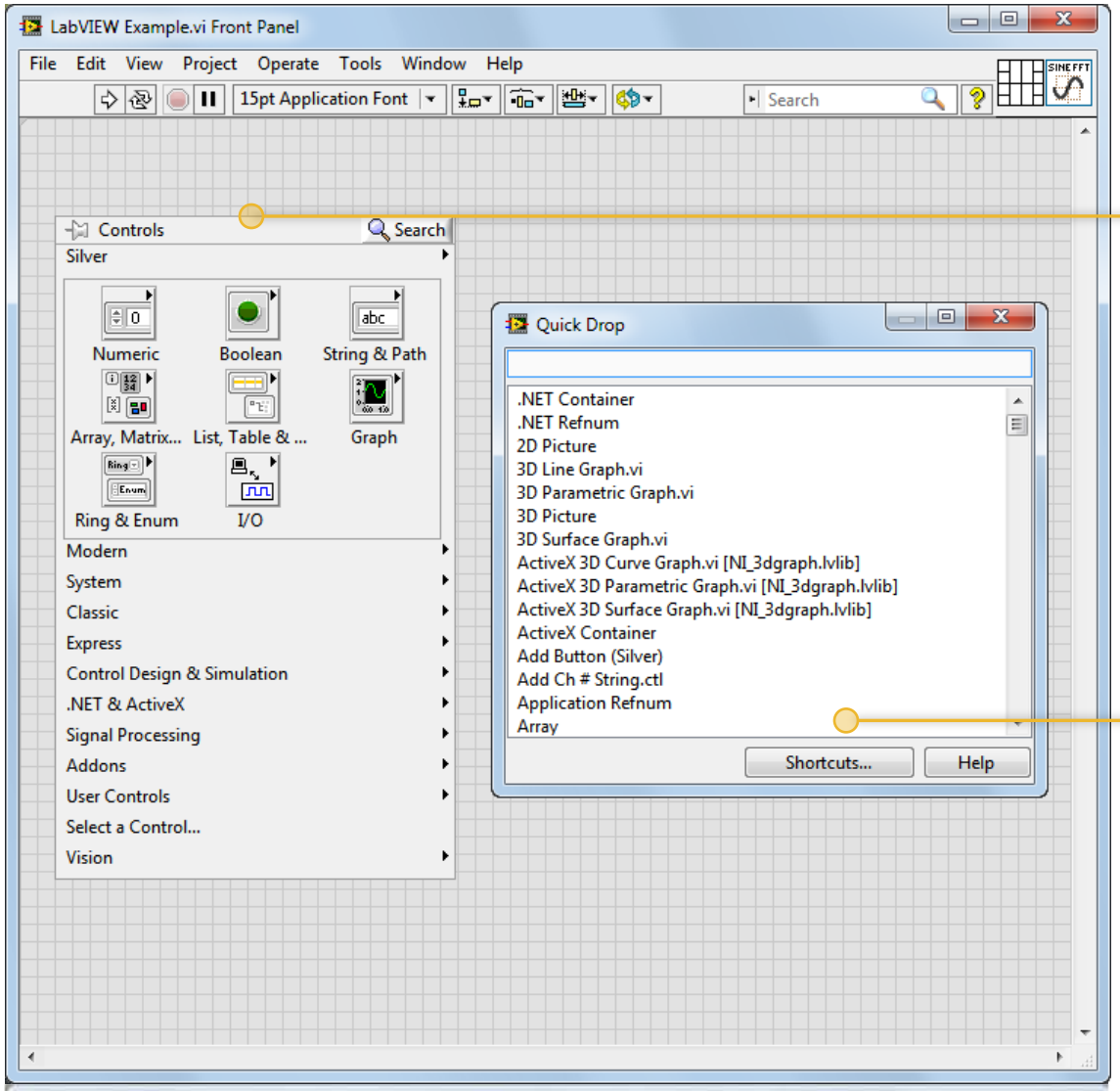
Icon / Connector Pane
Maps inputs and outputs

LabVIEW Front Panel
The user interface of a VI

LabVIEW Block Diagram
The source code of a VI

*Note: A *.vi file encapsulates all three elements*

Creating a LabVIEW Front Panel

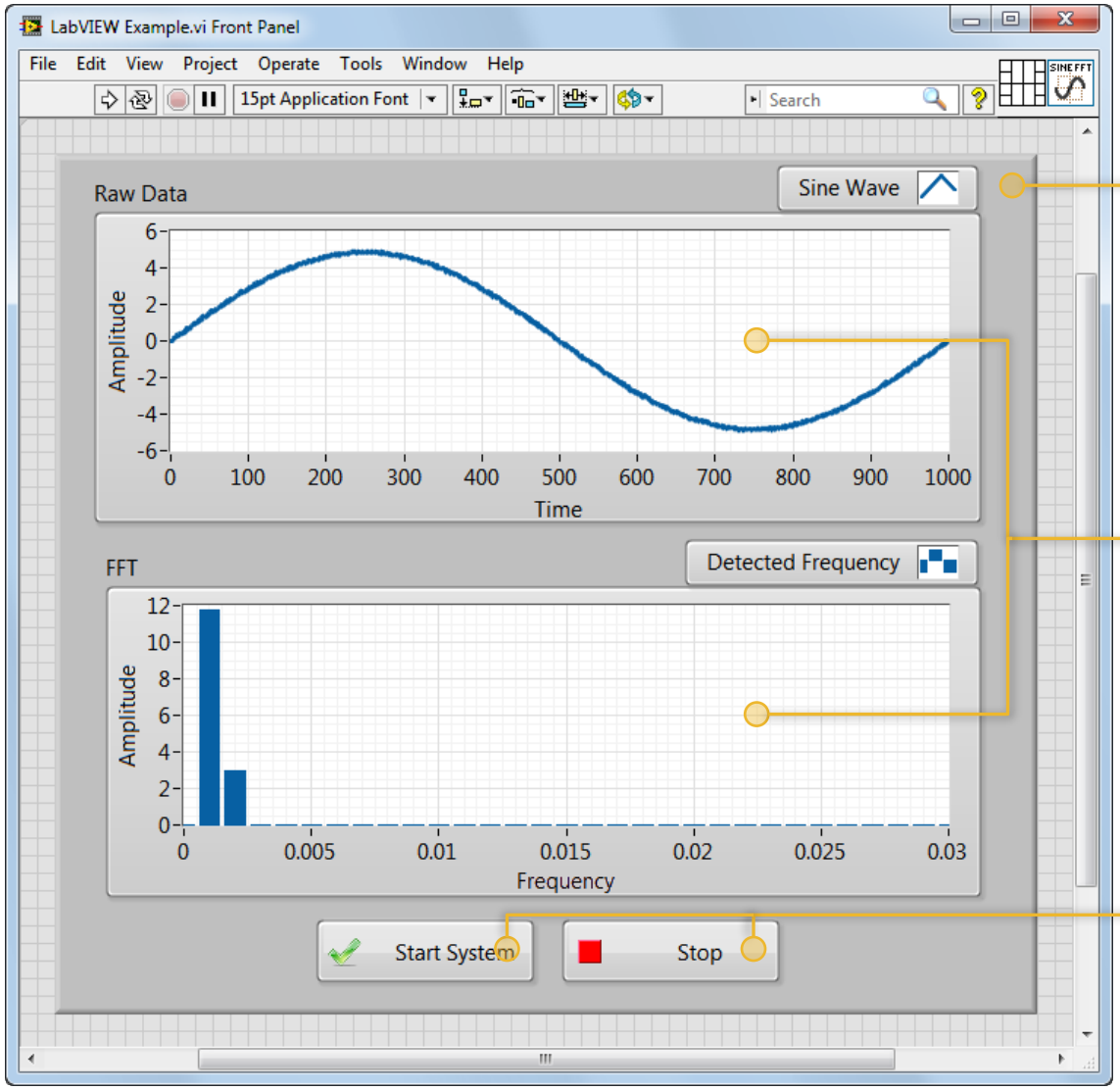


Controls Palette (Right-Click)
Access a hierarchical palette of all front panel elements.

Quick Drop (Ctrl + Space)
Search by object name.

NATIONAL INSTRUMENTS

Front Panel Objects



Decorations

Decorative elements and imagery

- Text
- Arrows
- Callouts
- Lines
- Images
- …and more

Customizable Indicators

Used to convey outputs to a user

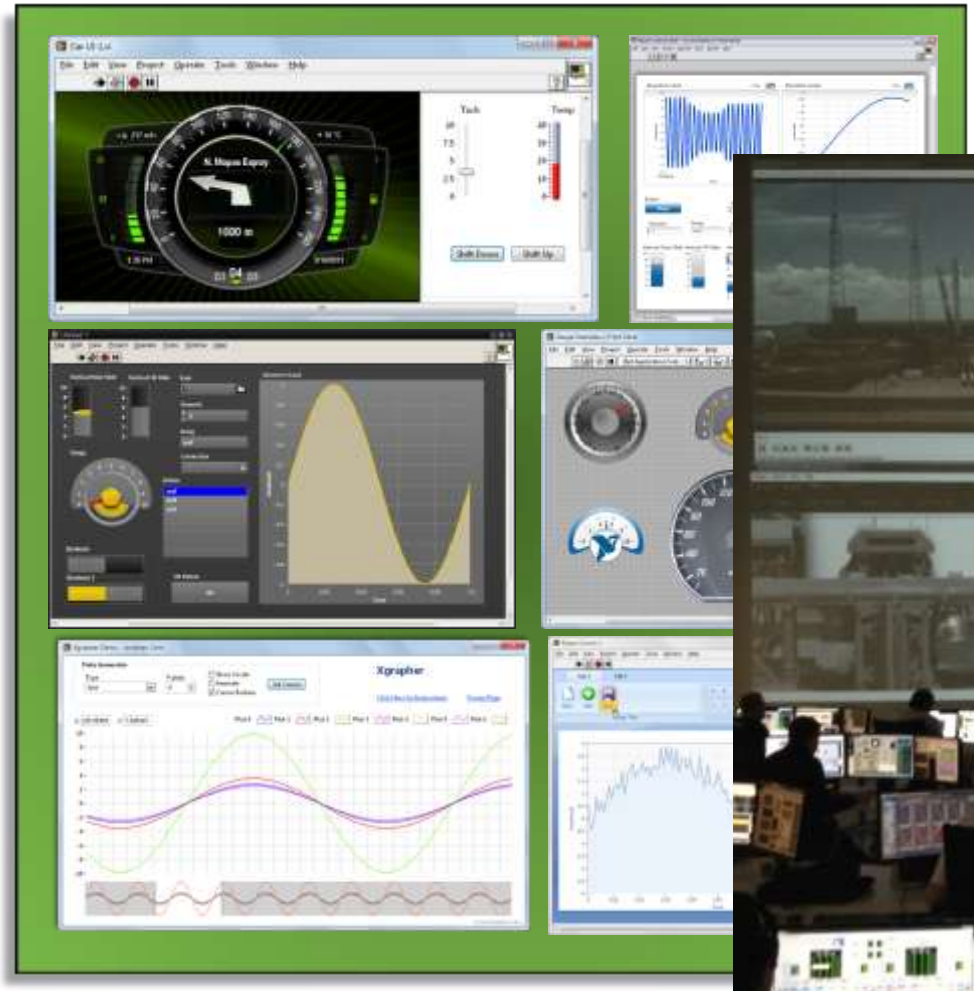
- Graphs and Charts
- Progress Bars
- Gauges and Meters
- LEDs
- Numerics
- Strings and Paths
- …and more

Customizable Controls

Used to receive input from a user

- Knobs and Dials
- Sliders
- Buttons
- Numerics
- Strings and Paths
- …and more

LabVIEW Front Panels in Action

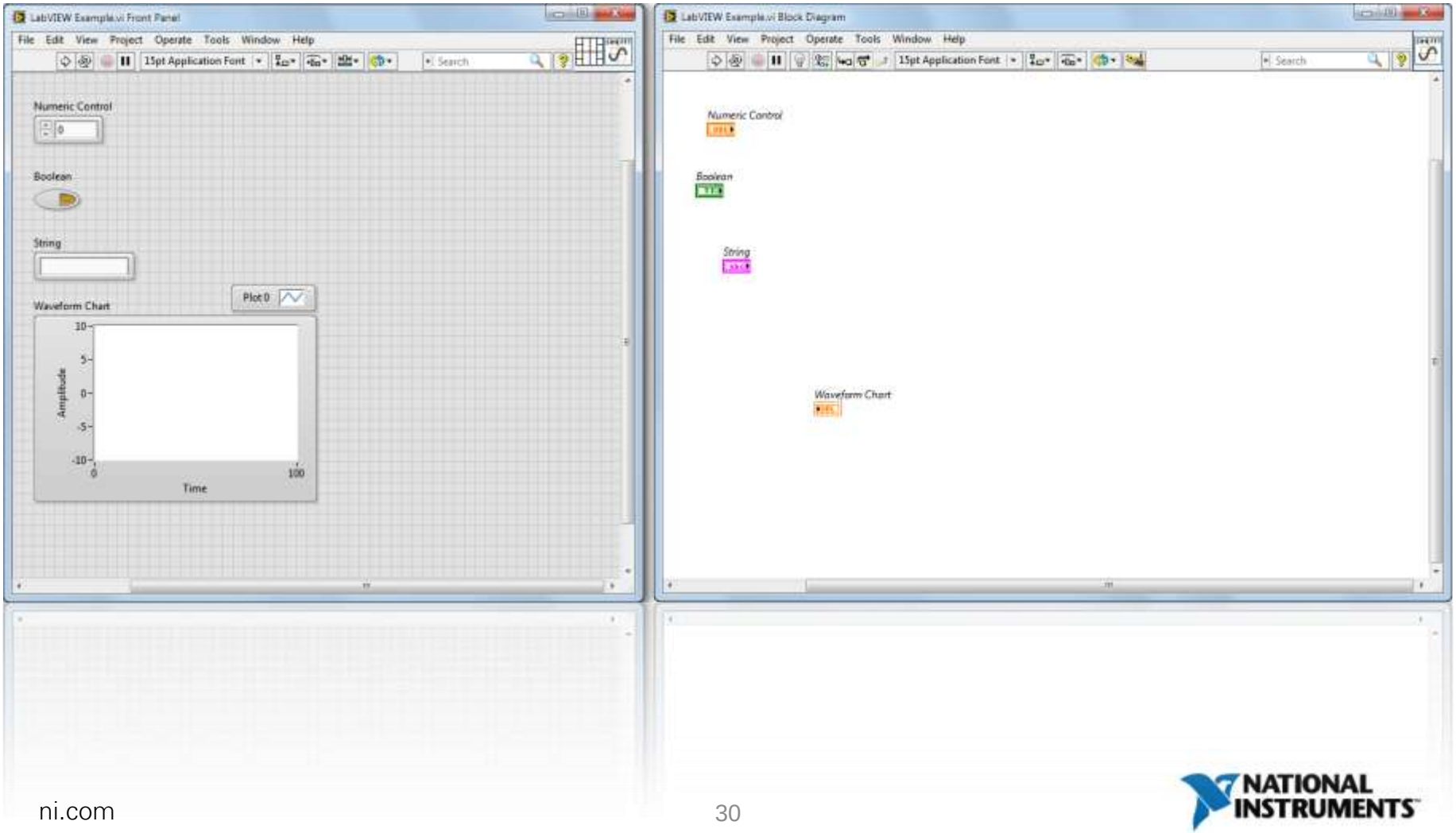


All of the front panels above were contributed for sharing and reuse by members of the global LabVIEW community.

Dozens of LabVIEW front panels at SpaceX Mission Control during successful launch of Dragon
Photo Credit: Elon Musk

All Front Panel Elements Have Block Diagram Terminals

Block diagram terminals provide access to front panel values

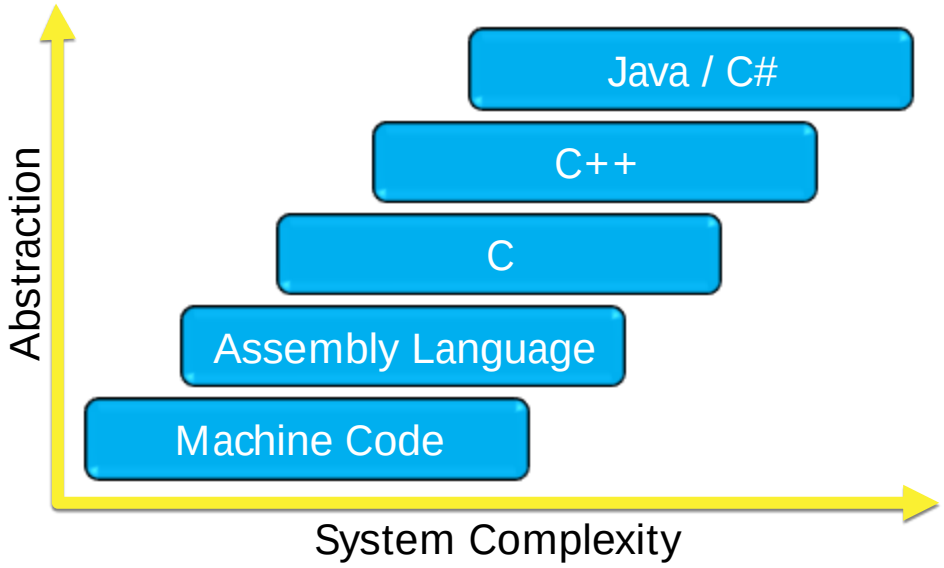


NATIONAL INSTRUMENTS

Examining Traditional Source Code

Humans use abstracted languages because machine code is too hard to comprehend

```
Bool32 CanProbeSignal(OHHandle h, ObjID s)
{
   ObjList *lp;
   ObjID tl, t = 0L;
   int16 *tdp=NULL;
   InstrHandle vi;
   if(s)
   {
      tl= SignalPtr(h,s)->termList;
      DAssert(tl, 0);
      lp= ObjListPtr(h, tl);
      t= lp->nObj? lp->obj[0] : 0;
   }
   if(!t)
      return FALSE;
   tdp= TDPtrFromID(h, oGetDataType(h, t));
   vi = HeapVI(h);
   return DefinedType(tdp) && ((*vi)->instrState & viDebugCapable)
         && !((*vi)->flags & parallelVIMask) && (!((*vi)->instrState & viRetrieveMask)
            || !(*(*vi)->retrieve)->wasEditing);
}

int32 SelectPopUpMethod(OHHandle h, ObjID s, Args *ap)
{
   int32 reply= 0L;
   if (reply = SelectSupPopUp(h, s, ap))
      return reply;
   ap->popup.level++;
   if (ap->popup.doWhat == kBuildPopUp)
   {
      Str255 buff;
      int32  i;
      AppendDiagramListMenu(h, s, ap->popup.level, ap->popup.menu, FALSE);
      if (lvConfigSettings.autoReInitSelectFeatureEnabled)
      {
         InsertObjMenuItem(ap->popup.level, ap->popup.menu, NULL, 0, -1);
         GetPopUpString(miSelNodeAutoInit, buff);
         i = InsertObjMenuItem(ap->popup.level, ap->popup.menu, buff, miSelNodeAutoInit, -1);
         MCheckItem(ap->popup.menu, i, (SelNodePtr(h,s)->flags & autoReInitSelMask));
      }
   }
   else if (ap->popup.doWhat == kSelectPopUp)
      reply= SeqSelPopUpSelect(h, s, ap, FALSE);
   return reply;
}
```

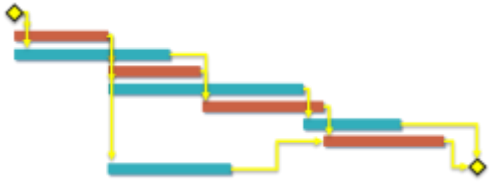


NATIONAL INSTRUMENTS

We Live in a Graphical, Parallel World

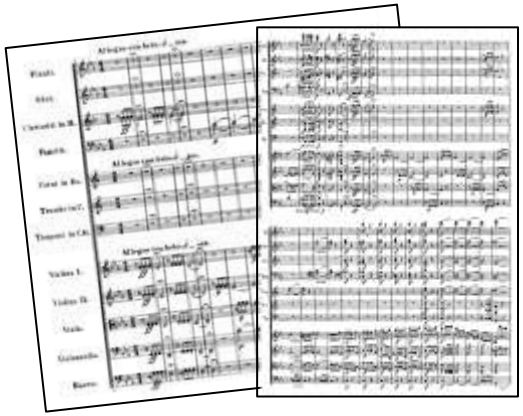
…But what if everything was represented using sequential, textual syntax?

Gantt Chart



Football Play

Musical Score



Our World

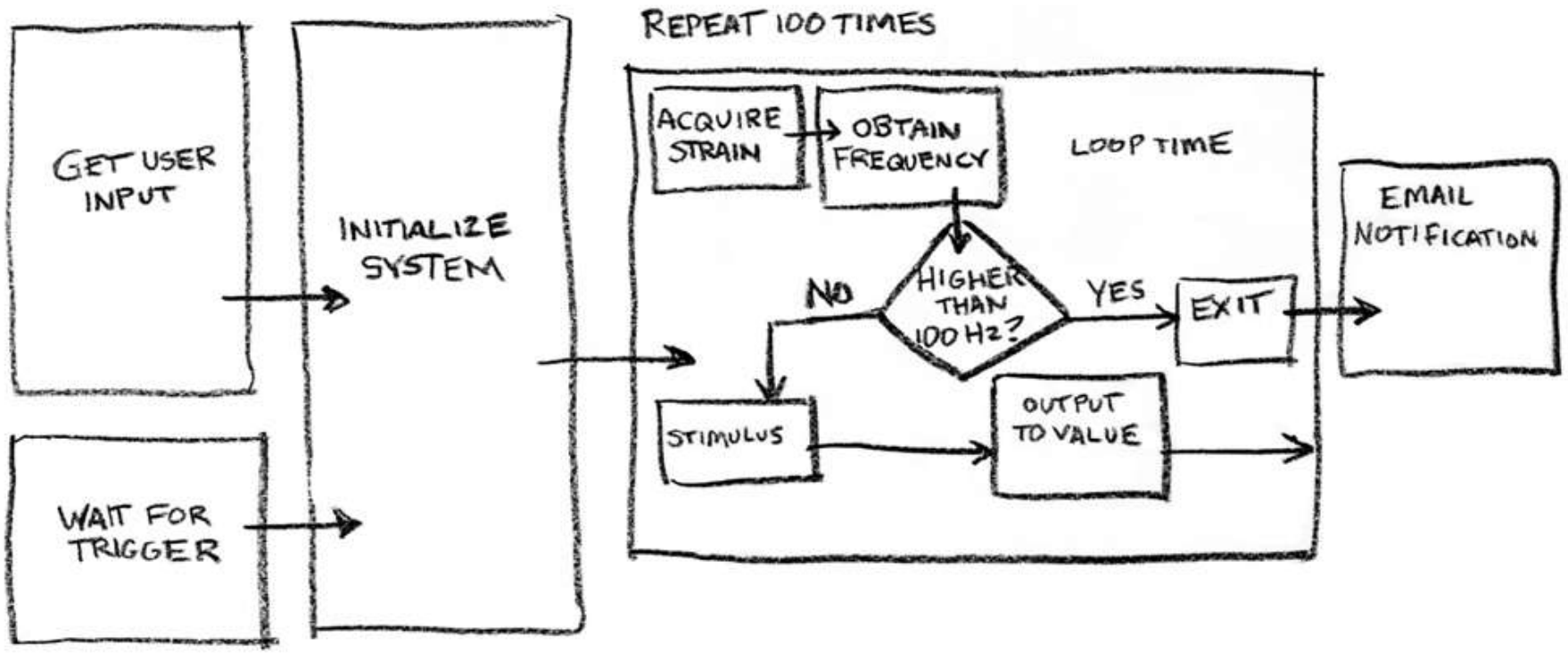
A World Without Graphical

Begin Project
Simultaneously Begin Tasks A and B
When Task A Ends,
Simultaneously Begin Tasks C, D, and H
When Tasks B and C Both End,
Begin Task E
When Task D Ends,
Begin Task F
When Task E Ends, If Task H has Ended,
Begin Task G
When Task F and G End,
Finish Project

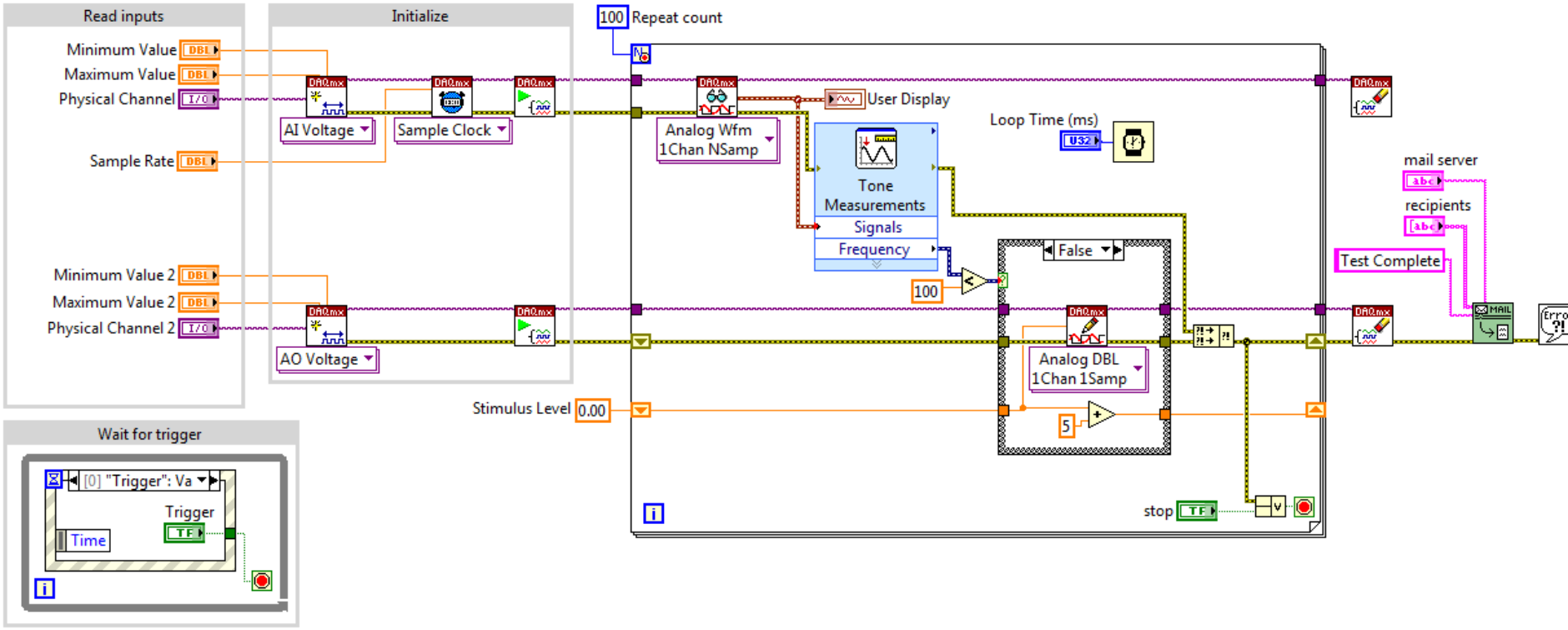
Align in Split-Back Formation
Center Hikes Ball to Quarterback
Simultaneously,
    Center Blocks Defensive Tackle
    Quarterback Hands Ball to Tailback
    Offensive Tackles 1-4 Block Defensive End
    Wide Receiver Right Runs In Route
    Wide Receiver Left Runs Screen Route
    Tight End Blocks Linebacker
    Tailback Runs Through Center Hole
    Fullback Blocks Middle Linebacker
End Play

