



UiO • University of Oslo

FYS3240

PC-based instrumentation and microcontrollers

Instrumentation and data acquisition

Spring 2013 – Lecture #6



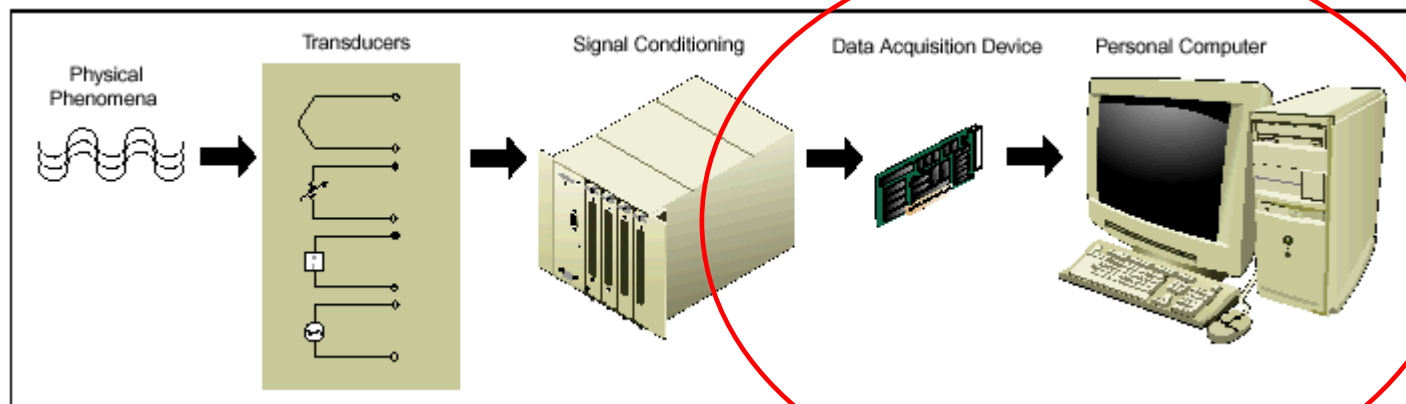
Overview

- Overview of Data acquisition
- DAQ signals
- Signal conditioning
- Sampling concepts
- DAQ hardware
 - PXI
- Device drivers
- Software development for PC-based DAQ (using LabVIEW)
- Hardware timing
- A few other examples (including VISION)

Data acquisition (DAQ)

- **Data acquisition involves measuring signals (from a real-world physical system) from different sensors, and digitizing the signals for storage, analysis and presentation.**
- Analog input channels can vary in number from one to several hundred or even thousands

Computer-based DAQ system:

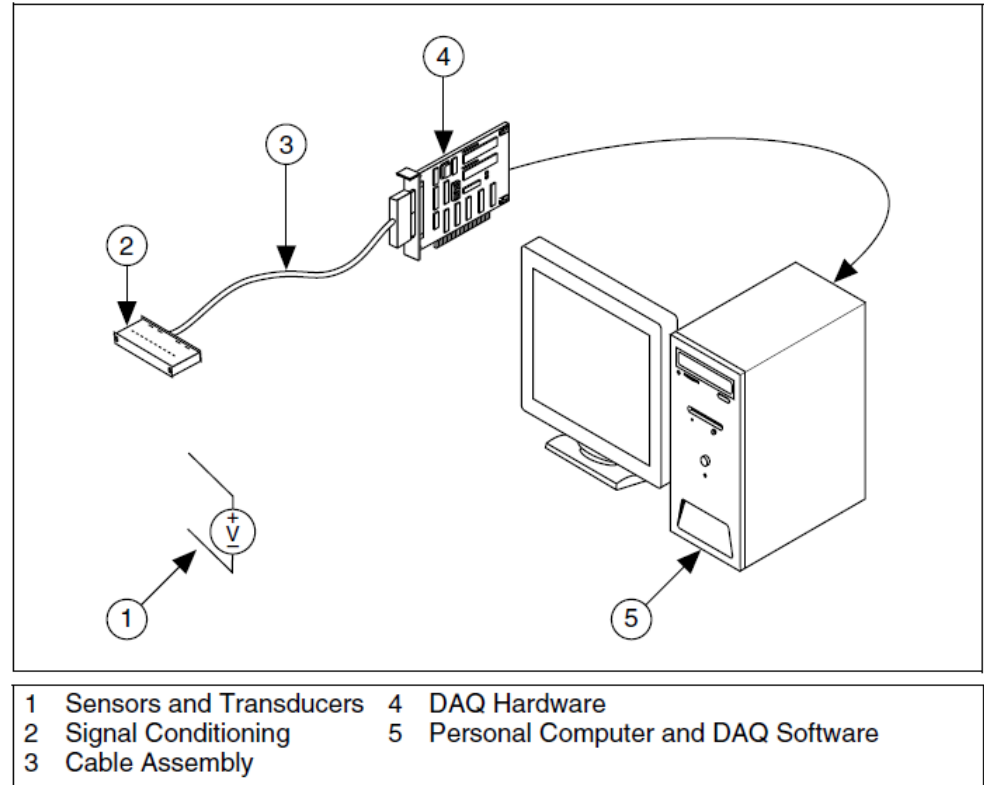


NI SC DAQ

Overview of Data acquisition (DAQ)

A DAQ system consists of:

- Sensors (transducers)
- Signal Conditioning
- Cables
- DAQ hardware
- Drivers
- Software



DAQ fundamentals tutorial from NI:
<http://zone.ni.com/devzone/cda/tut/p/id/3216>

Types of Data Acquisition Systems

- **DAQ types and some of their characteristics:**
 1. Laboratory DAQ
 - Permanent location
 - Can be a large and heavy installation
 - Often rack mounted (19-inch racks)
 - PC-based
 2. Portable DAQ
 - Small and light
 - PC-based (connected to a PC), or
 - Stand-alone units, like data loggers, that do not need a PC connection (e.g. a flight data recorder)
 3. Both 1. and 2. can be required to be ruggedized for use in the field
 - Shock and vibration tolerant
 - Water and dust protected



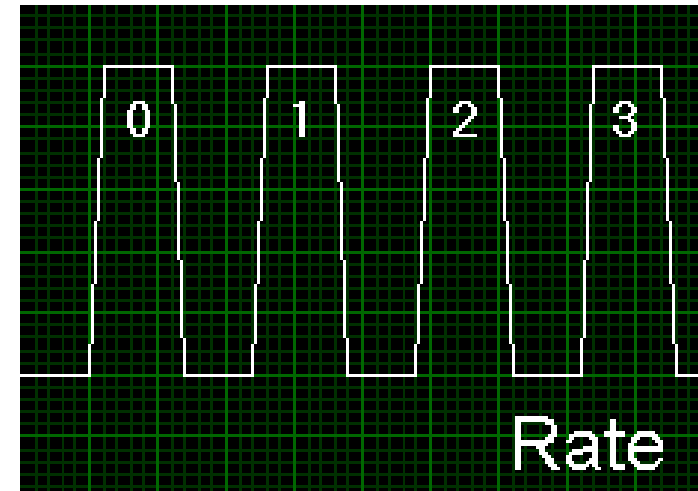
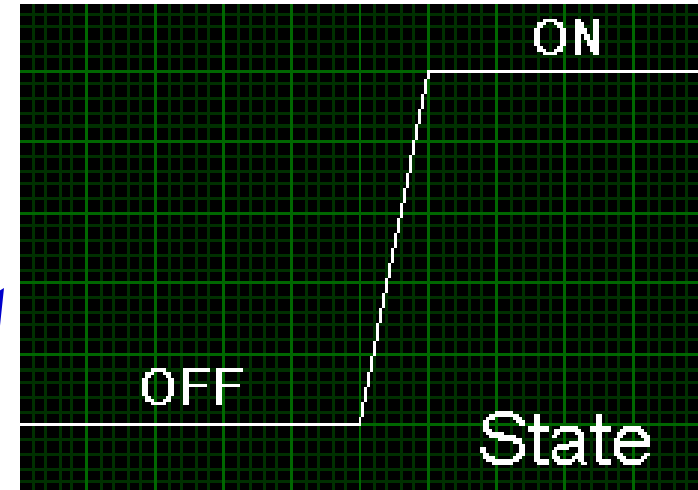
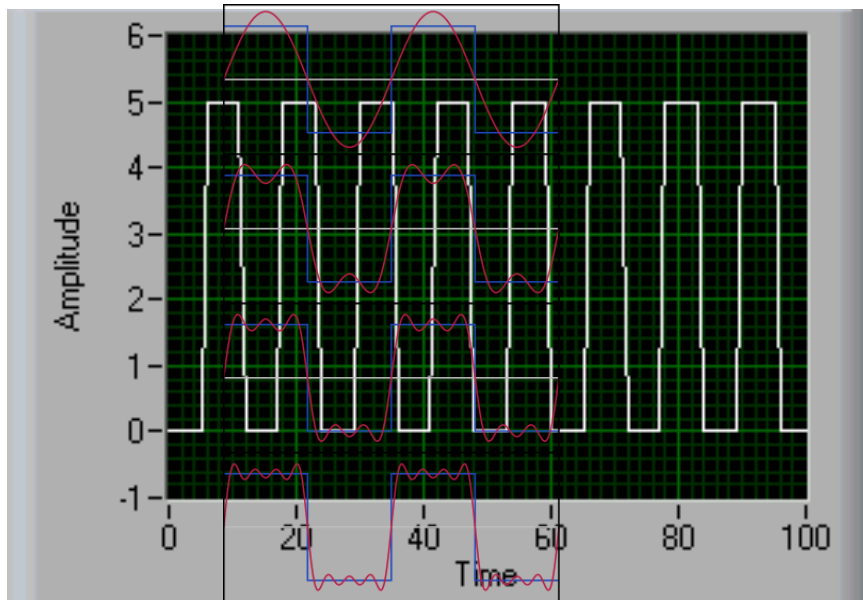
Categories of DAQ signals

- Analog input
- Analog output
- Digital I/O
- Counter
 - *Quadrature encoder input for rotational measurements*
 - *Digital edge counting*
 - *Frequency measurements*
- Other signals: bus-based and serial

Digital Signal Information

Two types of information:

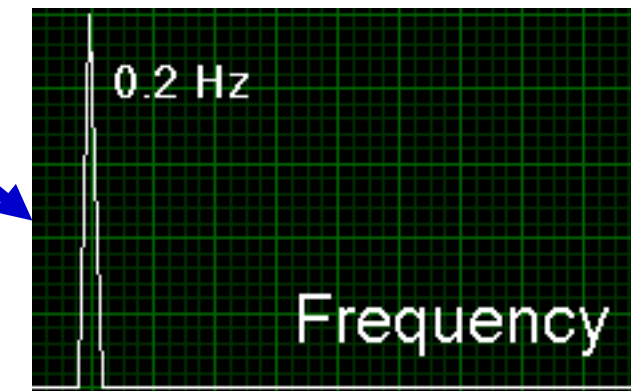
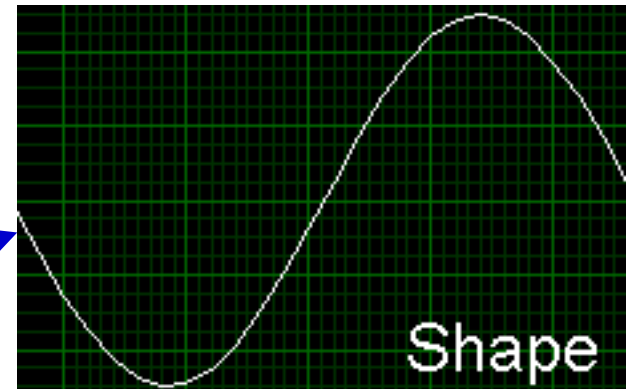
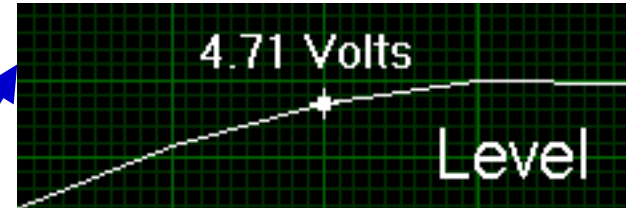
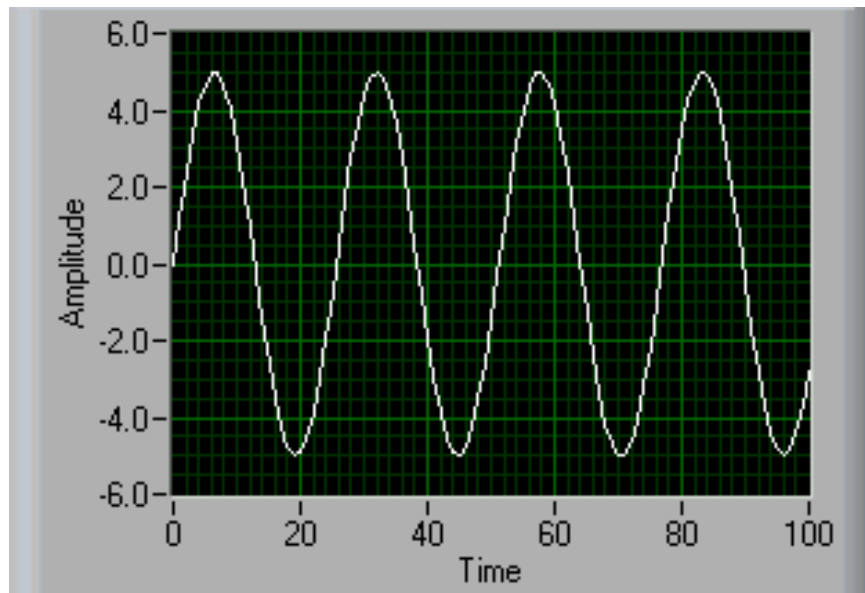
- State
- Rate



Analog Signal Information

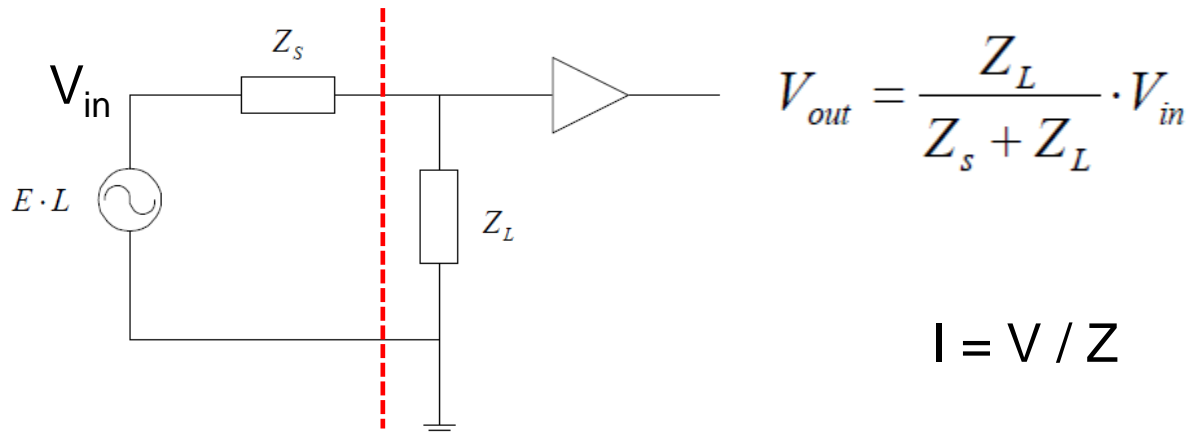
Three types of information:

- Level
- Shape
- Frequency



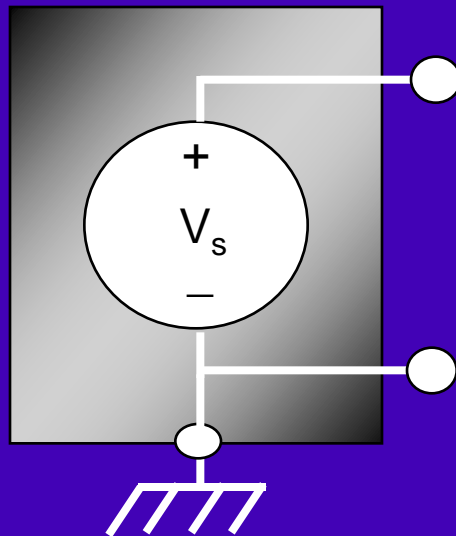
Considerations for analog signals

- Signal source - grounded or floating
- Source impedance
 - The DAQ device must have a much higher input impedance than the signal source
 - This is usually not a problem as the DAQ devices are designed to have a very high input impedance (GΩ range)
- Single-ended & differential signals

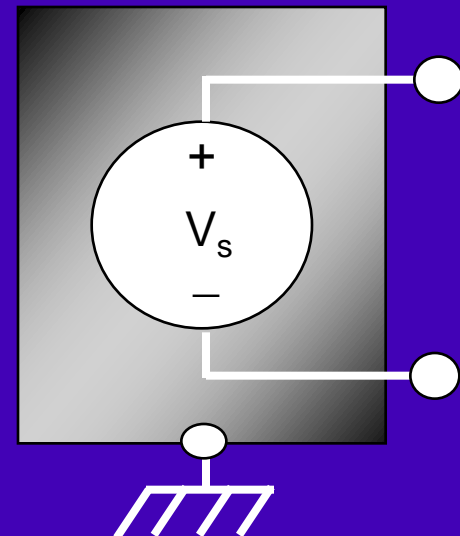


Signal Source Categories

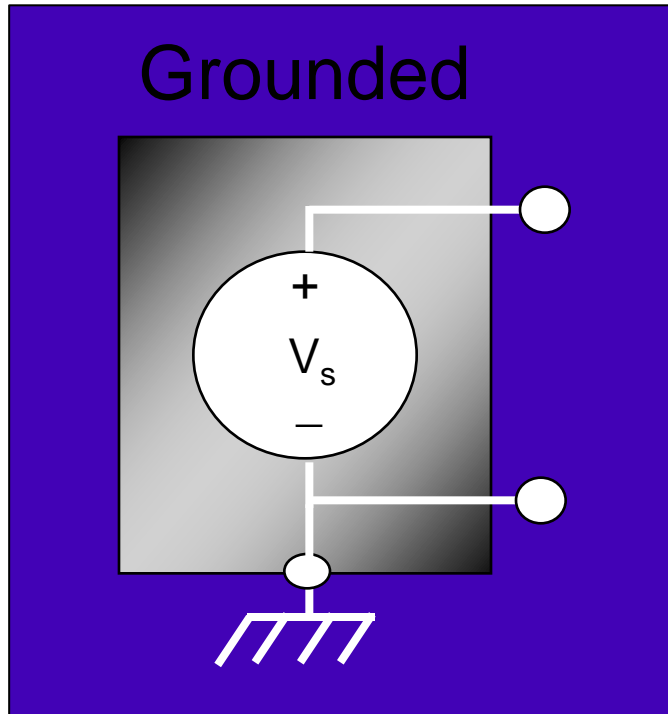
Grounded



Floating



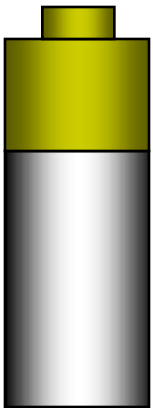
Grounded Signal Source



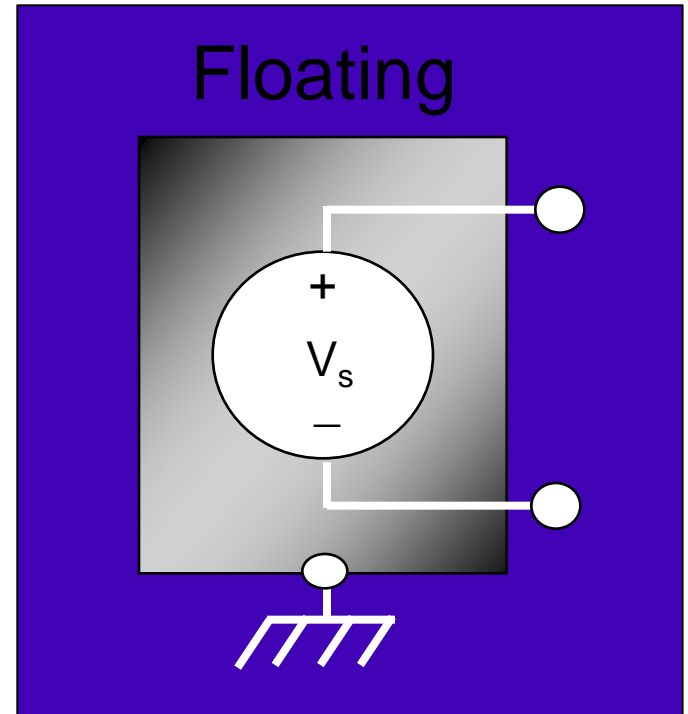
- Signal is referenced to a system ground
 - earth ground
 - building ground
- Examples:
 - Power supplies
 - Signal Generators
 - Anything that plugs into an outlet ground



Floating Signal Source



- Signal is NOT referenced to a system ground
 - earth ground
 - building ground
- Examples:
 - Batteries
 - Thermocouples
 - Transformers
 - Isolation Amplifiers

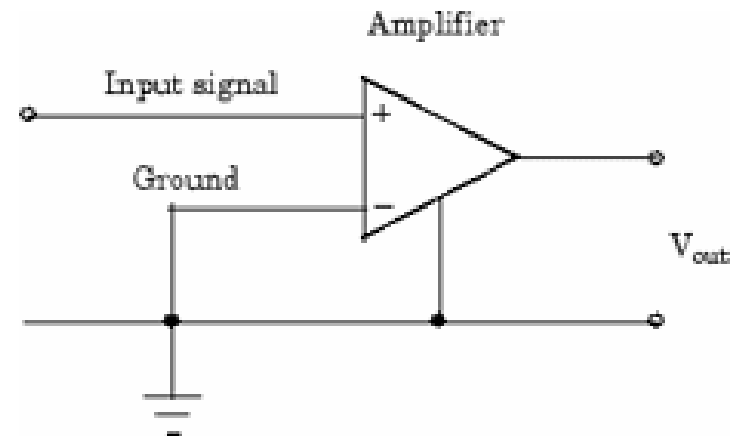


DAQ-card input signal configuration

- DAQ input channels can be configured in two ways:
 - Differential
 - Single-ended
 - Referenced Single-Ended (RSE)
 - Non-Referenced Single-Ended (NRSE)
- The optimal connection depends on how your signal is grounded

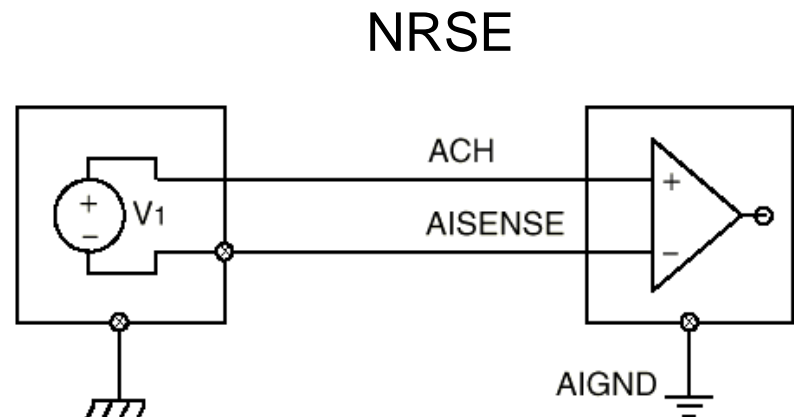
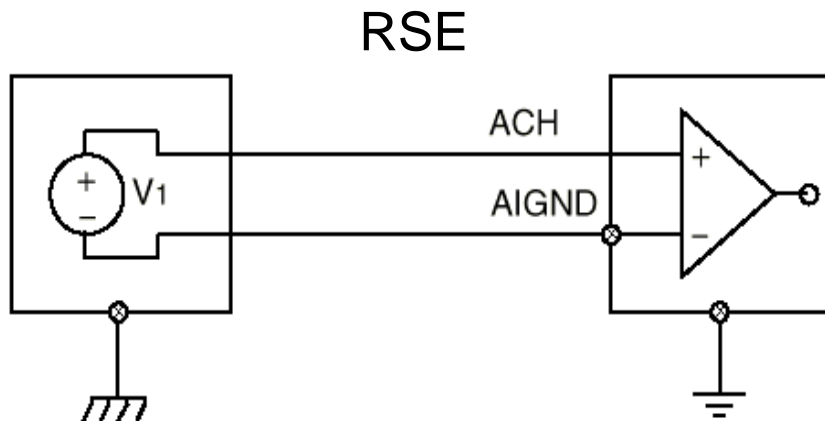
Single-ended (SE) signals

- One signal wire for each input signal
- Can be used for the following conditions:
 - High-level input signals (greater than 1 V)
 - Short cables
 - Properly-shielded cables or cables traveling through a noise-free environment
 - All input signals can share a common reference point (ground)
- Two types of connections:
 - Referenced Single-Ended (RSE)
 - Non-Referenced Single-Ended (NRSE)



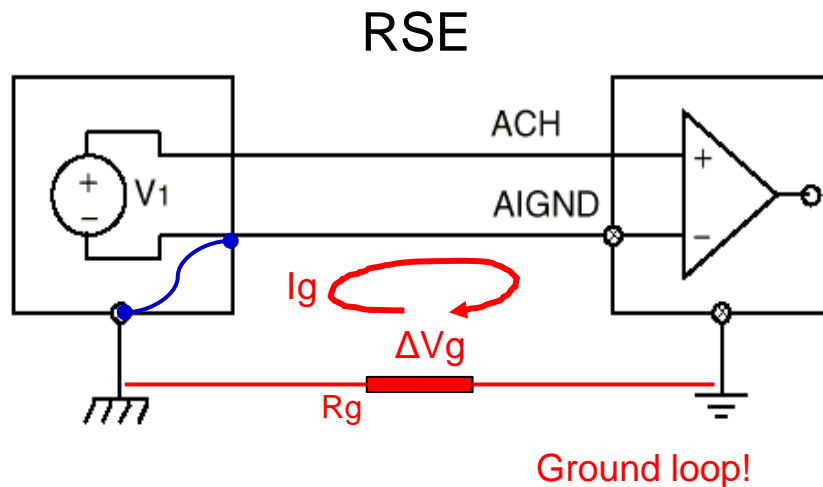
RSE vs. NRSE configuration

- The **RSE** configuration is used for floating signal sources. In this case, the DAQ hardware device itself provides the reference ground for the input signal.
- The **NRSE** input configuration is used for grounded signal sources. In this case, the input signal provides its own reference ground and the hardware device should not supply one.
 - Measurement made with respect to a common reference (AISENSE), not system ground (AIGND)
 - AISENSE is floating



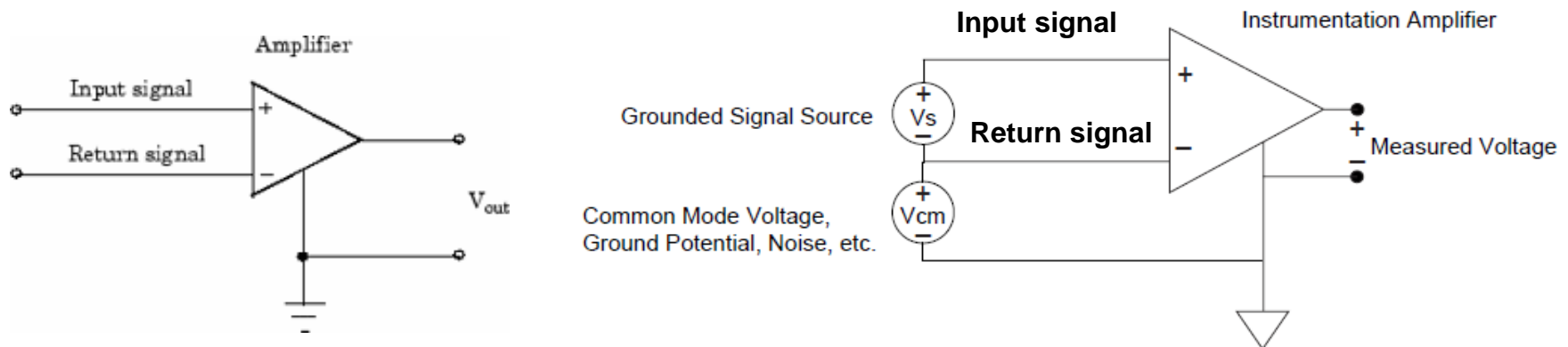
Ground loop illustration

The blue connection to ground must not be added, since it creates a ground loop

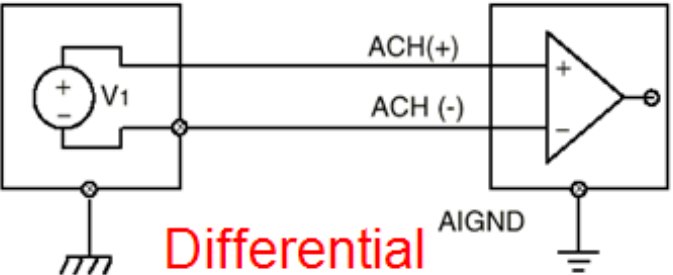
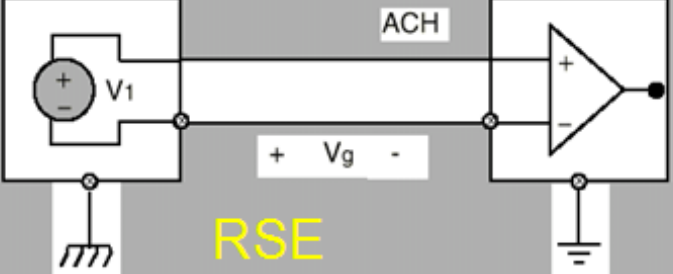
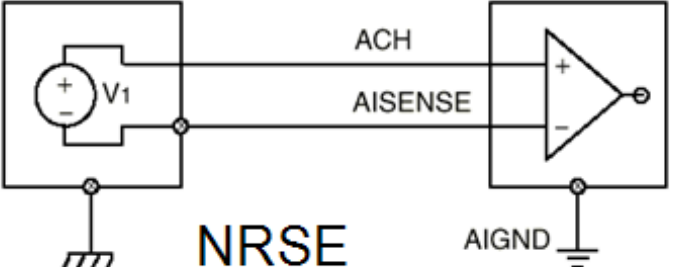


Differential signals

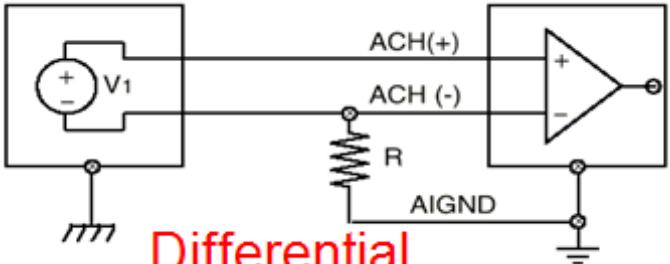
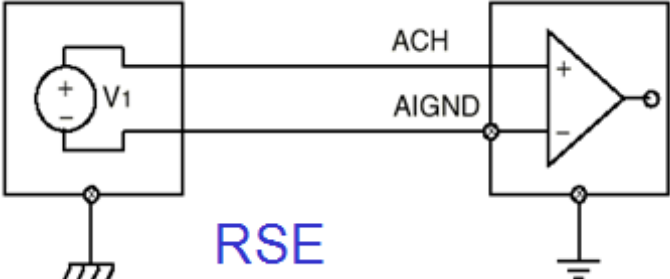
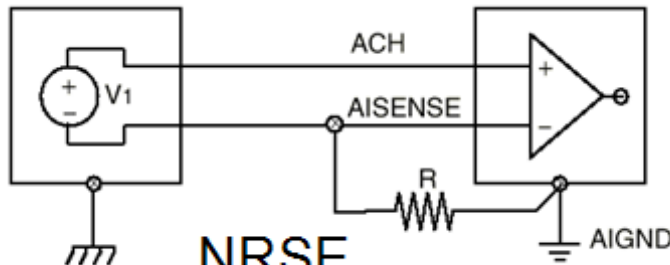
- Two signal wires for each input signal (input and return signals)
- The measurement is the voltage difference between the two wires
- Recommended for the following conditions:
 - Low-level signals (less than 1 V)
 - Long cables
 - The input signal requires a separate ground-reference point or return signal
 - The signal leads go through a noisy environment
- DAQ devices with instrumentation amplifiers can be configured as differential measurement systems
- Any voltage present at the instrumentation amplifier inputs with respect to the amplifier ground is called a common-mode voltage
- The instrumentation amplifier rejects common-mode voltage and common-mode noise



Options for Grounded Signal Sources

 <p>Differential</p>	<p><u>BETTER</u></p> <ul style="list-style-type: none"> + Rejects Common-Mode Voltage - Cuts Channel Count in Half
 <p>RSE</p>	<p><u>NOT RECOMMENDED</u></p> <ul style="list-style-type: none"> - Voltage difference (V_g) between the two grounds makes a ground loop that could damage the device
 <p>NRSE</p>	<p><u>GOOD</u></p> <ul style="list-style-type: none"> + Allows use of entire channel count - Doesn't reject Common-Mode Voltage

Options for Floating Signal Sources

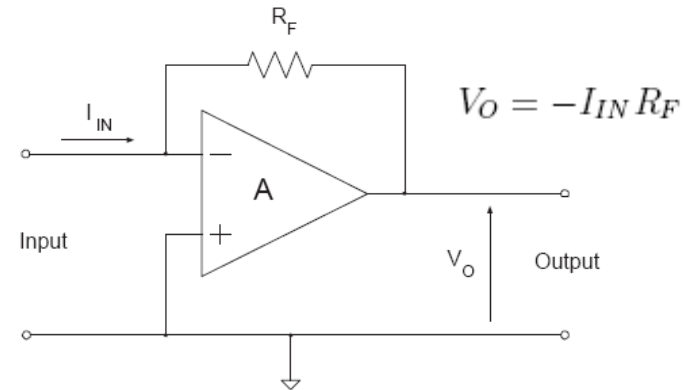
 <p>Differential</p>	<p><u>BEST</u></p> <ul style="list-style-type: none"> + Rejects Common-Mode Voltage - Cuts Channel Count in Half - Need bias resistors
 <p>RSE</p>	<p><u>BETTER</u></p> <ul style="list-style-type: none"> + Allows use of entire channel count + Don't need bias resistors - Doesn't reject Common-Mode Voltage
 <p>NRSE</p>	<p><u>GOOD</u></p> <ul style="list-style-type: none"> + Allows use of entire channel count - Need bias resistors - Doesn't reject Common-Mode Voltage

Signal conditioning

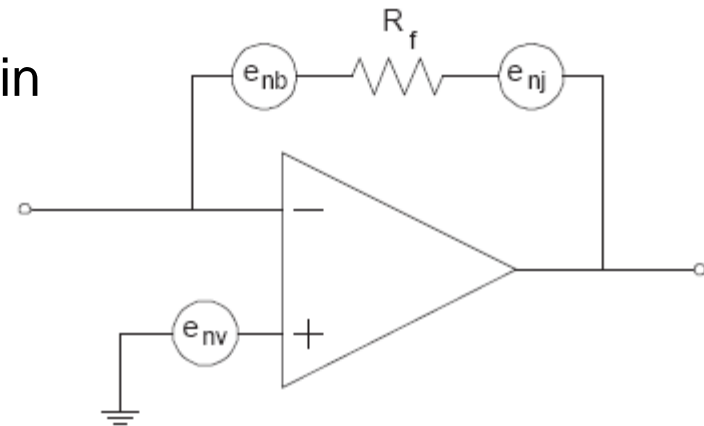
- Signal conversion
 - E.g. current-voltage converter
- Amplification
- Attenuation
 - Voltage divider
- Filtering
 - Anti-aliasing
- Isolation

Current-to-voltage converter

- Transimpedance amplifier (Feedback Ammeter)
- Recommended connection for small currents
- Sensitivity determined by R_f
- Add a capacitor C_f in parallel with R_f to avoid oscillations
- R_f usually large to achieve a large gain
- e_{nb} dominate for large R_f



Noise equivalent circuit:



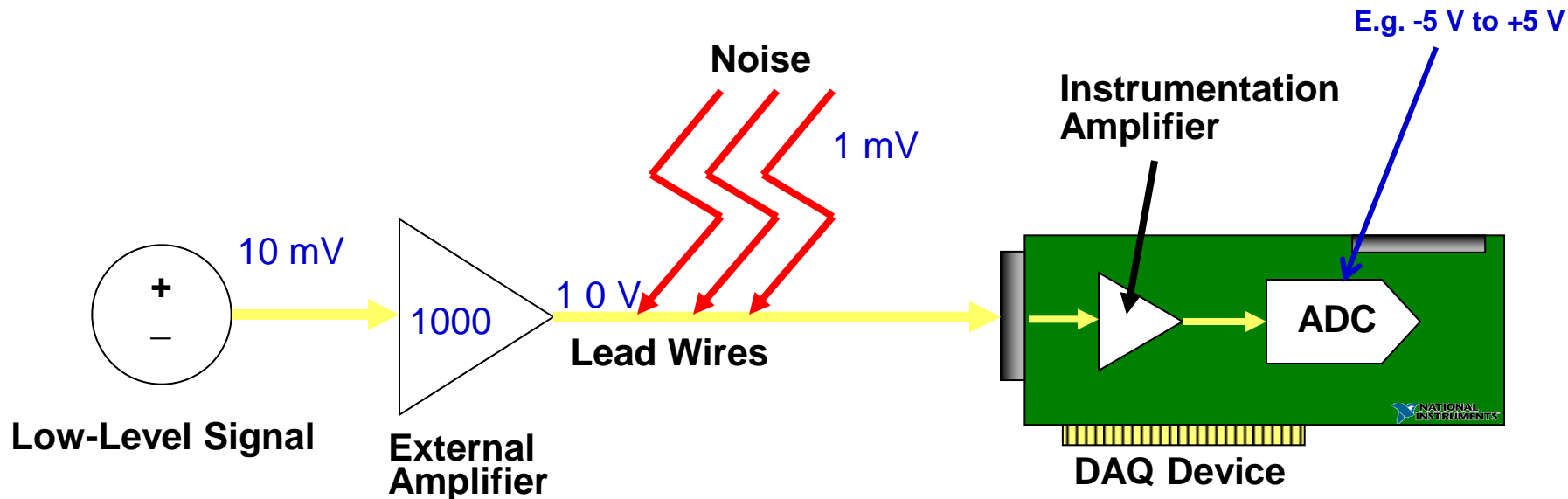
e_{nb} = input current noise * R_f

e_{nv} = input voltage noise

e_{nj} = thermal noise (voltage)

Amplification

- Used on low-level signals (less than around 100 mV)
- Maximizes use of Analog-to-Digital Converter (ADC) range and increases accuracy
- Increases Signal to Noise Ratio (SNR)