Begin Song
Rest Two Beats in ¾ Time
While Three Iterations Haven't Been Played,
    Left Hand Plays Low C, G, and Middle C
    And Right Hand E, G, and High C
    Hold for Two Beats
    Pause for One Beat
    Left Hand Plays Low A, D, F
    And Right Hand Plays High F, A, F
    Hold for Three Beats
Repeat
End Song

With LabVIEW, You Can Program the Way You Think



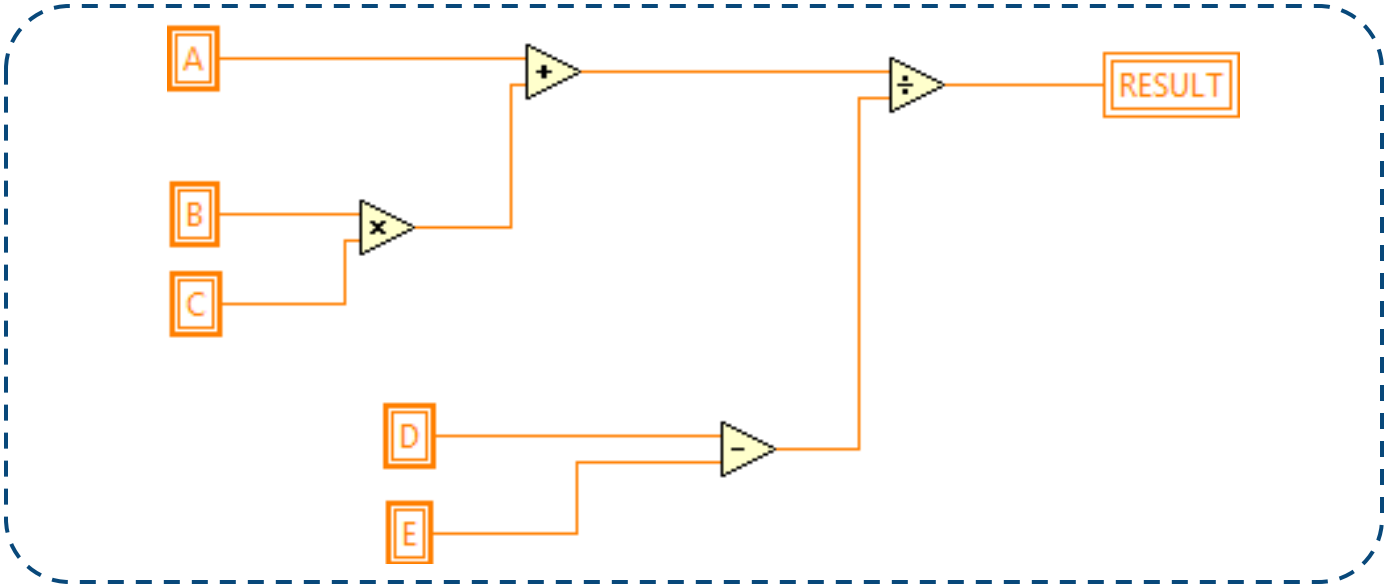
With LabVIEW, You Can Program the Way You Think



The graphical, dataflow-based G programming language is ideal for programming parallel data acquisition hardware.

What Is Data Flow?

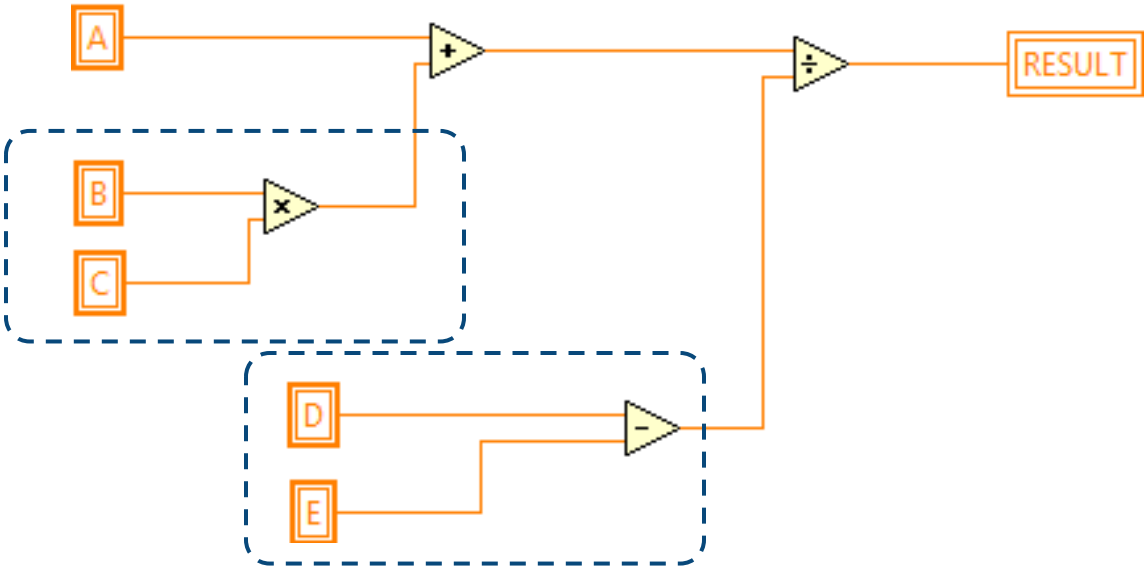
- Each block diagram node executes only when it receives all inputs
- Each node produces output data after execution
- Data flows along a path defined by wires
- The movement of data determines execution order



Formula: Result = (A+B*C) / (D-E)

What Is Data Flow?

- Each block diagram node executes only when it receives all inputs
- Each node produces output data after execution
- Data flows along a path defined by wires
- The movement of data determines execution order

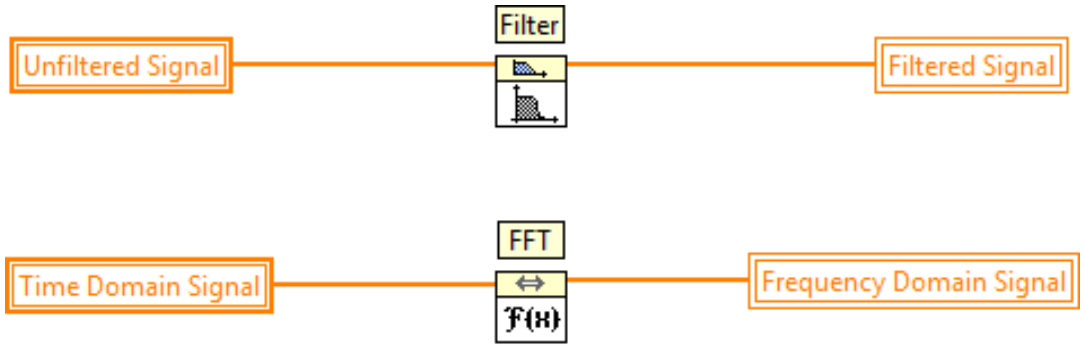


The [Multiply] and [Subtract] operations can execute at the same time since they don't have any data dependencies.

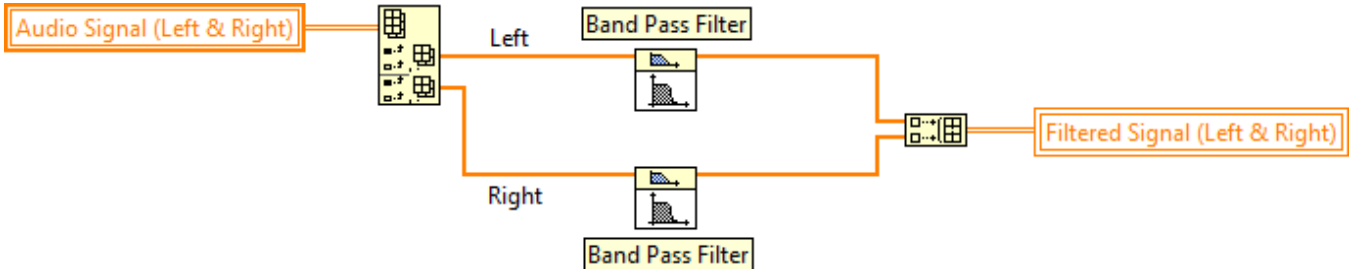
Dataflow Languages Naturally Express Parallelism

The LabVIEW compiler will automatically multithread code expressed in parallel

Task Parallelism

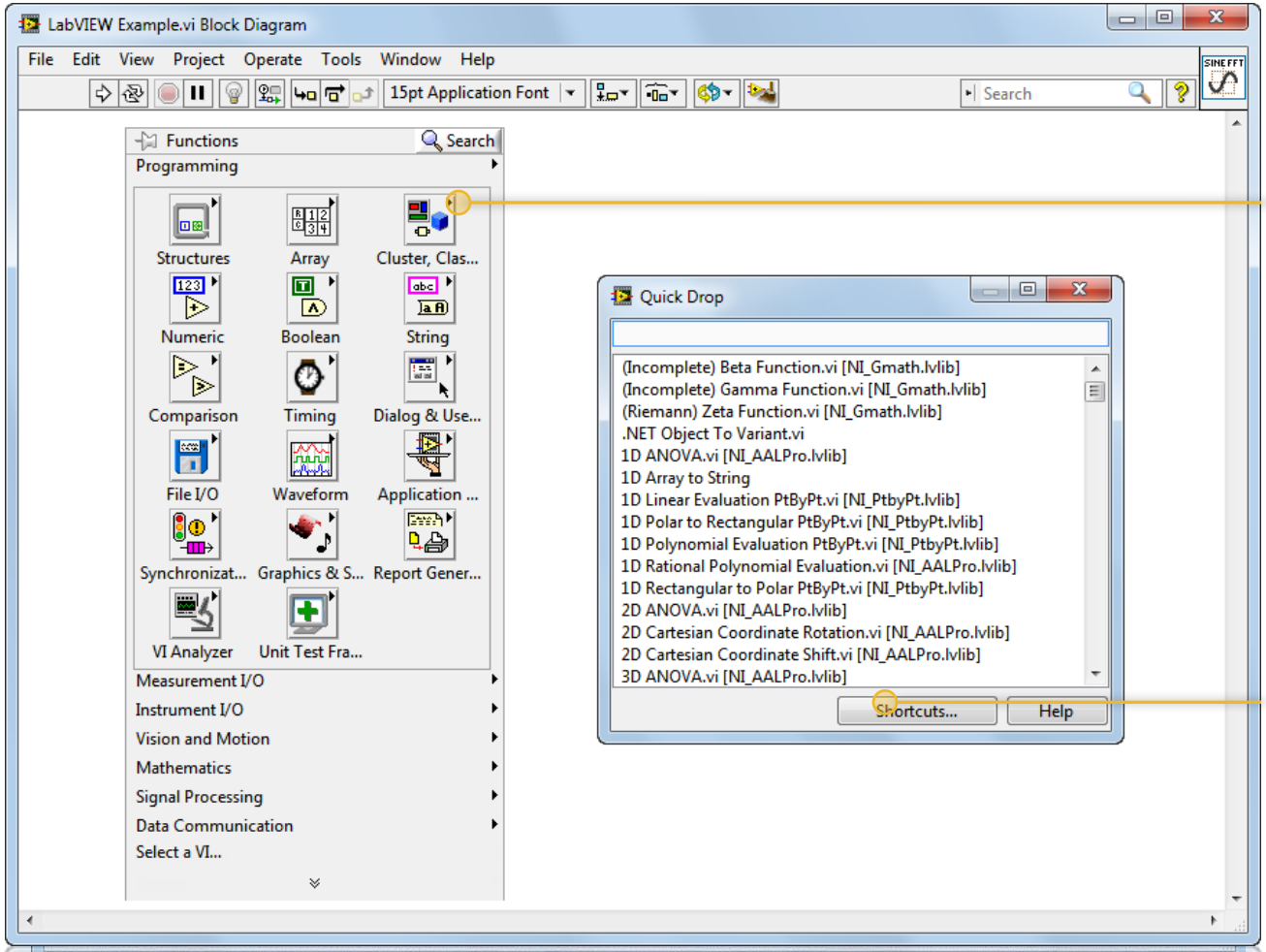


Data Parallelism



NATIONAL INSTRUMENTS

Creating a LabVIEW Block Diagram

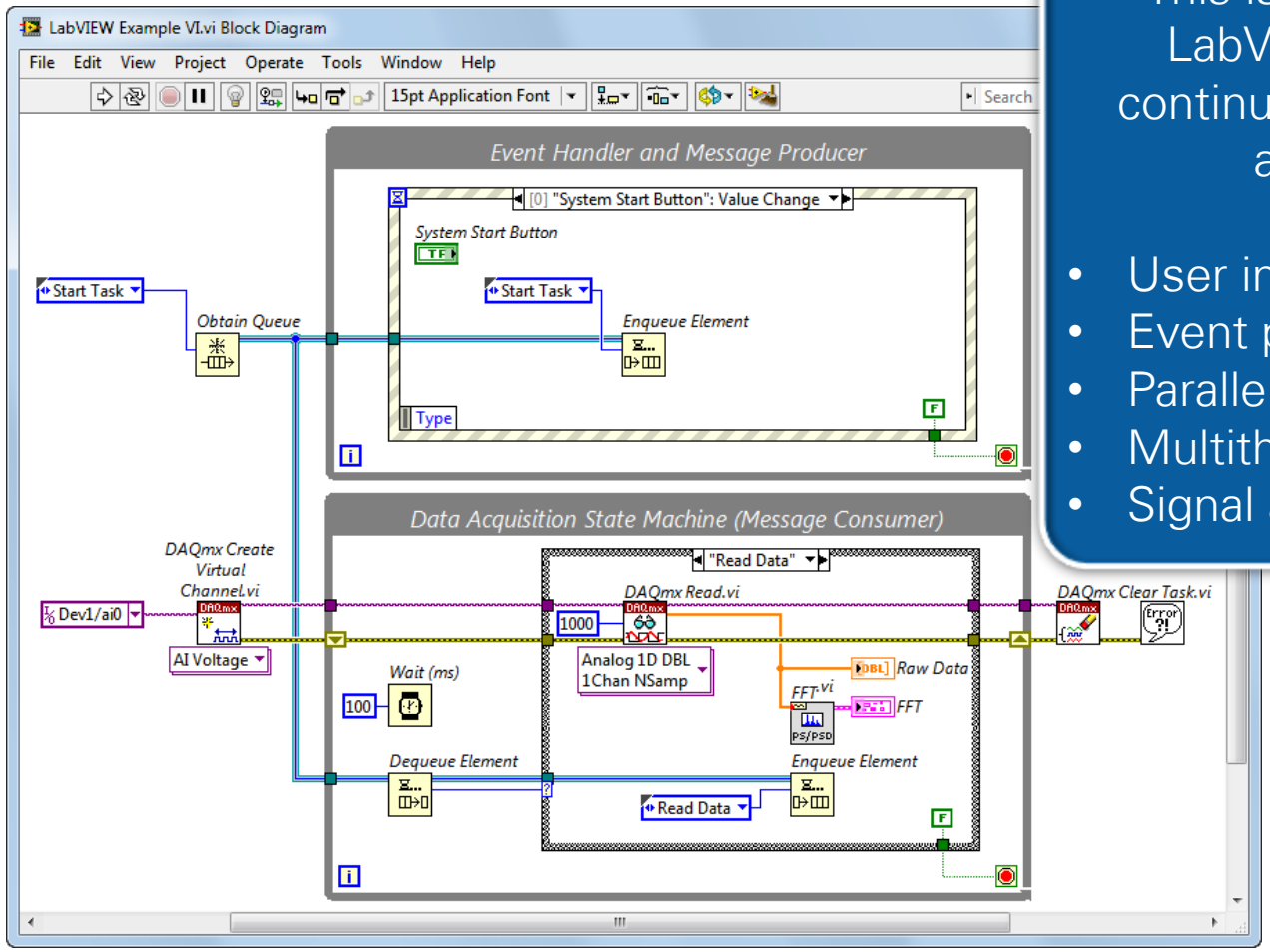


Functions Palette (Right-Click)
Access a hierarchical palette of all block diagram functions.

Quick Drop (Ctrl + Space)
Search by object name.

NATIONAL INSTRUMENTS

Exploring a LabVIEW Block Diagram

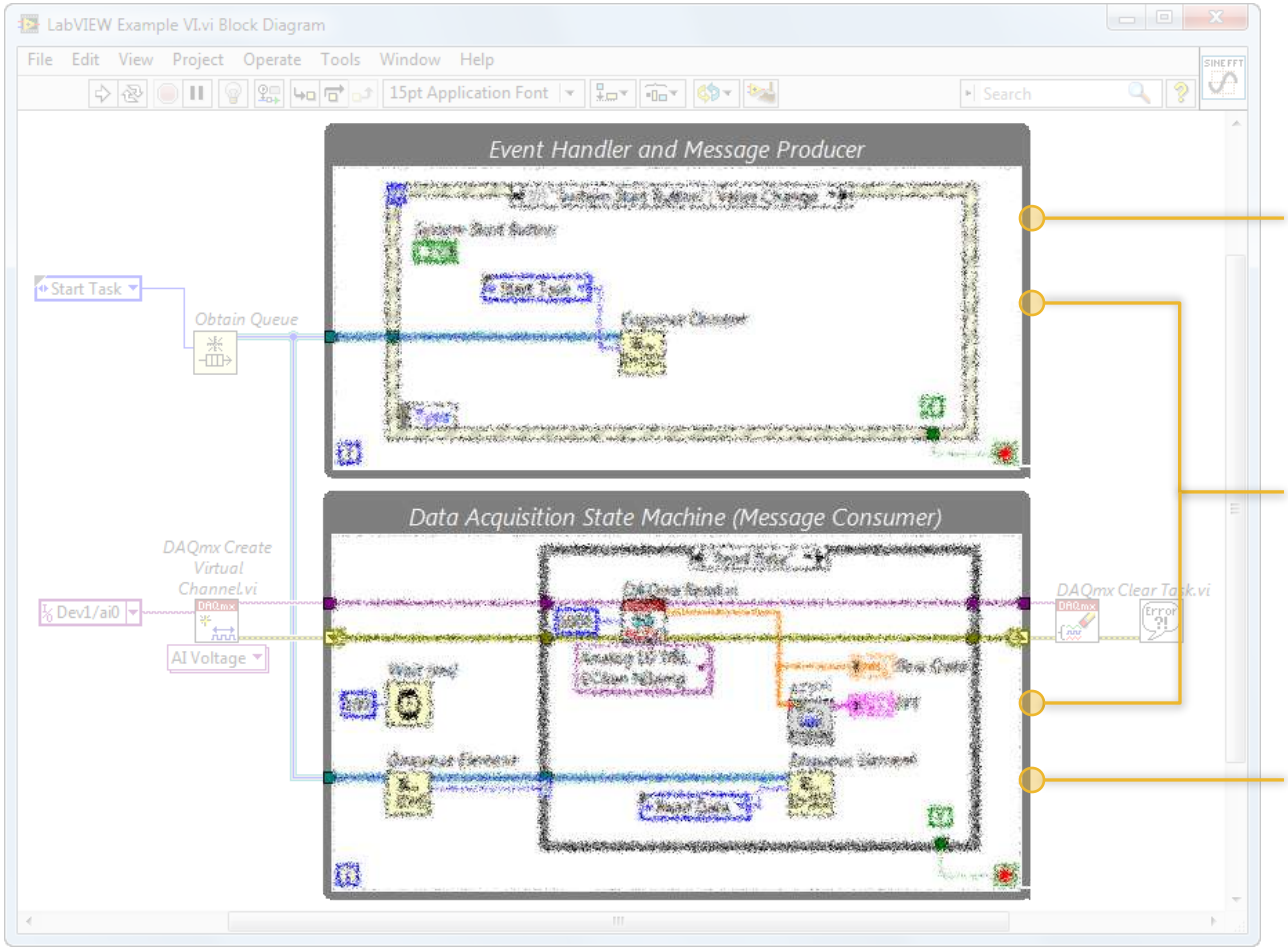


This is a typical, fully functional LabVIEW block diagram for a continuous voltage measurement application featuring:

- User interface handling
- Event processing
- Parallelism
- Multithreaded data transfer
- Signal analysis

We'll dissect the components of this source code in the following slides.

Exploring a LabVIEW Block Diagram



UI Thread
This thread handles events from the front panel (user interface).

Two Asynchronous Threads
Because no data is passed between entities, they execute in parallel.

DAQ Thread
This thread interacts with data acquisition hardware.

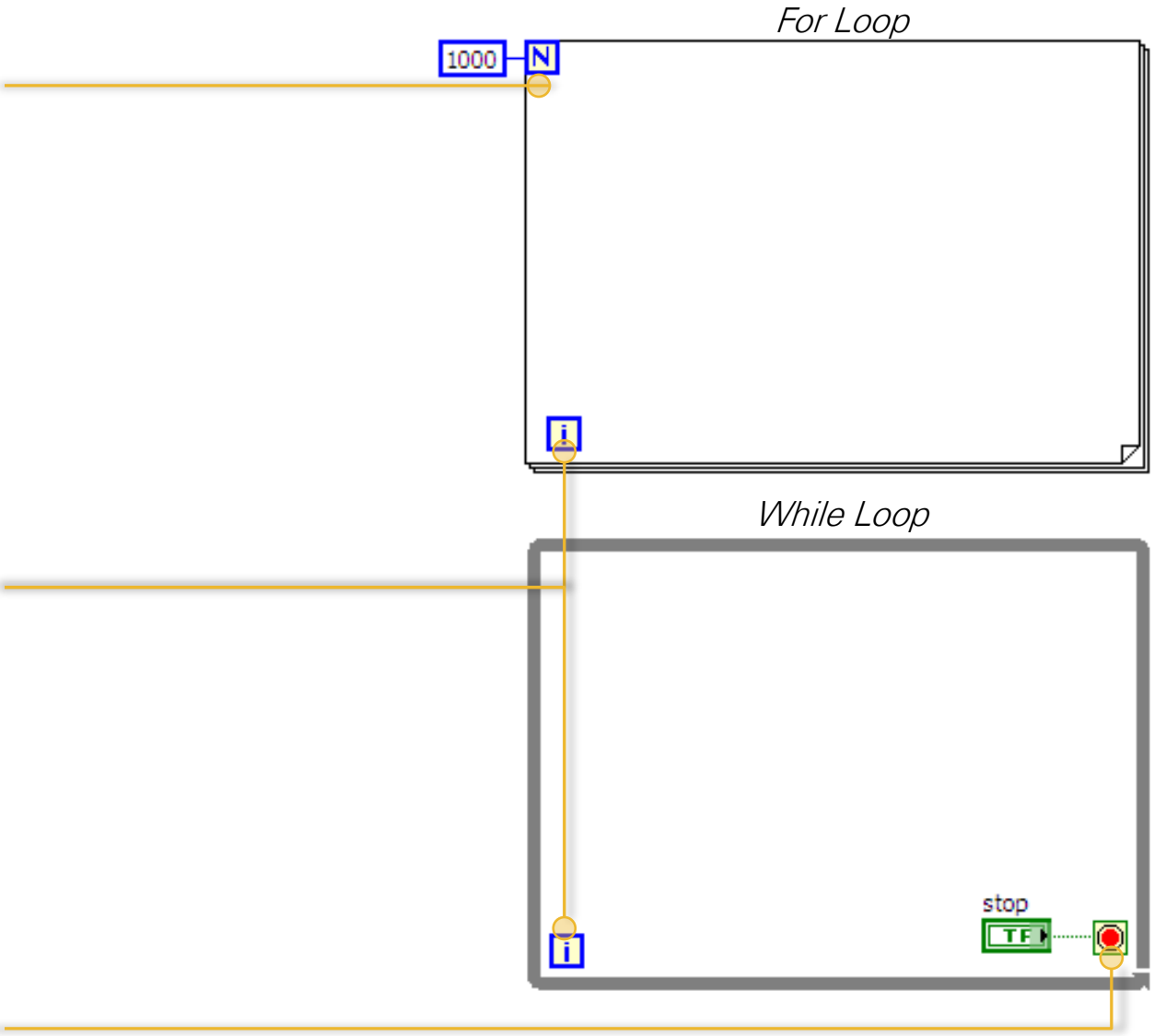
Any block diagram entity that can contain code within it is called a structure.

Execution Control Structures: Loops

Count Terminal
The code contained within this For Loop will execute N times.

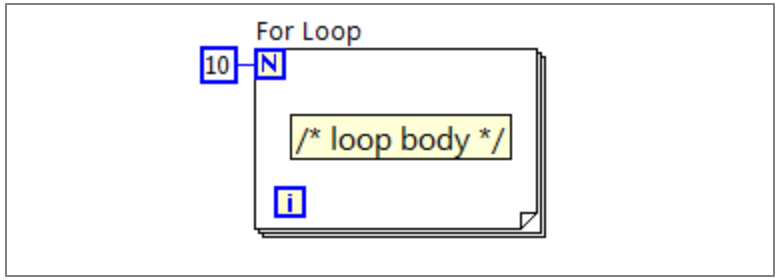
Loop Iteration Terminals
This provides the current loop iteration count, which ranges from 0 to N-1.

Conditional Terminal
The code within this While Loop will run until a True value is evaluated.

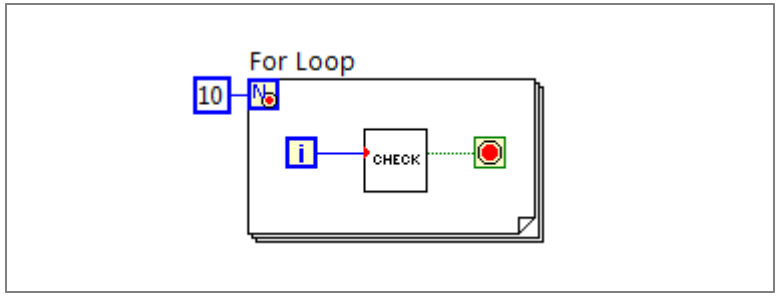


Text Loops and Their LabVIEW Equivalents

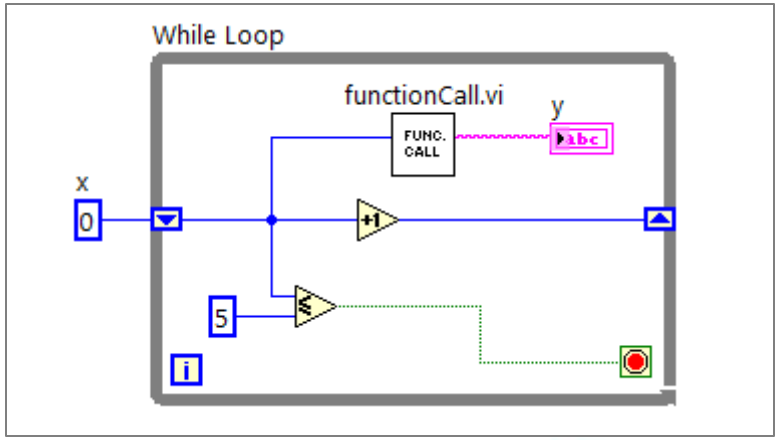
```
for (i = 0; i < 10; i++)
{
    /* loop body */
}
```




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




```
int x = 0;
String y;
while (x < 5)
{
    y = functionCall(x);
    printf(y);
    x++;
}
```



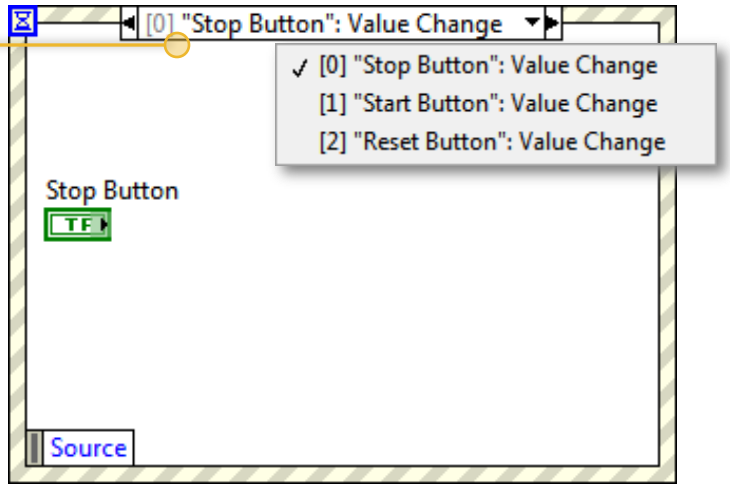
Event and Case Structures

Event Selector Label
This indicates which subdiagram is visible and details the event that the code within the diagram handles.

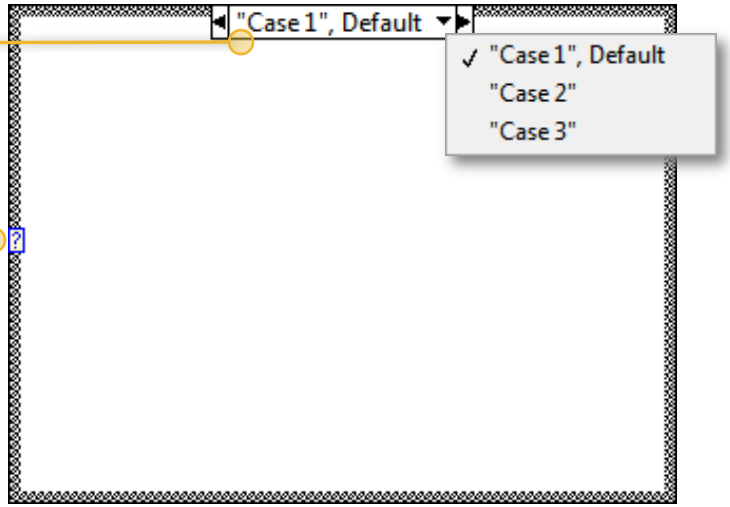
Case Selector Label
This indicates which subdiagram is visible.

Selector Terminal
The value wired to this terminal determines which of the subdiagrams, or cases, will execute.

Event Structure

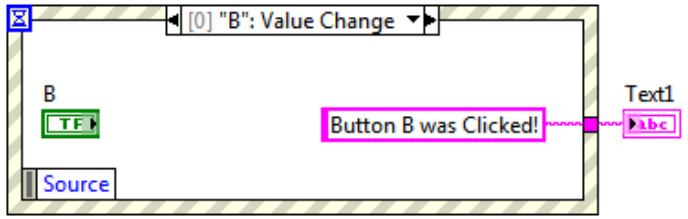


Case Structure

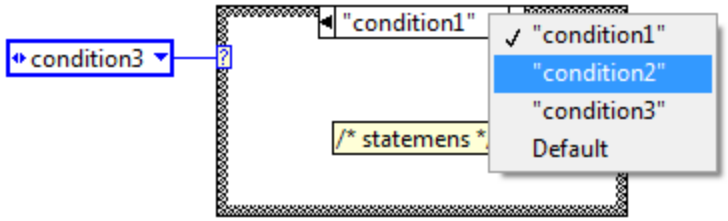


Text Events, Cases, and Their LabVIEW Equivalents

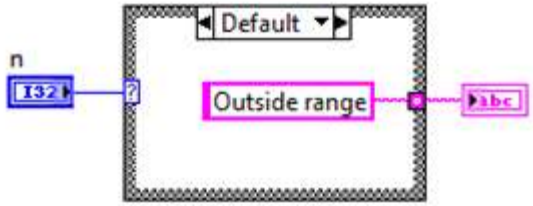
```
Button B = new Button();
B.Click += new RoutedEventHandler(OnBClick);

void OnBClick(object Source)
{
      Text1.Text = "Button B was Clicked!";
}
```



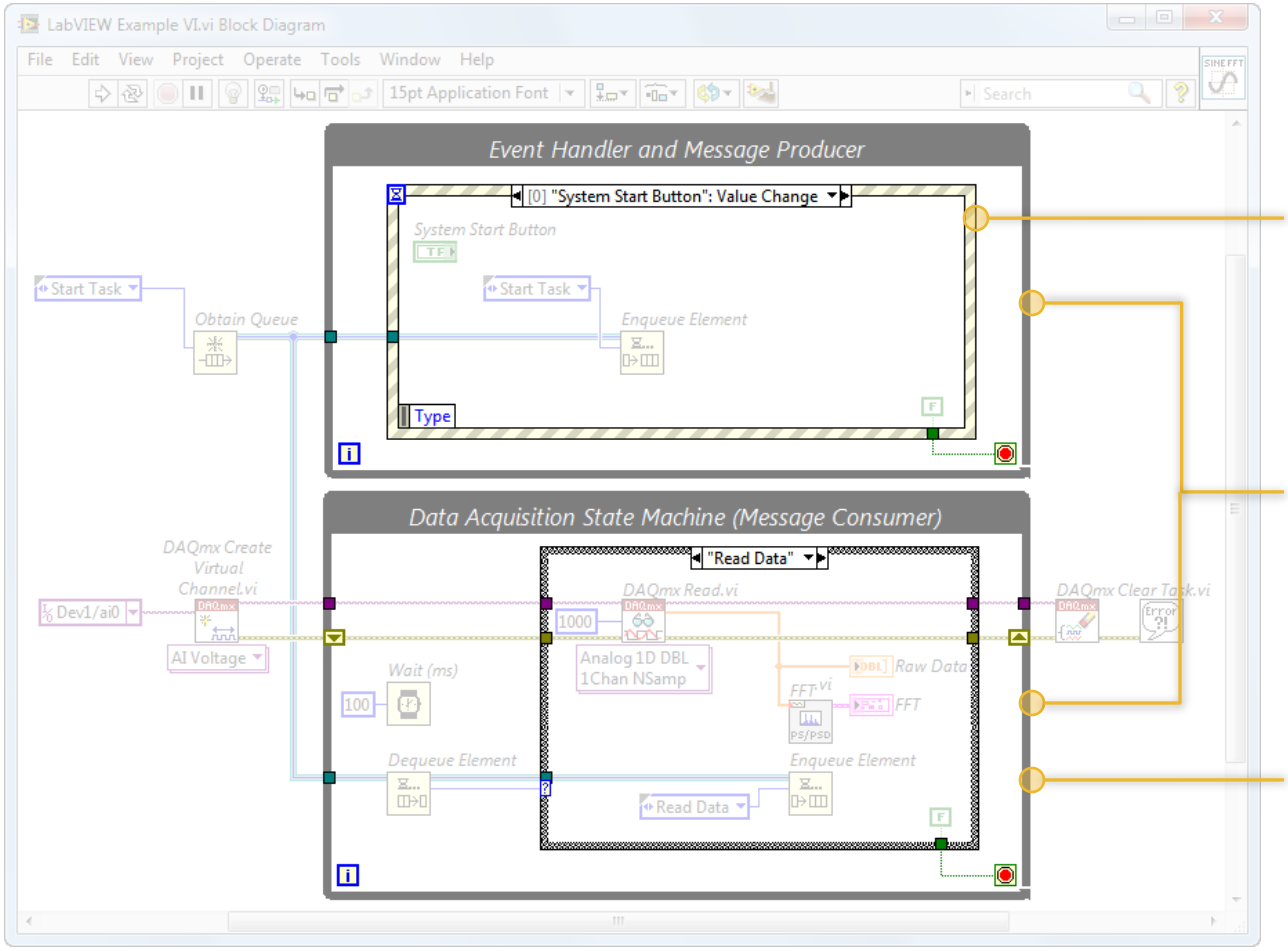

```
if condition1 then
  -- statements;
elseif condition2 then
  -- more statements
elseif condition3 then
  -- more statements;
else
  -- other statements;
end if
```




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



Exploring a LabVIEW Block Diagram



Event Structure

Executes different subdiagrams based on events and interrupts

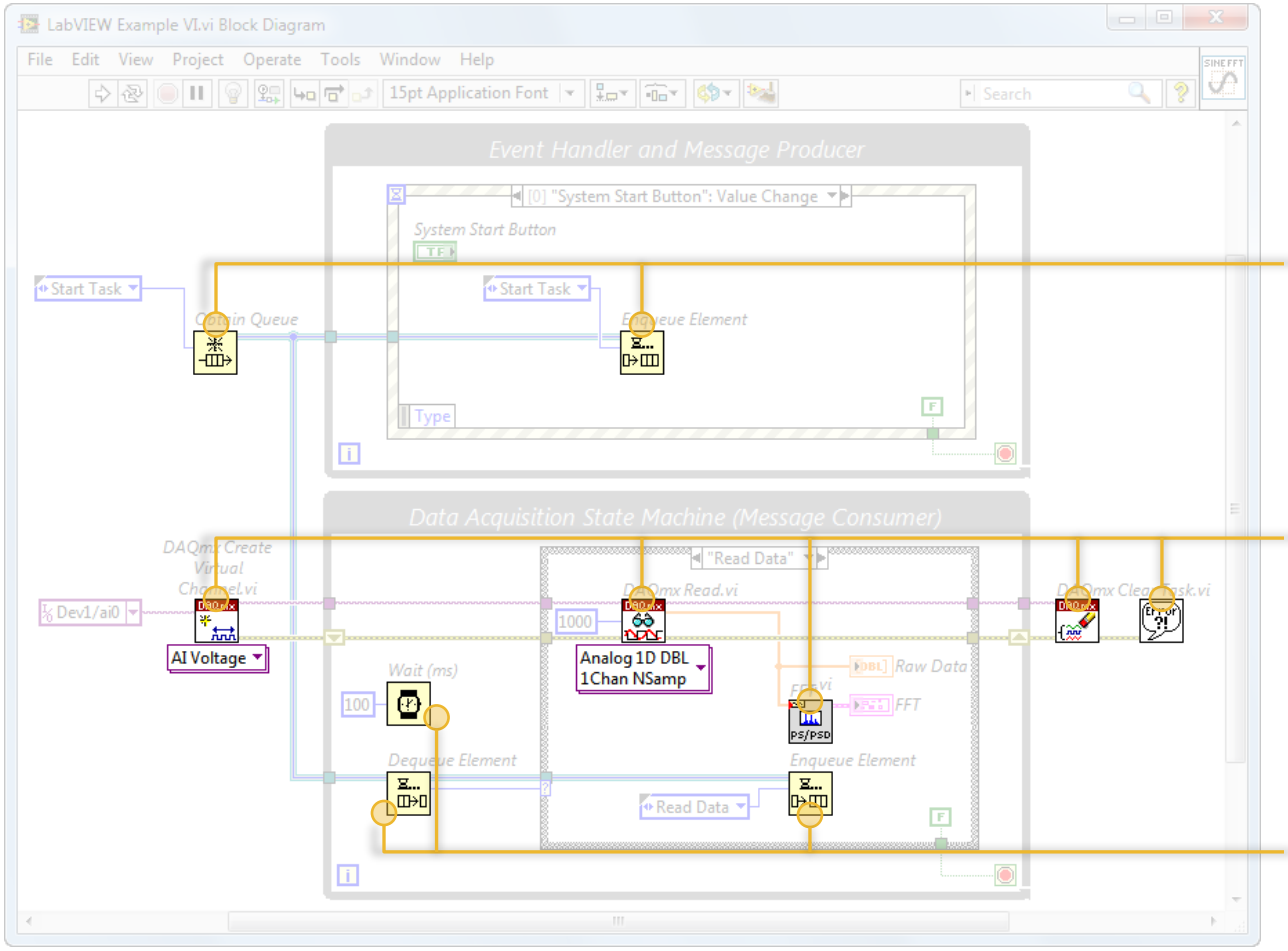
While Loops

Iterate continuously until a true value is passed to the stop terminal

Case Structure

Executes different subdiagrams based on the value of its selector terminal

Exploring a LabVIEW Block Diagram



Primitive Functions
Yellow subVIs are a native part of the G language and cannot be modified.

Standard Functions
These subVIs could be user created or may be part of a driver, library, or toolkit.

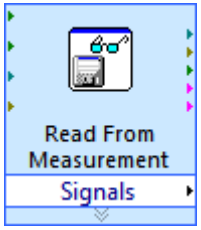
Primitive Functions
Yellow subVIs are a native part of the G language and cannot be modified.

These static calls to LabVIEW functions (SubVIs), do not execute until data has arrived at all input terminals. When complete, outputs will be populated with values so that execution can continue.

LabVIEW Functions Are as Complex as You Need

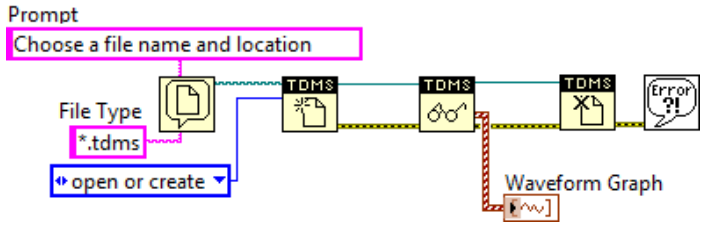
Express VIs

- Quick and Easy
- Configuration-Based
- Limited



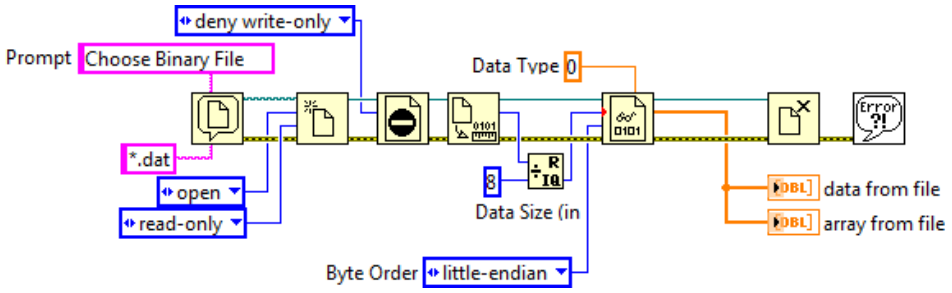
Regular VIs

- Hides Unnecessary Details
- Retains Power and Flexibility



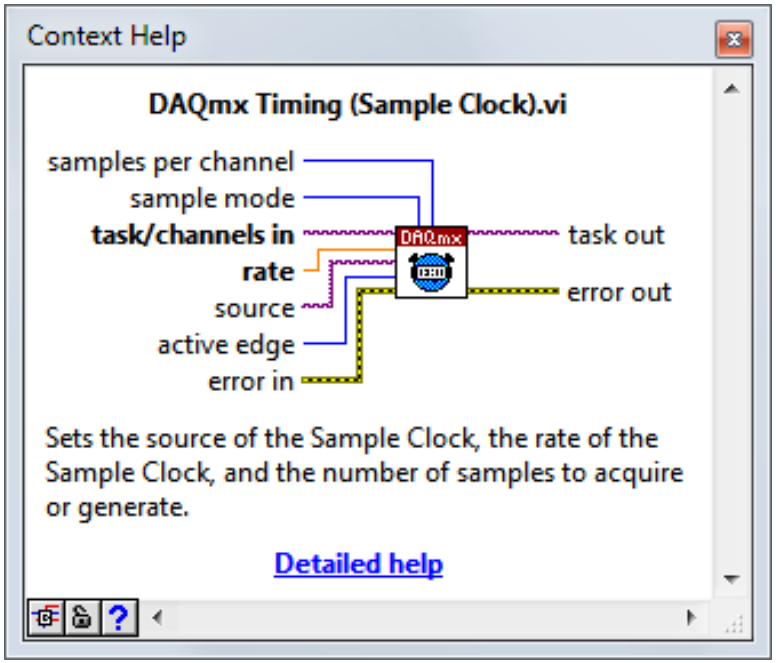
Low-Level VIs

- Powerful, Flexible
- Difficult, Time-Consuming



Understanding SubVI (Function) Behavior

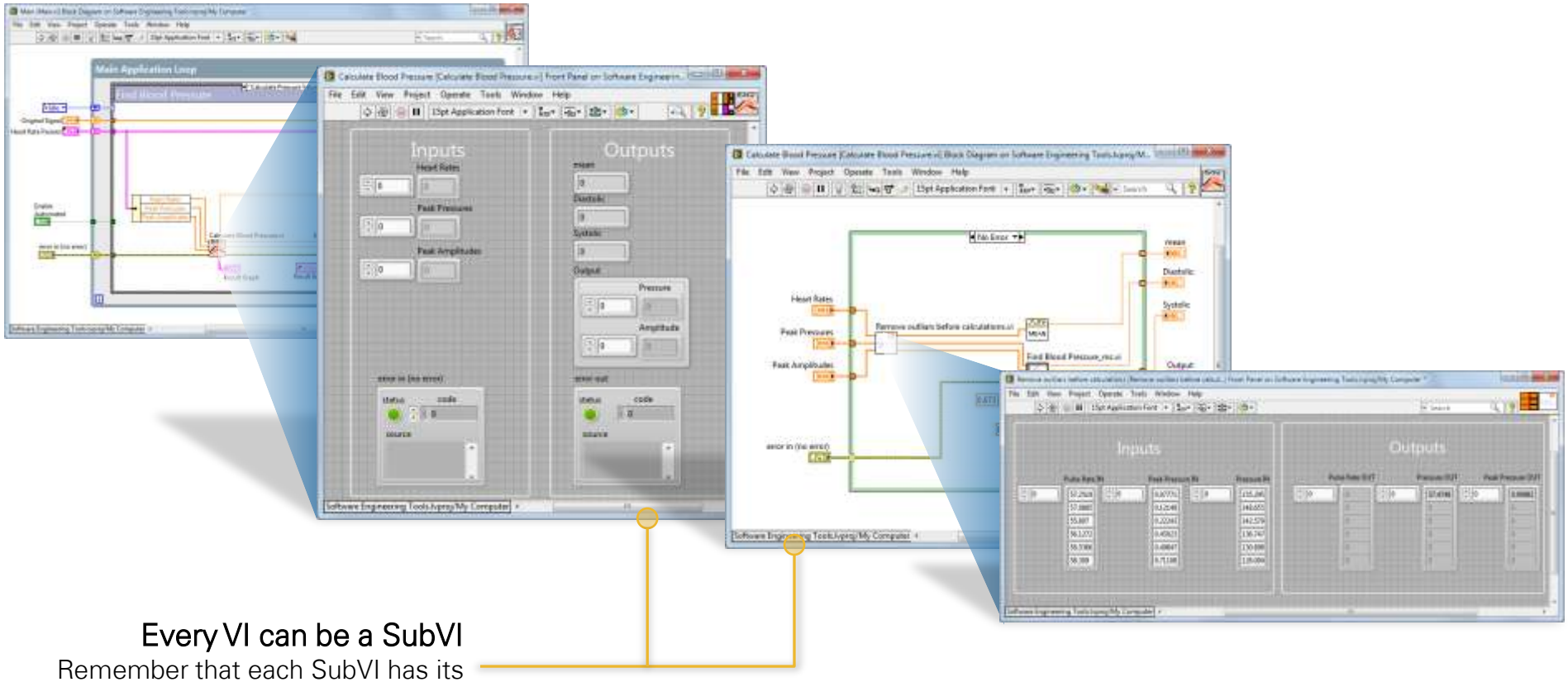
- Code will only compile if required inputs are wired
- Required inputs are **Bold**
- If an optional input is not supplied, a default value will be used for execution



Tip: Access the Context Help using Ctrl+H

Understanding Application Hierarchy

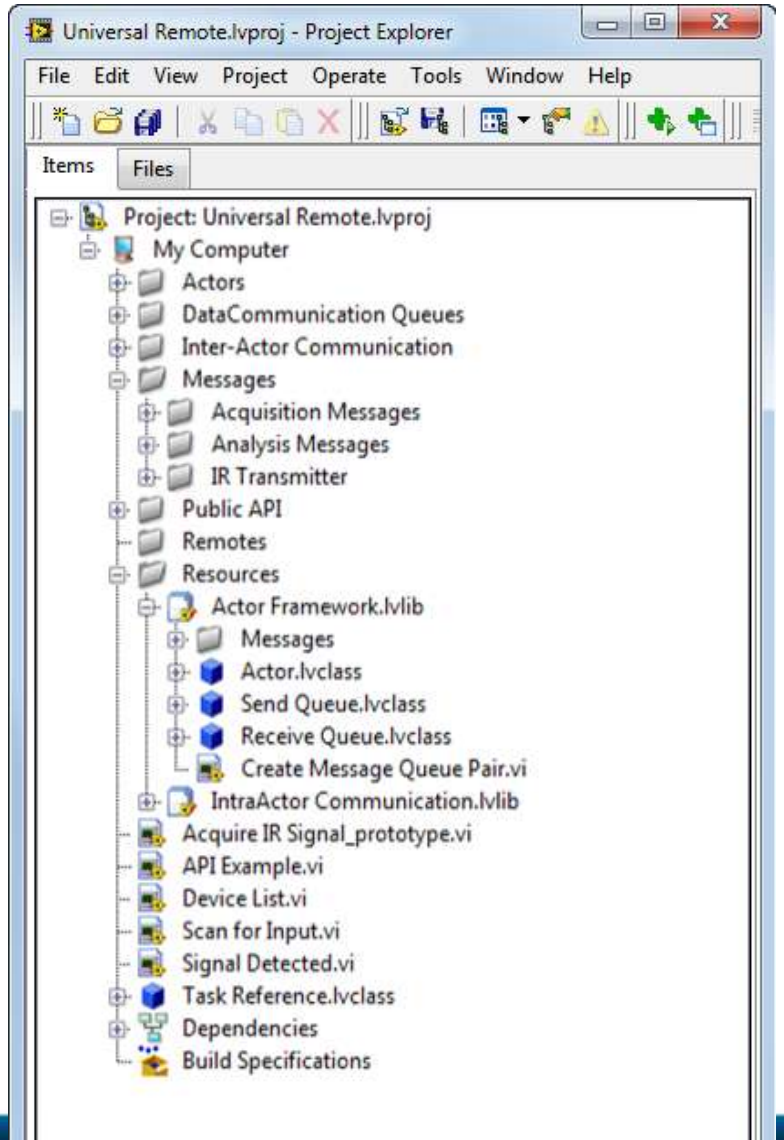
Double-clicking a nonprimitive SubVI opens the function



Every VI can be a SubVI

Remember that each SubVI has its own front panel and block diagram.