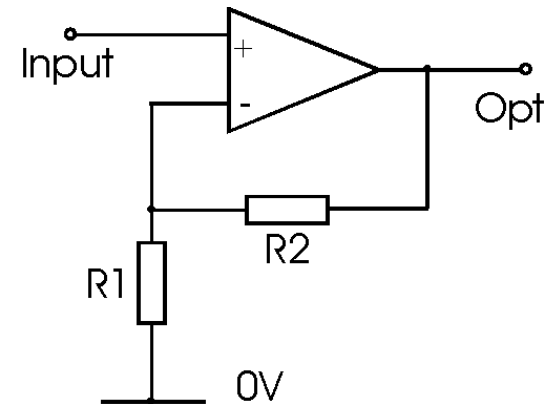
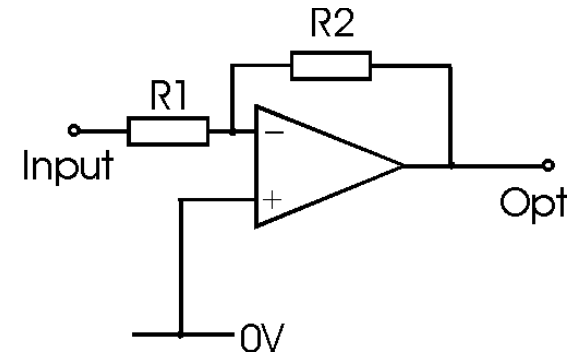


$$SNR = V_{\text{signal}} / V_{\text{noise}} = (10 \text{ mV} * 1000) / 1 \text{ mV} = 10\,000$$

$$SNR = 20 \log \left(\frac{V_{\text{signal}}}{V_{\text{noise}}} \right)$$

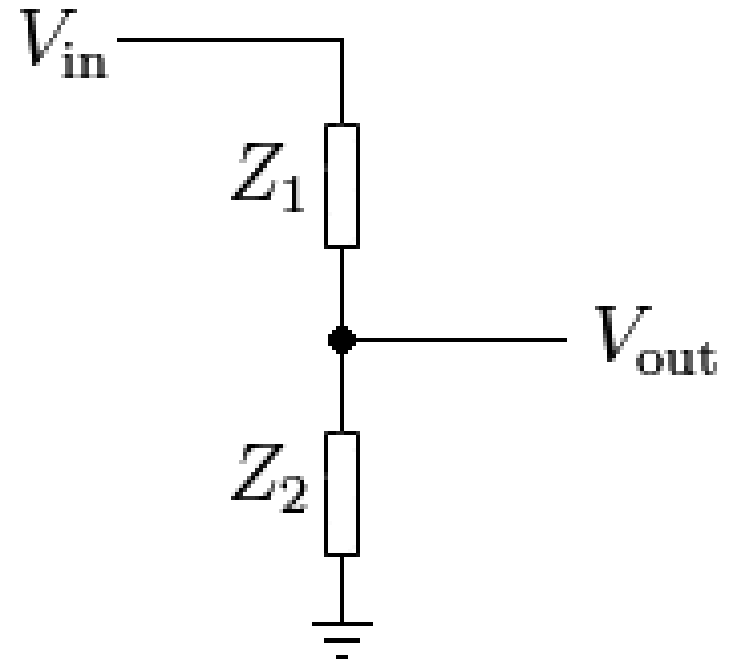
Operational amplifier (Op-amp)

- Inverting op-amp amplifier
 - $V_o = -R_2/R_1 * V_i$
- Non-inverting op-amp amplifier
 - $V_o = (1+R_2/R_1) * V_i$
- Non-inverting op-amp amplifier useful when a **high impedance** input is needed
- Inverting op-amp amplifier useful when a **low impedance** input is needed
- Non-inverting op-amp amplifier gives less noise (due to $G = 1+R_2/R_1$ instead of $G = -R_2/R_1$)



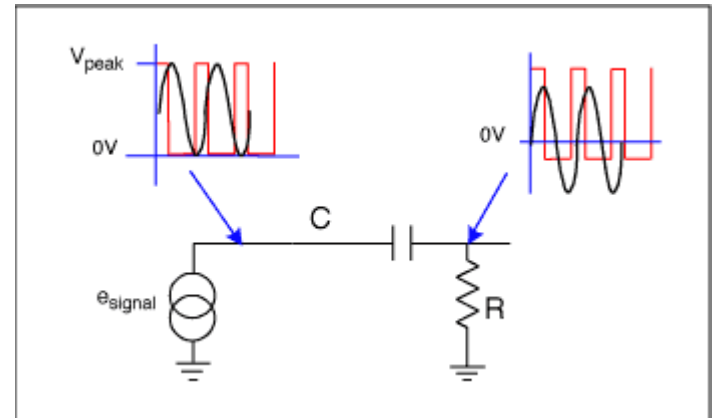
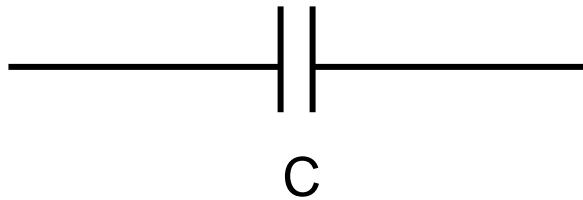
Attenuation

- Voltage divider
- A circuit that produces an output voltage (V_{out}) that is a fraction of its input voltage (V_{in})
- Can be needed to get a high-level signal down to the acceptable DAQ-card range



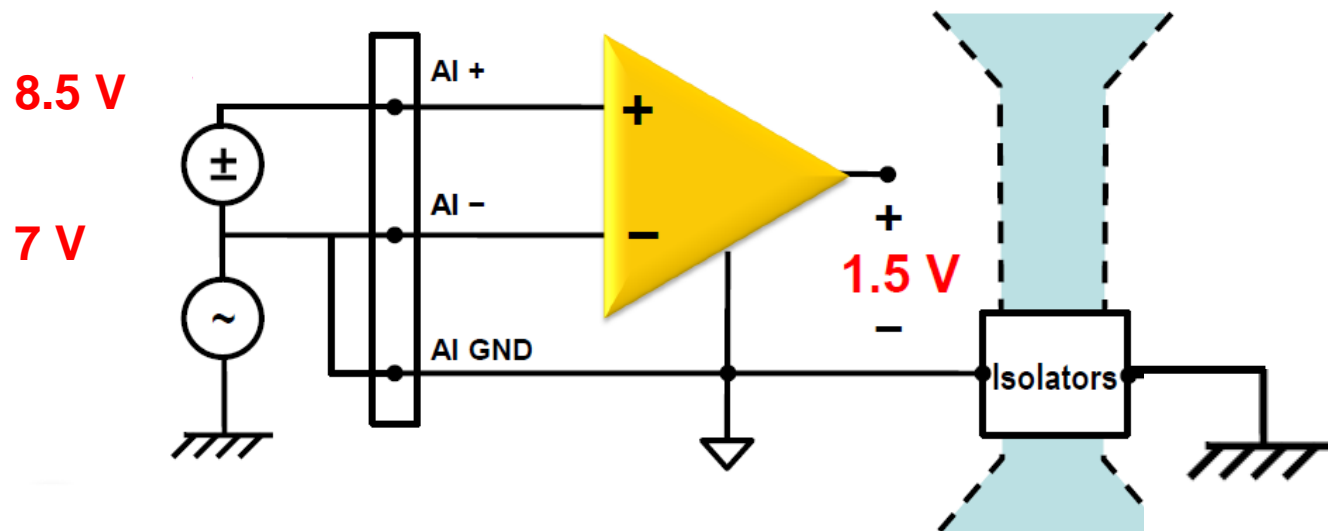
Input Coupling

- Use AC coupling when the signal contains a large DC component. If you enable AC coupling, you remove the large DC offset for the input amplifier and amplify only the AC component. This configuration makes effective use of the ADC dynamic range



Isolation amplifiers

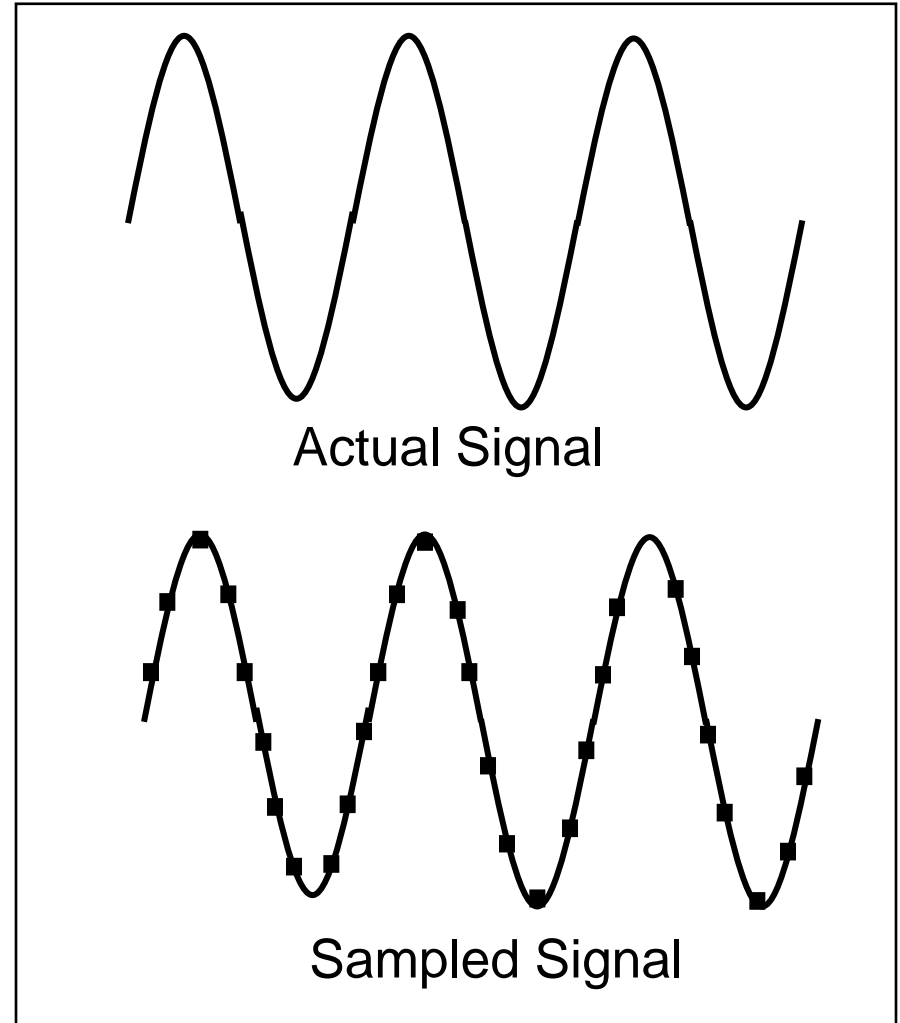
- Isolation electrically separates two parts of a measurement device
- Protects from high voltages
- Prevents ground loops
 - when two connected points are at a different ground potential, creating a current flow in the interconnection, which produces an offset error
- Separate ground planes of data acquisition device and sensor
- Isolation techniques: Optical, Capacitive, Inductive Coupling



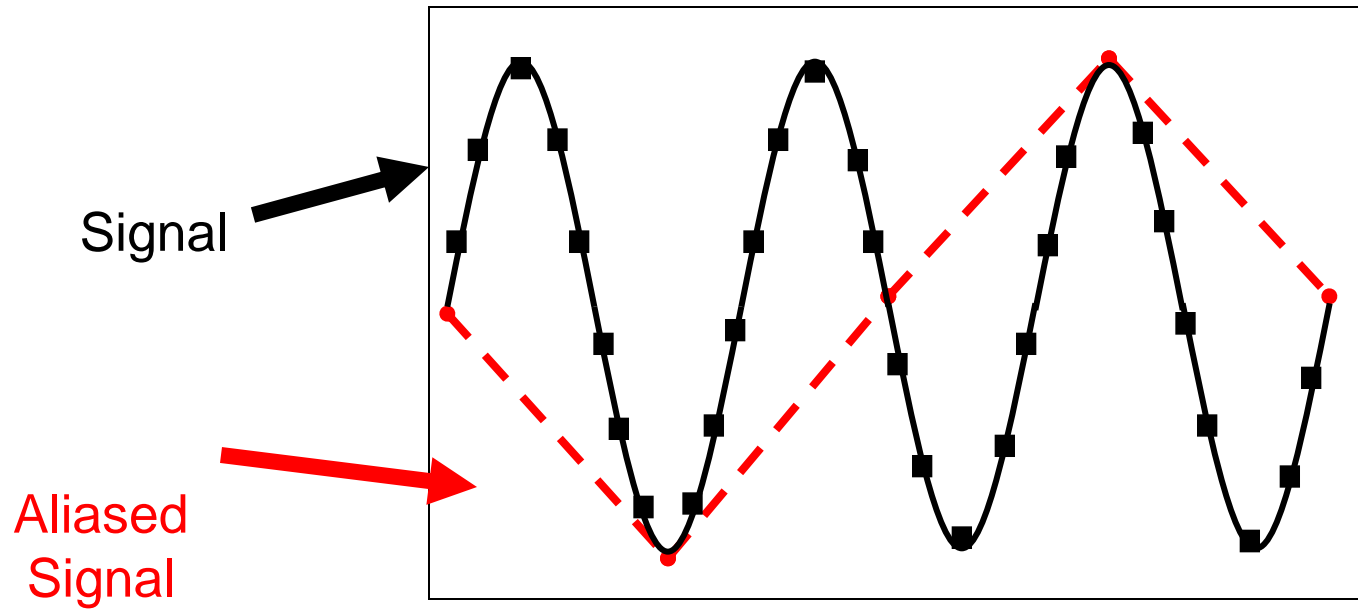
- Before we look at hardware filtering, lets have a look at sampling concepts

Sampling Considerations

- An analog signal is continuous
- A sampled signal is a 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



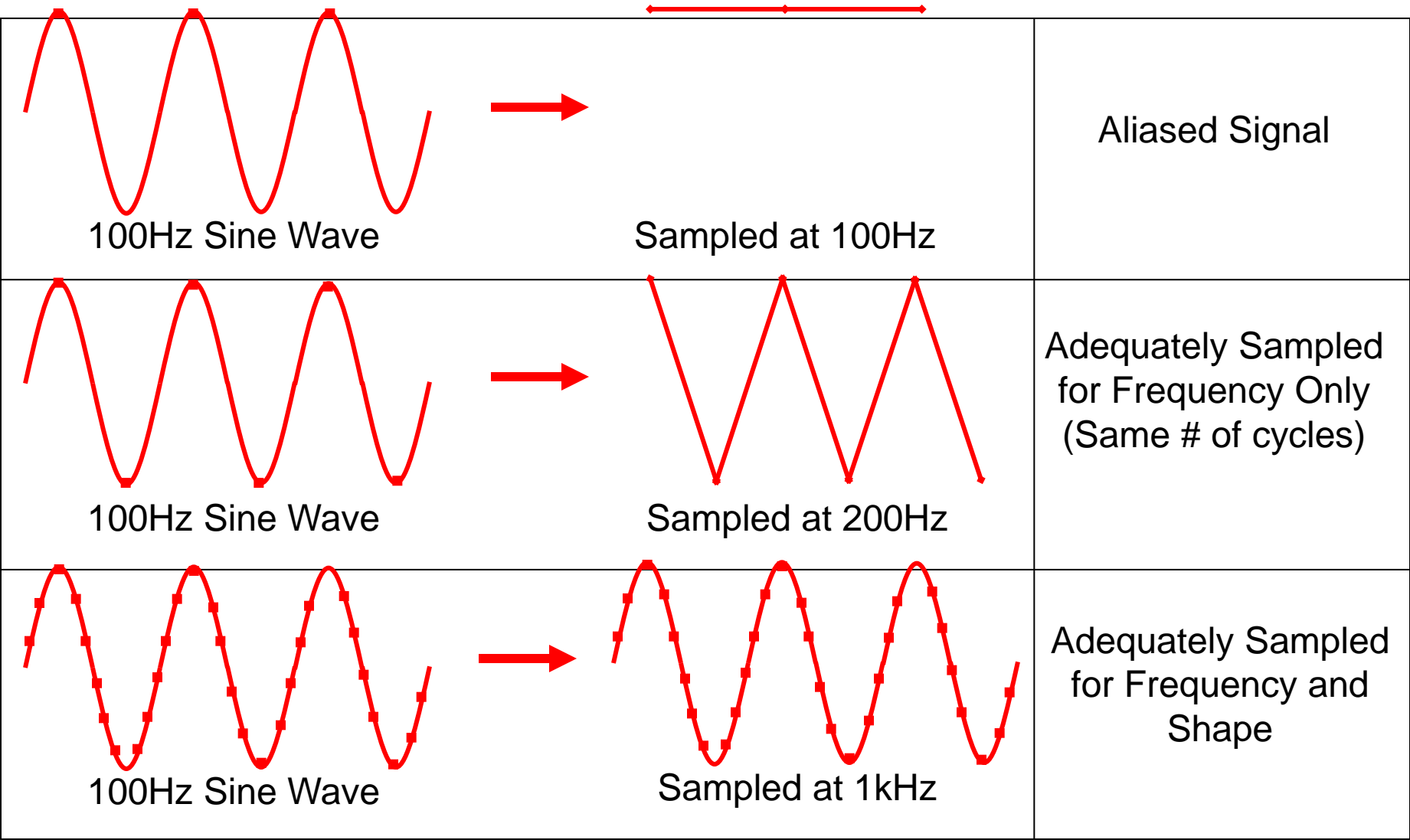
Aliasing

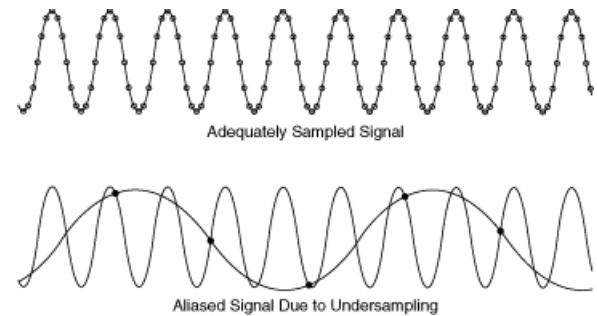


Sampling & Nyquist's Theorem

- Nyquist's Theorem
 - You must sample at greater than 2 times the maximum frequency component of your signal to accurately represent the frequency of your signal
- NOTE: You must sample between 5 - 10 times greater than the maximum frequency component of your signal to accurately represent the shape of your signal

Sampling Example

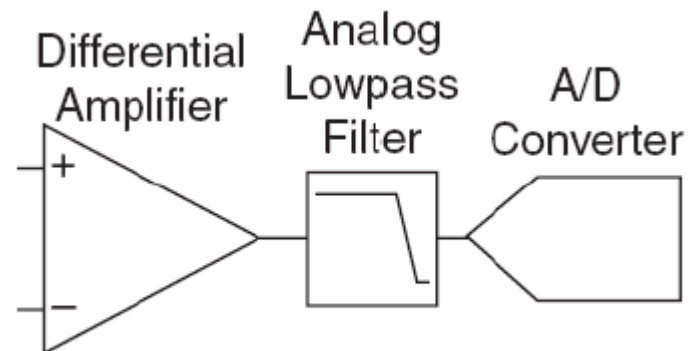
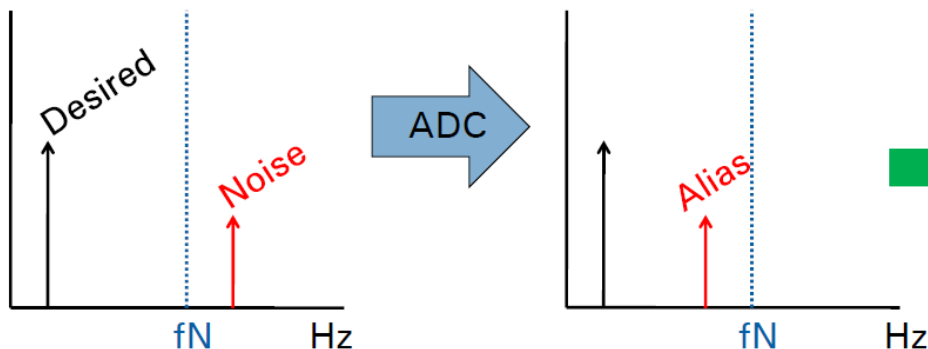




Hardware Filtering

- Filtering
 - To remove unwanted signals from the signal that you are trying to measure
- Anti-aliasing low-pass filtering (before the A/D converter)
 - To remove all signal frequencies that are higher than the input bandwidth of the device. If the signals were not removed, they would erroneously appear as signals within the input bandwidth of the device (known as aliasing)

Frequency Domain:



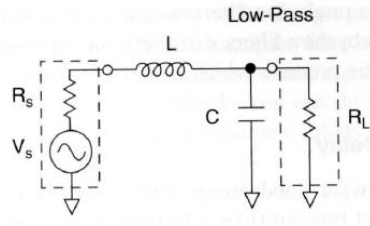
Analog filters

- Filter types: **LP**, HP, BP, BS, Notch

- Passive filters:

- RC, LCR

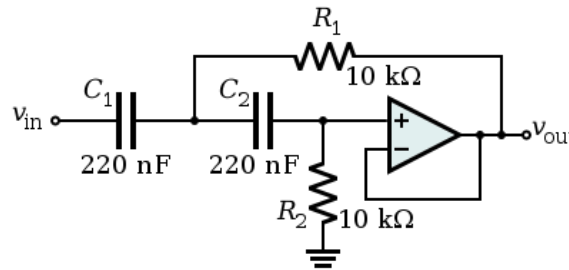
- (often inductors L are avoided, but they are needed for high Q-factor)



- Active filters

- opamp + R and C

- (can avoid L)



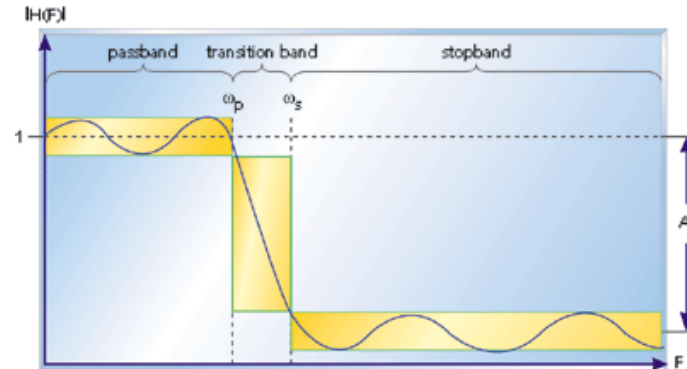
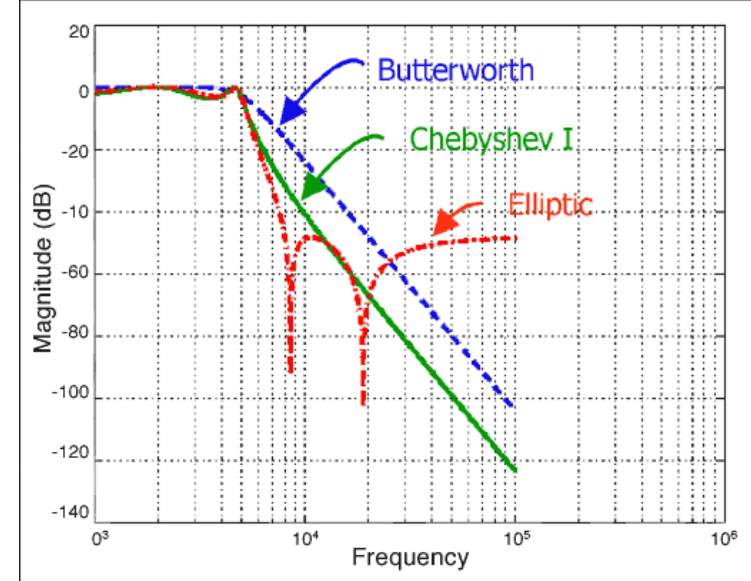
- Some common filter characteristics

- Butterworth

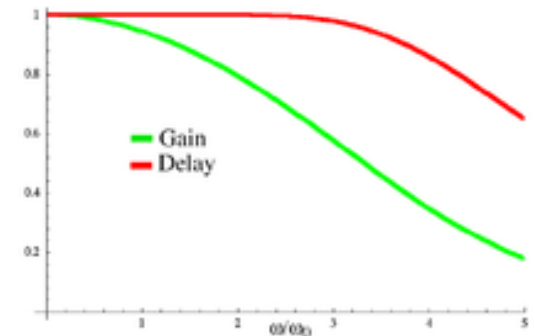
- Chebyshev

- Bessel (constant group delay in pass band)

- Elliptic

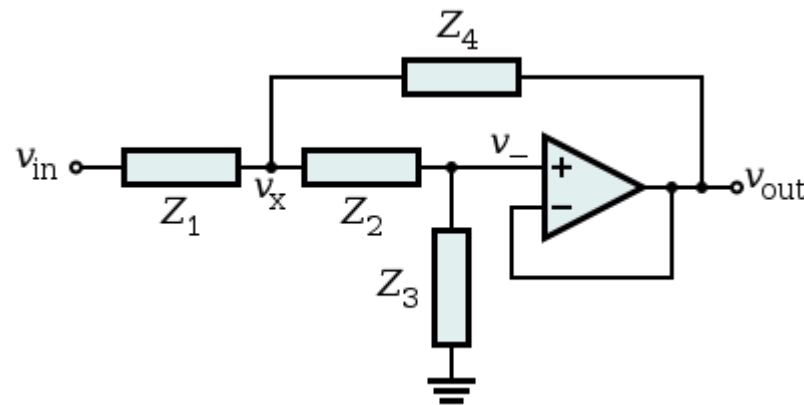


Bessel

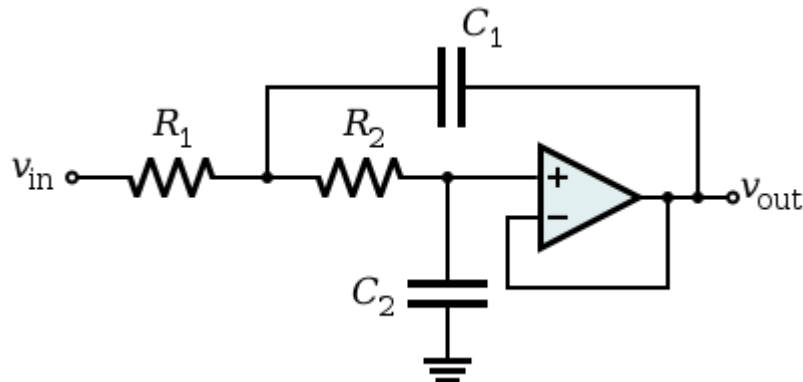


Sallen-Key - Active analog filter

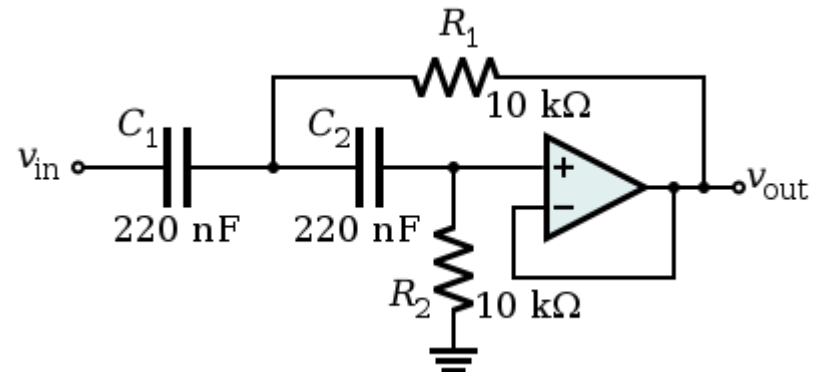
- Free program: FilterPro from Texas instruments
 - <http://focus.ti.com/docs/toolsw/folders/print/filterpro.html>



LP

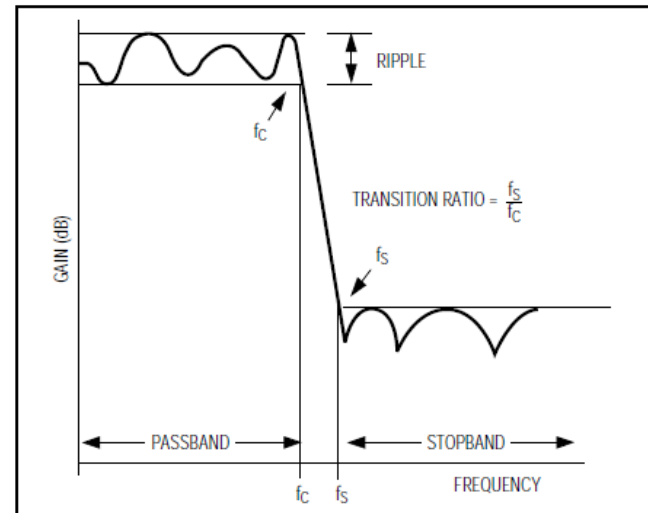
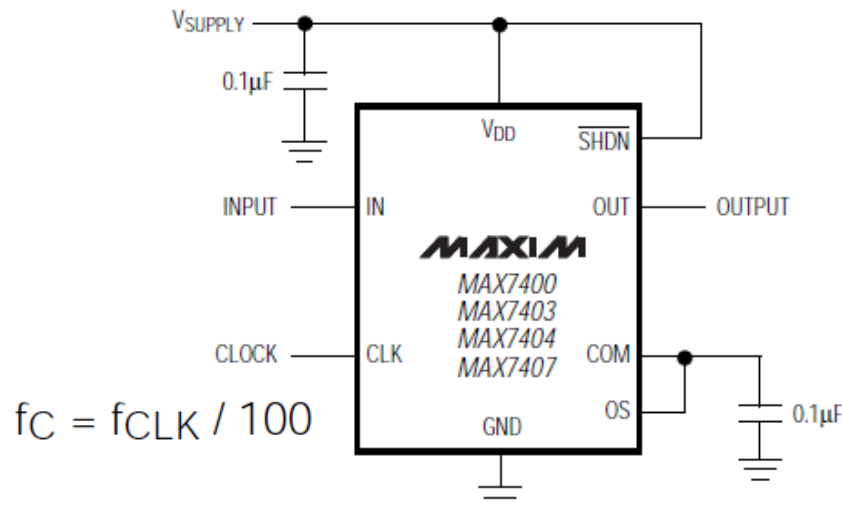


HP



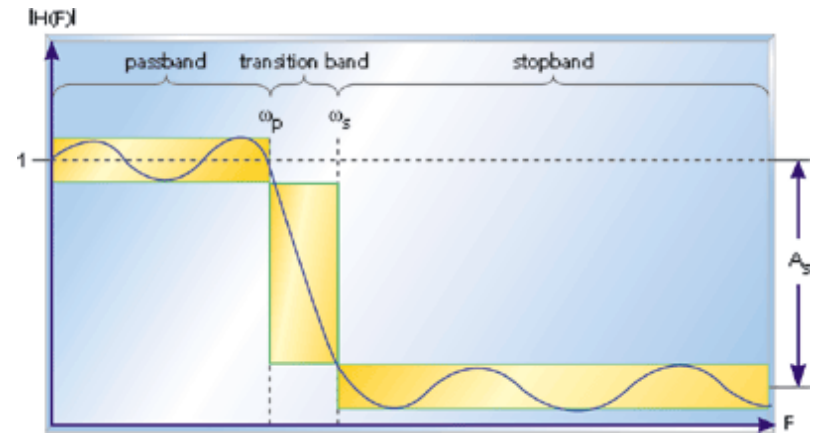
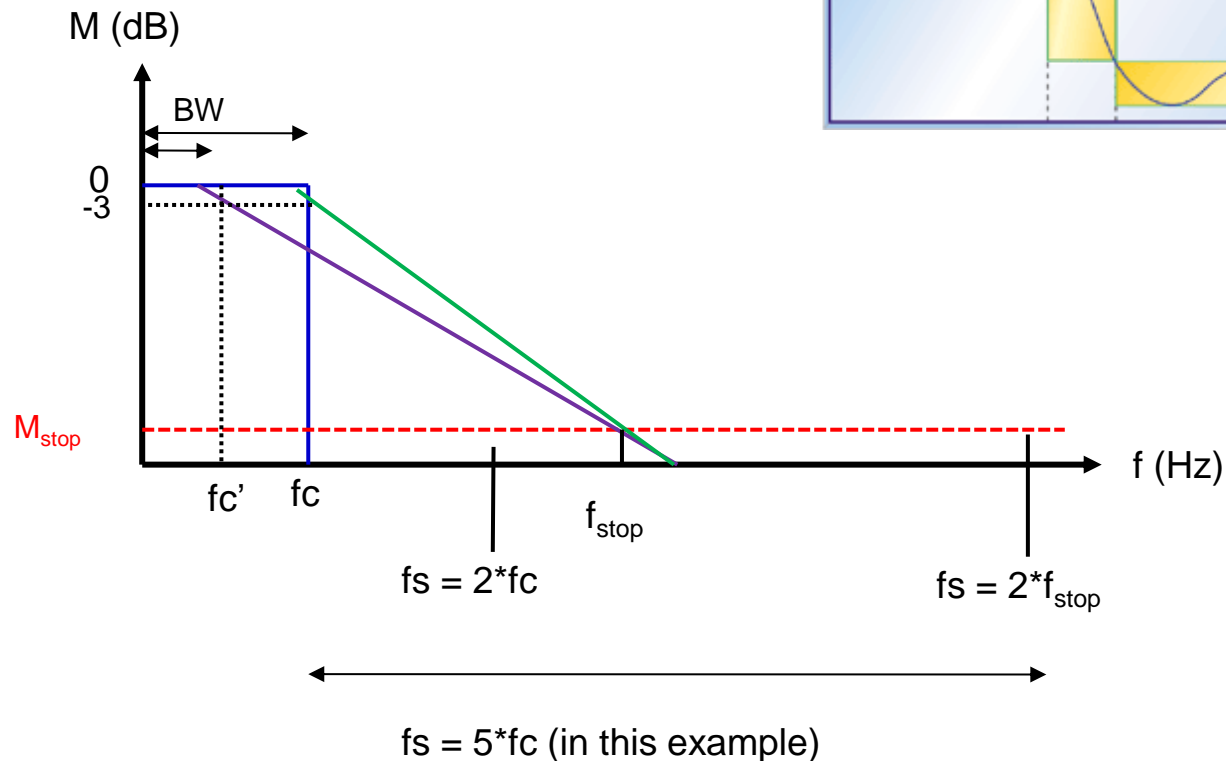
Switched-Capacitor Filter

- Can be suitable as an ADC anti-aliasing filter if you build your own electronics
- Be aware of possible clock noise (add RC-filters before and after)
- The corner frequency (cut-off) f_c is “programmable” using an external clock
- Example:
 - MAX7400 8th-order, lowpass, elliptic filter
 - MAX7400 has a transition ratio (f_s/f_c) of 1.5 and a typical stop band rejection of 82dB



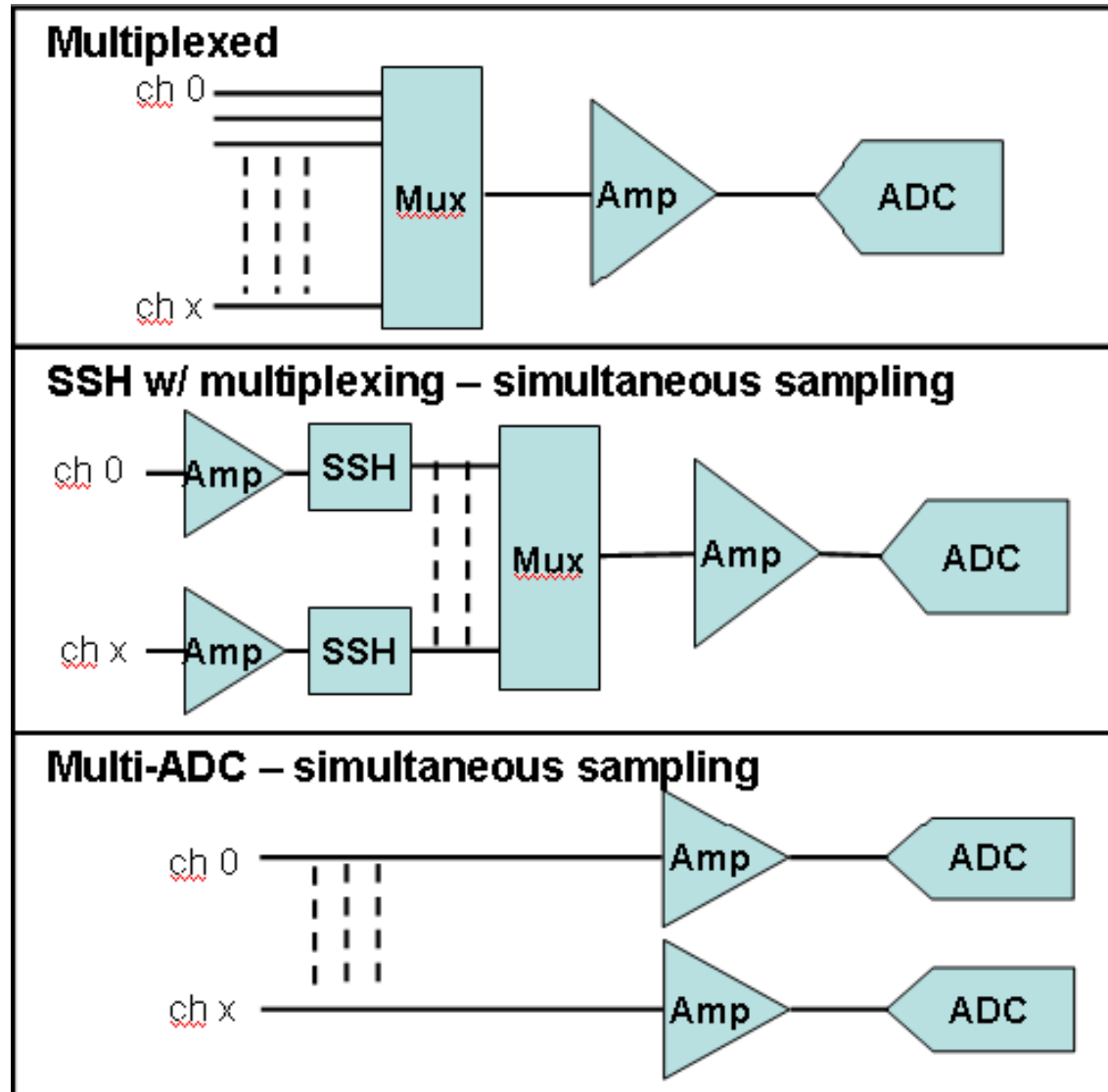
Importance of LP-filter selection for DAQ bandwidth