Managing Application Resources in Larger Applications



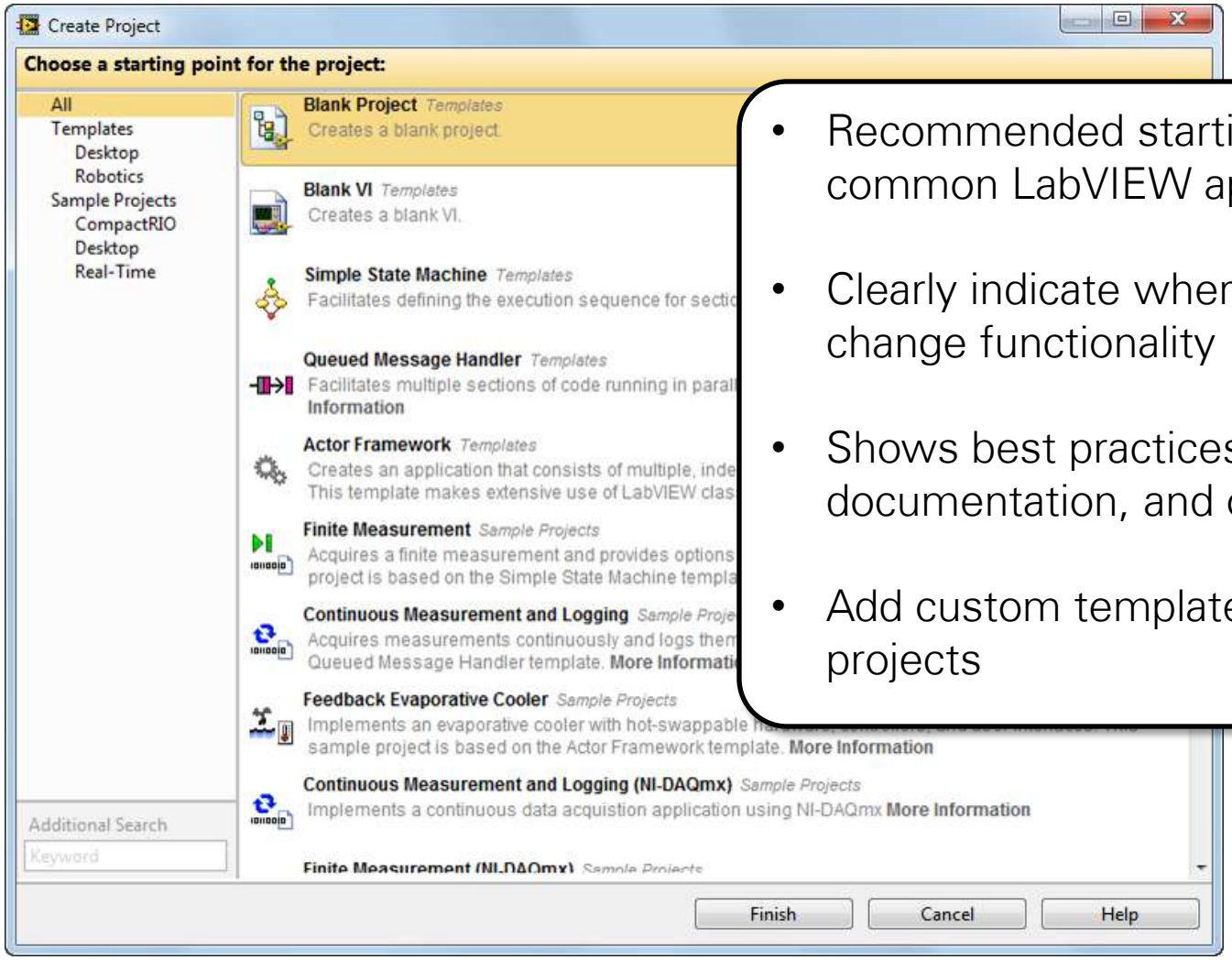
LabVIEW Project Explorer

- Organize application resources
- Create classes and libraries
- Build executables and installers
- Define access scope of components
- Create and manage hardware deployment targets

NATIONAL INSTRUMENTS

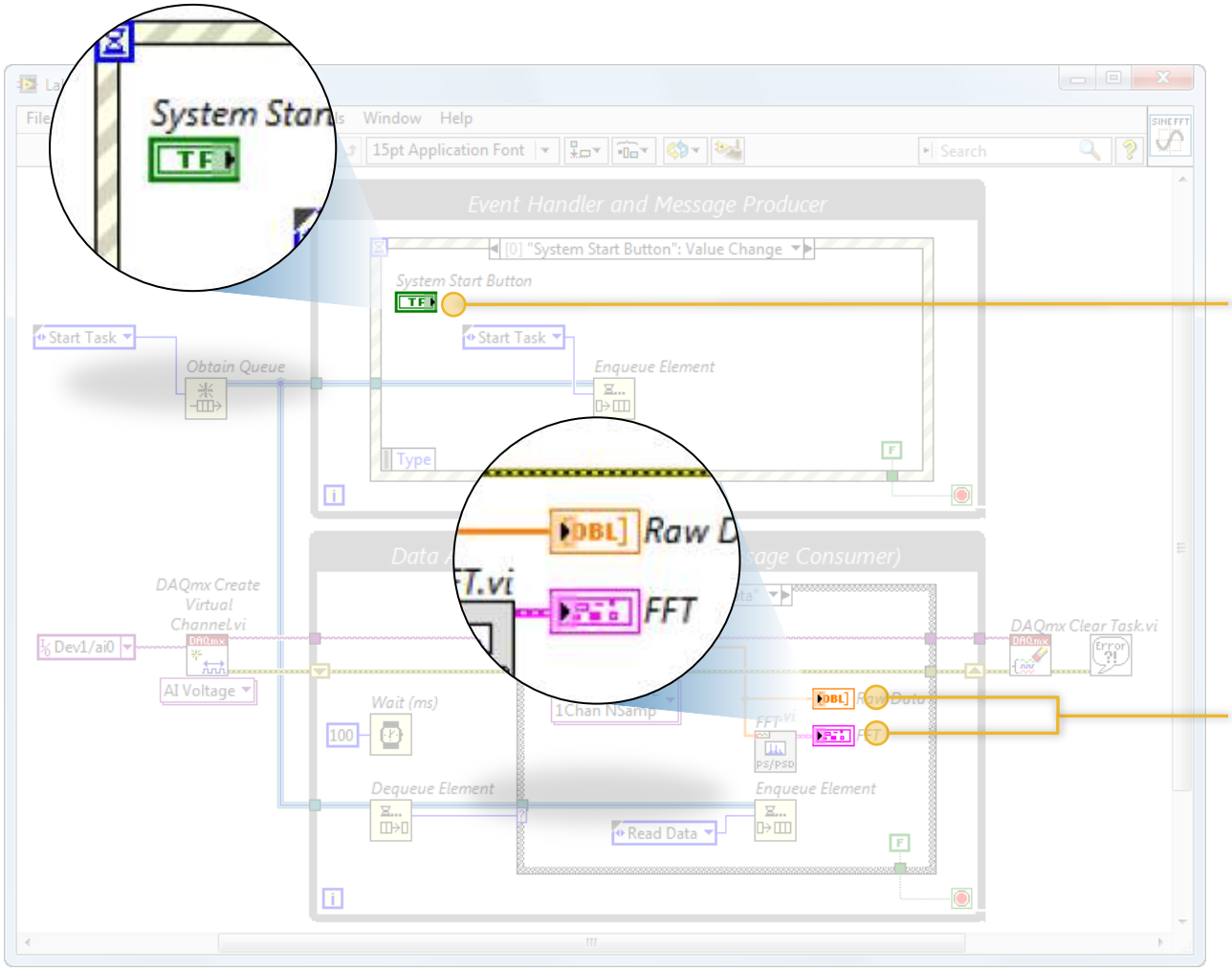
Never Start a LabVIEW Project From Scratch

Abundant sample projects and templates provide a scalable starting point



- Recommended starting points for common LabVIEW applications
- Clearly indicate where to add or change functionality
- Shows best practices for code design, documentation, and organization
- Add custom templates and sample projects

Exploring a LabVIEW Block Diagram



Input Terminal

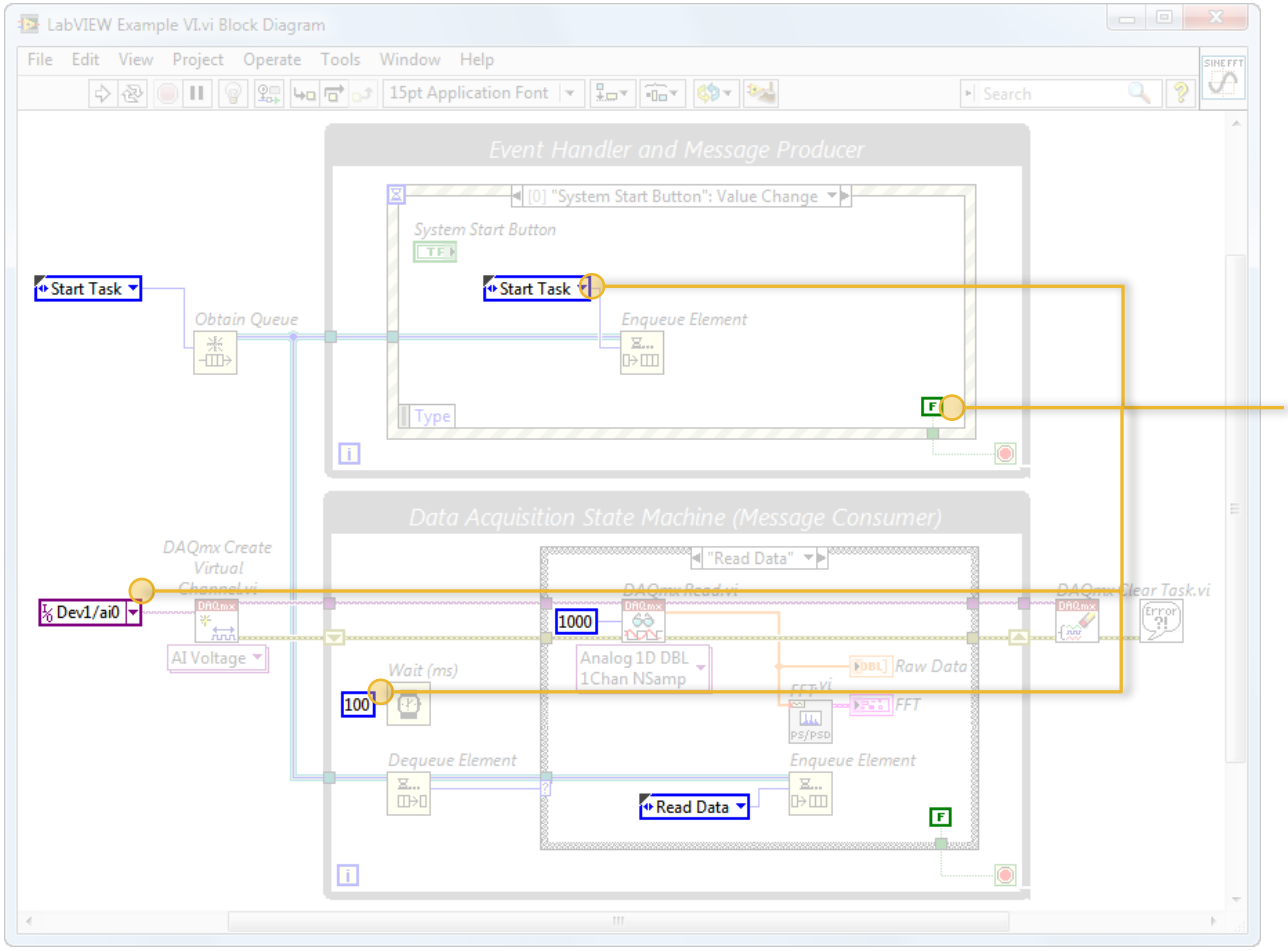
Input terminals are connected to front panel controls and receive input data from the user interface.

Output Terminals

Output terminals are connected to front panel indicators and display data as output to the user interface.

You can tell whether a terminal is a control or indicator by examining the direction it faces.

Exploring a LabVIEW Block Diagram

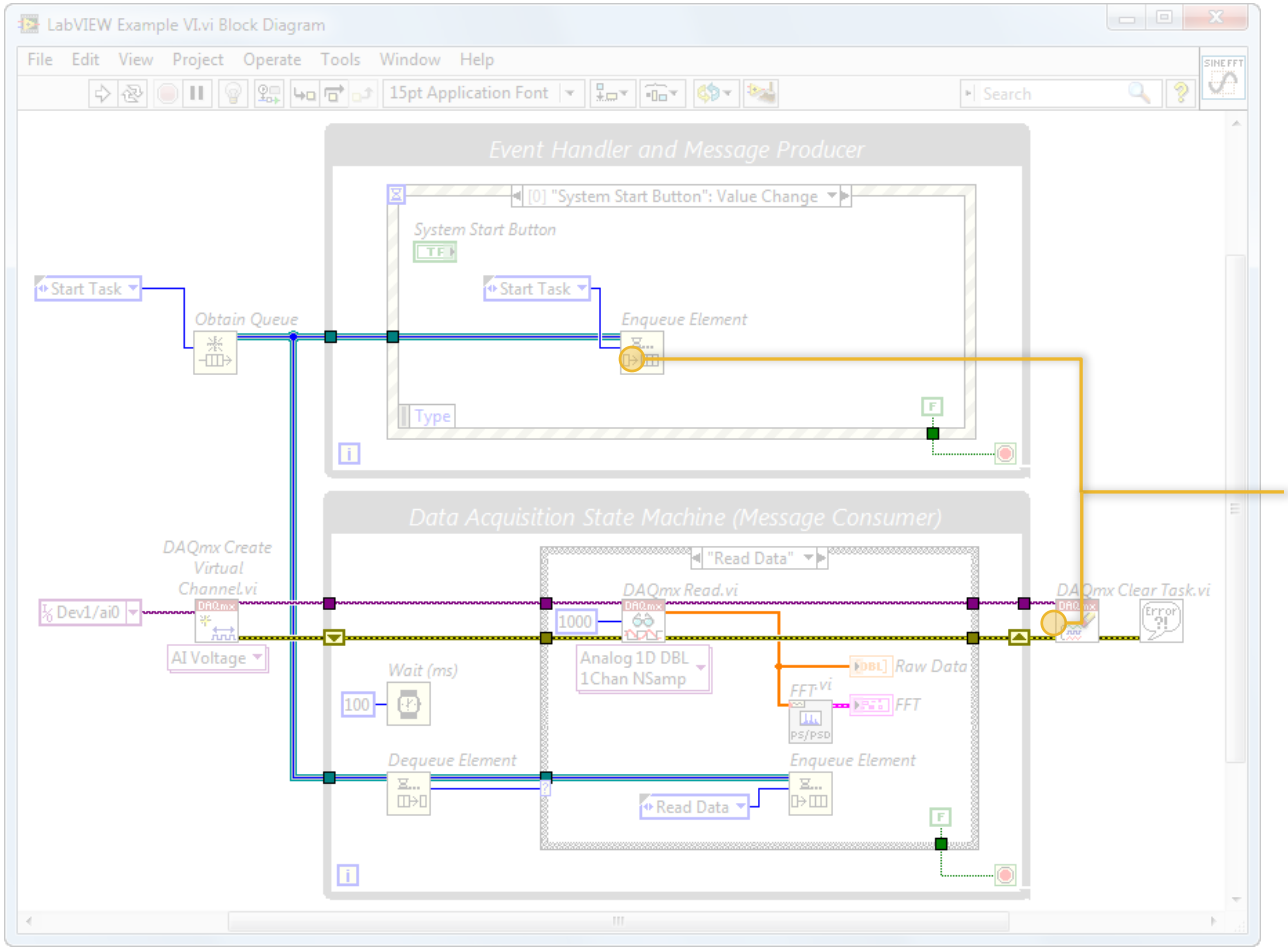


Constants

These constant values are hard-coded on the block diagram and can only be modified at edit-time.

The color of the constant indicates the type of data represented.

Exploring a LabVIEW Block Diagram

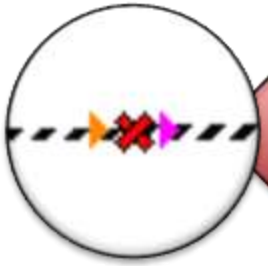


Wires
Data flows on wires between nodes on the block diagram.

The color of the wire indicates its data type, which is strictly enforced at edit-time.

The Color, Style, and Thickness of Common Wires

Wire Type	Scalar	1D Array	2D Array	Color
Floating Point				Orange
Integer				Blue
Boolean				Green
String				Pink
Error				Yellow

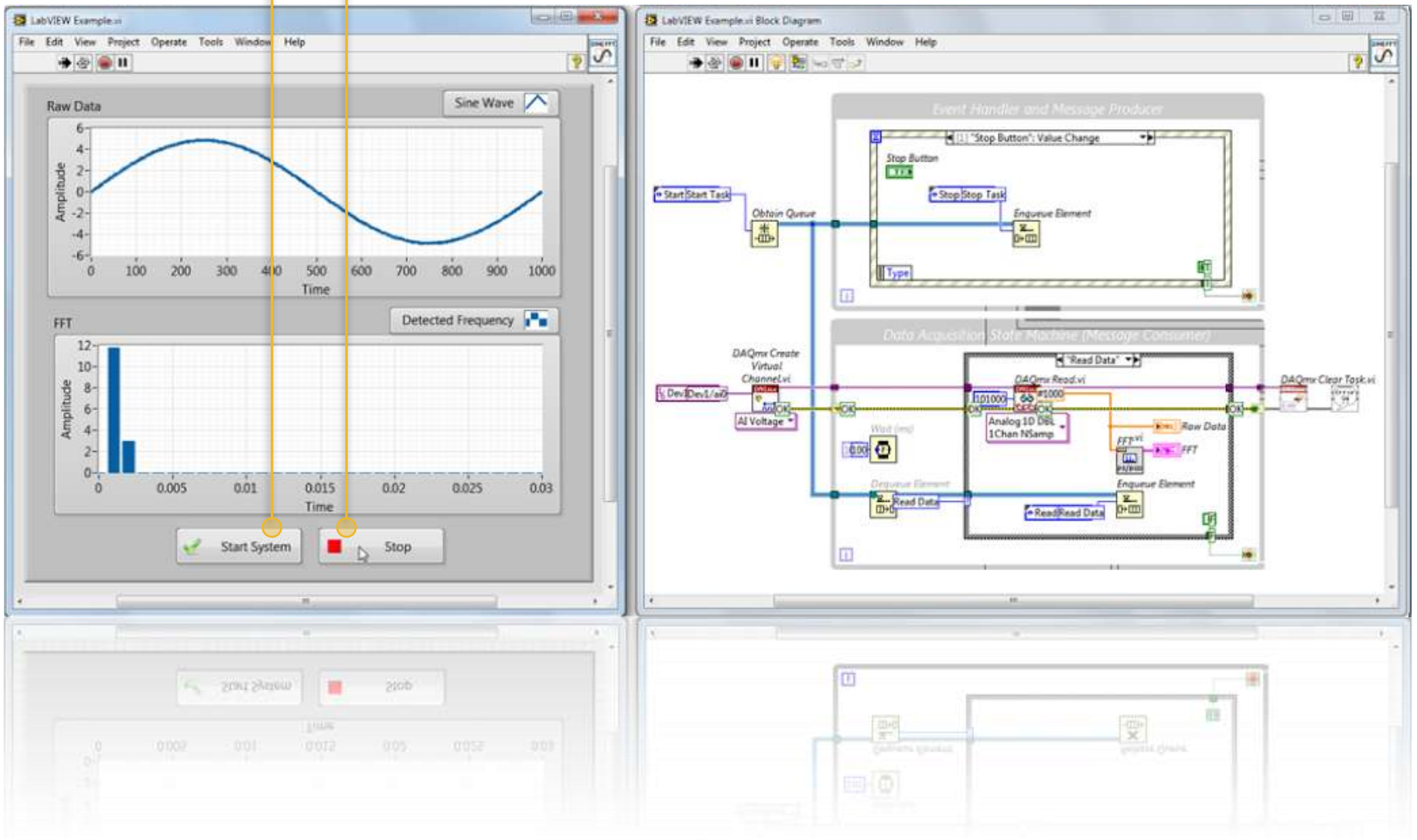


A "broken wire" represents a data type conflict that LabVIEW cannot automatically resolve. Fix it, or your code won't run!

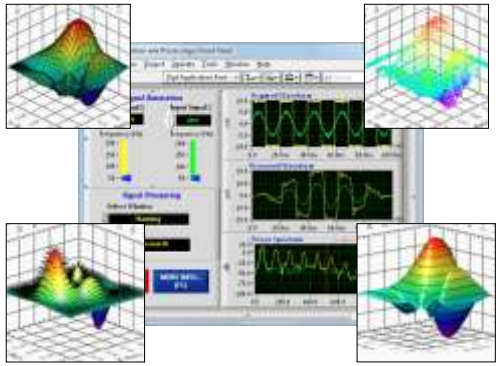
Visualizing Data Flow Along Wires: Highlight Execution

User presses the "Start" button to fire the first event

User presses the "Stop" button to fire the second event



Extending LabVIEW Beyond Data Acquisition



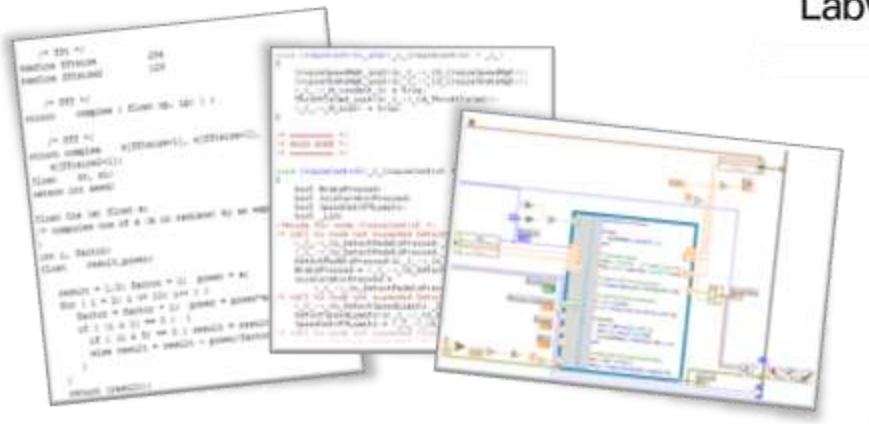
Advanced Analysis

External Code Integration

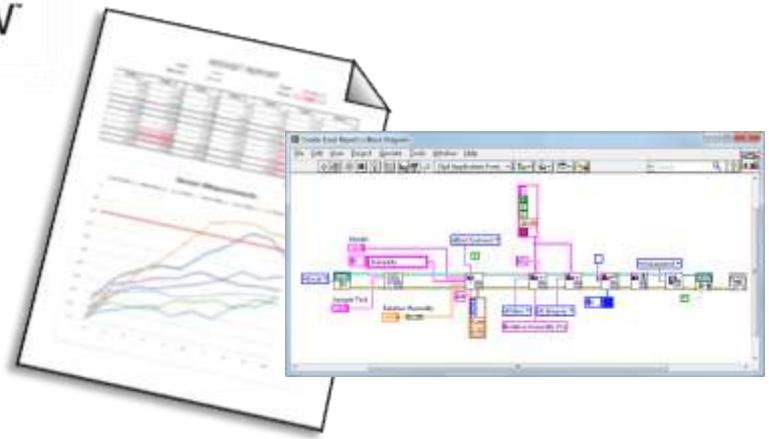
NATIONAL INSTRUMENTS
LabVIEW

Complex Visualization

Automated Reporting



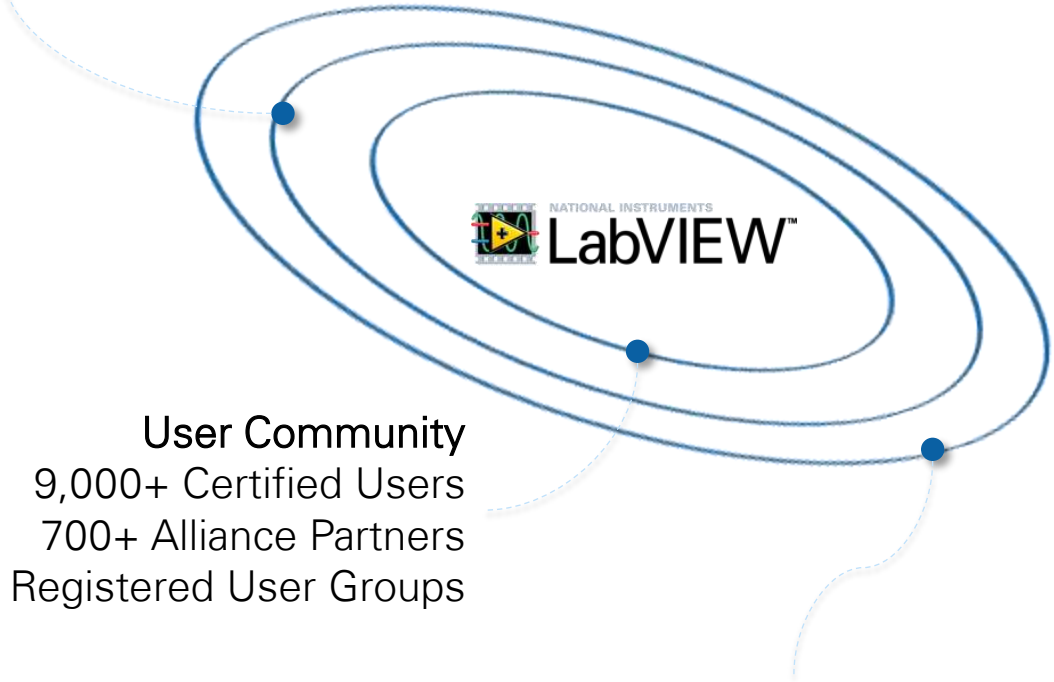
.NET Assembly, C DLL, .m File



Leveraging the LabVIEW Ecosystem

LabVIEW Tools Network

1,000,000+ Add-Ons Downloaded
26+ Certified Add-Ons
100+ Available Add-Ons



User Community

9,000+ Certified Users
700+ Alliance Partners
60+ Registered User Groups

Modules and Toolkits

40+ Toolkits and Modules Including:

LabVIEW Real-Time Module
LabVIEW FPGA Module
LabVIEW Embedded Module for ARM
LabVIEW Touch Panel Module
LabVIEW Wireless Sensor Network Module
LabVIEW C Code Generator
NI Real-Time Hypervisor
Vision Development Module for LabVIEW
Sound and Vibration Measurement Suite
Sound and Vibration Toolkit
LabVIEW Advanced Signal Processing Toolkit
LabVIEW Adaptive Filter Toolkit
LabVIEW Digital Filter Design Toolkit
LabVIEW MathScript RT Module
Spectral Measurements Toolkit
Modulation Toolkit for LabVIEW
LabVIEW Robotics Module
LabVIEW Biomedical Toolkit
ECU Measurement and Calibration Toolkit
GPS Simulation Toolkit for LabVIEW
Measurement Suite for Fixed WiMAX
WLAN Measurement Suite
Automotive Diagnostic Command Set
LabVIEW GPU Analysis Toolkit
Multicore Analysis and Sparse Matrix Toolkit
LabVIEW PID and Fuzzy Logic Toolkit
LabVIEW Control Design and Simulation Module
LabVIEW System Identification Toolkit
LabVIEW Simulation Interface Toolkit
LabVIEW SoftMotion Module
LabVIEW Datalogging and Supervisory Control Module
LabVIEW Report Generation Toolkit for Microsoft Office
LabVIEW Database Connectivity Toolkit
LabVIEW DataFinder Toolkit
LabVIEW SignalExpress
LabVIEW VI Analyzer Toolkit
LabVIEW Statechart Module
LabVIEW Desktop Execution Trace Toolkit
NI Requirements Gateway
NI Real-Time Execution Trace Toolkit
LabVIEW Unit Test Framework Toolkit
LabVIEW Application Builder for Windows

Benefits to LabVIEW Software Maintenance

Future Software Updates and Upgrades
» Always Leverage the Latest Technologies «

Phone and Email Support From Applications Engineers
» Save Time Troubleshooting «

Download Older Versions of Media
» Ensure Quick Access to Existing Software «

Exclusive Self-Paced Training Modules
» Increase Proficiency at Your Pace «

The NI Software Standard Service Program (SSP) lowers the total cost of ownership and gets you on the path to success faster.