- f_c = cut-off frequency
- f_s = sampling frequency
- BW = bandwidth



ADC architectures

- Multiplexed
- Simultaneous sampling



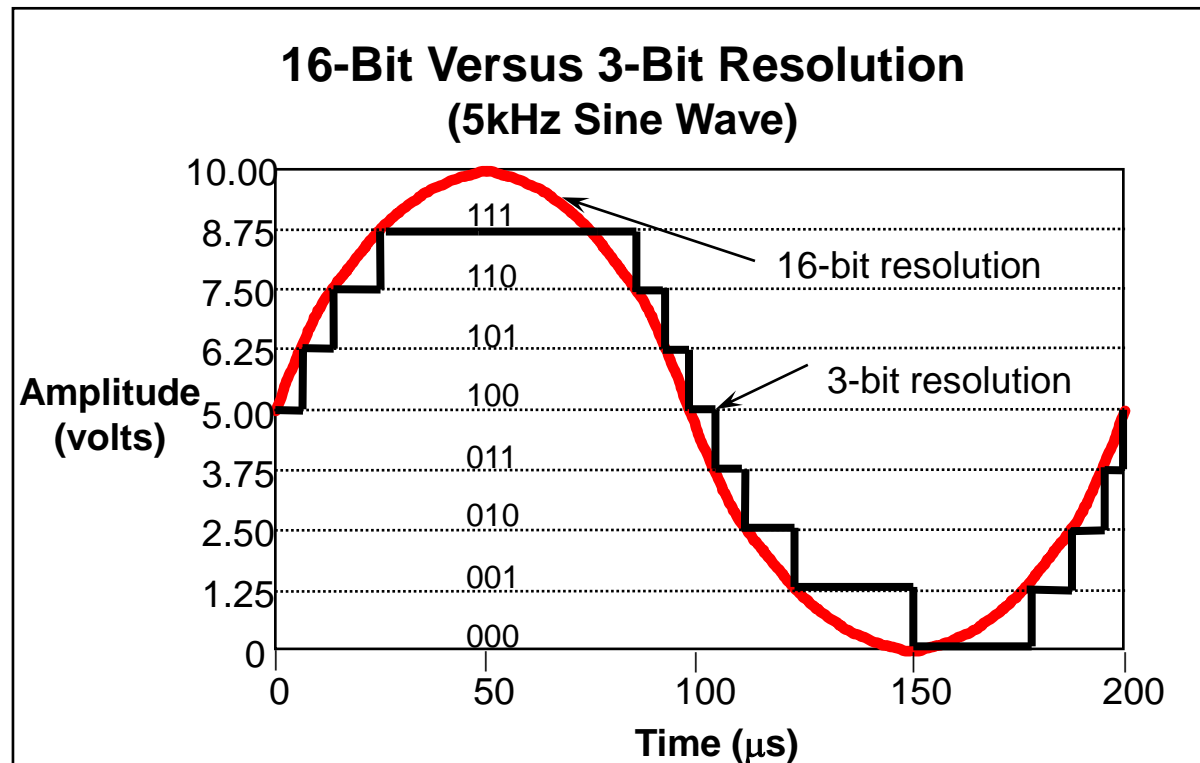
ADC resolution

- The number of bits used to represent an analog signal determines the resolution of the ADC
- Larger resolution = more precise representation of your signal
- The resolution determine the smallest detectable change in the input signal, referred to as code width or LSB (least significant bit)

$$\text{code width} = \frac{\text{device range}}{2^{\text{resolution}}}$$

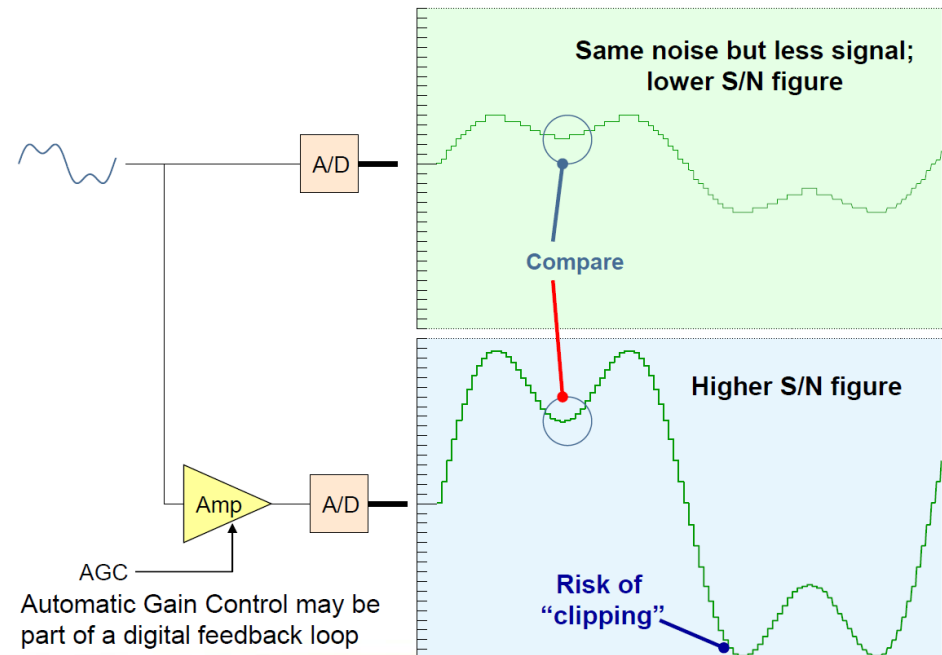
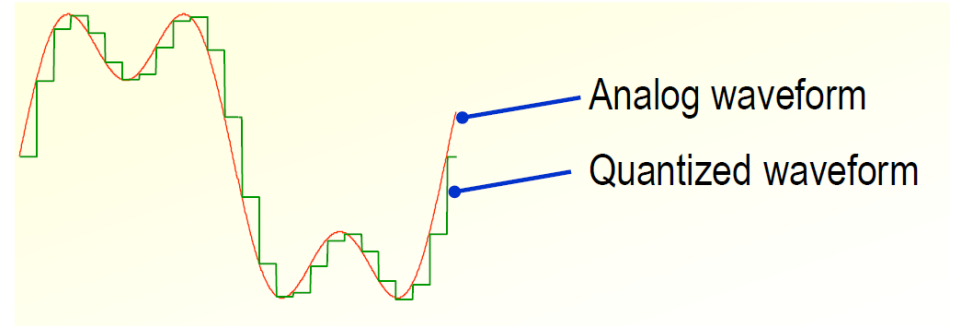
Example:

$$\frac{\text{device range}}{2^{\text{resolution}}} = \frac{10}{2^{16}} = .15 \text{ mV}$$

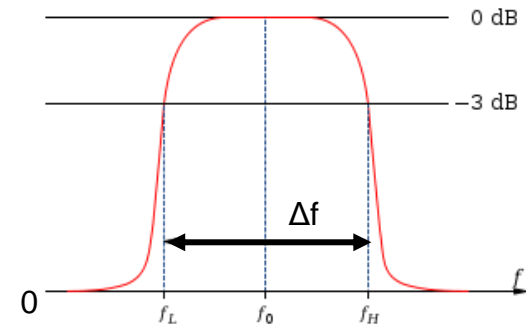


Digital signals: Bits, dynamic range, and SNR

- SNR = signal to noise ratio
- The number of bits used determines the maximum possible signal-to-noise ratio
- Using the entire ADC range (using an amplifier) increases the SNR
- The minimum possible noise level is the error caused by the quantization of the signal, referred to as **quantization noise**.



ADC oversampling



- The SNR of an ideal N-bit ADC (due to quantization effects) is:

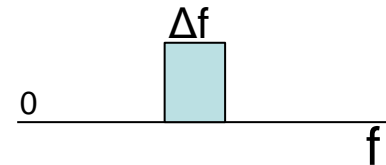
$$\text{SNR(dB)} = 6.02 \cdot N + 1.76$$

- If the sampling rate is increased, we get the following SNR:

- $\text{SNR(dB)} = 6.02 \cdot N + 1.76 + 10 \cdot \log_{10}(\text{OSR})$

- $\text{OSR} = f_s / f_{\text{nyquist}}$


- Nyquist sampling theorem: $f_s \geq 2 \cdot \Delta f_{\text{signal}}$
- Oversampling makes it possible to use a simple RC anti-aliasing filter before the ADC
- After A/D conversion, perform digital low-pass filtering and then down sampling to f_{nyquist}
- Effective resolution with oversampling = $N + 1/2 \cdot \log_2 (f_s / f_{\text{nyquist}})$, where N is the resolution of an ideal N-bit ADC at the Nyquist rate
 - If $\text{OSR} = f_s / f_{\text{nyquist}} = 1024$, an 8-bit ADC gets an effective resolution equal to that of a 13-bit ADC at the Nyquist rate (which is $2 \cdot \Delta f_{\text{signal}}$)



ADC range

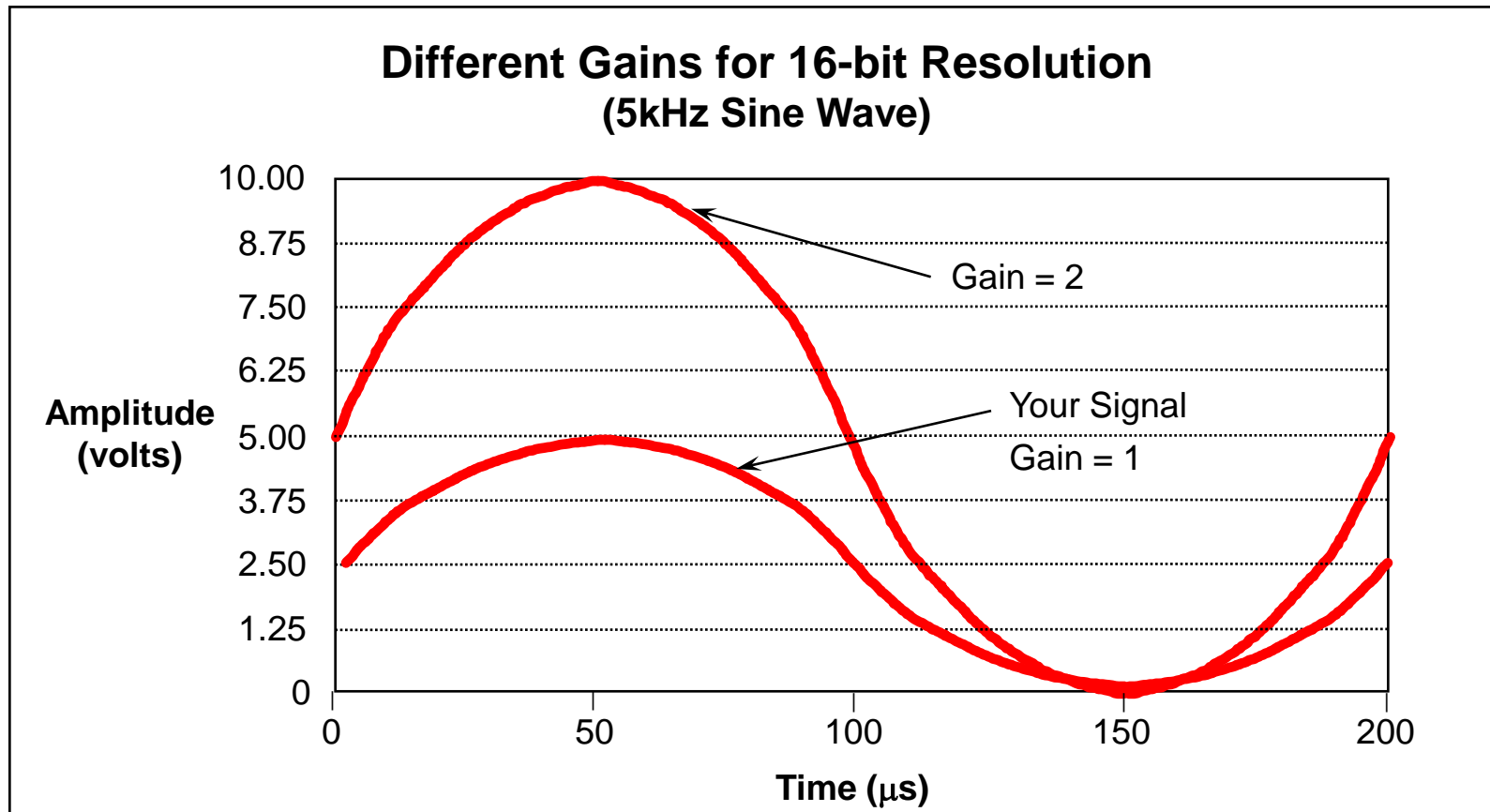
- Range refers to the minimum and maximum analog signal levels that the ADC can digitize (± 5 or ± 10 typical for many DAQ-cards)
- Pick a range that your signal fits in
 - Smaller range gives a more precise representation of your signal, given that the signal is not clipped (saturated)
 - If your signal is clipped you can in general not determine the absolute signal levels
- Bipolar signals are signals that range from a negative to a positive value (e.g. -5 to $+5$ V)
- Unipolar signals are signals that range from 0 value to a positive value (e.g. $0 - 5$ V)

Gain

- Gain setting amplifies the signal for best fit in ADC range
- Gain settings are 0.5, 1, 2, 5, 10, 20, 50, or 100 for most DAQ cards.
- You don't choose the gain directly in LabVIEW
 - **Choose the input limits of your signal**

 - Maximum gain possible is selected
 - Maximum gain possible depends on the limits of your signal and the range of your ADC
- Proper gain = more precise representation of your signal
 - Allows you to use all of your available resolution

Gain Example

- Input limits of the signal: 0 to 5 Volts
- Range setting for the ADC: 0 to 10 Volts
- Gain setting applied by instrumentation amplifier: 2

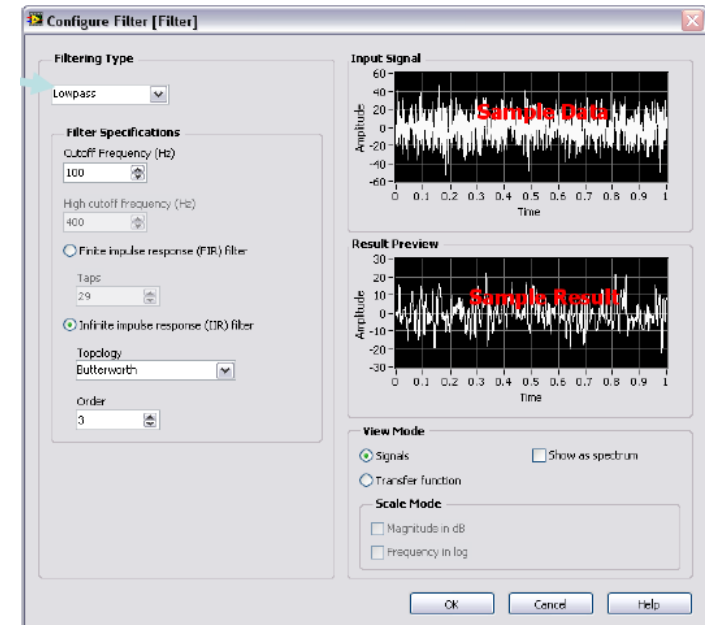
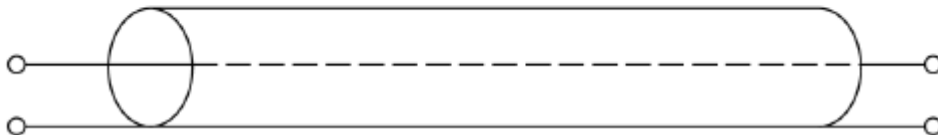


Other noise reduction techniques in DAQ systems

- Position noise sources (e.g. motors and power lines) away from data acquisition device, cable, and sensor if possible
- Place data acquisition device as close to sensor as possible to prevent noise from entering the system
- Twisted pairs, coax cable, shielding
- Software Filtering (e.g. averaging)



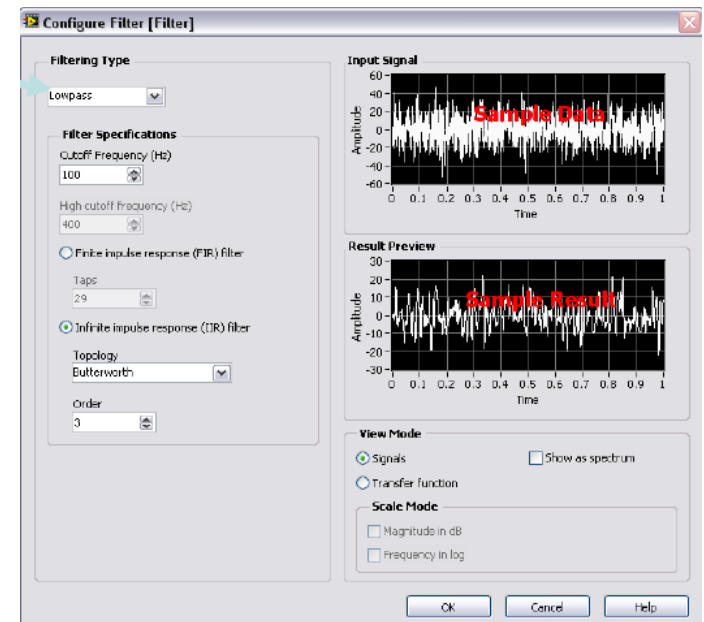
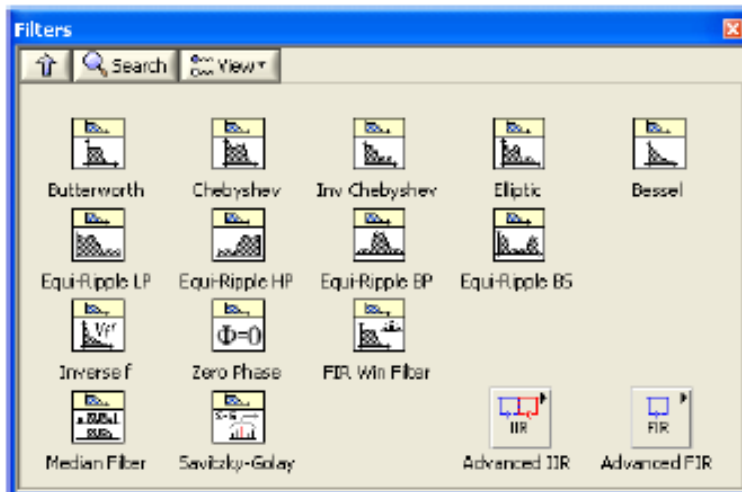
(a) Twisted-pair cable



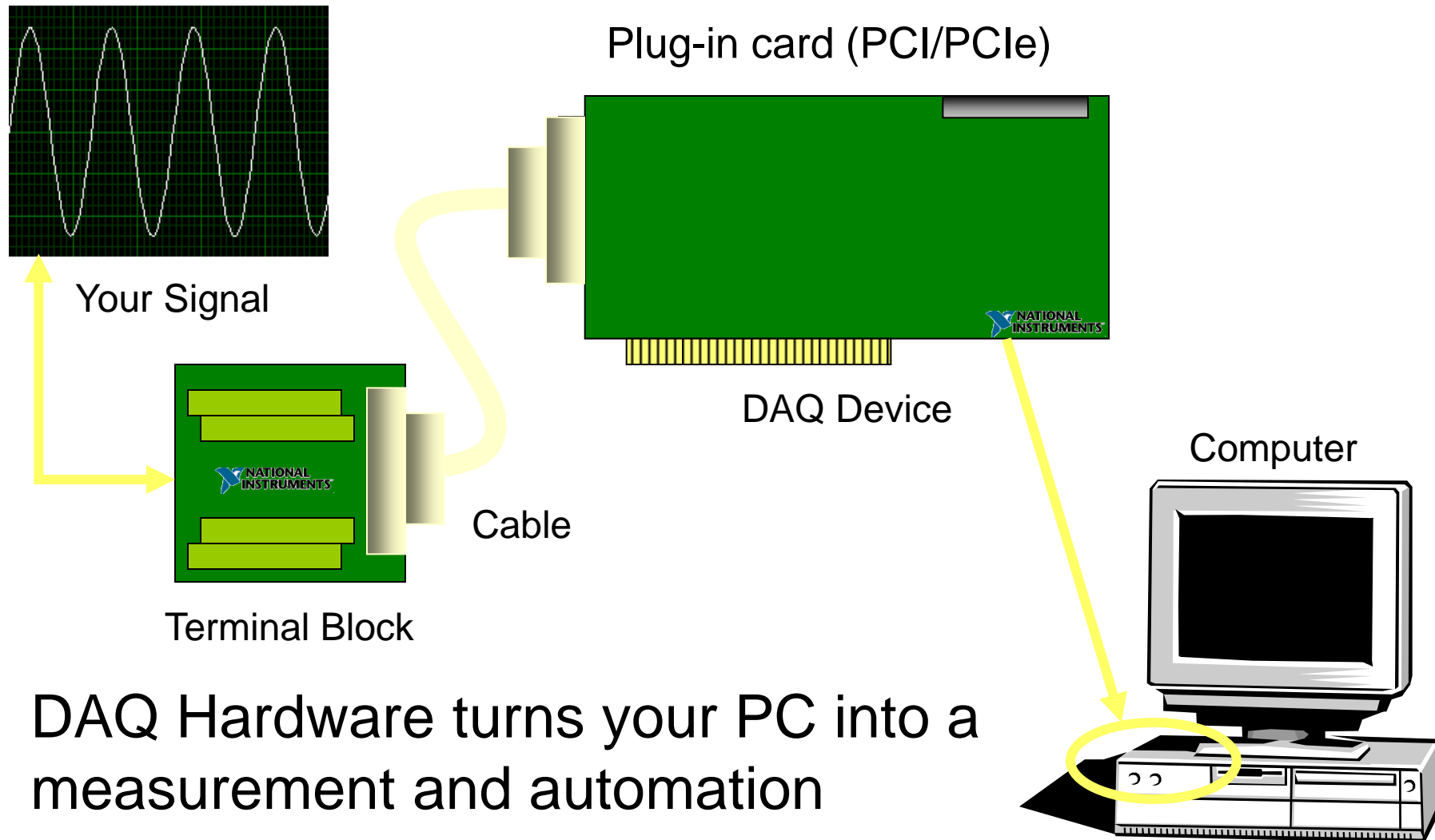
Software filtering

- Easy, flexible, predictable, inexpensive
- Unable to distinguish aliased signals from true ones
 - *Need a hardware anti-aliasing filter before the A/D conversion!*
- Modern PCs have plenty of CPU speed for software filtering

Lower-Level Filter VIs



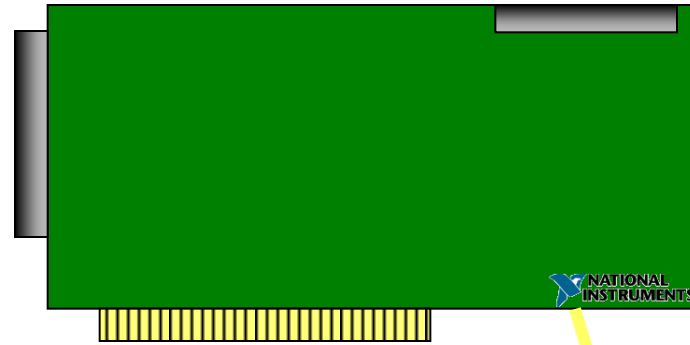
Data Acquisition Hardware



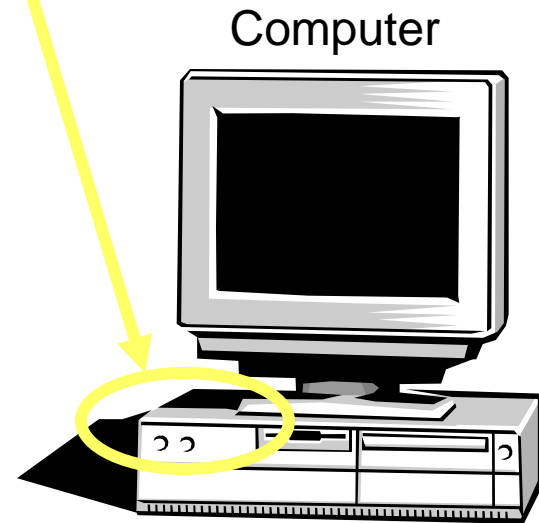
- DAQ Hardware turns your PC into a measurement and automation system

DAQ Device

- Most DAQ devices have:
 - Analog Input
 - Analog Output
 - Digital I/O
 - Counters
 - Frequency measurements
 - Angular measurements from angular encoders
- Connects to the bus of your computer
- Compatible with a variety of bus protocols
 - PCI, PXI, CompactPCI, PCIe, PXIe, PCMCIA, USB



DAQ Device



Computer

PXI

- PXI = PCI eXtensions for Instrumentation.
- PXI is a high-performance PC-based platform for measurement and automation systems.
- PXI was developed in 1997 and launched in 1998.
- Today, PXI is governed by the PXI Systems Alliance (PXISA), a group of more than 70 companies chartered to promote the PXI standard, ensure interoperability, and maintain the PXI specification.

PXI

- PXI systems are composed of three basic components:
 - Chassis
 - Controller
 - Peripheral modules



PXI chassis

- The PXI chassis contains the backplane for the plug-in DAQ cards
- The chassis provides power, cooling, and communication buses for the PXI controller and modules.
- Chassis are available both with PCI and PCI Express
- 4 – 18 slots chassis are common



PXI controllers

- PXI Embedded Controller
- Laptop Control of PXI
 - Using e.g. ExpressCard serial bus
- Desktop PC Control of PXI

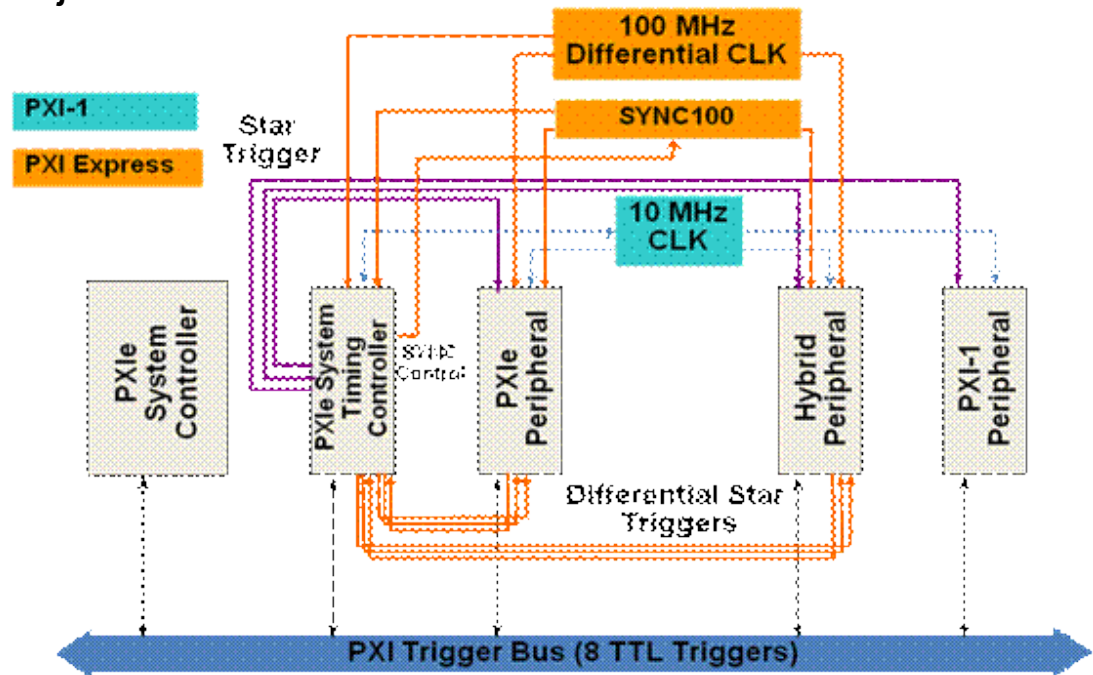


PXI-based DAQ systems

- The benefits of PXI-based data acquisition systems include rugged packaging that can withstand the harsh conditions that often exist in industrial applications.
- PXI systems also offer a modular architecture, which means that you can fit several devices in the same space as a single stand-alone instrument, and you have the ability to expand your system far beyond the capacity of a desktop computer with a PCI bus.

PXI triggering and timing

- One of the key advantages of a PXI system is the integrated timing and synchronization.
- The PXI chassis includes reference clocks, triggering buses and slot-to-slot local bus.
 - Any module in the system can set a trigger that can be seen from any other module.
 - The local bus provides a means to establish dedicated communication between adjacent modules.



Device Drivers

- In computing, a device driver or software driver is a computer program allowing higher-level computer programs to interact with a hardware device.
- A driver typically communicates with the device through the computer bus or communications subsystem to which the hardware connects. When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke routines in the original calling program. Drivers are hardware-dependent and operating-system-specific.

NI Instrument Drivers - IDNET

- All NI hardware is shipped with LabVIEW driver software
- Driver upgrade to the latest version available at www.ni.com
- Many third-party vendors also ship LabVIEW drivers with their instruments

ni.com/idnet

Instrument Driver Network

Instrument Driver Network makes available more than 8,000 drivers for instruments from over 275 third-party vendors. Drivers are available for use in LabVIEW, MATLAB/Simulink, and Measurement Studio for Visual Studio.

Looking for NI drivers such as NI-DAQmx, NI-488.2, NI-IMAQ, etc.? Hardware drivers, firmware updates, and application software (e.g., LabVIEW, MATLAB/Simulink, and Measurement Studio) are available in NI Drivers and Updates. [Visit NI Drivers and Updates for more options](#)

Find an Instrument Driver

[Browse All Drivers](#) [Submit New Driver](#) [Request New Driver](#) [Request Support](#)

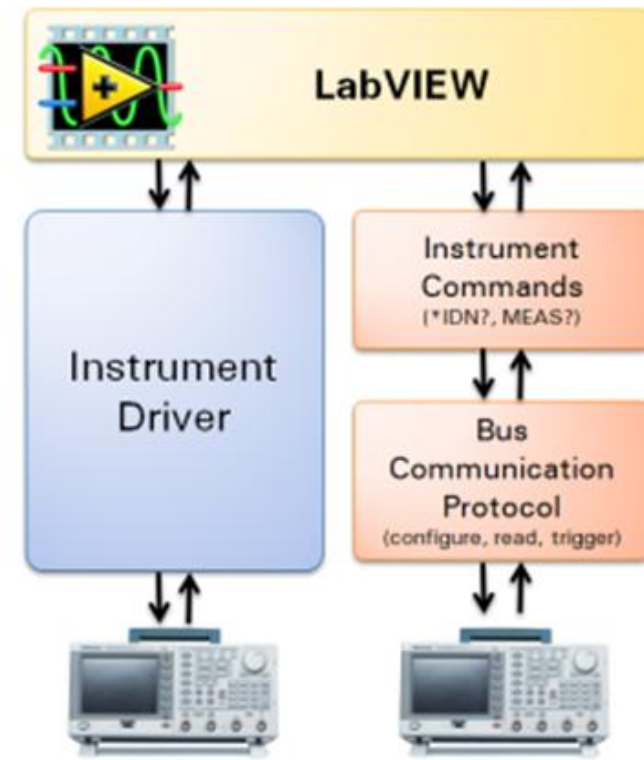
Browse All Drivers

Search within results

Manufacturer	Interface	Development Environment
A	IEEE 488.2 (GPIB)	LabVIEW
B-C	USB	LabVIEW Real-Time
D-E	Ethernet	LabWindows/CVI
F-H	Serial	Measurement Studio for Visual Studio
I-K	LXI	
L-M	PXI/CompactPCI	
N-O	PCI	Application Area
P-Q	VXI/VME	Automation
R-S	Other	RF
T-V		Test and Measurement
W-Z		
	Technology	

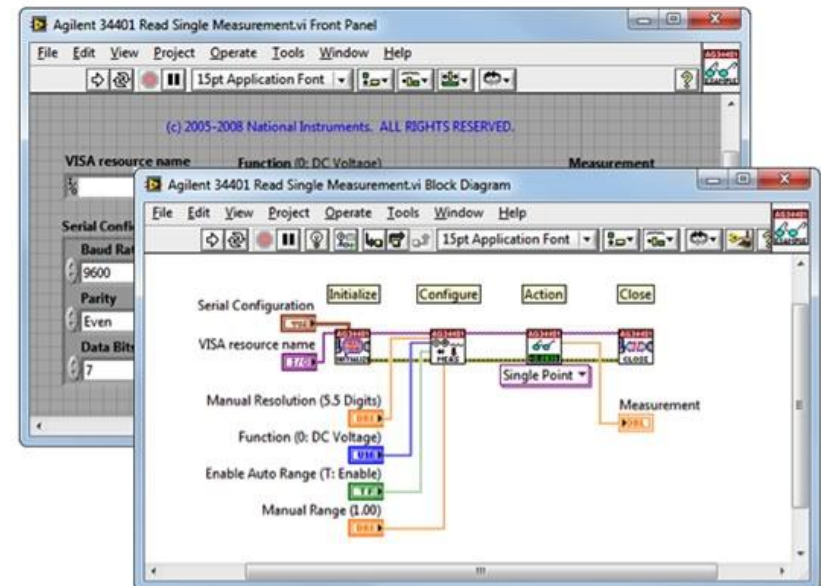
Connect to any instrument using LabVIEW instrument drivers

- Every programmable test and measurement instrument has a set of commands that it understands. Typically, a programmer's manual that comes with the instrument documents these commands, and it is up to you to find the commands you need
- Instrument drivers simplify this process by **abstracting the low-level commands** for each instrument and providing a familiar API for all instruments. By using an instrument driver, you can focus on the application you are developing



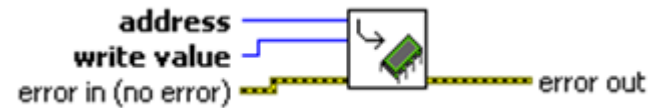
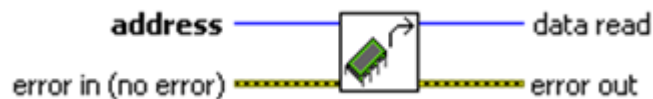
Using device drivers in LabVIEW

- The flow of an application typically starts with opening a connection to the hardware, configuring hardware settings, reading and writing measured data to and from the hardware, and finally closing the connection to the hardware.
- Since most drivers follow this framework, learning a new driver is relatively easy
- To “modes” to choose:
 - High-level, easy-to-understand operations (Plug and Play)
 - Lower-level operations required to use more advanced features



“In Port” and “Out Port” in LabVIEW

- Use these VIs only for 16-bit I/O addresses
- **These VIs are not available in Windows Vista or Windows 7, because it allows read/write access to any I/O port on the system, which is discouraged for security reasons.**
- **Solution: Use VISA ...**

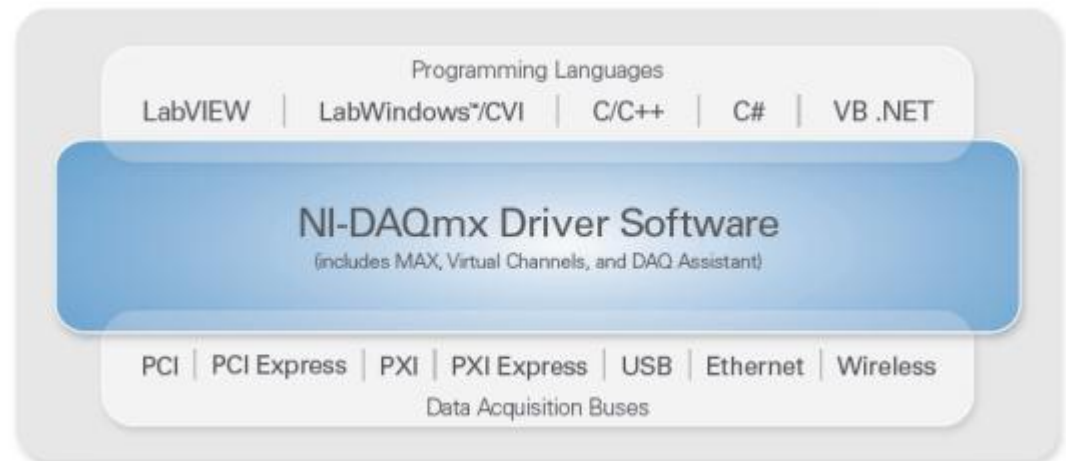


VISA

- VISA = Virtual Instrument Software Architecture.
- NI-VISA is the NI implementation of the VISA standard.
- LabVIEW instrument drivers are based on the VISA standard, which makes them bus- and platform-independent.
- Supports communication with instruments via:
 - GPIB
 - Serial
 - Ethernet
 - USB
 - PXI

NI-DAQmx

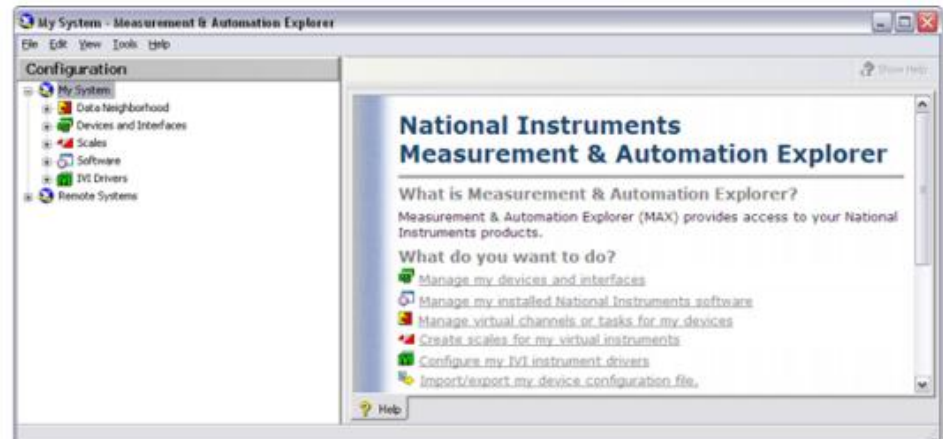
- NI-DAQmx (multithreaded driver) software provides ease of use, flexibility, and performance in multiple programming environments
- Driver level software
 - DLL that makes direct calls to your DAQ device
- Supports the following software:
 - NI LabVIEW
 - NI LabWindows CVI
 - C/C++
 - C#
 - Visual Basic .NET.





NI Measurement & Automation Explorer (MAX)

- All NI-DAQmx devices include MAX, a configuration and test utility
- You can use MAX to
 - Configure and test NI-DAQmx hardware with interactive test panels
 - Perform self-test sequences
 - Create simulated devices
 - Reference wiring diagrams and documentation
 - Save, import, and export configuration files
 - Create NI-DAQmx virtual channels that can be referenced in any programming language



MAX Example

The screenshot displays the NI USB-9215A (BNC) "Dev1" - Measurement & Automation Explorer interface. The left pane shows the system tree with "NI USB-9215A (BNC) 'Dev1'" selected under "Devices and Interfaces". The right pane shows the "Test Panels" dialog for "NI USB-9215A (BNC): 'Dev1'".