The Fundamentals of Data Acquisition (DAQ)

The Basics of Making PC-Based Measurements

NATIONAL INSTRUMENTS

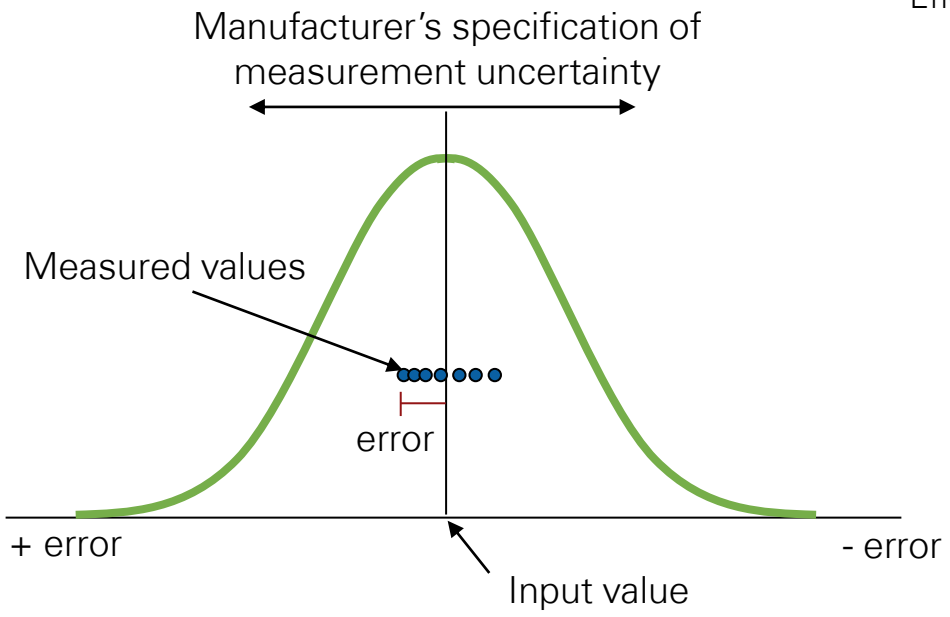
What Is Data Acquisition (DAQ)?

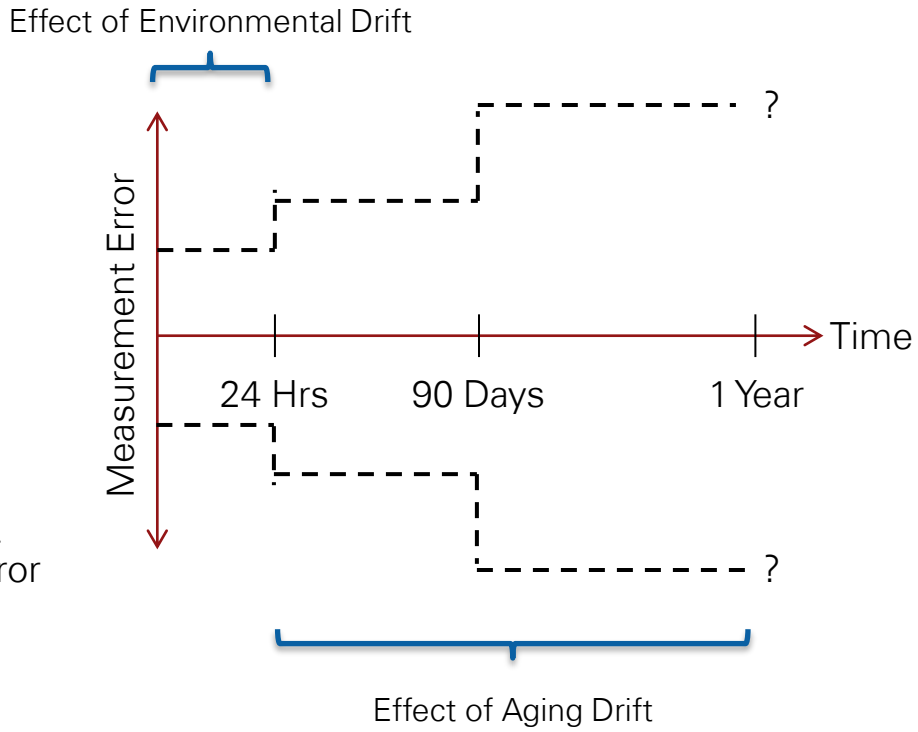
Data acquisition (DAQ) is the process of measuring an electrical or physical phenomenon such as voltage, current, temperature, pressure, or sound with a computer.

Compared to traditional measurement systems, PC-based DAQ systems exploit the processing power, productivity, display, and connectivity of industry-standard computers providing a more powerful, flexible, and cost-effective measurement solution.

All Measurements Are Technically Inexact

Electronic components naturally drift over time and require calibration





NI Offers a Range of Hardware Calibration Services

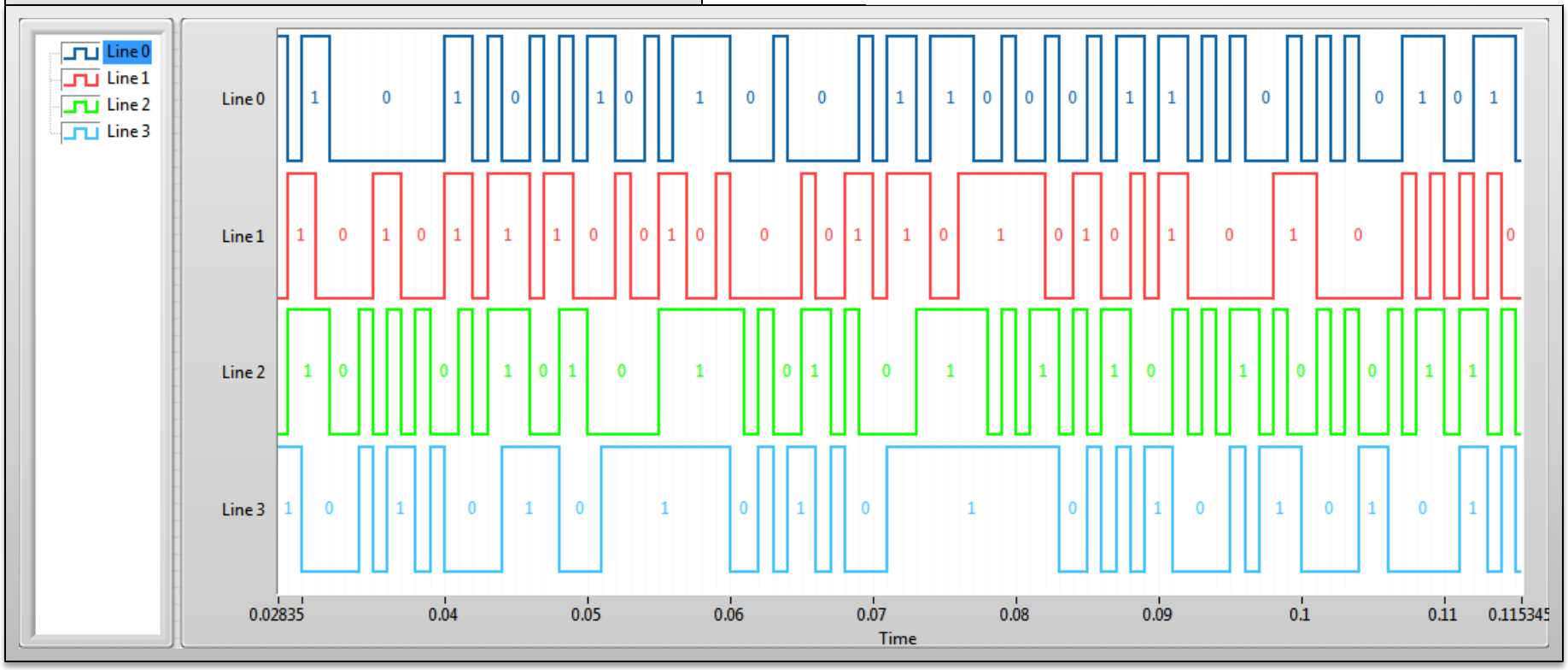
Verify and adjust measurement performance using NI-approved calibration procedures	✓	✓	✓
Detailed measurement data for all channels	✓	✓	✓
Available at point of sale	✓	✓	✓
Uncertainty evaluation		✓	✓
Performed at laboratory accredited to ISO 17025		✓	✓
Calculated measurement uncertainties (includes ISO 17025 accreditation body logo)			✓

**May be performed at NI Certified Calibration Center operated by third-party provider.*

Signals Come in Two Forms: Digital and Analog

Digital

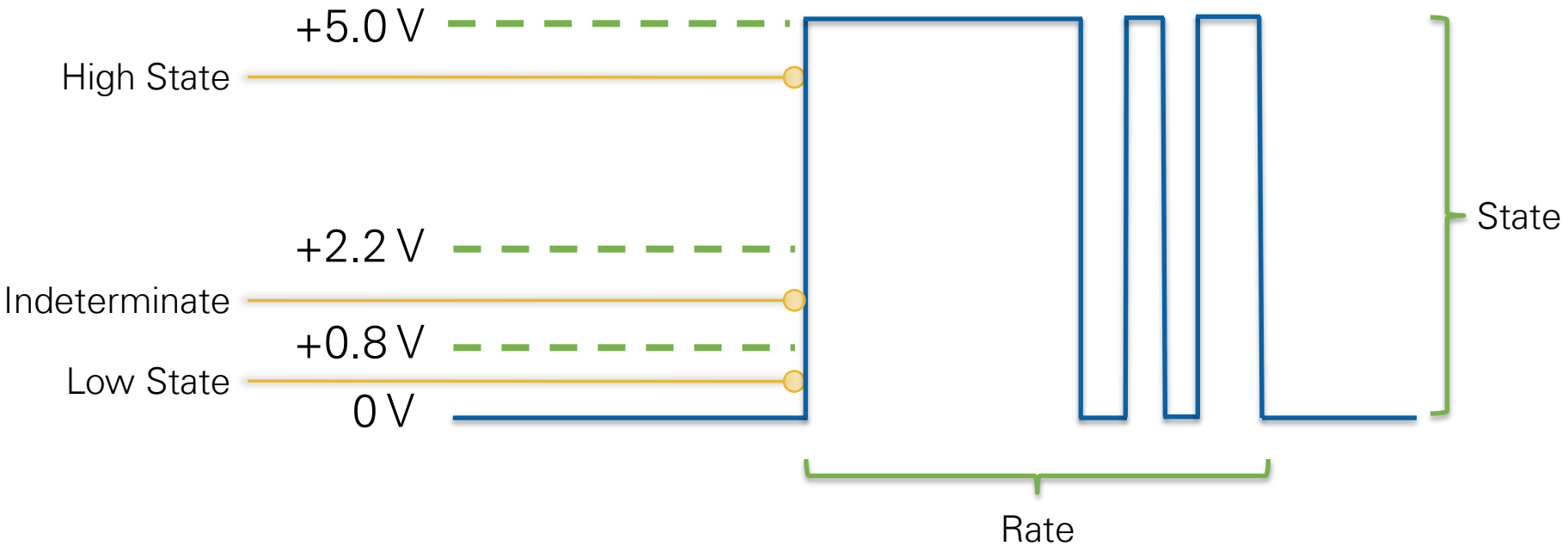
Analog



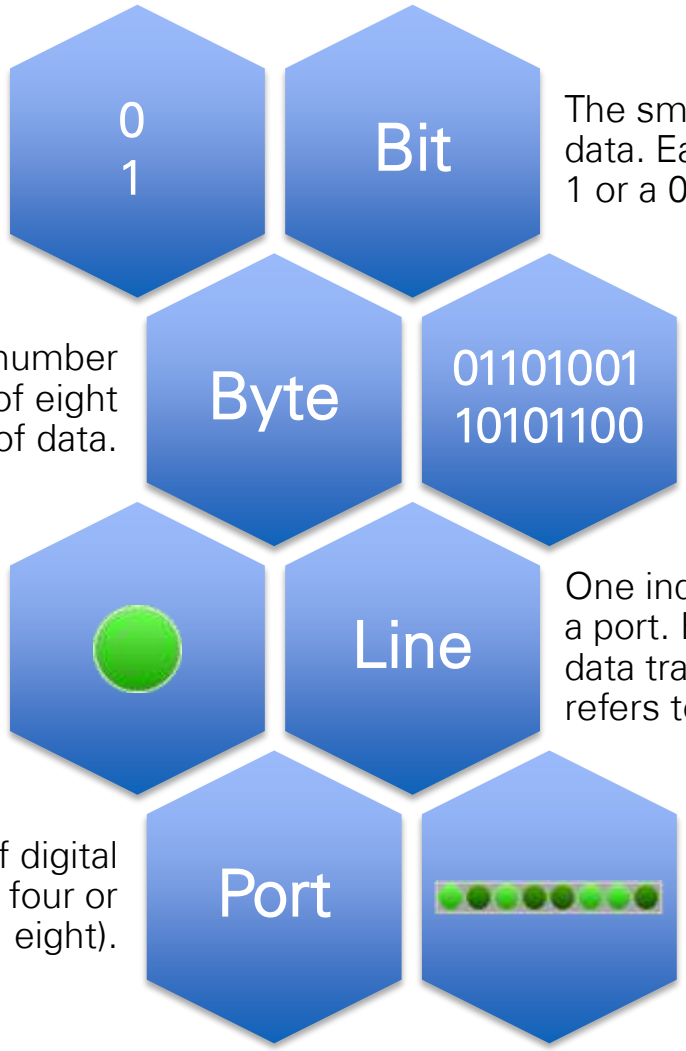
NATIONAL INSTRUMENTS

Digital Signals

- Digital signals have two states: high and low
- Digital lines on a DAQ device accept and generate transistor-transistor logic (TTL) compatible signals



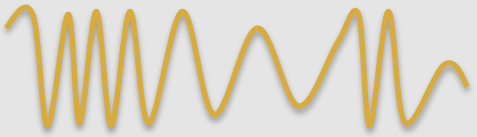
Digital Terminology

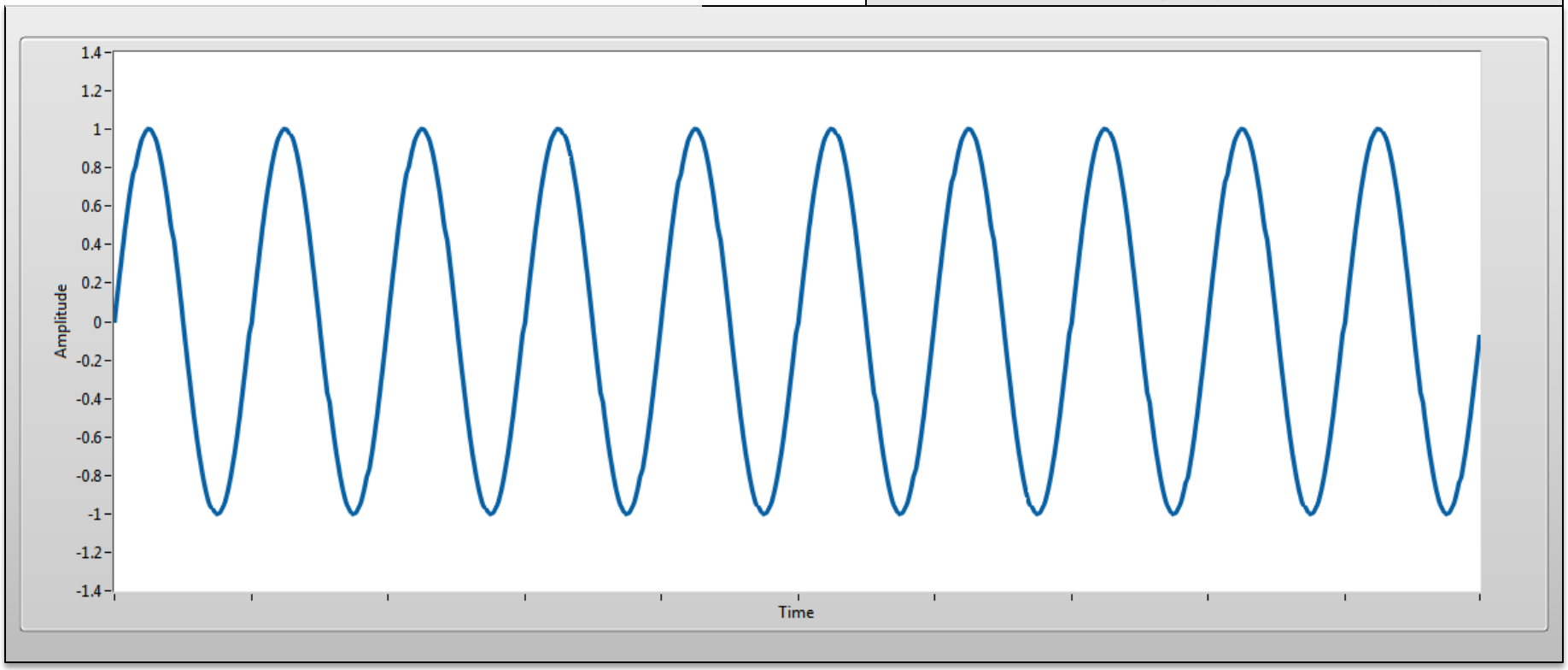


Signals Come in Two Forms: Digital and Analog

Digital

Analog

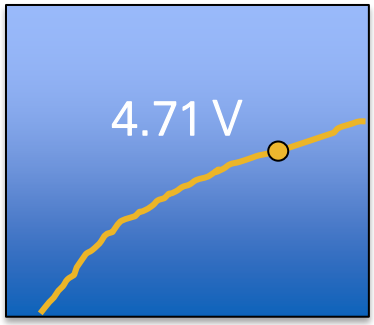




Analog Signals

Analog signals are continuous signals that can be any value with respect to time.

Analog Terminology

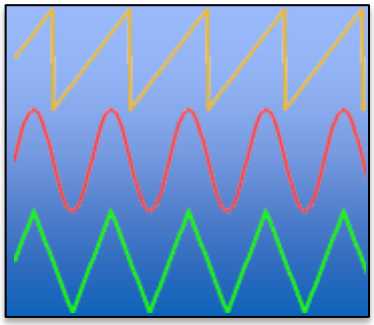


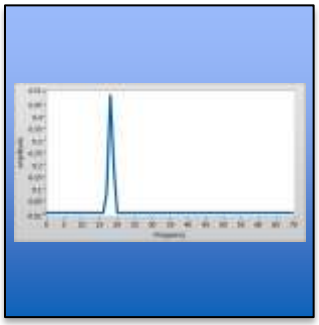
Level

The instantaneous value of the signal at a given point in time.

Shape

The form that the analog signal takes, which often dictates further analysis that can be performed on the signal.



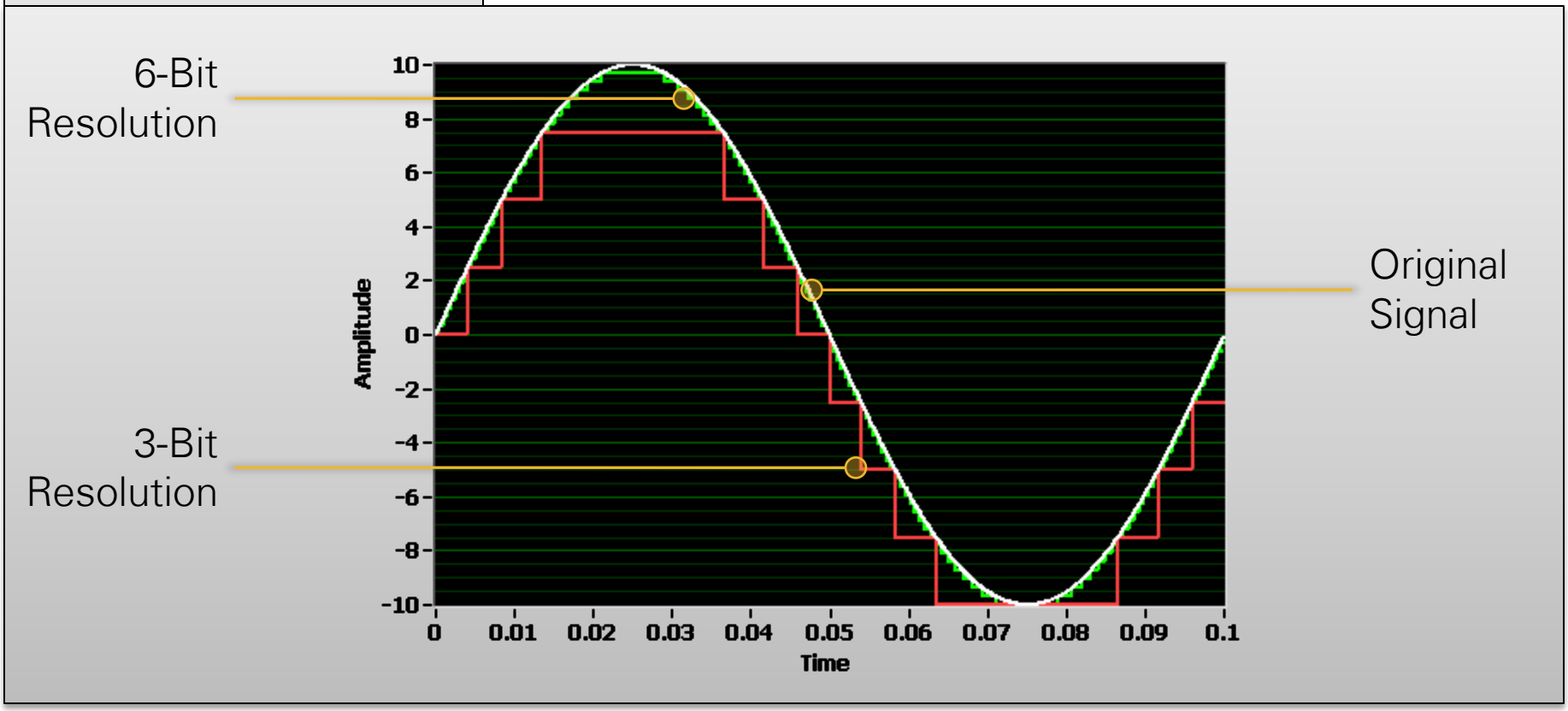


Frequency

The number of occurrences of a repeating event over time.

The Three R's of Data Acquisition: Resolution

Resolution | Range | Rate



The Three R's of Data Acquisition: Range

Resolution | **Range** | Rate

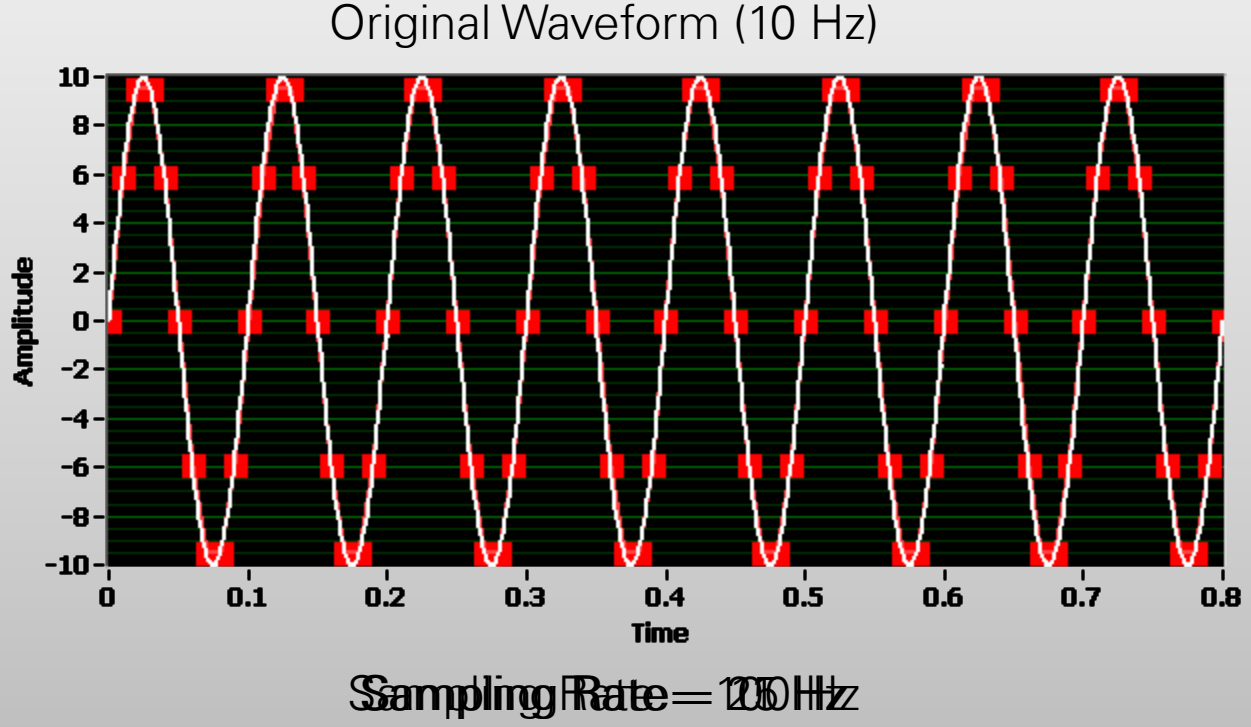
Range of
-10V to 10V

Original Signal

3-Bit Resolution

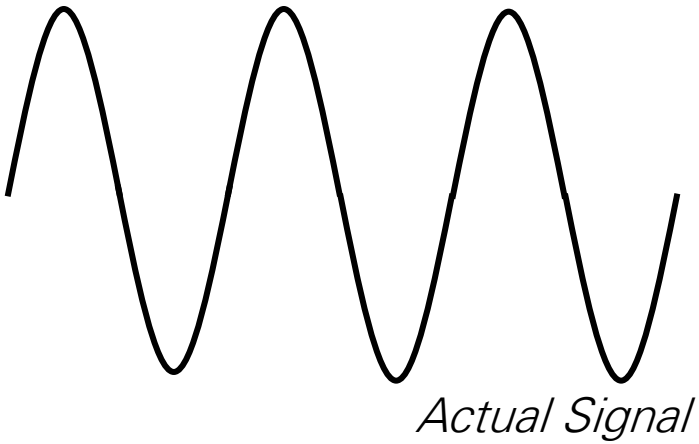
The Three R's of Data Acquisition: Rate

Resolution Range **Rate**

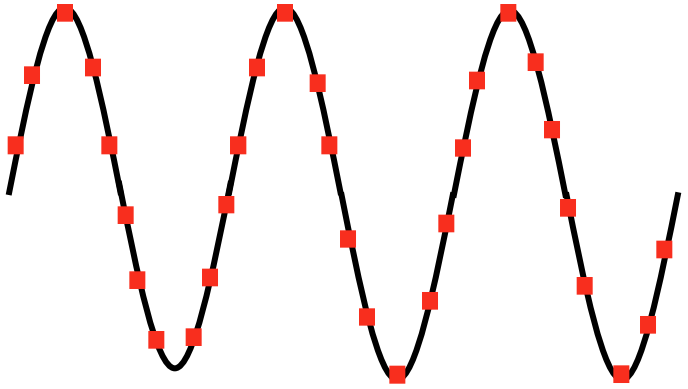


Sampling Rate Considerations

- An analog input signal is continuous with respect to time.
- Sampled signal is series of discrete samples acquired at a specified sampling rate.
- The faster we sample, the more our sampled signal will look like our actual signal.
- If not sampled fast enough, a problem known as aliasing will occur.



Actual Signal

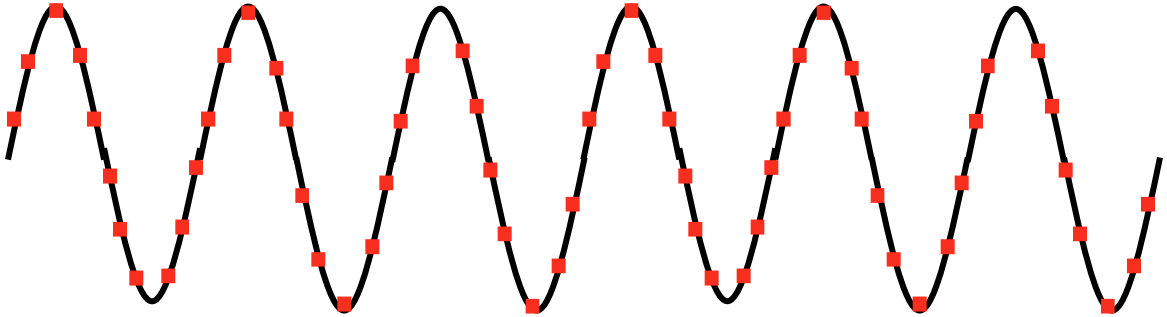


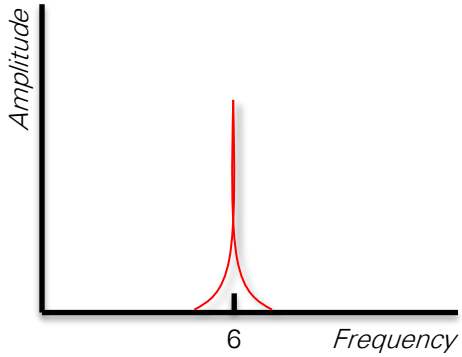
Sampled Signal

Aliasing

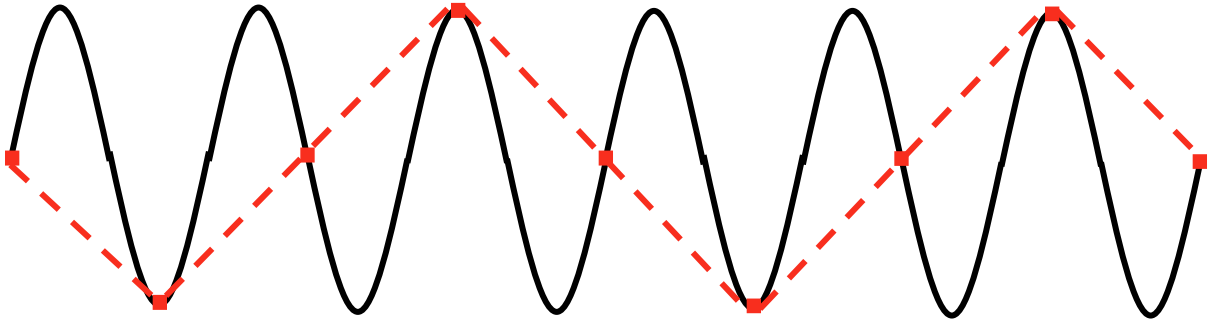
- Sample rate: how often an A/D conversion takes place
- Alias: misrepresentation of a signal

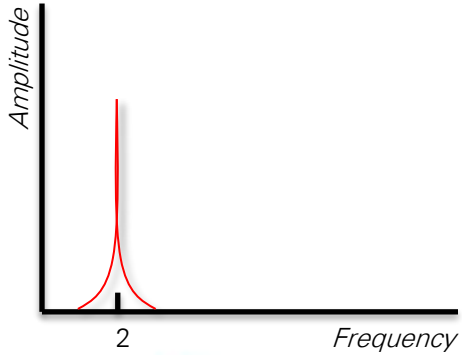
Adequately Sampled





Aliased Due to Undersampling





Following the Nyquist Theorem Prevents Aliasing

Frequency

To accurately represent the *frequency* of your original signal…

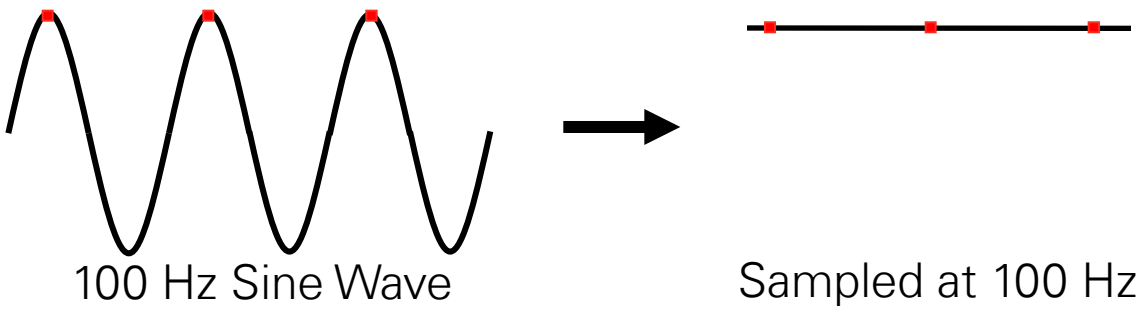
You must sample at greater than 2 times the maximum frequency component of your signal.

Shape

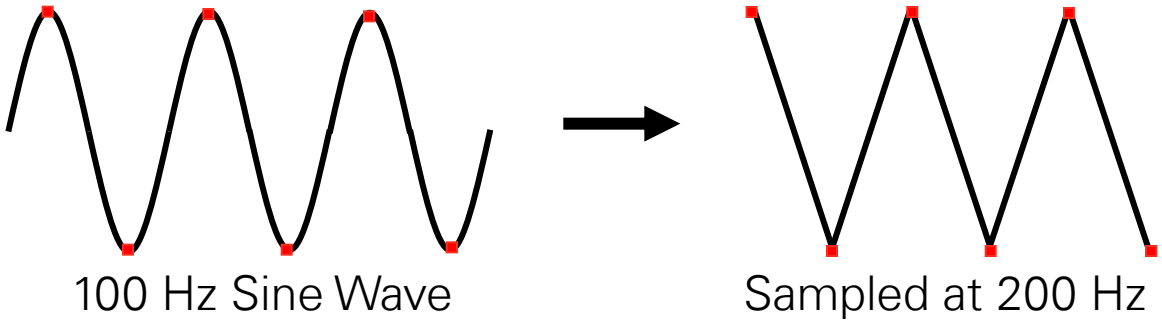
To accurately represent the *shape* of your original signal…

- You must sample between 5–10 times greater than the maximum frequency component of your signal .

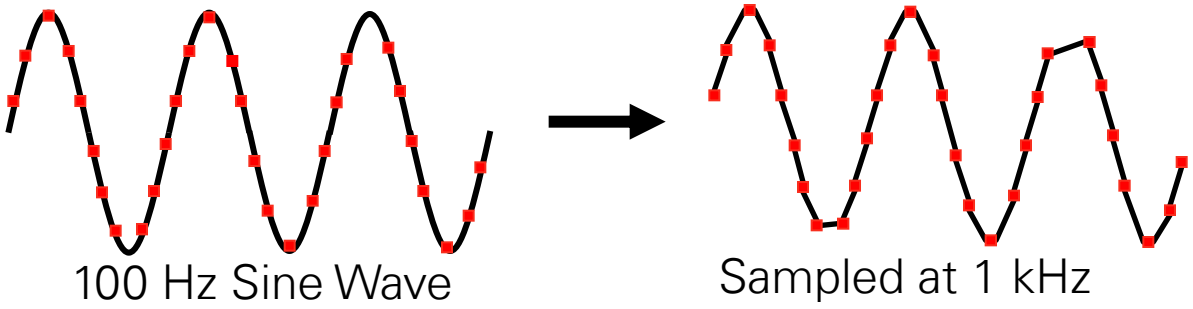
The Nyquist Theorem in Action



Aliased Signal



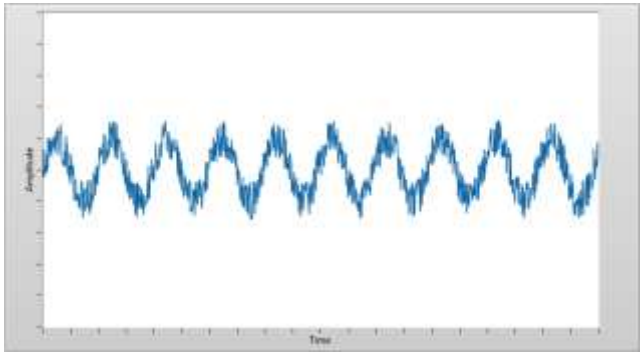
Adequately Sampled for Frequency Only



Adequately Sampled for Both Frequency and Shape

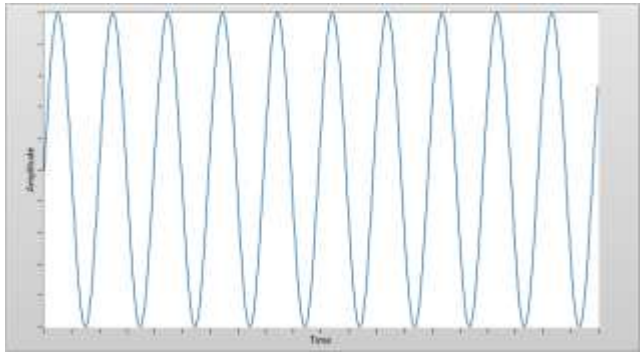
Conditioning Signals for Quality Measurements

- Signal conditioning improves a signal that is difficult for your DAQ device to measure
- Signal conditioning is not always required



Noisy, Low-Level Signal

Signal Conditioning



Filtered, Amplified Signal

Common Signal Conditioning Examples

Transducer/Signals	Signal Conditioning
Thermocouples	Amplification, Linearization, Cold-Junction Compensation
RTD (Resistance Temperature Detector)	Current Excitation, Linearization
Strain Gage	Voltage Excitation, Bridge Configuration, Linearization
Common Mode or High Voltage	Isolation Amplifier
Loads Requiring AC Switching or Large Current Flow	Electromechanical Relays or Solid-State Relays
High-Frequency Noise	Low-Pass Filters

Examining Common Signal Conditioning for Voltage Measurements

Amplification

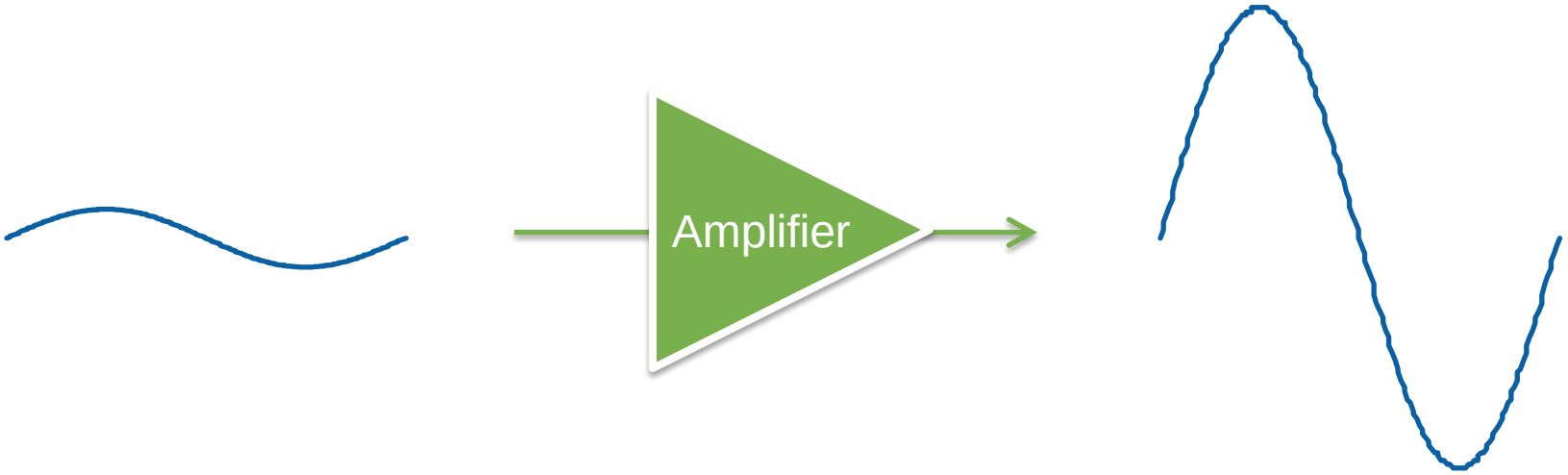
Attenuation

Filtering

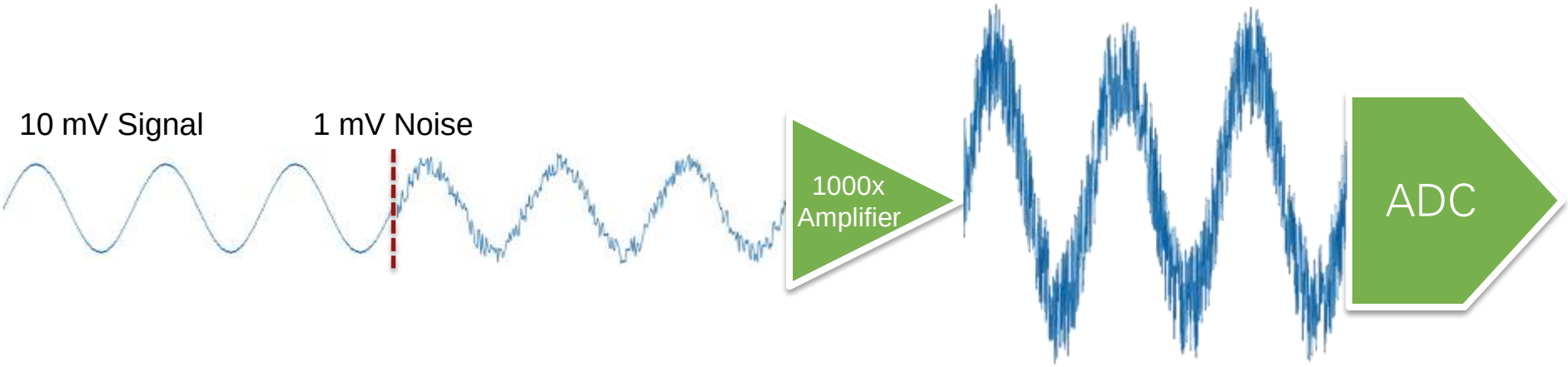
Isolation

Amplification

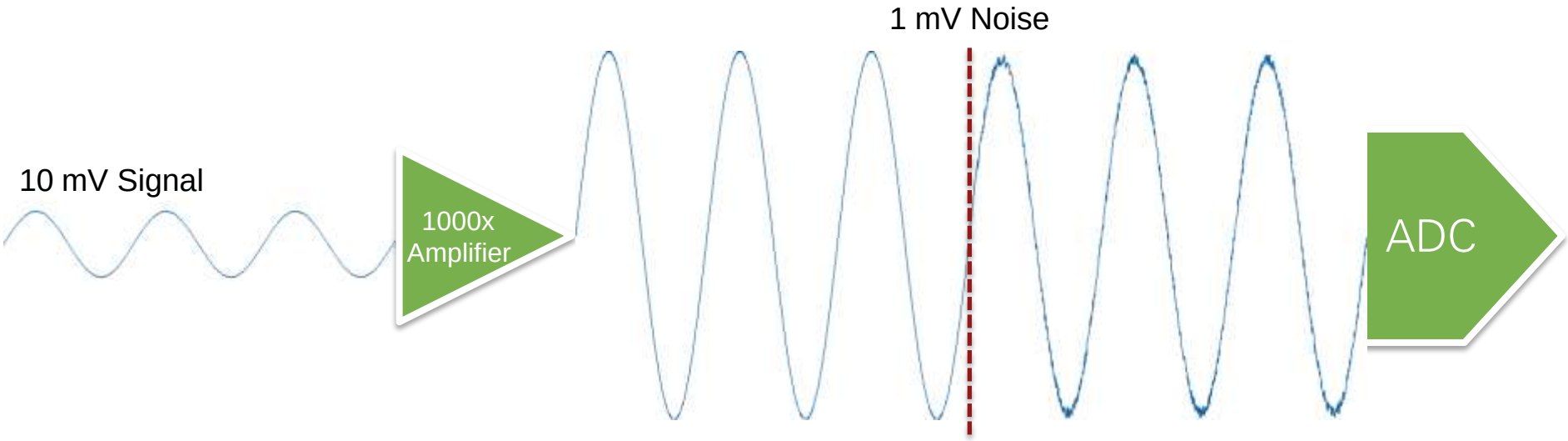
- Used on low-level signals
- Maximizes use of analog-to-digital converter (ADC) range and increases accuracy
- Increases signal-to-noise ratio (SNR)



Example: Amplification and the Signal-to-Noise (SNR) Ratio



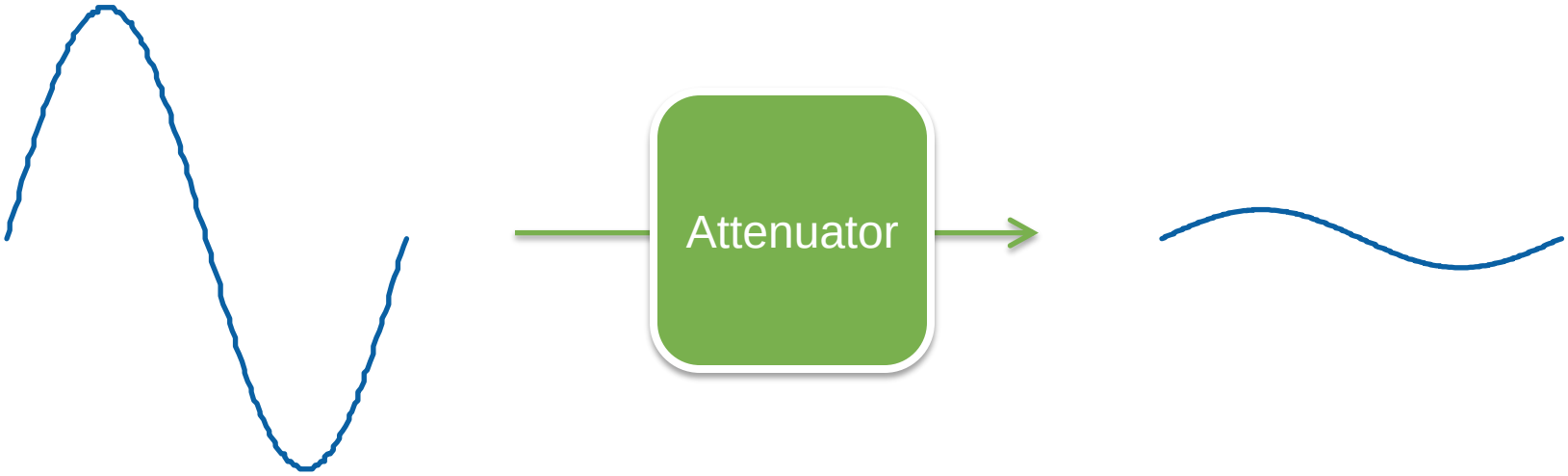
» *SNR = 10*



» *SNR = 10,000*

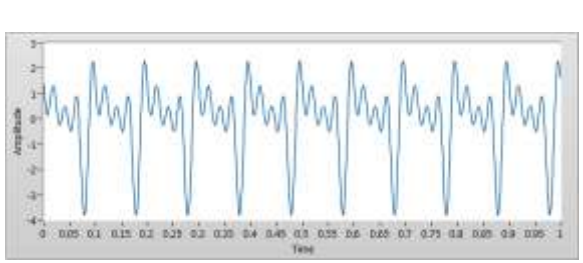
Attenuation

- Decreases the input signal amplitude to fit within the range of the DAQ device
- Necessary when input signal voltages are beyond the range of the DAQ device

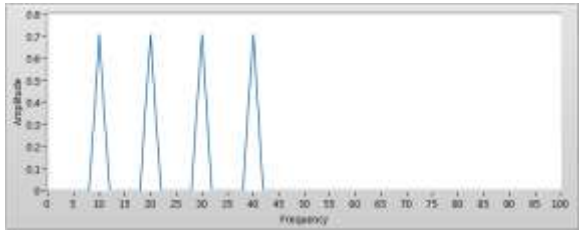


Filtering

Filters remove unwanted noise from a measured signal and block unwanted frequencies

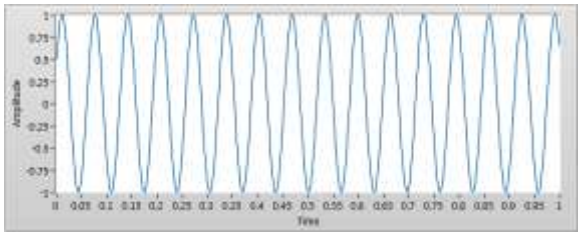


Time Domain

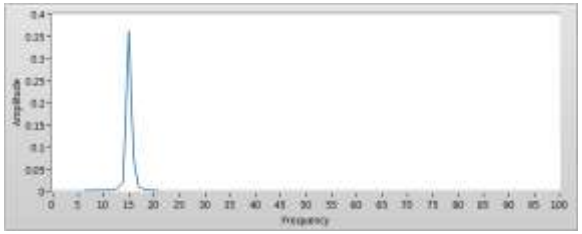


Frequency Domain

Lowpass Filter



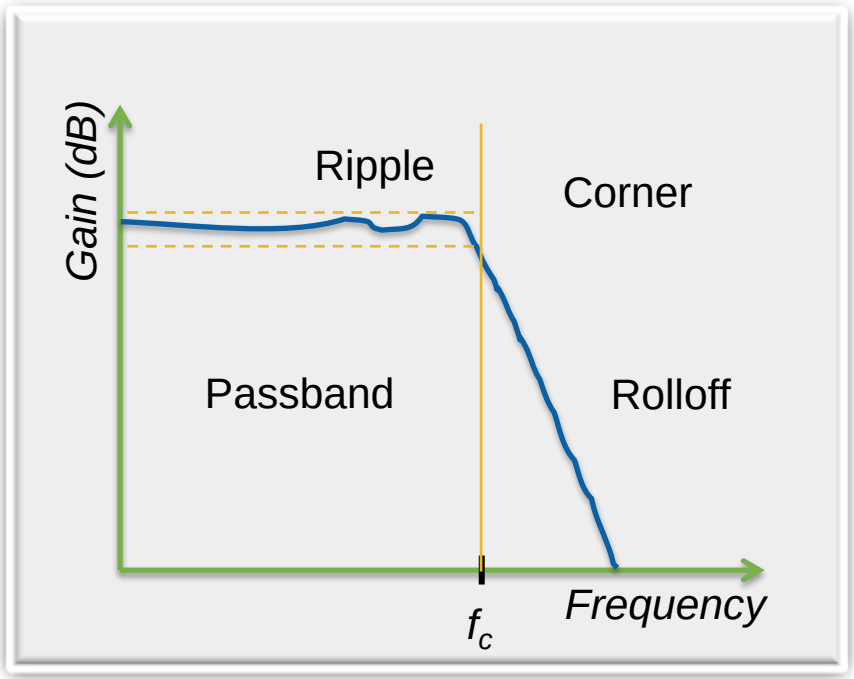
Time Domain



Frequency Domain

Filtering

- Passband
  - Frequencies the filter lets pass
- Ripple
  - Filter's effect on the signal's amplitude
- Corner
  - Frequency where the filter begins blocking the signal
- Rolloff
  - How sharply the filter cuts off unwanted frequencies



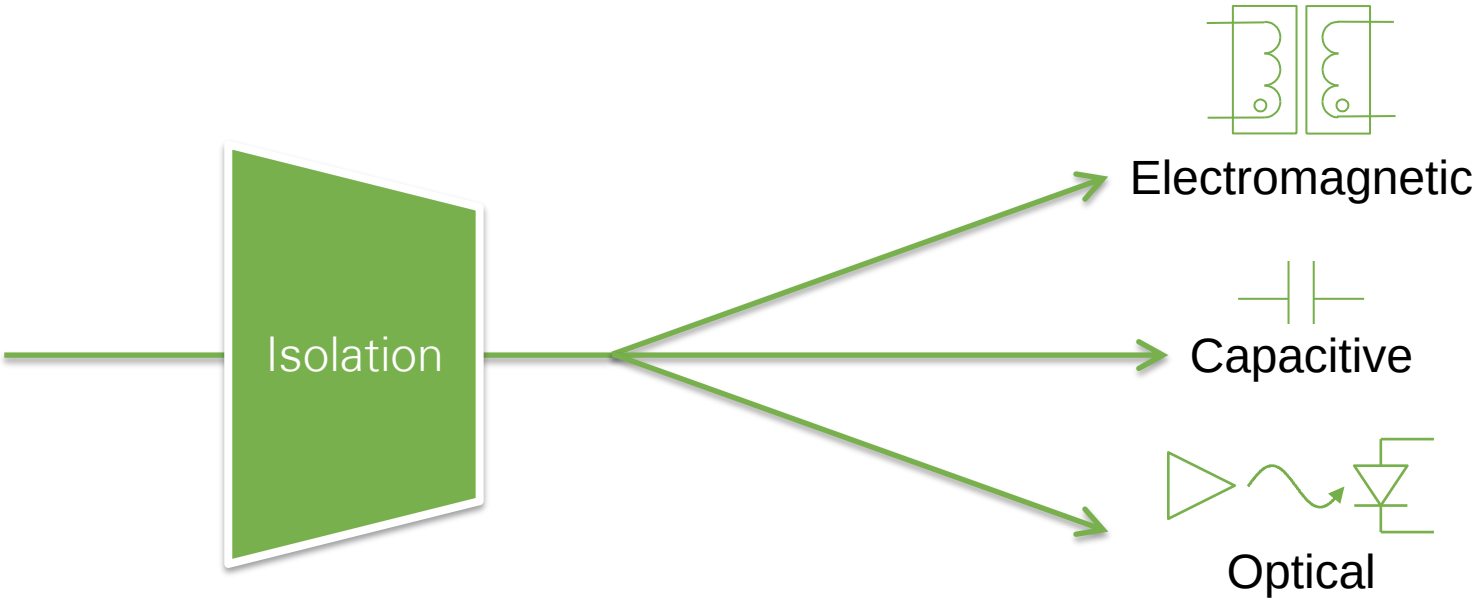
Example Bode Plot

A filter's attributes are typically described using Bode Plots

Isolation

Isolation helps to pass a signal from its source to a measurement device without a direct physical connection

- Blocks high common-mode signals
- Breaks ground loops
- Protects your instrumentation



Architecture of an Integrated Measurement System

NI CompactDAQ hardware combines a 1-, 4-, or 8-slot chassis with over 50 measurement-specific NI C Series I/O modules and can operate stand-alone with a built-in controller or connect to a host computer over USB, Ethernet, or 802.11 Wi-Fi.