The "Test Panels" dialog has a "Name" field with "Serial Number" and a "Value" field with "0xE505AF". Below this, the "Analog Input" tab is active, showing the following settings:

Channel Name	Max Input Limit	Rate (Hz)
Dev1/ai0	10	1000

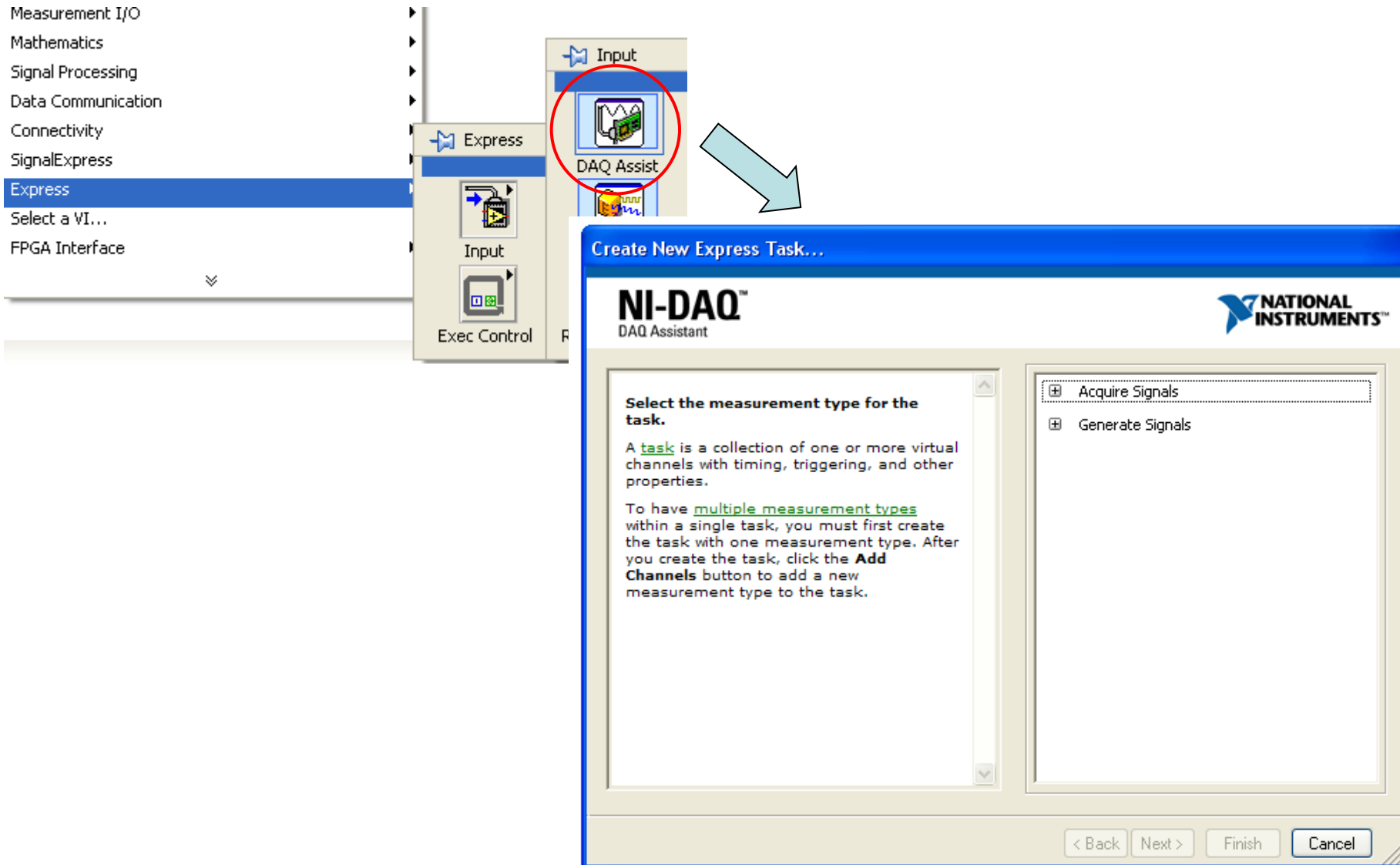
Mode	Min Input Limit	Samples To Read
On Demand	-10	1000

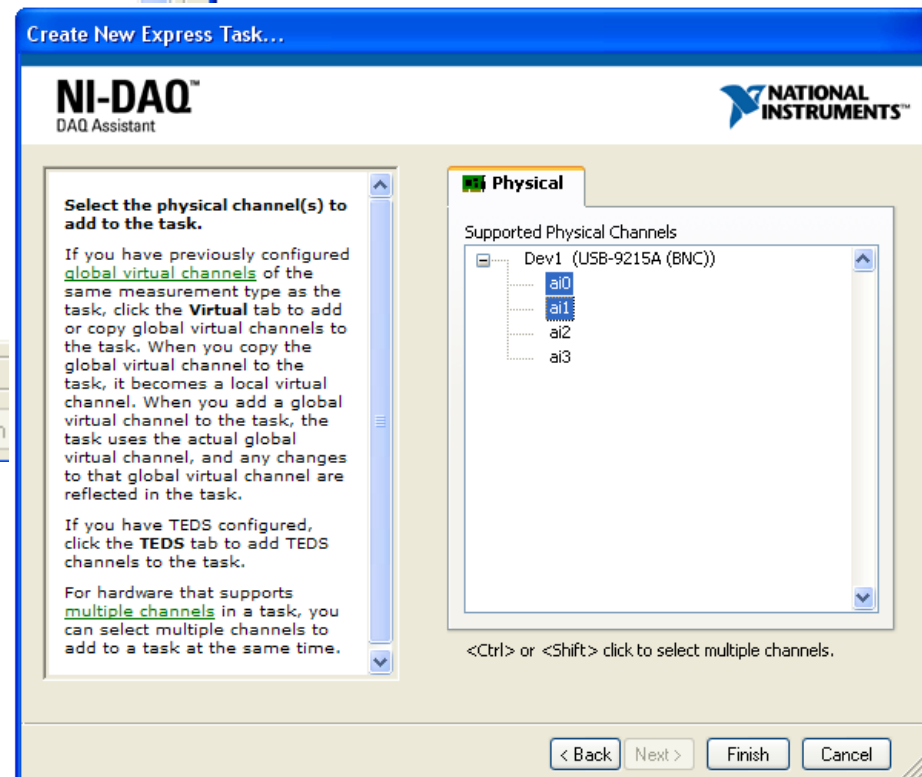
Input Configuration: Differential

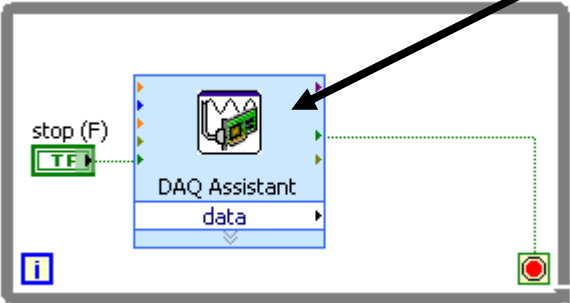
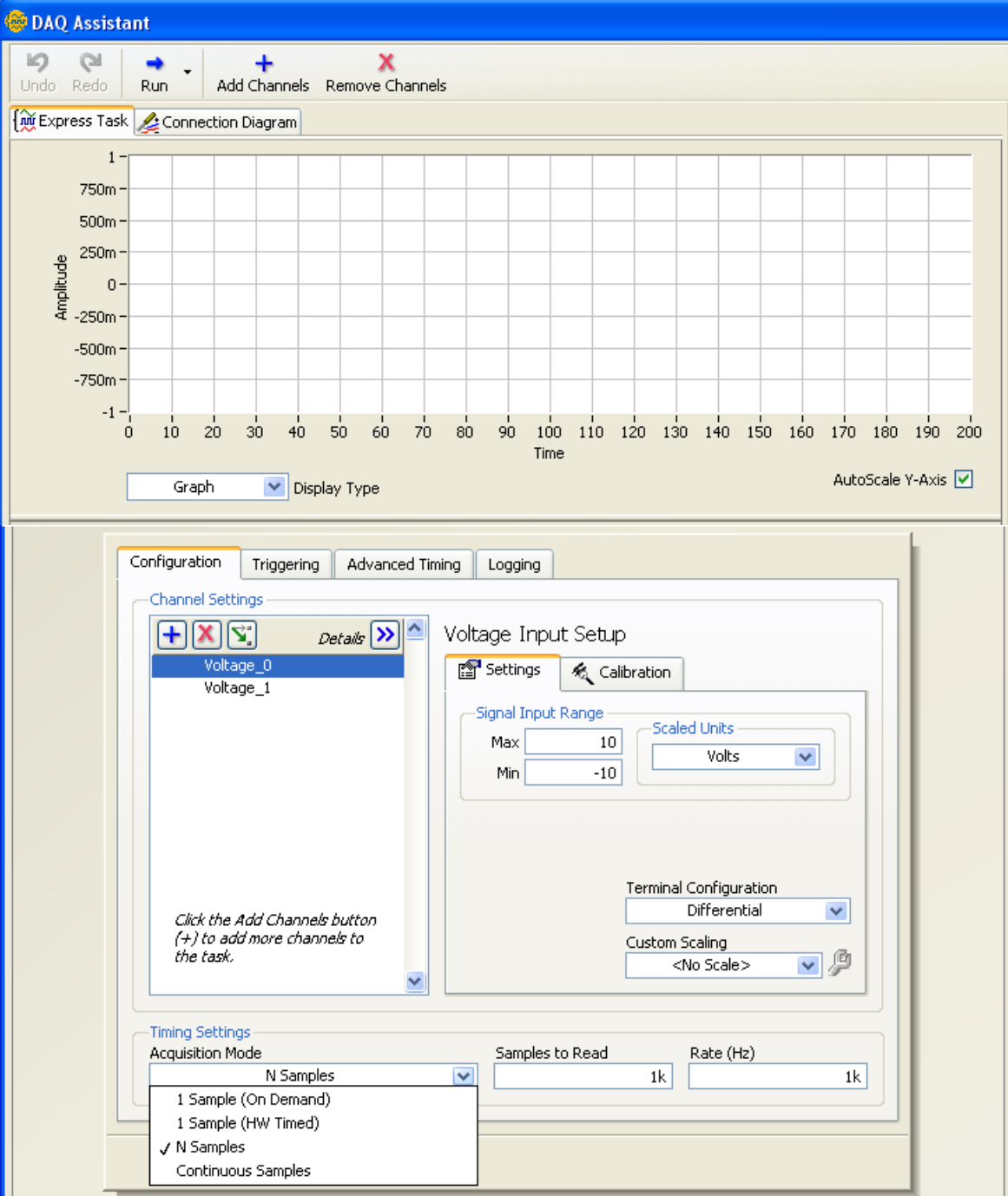
Amplitude vs. Samples Chart: Auto-scale chart ☒. The chart shows a black area with a y-axis from -10 to 10 and an x-axis from 0 to 99. The "Value" field shows 0.

Buttons: Start, Stop, Help, Close.

LabVIEW Express: DAQ assistant







LabVIEW - Sequential DAQ design

- Configure
- Acquire data
- Analyze data
- Visualize data
- Store data

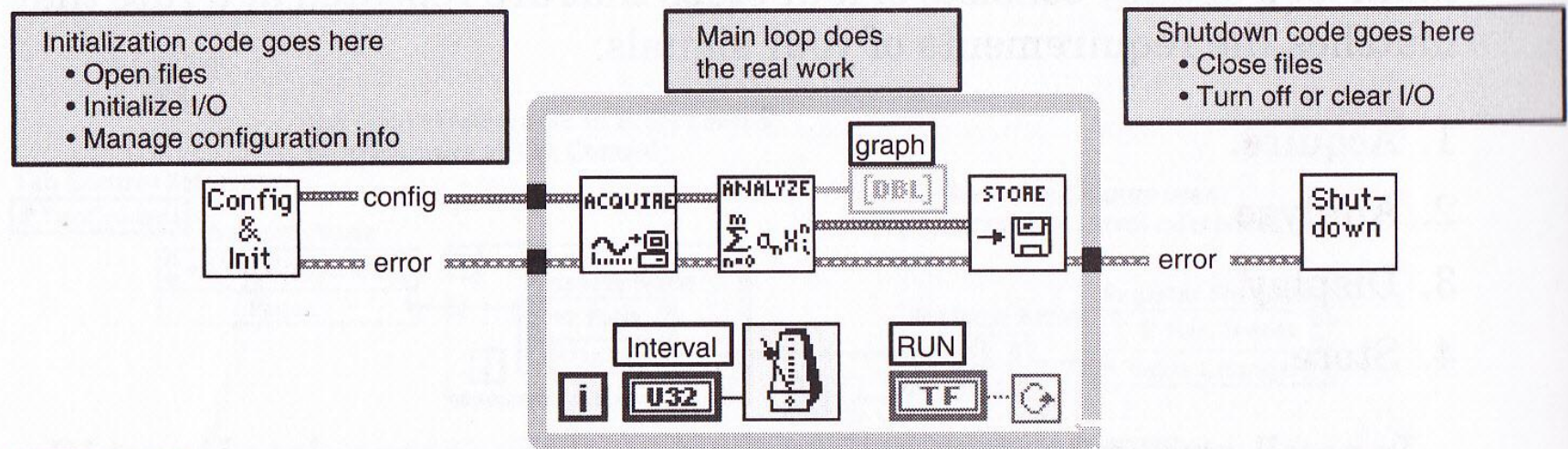
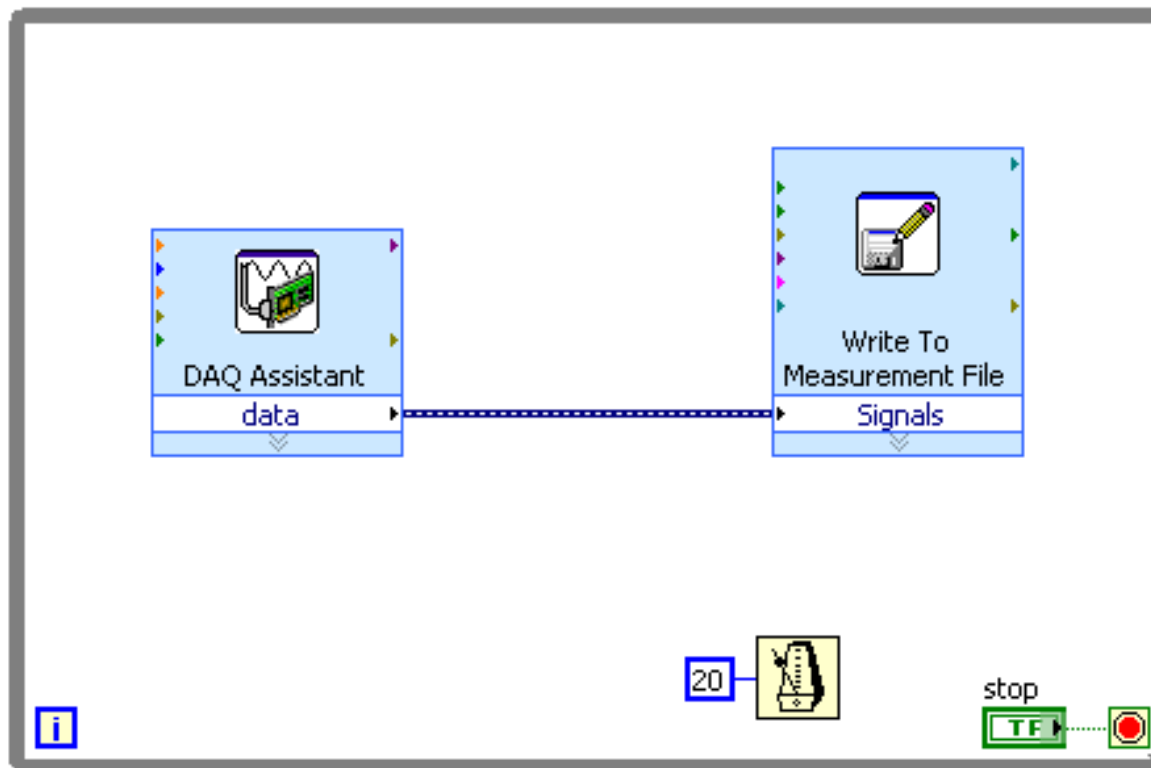


Figure 3.25 Dataflow instead of Sequence structure enhances readability. It has the same functionality as Figure 3.24. Note that the connections between tasks (subVIs) are not optional: they force the order of execution.

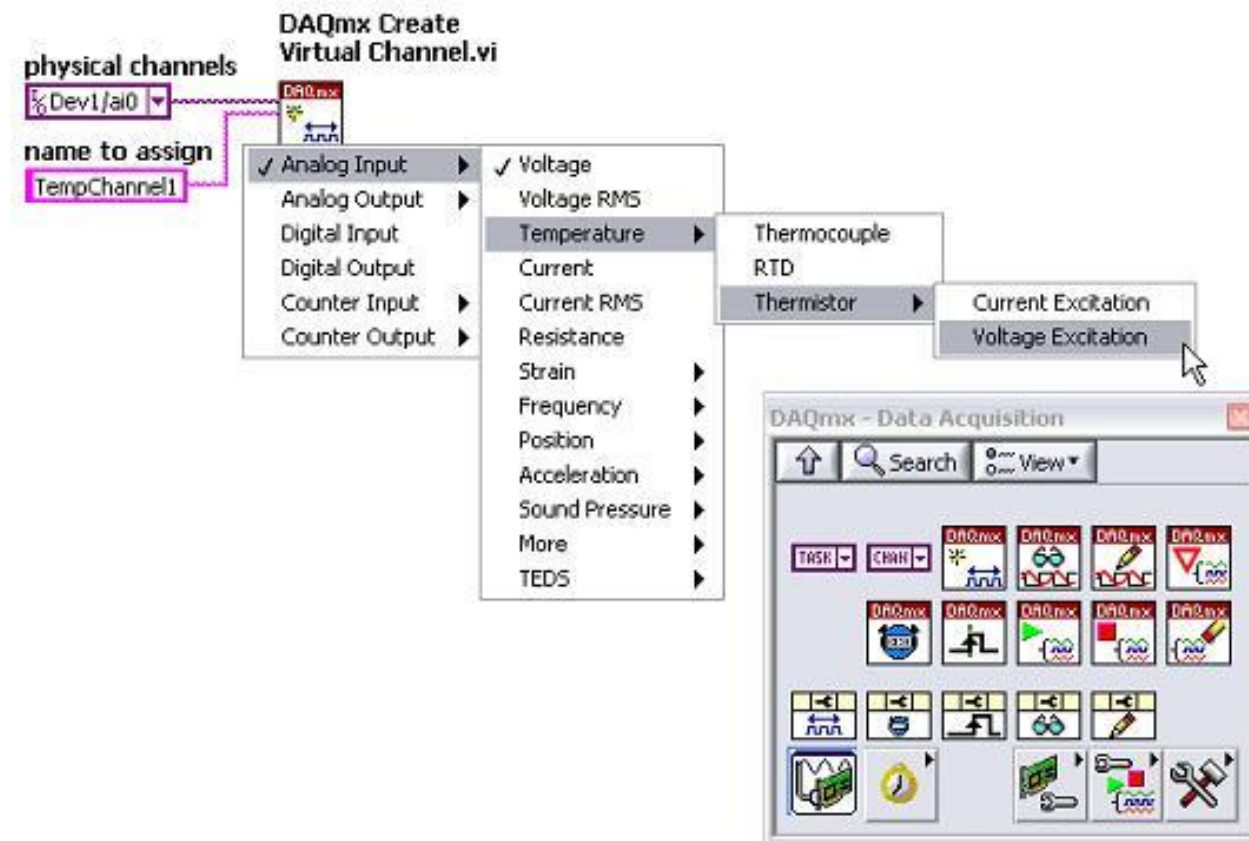
LabVIEW: Low-speed DAQ

- **DAQ assistant Express VI** used in the block diagram
- Data written to file using the **Write to Measurement File Express VI**