Sensor

Measurement Device

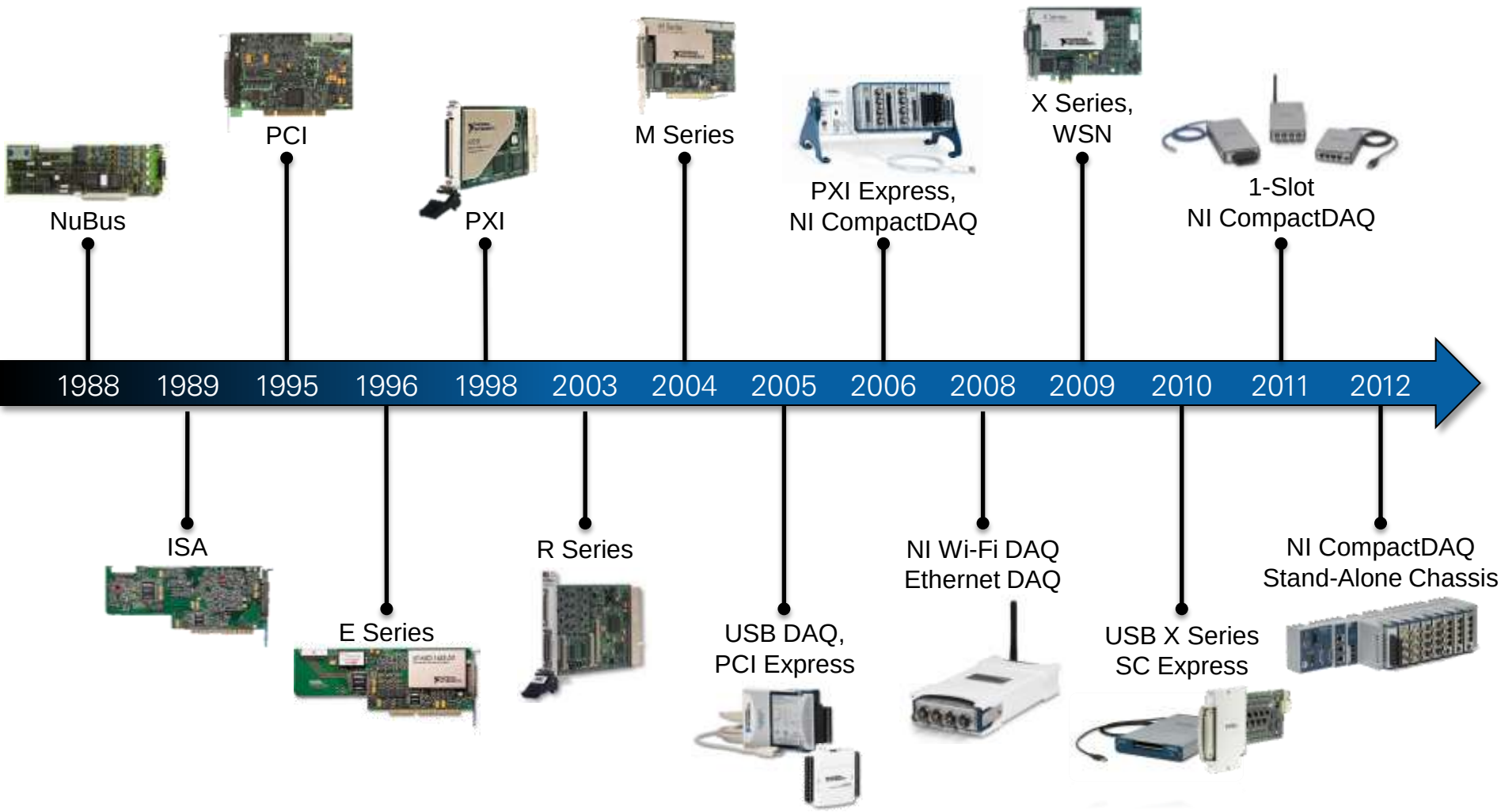
- Signal Conditioning
- Analog-to-Digital Converter

Software

- Driver Software
- Application Software

NI Is the Global Leader in Data Acquisition

With more than 20 years of DAQ hardware history and millions of channels sold



NATIONAL INSTRUMENTS

NI Data Acquisition Hardware Families

System

PXI
Optimized for high channel counts and tight synchronization

NI CompactDAQ
Customize with a variety of chassis and module types

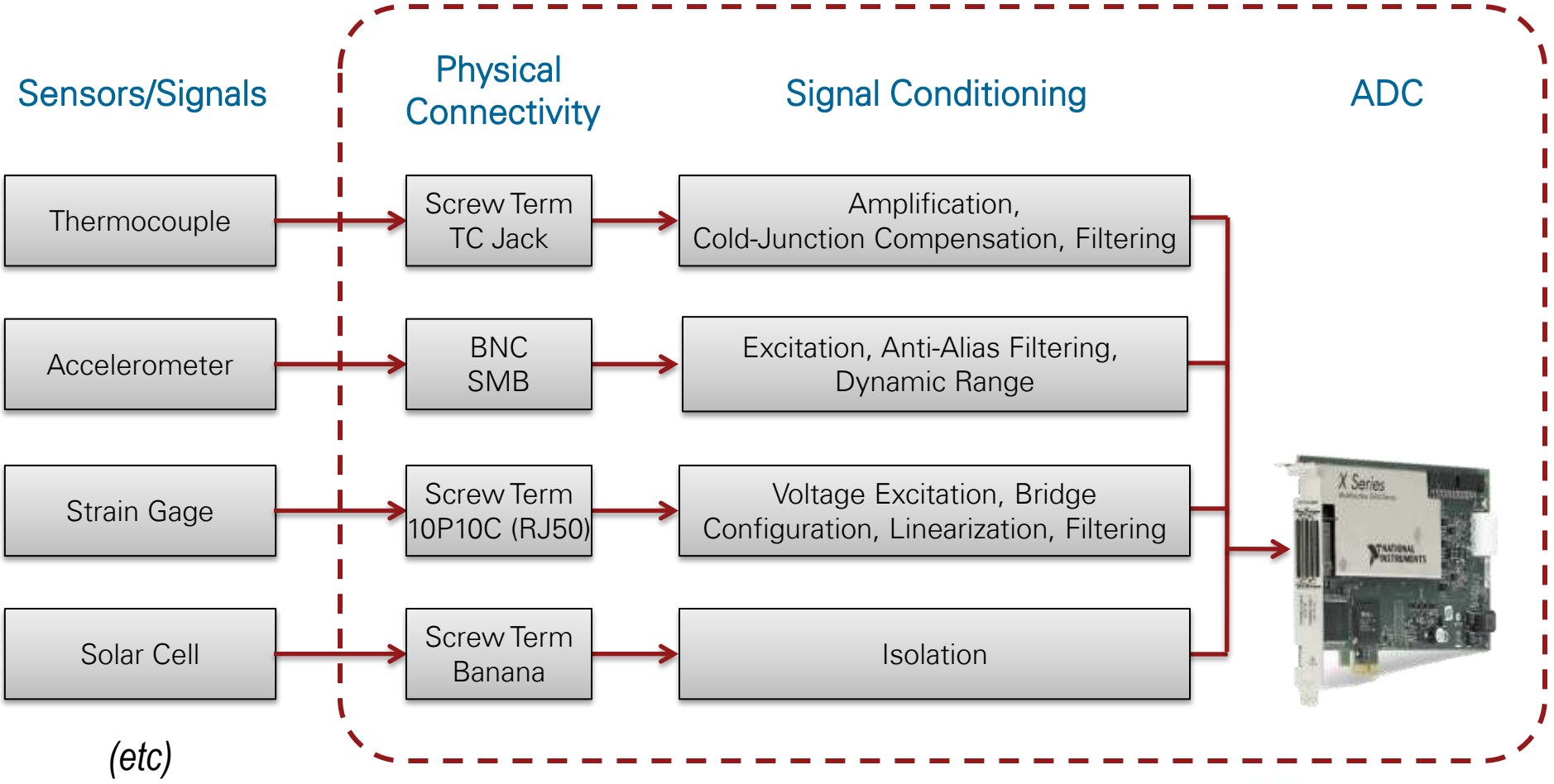
Desktop DAQ
Install in a desktop PC slot for maximum data throughput

Portable DAQ
Easily connect to any laptop or desktop with simple setup

Single Device

Traditional Hardware Components

Mixed-Measurement Systems Usually Involve Additional Connectivity and Conditioning



NI CompactDAQ Is an Integrated, Modular Solution

Sensors/Signals

- Thermocouple
- Accelerometer
- Strain Gage
- Solar Cell

(etc)

C Series Modules

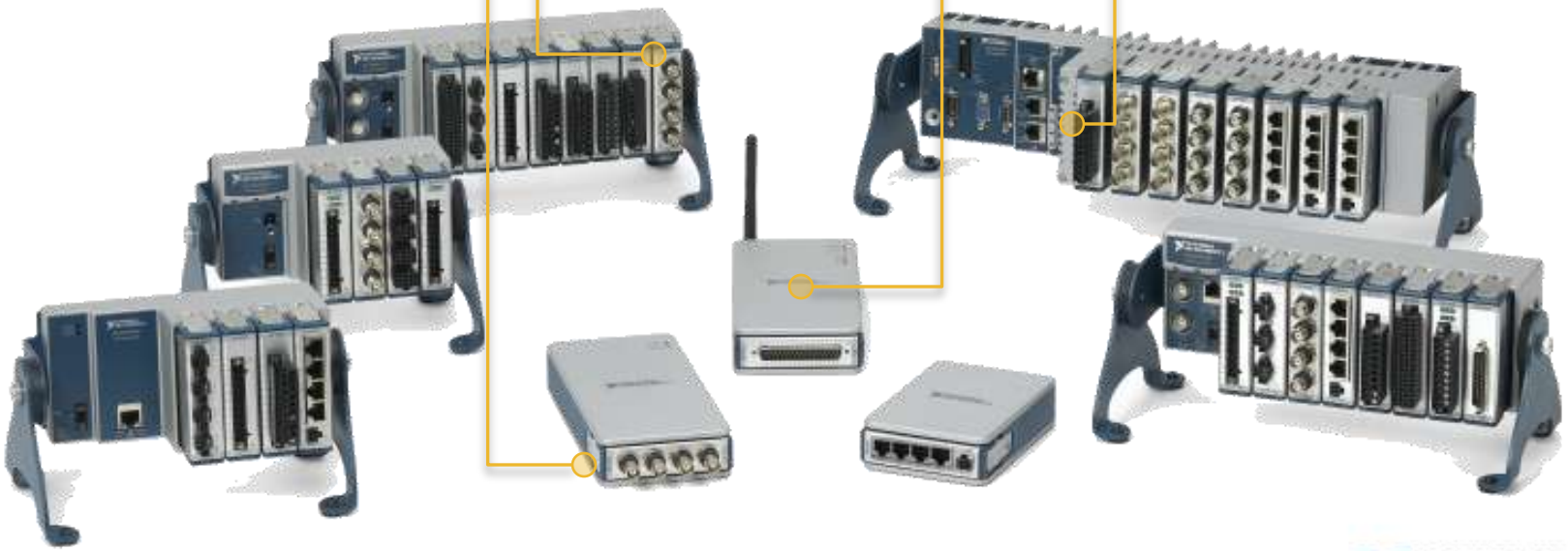
The NI CompactDAQ Family

A Custom System for Your Application
Mix and match from the entire family of measurement-specific, auto-detected, hot-swappable C Series modules.

A Module for Any Measurement
Over 50 measurement-specific modules integrate everything you need for a range of signal types, channel counts, and rates.

Same Code, Any Bus
Whether you've chosen to use USB, Ethernet, or Wi-Fi, identical code will run across each bus making scalability simple.

Choose the Right Form Factor for You
Available 1-, 4-, and 8-slot chassis accommodate up to 256 channels per chassis in tethered or stand-alone form.



Family Highlight: Stand-Alone NI CompactDAQ

Embedded Measurements and Logging

- >50 I/O modules
- Up to 24-bit, Up to 1 MS/s
- Dual-core processor
- 32 GB nonvolatile storage
- 0 to 55 °C Operating Temp
- 5g shock, 30g vibration
- Windows or Real-Time OS
- LabVIEW and NI-DAQmx

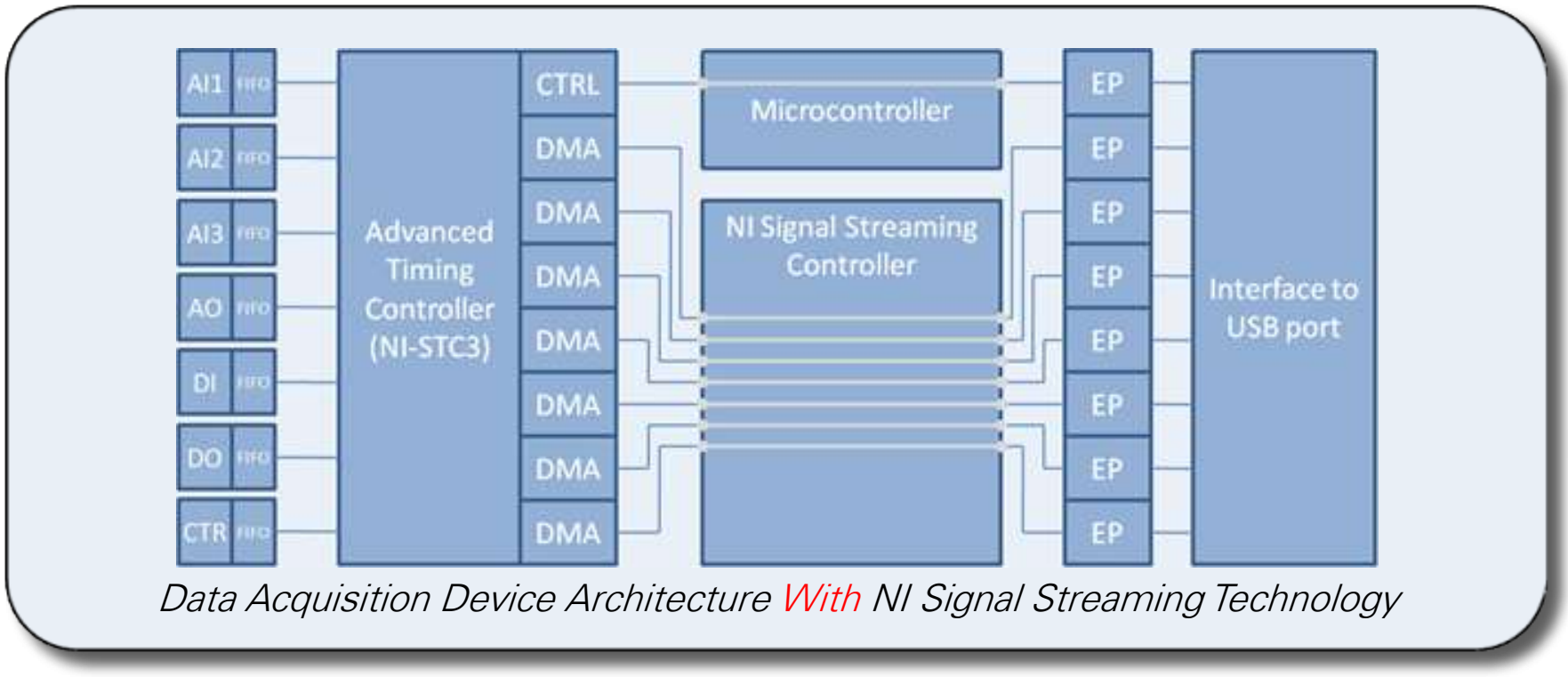
C Series I/O Modules

- Over 100 NI and Partner Modules
  - Analog Input
  - Analog Output
  - Digital I/O
  - Relay Output
  - Counter, Pulse Generation
  - Communication
    - CAN
    - LIN
    - PROFIBUS
  - Motion Control
  - Wireless
  - Engine Control
- Signal Conditioning
- Rugged Mechanicals
- Signal Conditioning/Filtering
- Isolation Barrier

The Most Trusted DAQ Hardware on the Market

NI hardware delivers greater value and performance through innovative technologies

NI DAQ hardware includes built-in technologies such as signal streaming that unlock performance for data streaming, logging, timing, and synchronization.



Data Acquisition Device Architecture With NI Signal Streaming Technology

Increasing Proficiency in NI Software and Hardware

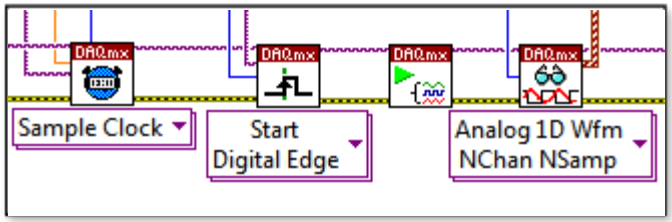
	Self-Paced	Instructor-Led		
Format Features	**Online** Prerecorded modules viewable at ni.com	**Online** 1- to 4-day classes held live remotely	**Regional** 1- to 3-day classes held at training facilities	**On-Site** 1- to 3-day classes held at your location
Learn from a certified instructor	—	✓	✓	✓
Access relevant hardware	—	✓	✓	✓
Eliminate distractions with a classroom setting	—	—	✓	✓
Interact with other students	—	—	✓	✓
Content modified to meet your group's needs	—	—	—	✓
Avoid travel expenses	✓	✓	—	✓
Printed manual that accompanies the course	✓	✓	✓	✓
Exercises to practice concepts you learn	✓	✓	✓	✓
Multimedia training	✓	✓	—	—
Concept review quizzes	✓	✓	✓	✓
Class duration	—	Half-Day	Full-Day	Full-Day
Price	$	$$	$$$	$$$

Combining Software and Hardware Into an Integrated System

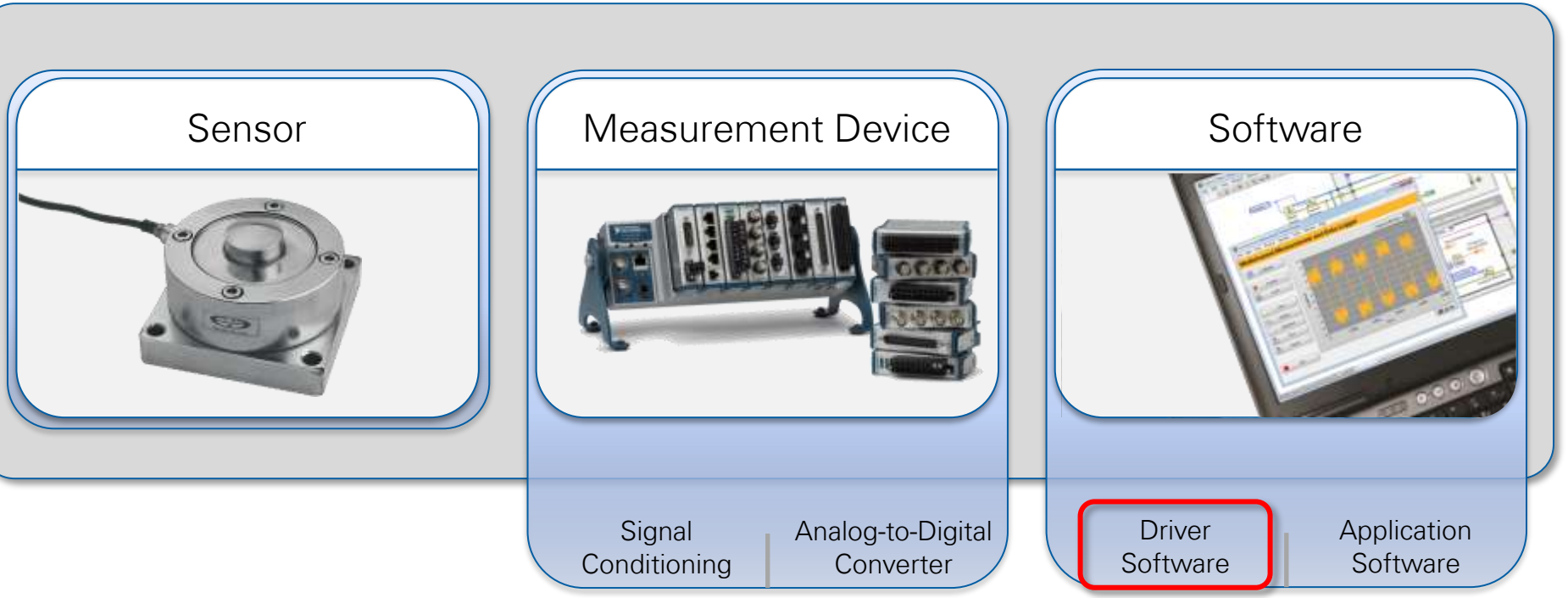
Automated Measurement Solutions With LabVIEW and NI DAQ

NATIONAL INSTRUMENTS

Architecture of an Integrated Measurement System



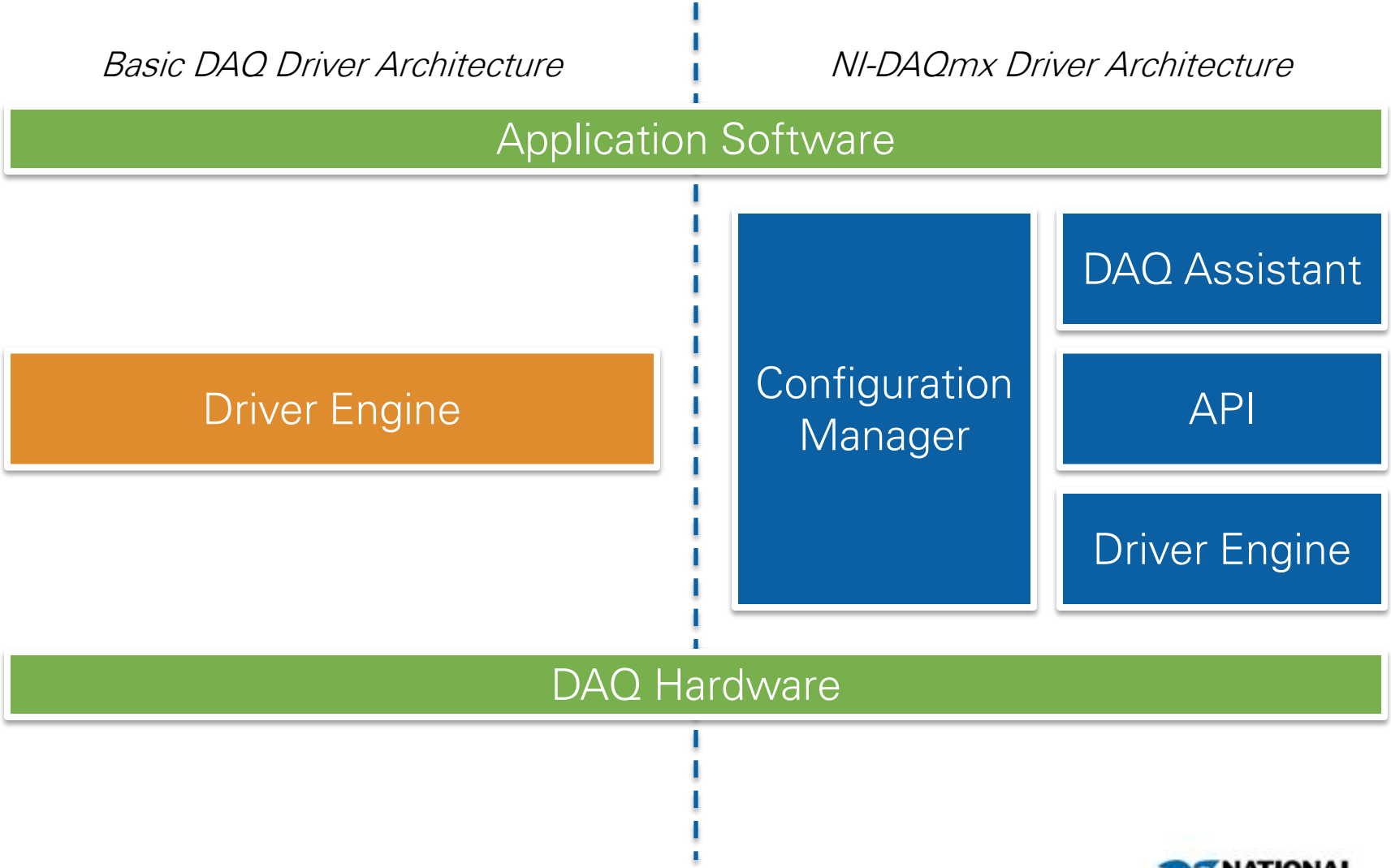
NI-DAQmx is free driver software that can be used in conjunction with several different programming languages to control thousands of different data acquisition devices with a consistent API.



Bridging the Hardware and Software Gap with NI-DAQmx

NI-DAQmx is a single, free hardware driver that supports various development languages and hundreds of NI data acquisition hardware platforms.

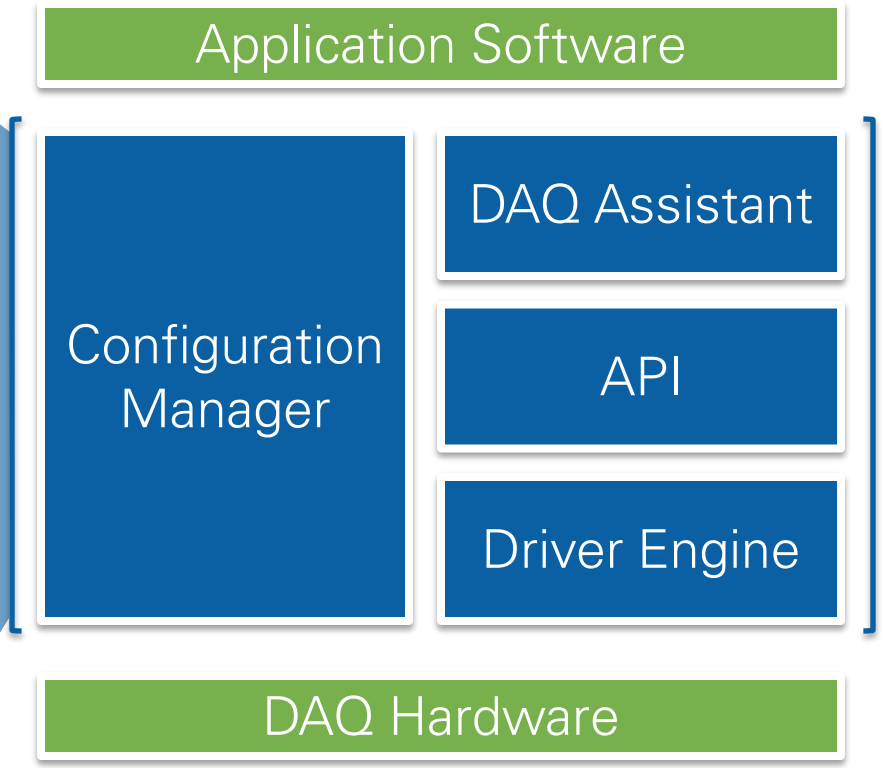
Comparing Basic DAQ Drivers to NI-DAQmx



Measurement Services With NI-DAQmx

- Streamlined API
  - Polymorphic functions
  - Automatic code generation
- Improved Architecture
  - Straightforward paradigm
  - Multithreaded measurements
  - Instant calibration
  - Optimized, even for single-point I/O

NI-DAQmx Driver Architecture



Measurement & Automation Explorer (MAX)

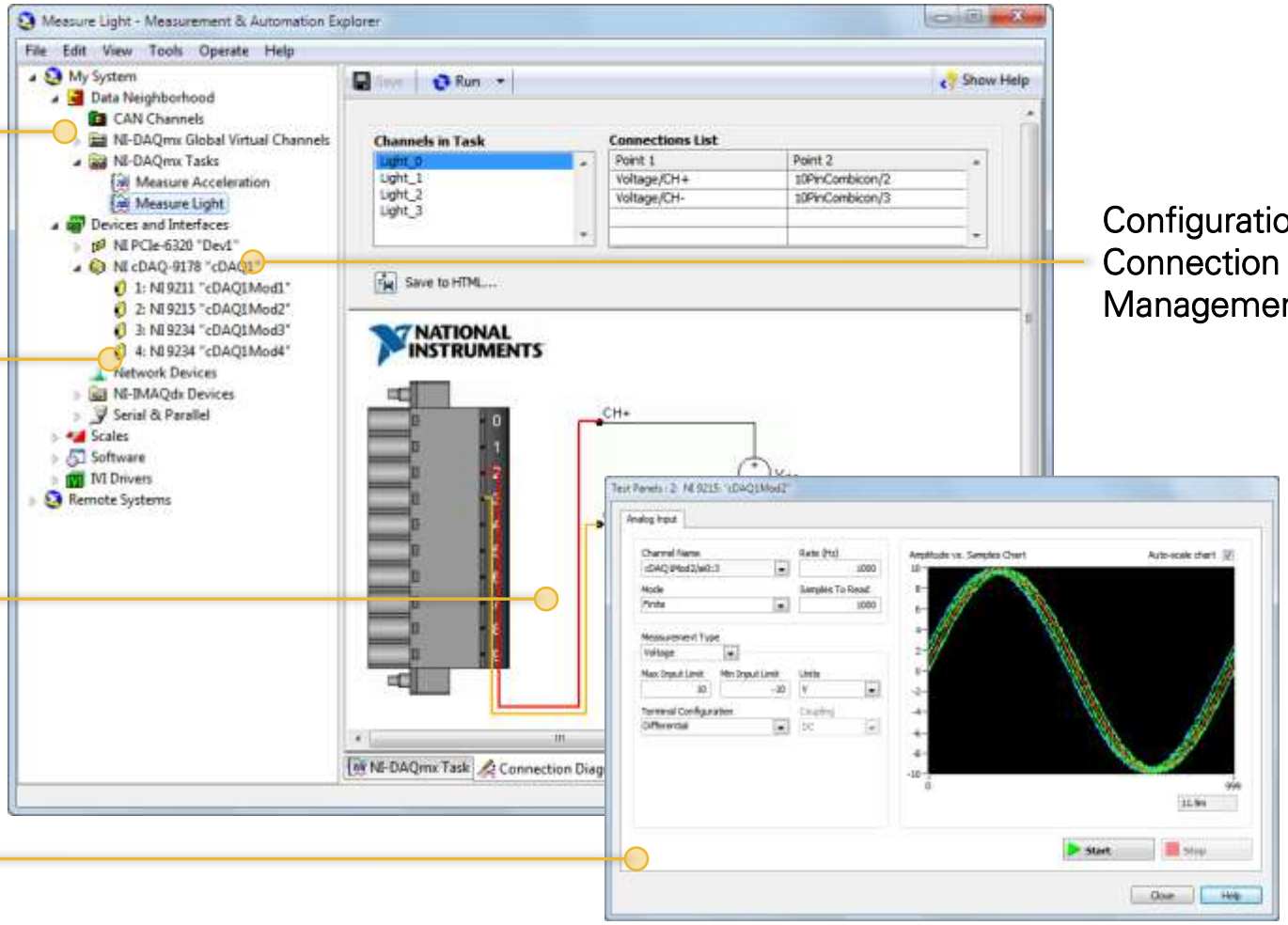
Free, unified configuration management utility for NI hardware

Task and Channel Creation

Simulated Devices

Built-In Signal Connection Diagrams

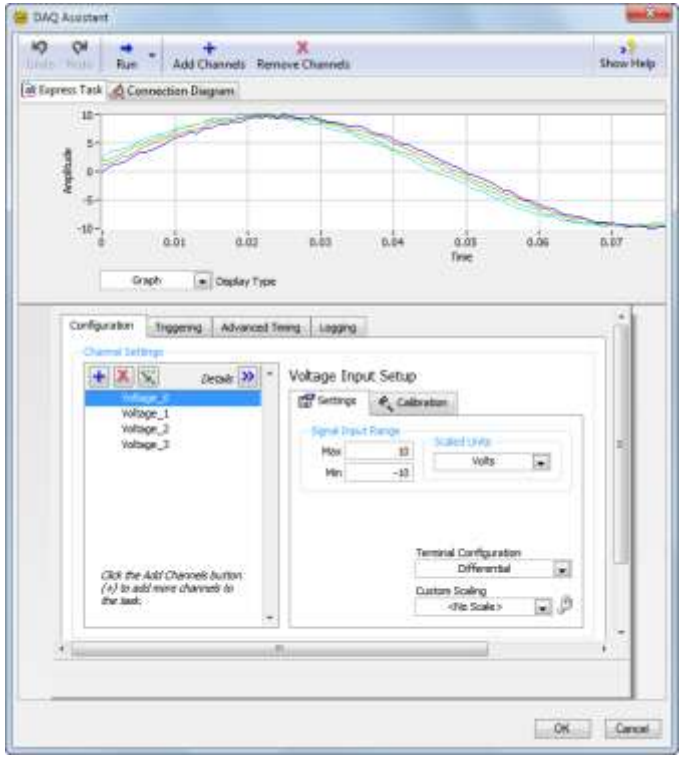
Test Panel Windows

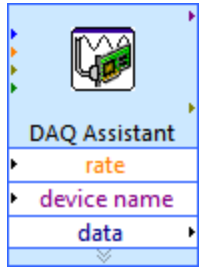


Configuration and Connection Management

NI-DAQmx API: Configuration-Based DAQ Assistant

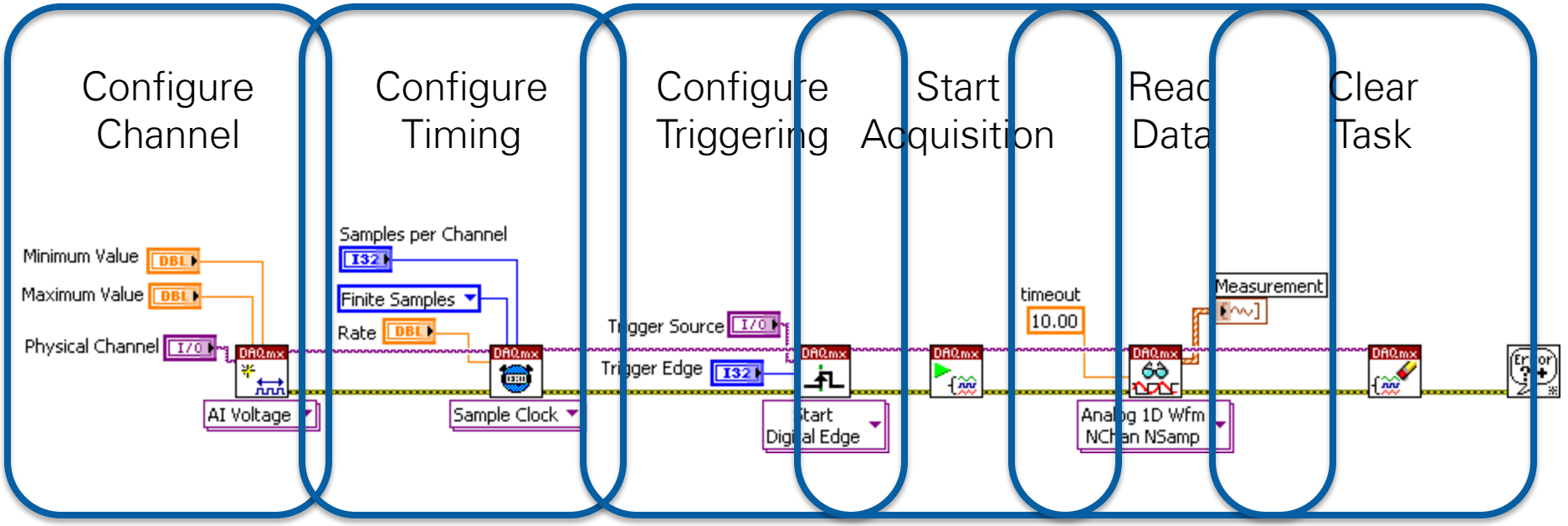
- Enables quick, configuration-based measurements
- Usable across multiple channels, multiple devices
- Maximum ease of use with some sacrificed flexibility
- Supported across multiple programming languages
- Automatically generates lower-level code





NI-DAQmx API: Low-Level LabVIEW VIs

- Maximizes flexibility and enables low-level control
- The basic flow:



NI-DAQmx C API

```
DAQmxCreateAIVoltageChan( taskHandle, "Dev1/ai0", "", DAQmx_Val_Cfg_Default,
    -10.0, 10.0, DAQmx_Val_Volts, NULL );
```

Configure Channel

```
DAQmxCfgSampClkTiming( taskHandle, "", 10000.0, DAQmx_Val_Rising,
    DAQmx_Val_FiniteSamps, 1000 );
```

Configure Timing

```
DAQmxStartTask( taskHandle );
```

Start Acquisition

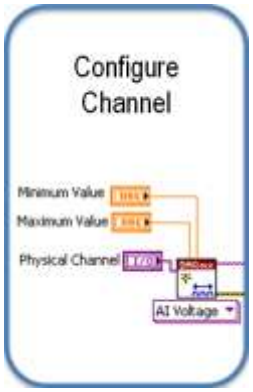
```
DAQmxReadAnalogF64( taskHandle, -1, 10.0, 0, data, 1000, &read, NULL );
    printf( "Acquired %d samples. %d", read );
```

Read Data

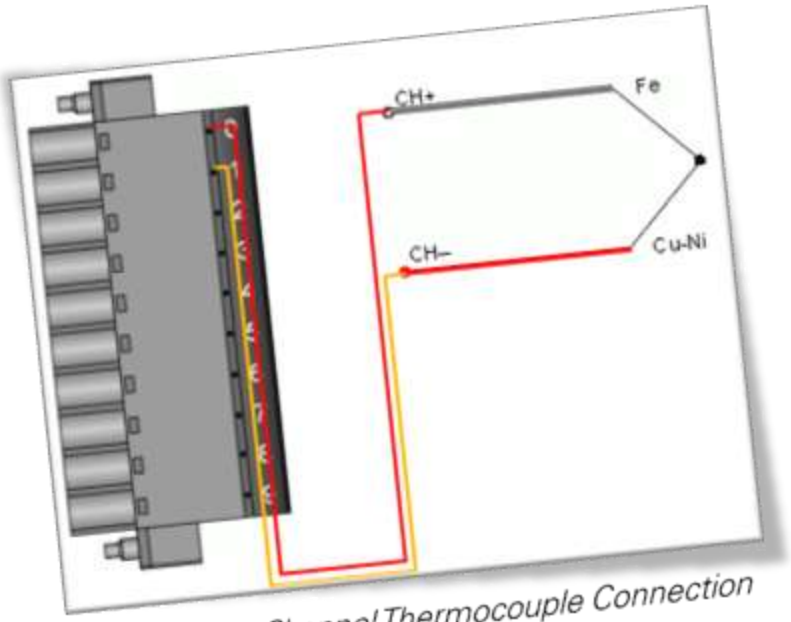
```
DAQmxClearTask( taskHandle );
```

Clear Task

NI-DAQmx Channels

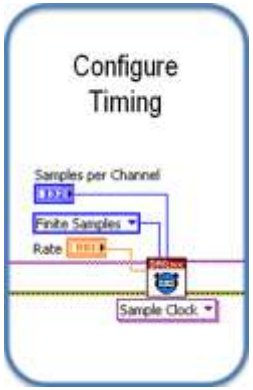


- NI-DAQmx channels encompass:
  - Measurement type, sensor/signal type
  - Terminal configuration
  - Physical connection settings
  - Name
  - Min/Max Value
    - Used to determine amplification level
  - Custom Scaling
    - Ex: thermocouple generates a mV signal; NI-DAQmx upscales to °C



NI-DAQmx Channel Thermocouple Connection Diagram

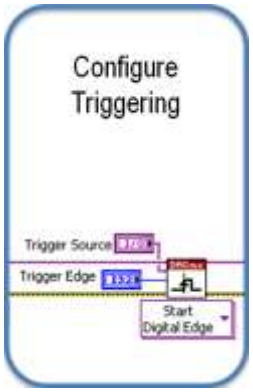
Timing



- Allows you to configure acquisition timing
- Set sample clock, rate of acquisition, and number of samples to acquire or generate

Timing Option	Description
Finite Samples	Acquire or generate a configurable number of samples at a configurable rate.
Continuous Samples	Acquire or generate samples continuously, until explicitly stopped by the API.
Hardware-Timed Single Point	Acquire or generate samples continuously on the edge of a hardware clock.

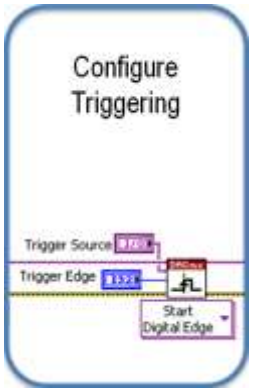
Triggering



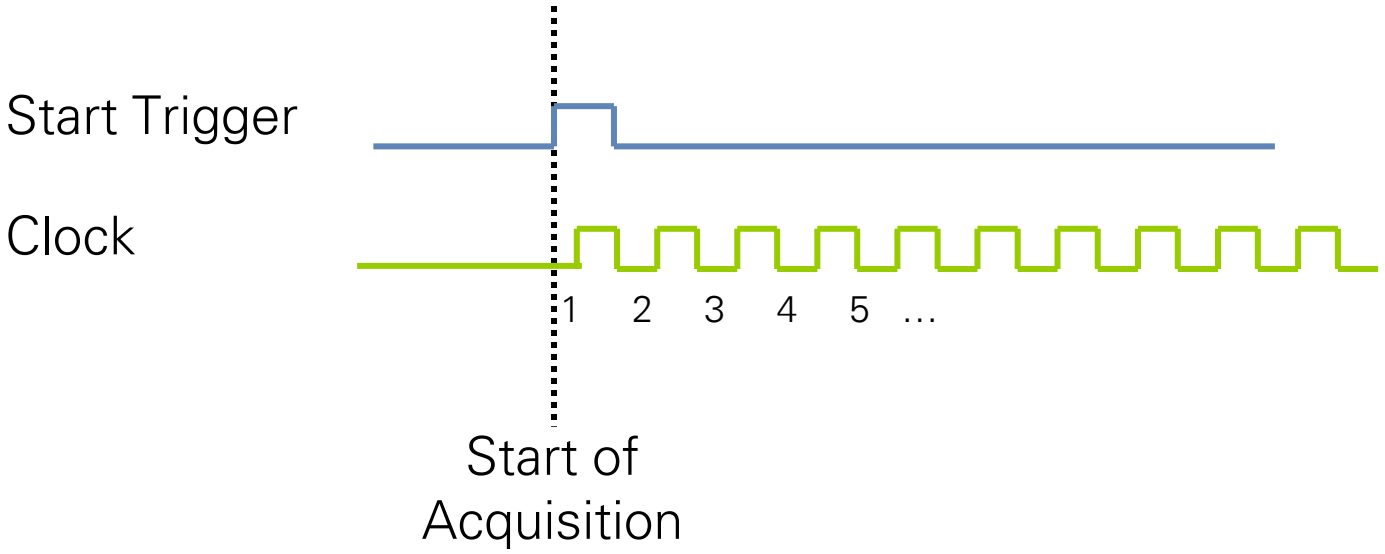
- Produces an action based on a stimulus
  - Ex: generate a waveform after receiving a digital pulse
- NI-DAQmx supports several different action types:

Advance	Pause	Reference	Start
• Switch to the next device in a list	• Pause when a trigger is low • Resume when a trigger is high	• Acquisition starts with software • Circular buffer is used until reference trigger is received • Returns pre- and post-trigger samples	• Begin acquisition • Begin generation

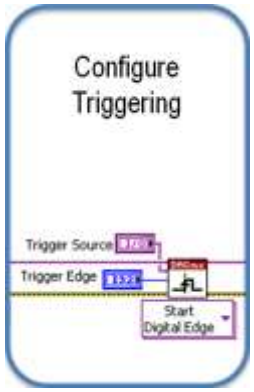
Triggering



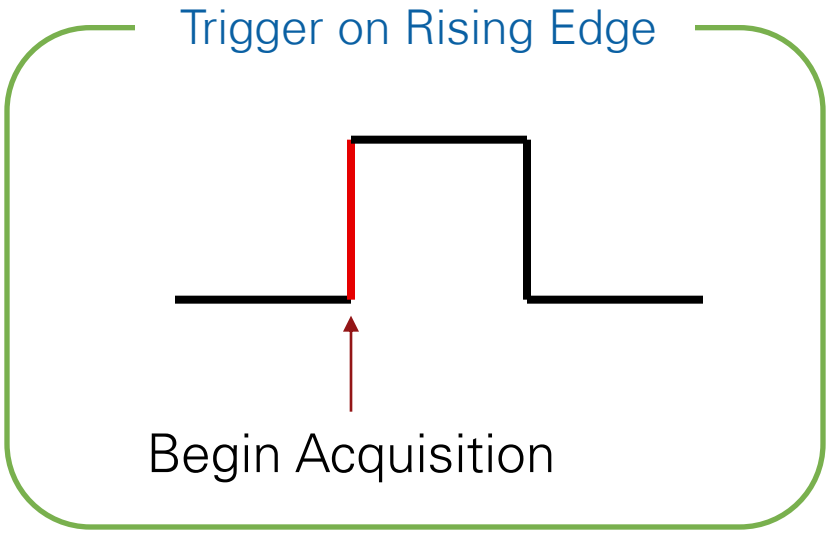
- Event-driven acquisition or generation
- Valid for finite or continuous operations
- Example: acquire 5 samples on a start trigger:

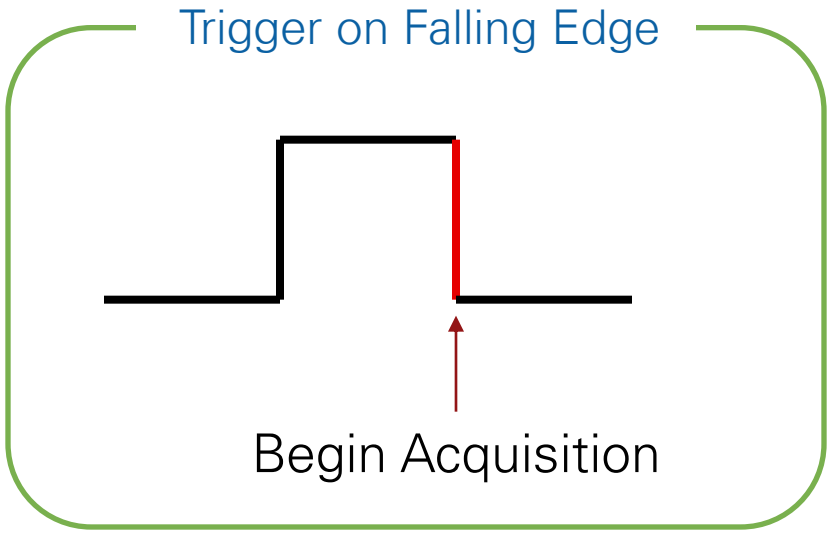


Trigger Types—Digital Edge Triggering

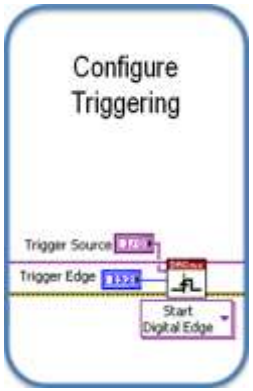


- Accepts TTL/CMOS-compatible signals
  - 0 to 0.8 V = logic low
  - 2.2 to 5 V = logic high
- Trigger on rising or falling edge of signal

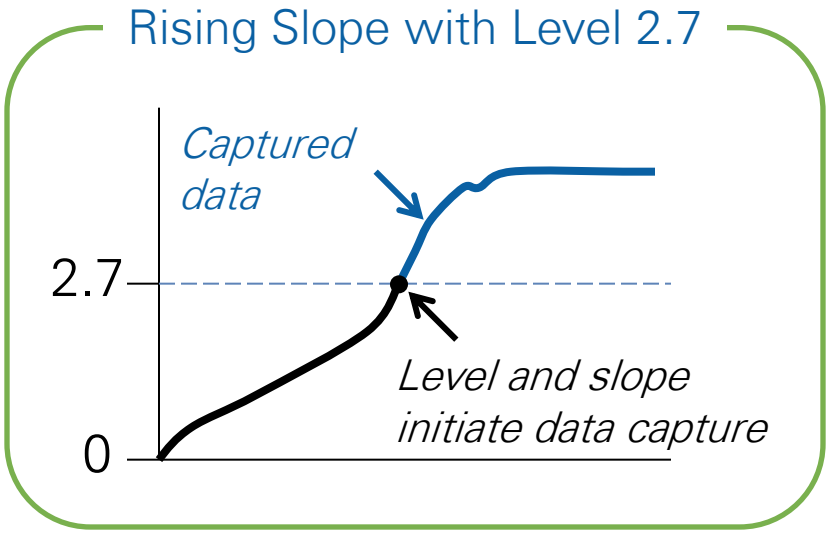


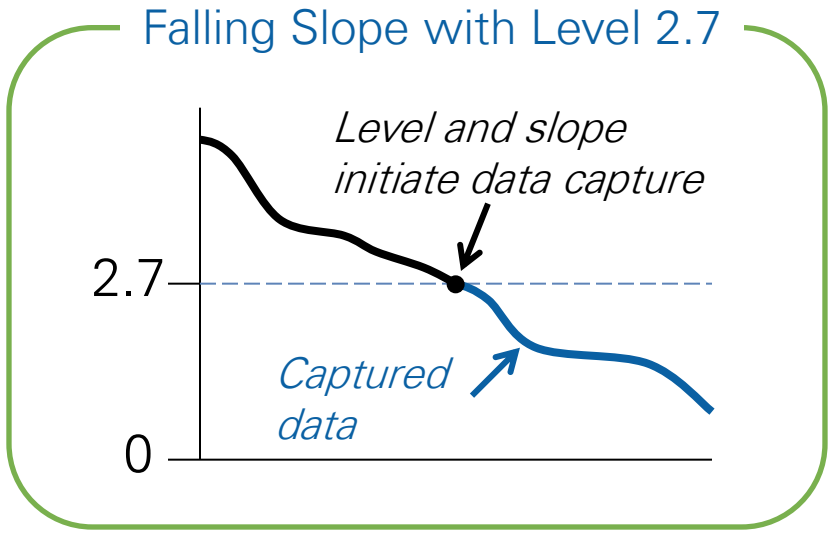


Trigger Types—Analog Edge Triggering



- Trigger off signal level and slope
- Slope can be rising or falling





Simplifying Code With NI-DAQmx Tasks

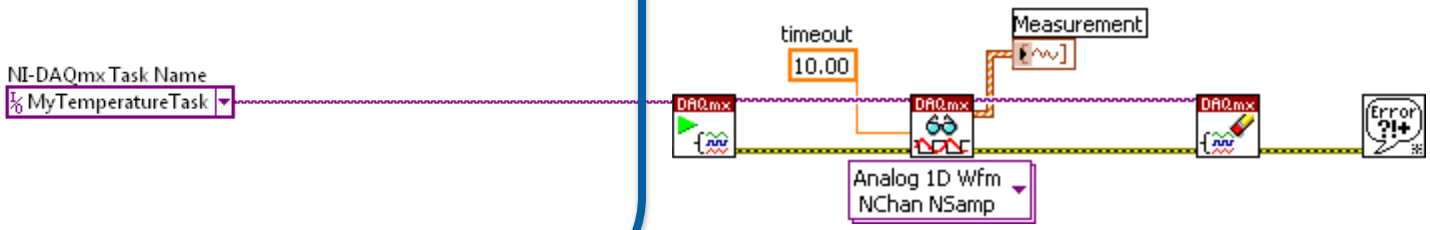
A *Task* is a collection of **channels** with homogenous **timing** and **triggering.**

Configure Channel

Configure Timing

Configure Triggering

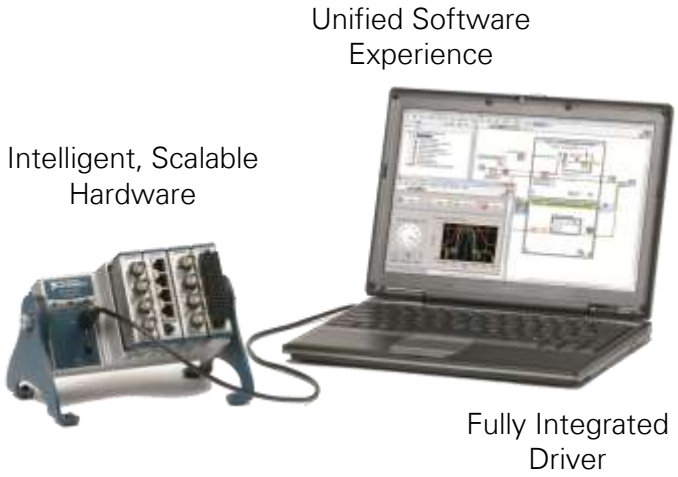
NI-DAQmx Task



NATIONAL INSTRUMENTS

The Most Productive, Flexible Approach

to building accurate and reliable automated measurement systems



1. Accelerated Productivity
2. Proven Performance and Accuracy
3. Scalability, Adaptability, and Flexibility