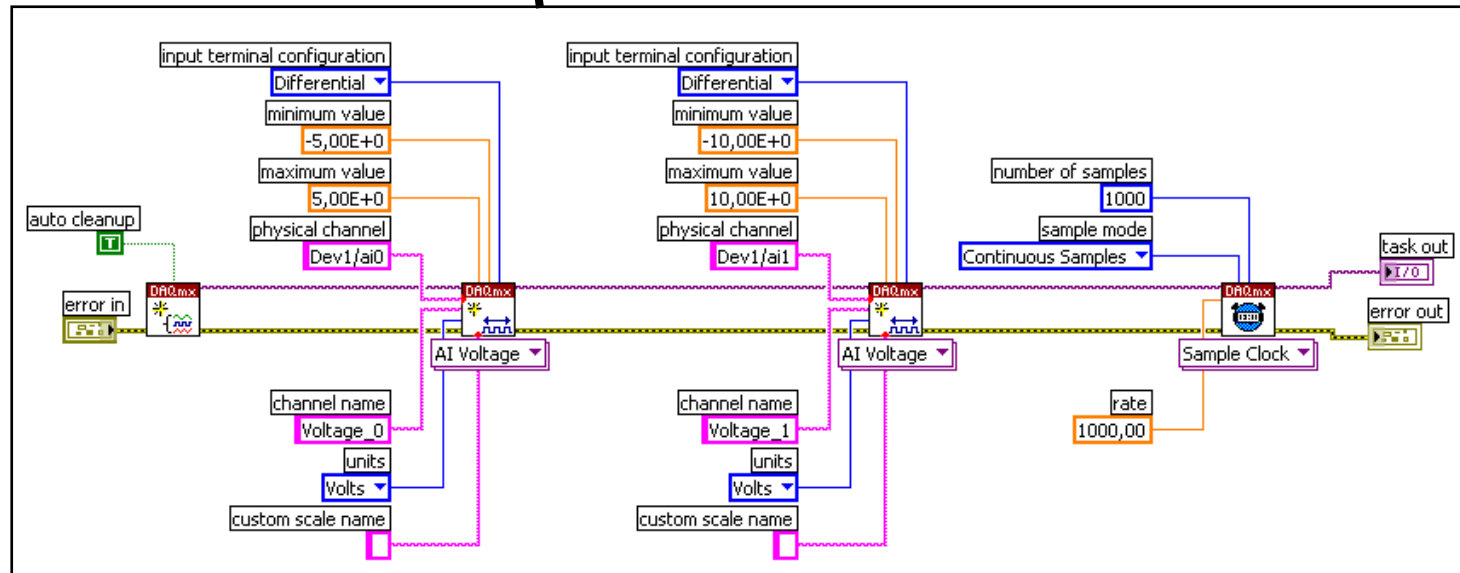
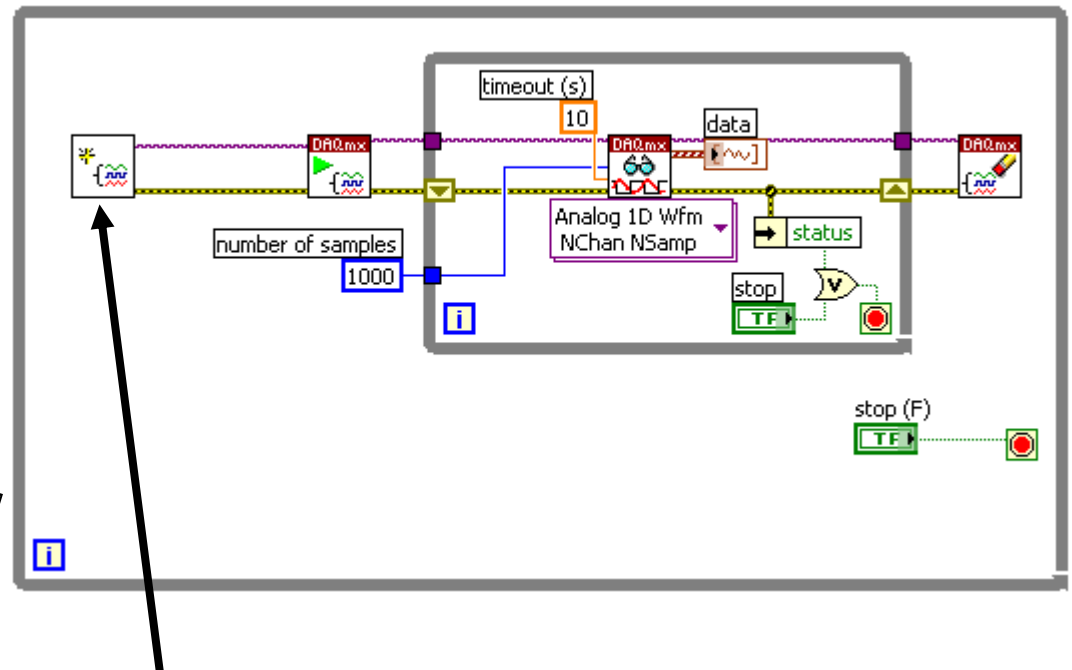
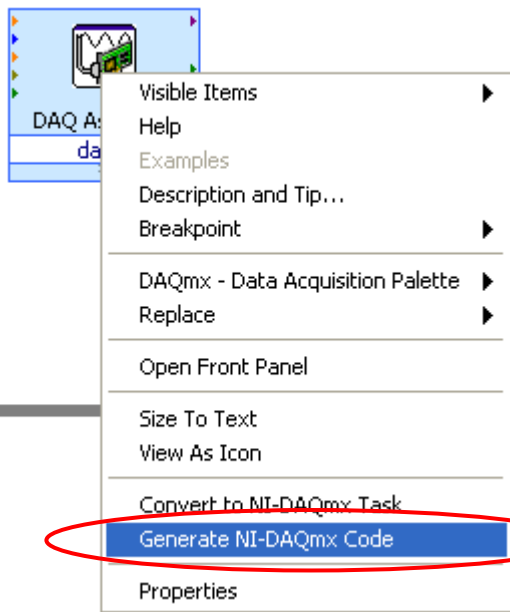


Using the NI-DAQmx API

- Create virtual channels programmatically
- Use graphical functions and structures to specify timing, triggering, and synchronization parameters

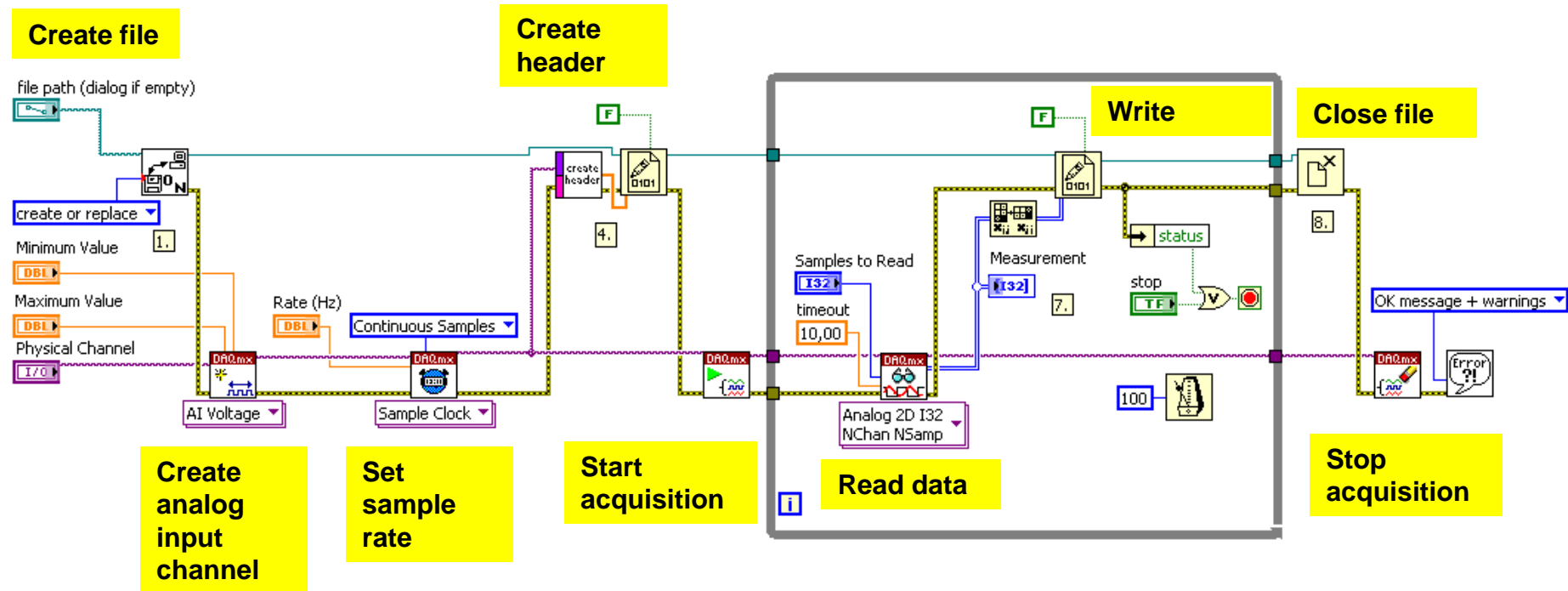


DAQ Assistant -> standard VIs



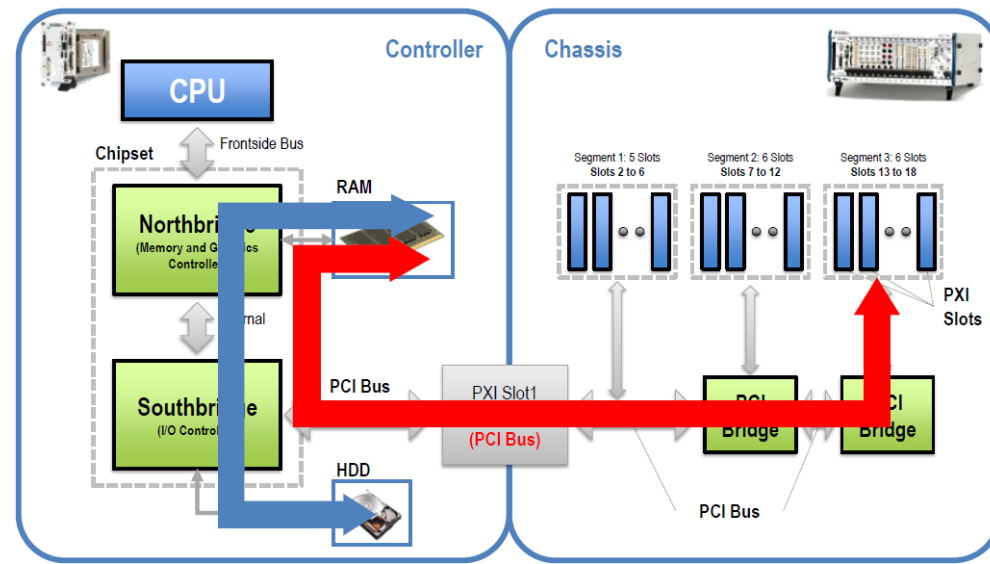
LabVIEW: Medium-speed DAQ

- Example: Cont Acq&Graph Voltage -To File (Binary).vi
- **Standard VIs used, and data written to a binary file**



Transferring Data from DAQ-card to System Memory

- The transfer of acquired data from the hardware to system memory follows these steps:
 - Acquired data is stored in the hardware's first-in first-out (FIFO) buffer.
 - Data is transferred from the FIFO buffer to system memory using interrupts or DMA.
 - The samples are transferred to system memory via the system bus (PCI/PCIe).

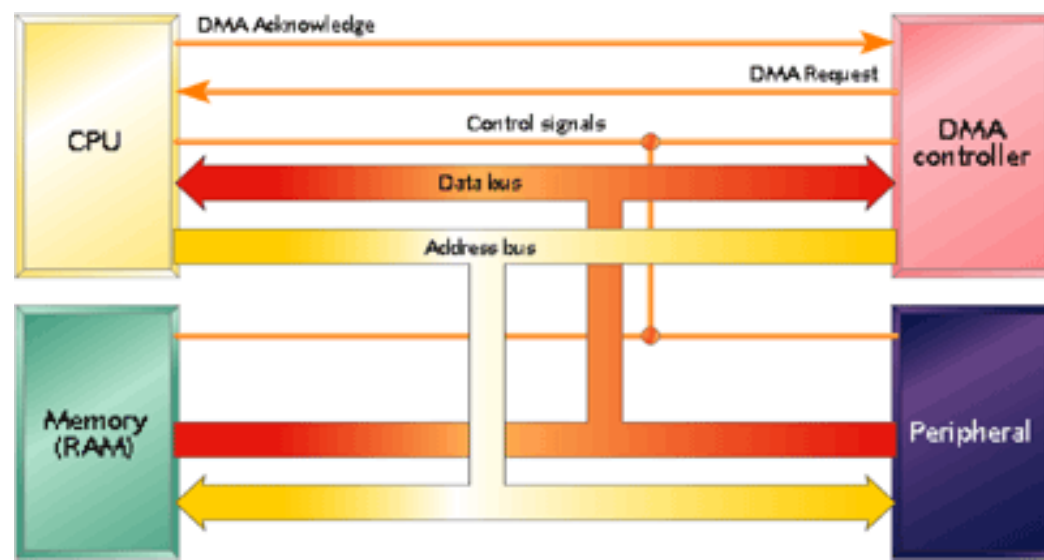


Interrupts

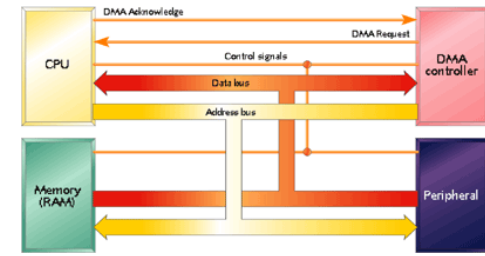
- The slowest method to move acquired data to system memory is for the DAQ-card to generate an interrupt request (IRQ) signal. This signal can be generated when one sample is acquired or when multiple samples are acquired. The process of transferring data to system memory via interrupts is given below:
 - When data is ready for transfer, the CPU stops whatever it is doing and runs a special interrupt handler routine that saves the current machine registers, and then sets them to access the board.
 - The data is extracted from the board and placed into system memory.
 - The saved machine registers are restored, and the CPU returns to the original interrupted process.
- The actual data move is fairly quick, but there is a lot of overhead time spent saving, setting up, and restoring the register information.

DMA introduction

- Direct memory access (DMA) is a system whereby samples are automatically stored in system memory while the processor (CPU) does something else.
- A computer usually supports several different DMA channels.



DMA (direct memory access) I



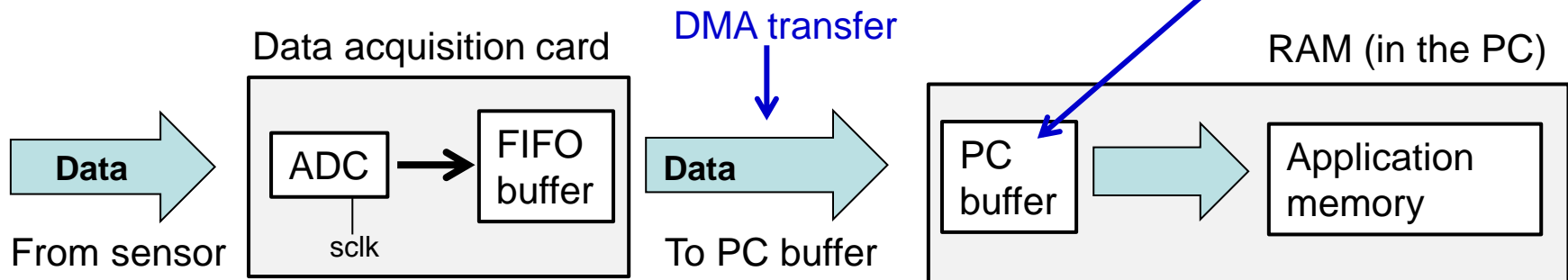
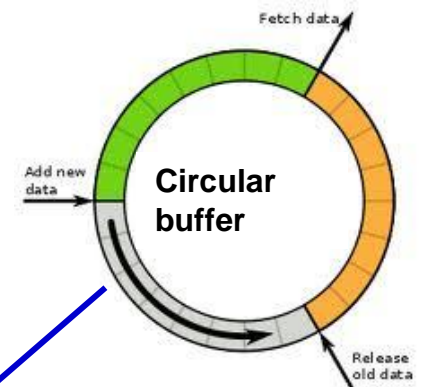
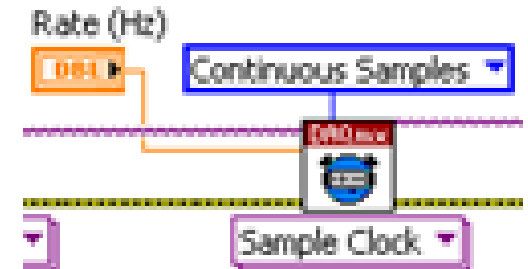
- DMA permits peripherals, such as a DAQ-card, to transfer data directly to or from memory without having each byte handled by the processor (CPU). Thus DMA enables more efficient use of interrupts, and increases data throughput.
- The process of transferring data via DMA is given below:
 - When data is ready for transfer, the DAQ-card notifies the DMA controller.
 - The DMA controller then asserts a DMA request signal to the CPU, asking its permission to use the bus (data bus, address bus, control bus).
 - The CPU completes its current bus activity, stops driving the bus, and returns a DMA acknowledge signal to the DMA controller.
 - The DMA controller then reads and writes one or more memory bytes, driving the address, data, and control signals as if it were itself the CPU.

DMA (direct memory access) II

- When the transfer is complete, the DMA controller stops driving the bus and deasserts the DMA request signal. The CPU can then remove its DMA acknowledge signal and resume control of the bus.
- In **single-cycle mode**, the DMA controller gives up the bus after each transfer. This minimizes the amount of time that the DMA controller keeps the processor off of the memory bus, but it requires that the bus request/acknowledge sequence be performed for every transfer. This overhead can result in a drop in overall system throughput if a lot of data needs to be transferred.
- In **burst mode**, the DMA controller keeps control of the bus until all the data buffered by the requesting device has been transferred to memory (or when the output device buffer is full, if writing to a peripheral).

LabVIEW DAQ - hardware setup

- When the **sample clock** (DAQmx Timing.vi) is configured, DAQmx configures the board for hardwared-timed I/O
 - *DAQ card sample clock or external sample clock*
- By enabling **continuous sampling** DAQmx automatically sets up a circular buffer
- DMA** is the default method of data transfer for DAQ devices that support DMA



DAQ data overwrite and overflow

- An **overwrite error** indicates that information is lost and occurs when the software program does not read data from the PC buffer quickly enough. Samples that are written to the circular PC buffer are overwritten before they are read into the application memory.
 - Solution: use **Producer-Consumer** architecture.
- An **overflow error** indicate that information has been lost earlier in the data acquisition process. Overflow errors indicate that the First In First Out (FIFO) memory buffer onboard the data acquisition card has reached its maximum capacity for storing acquired samples and can no longer accept new samples. An overflow error is symptomatic of a bus transfer rate that falls short of the requested data input rate.
 - Solution: use a Direct Memory Access (**DMA**) transfer mechanism.

How Is Buffer Size Determined in LabVIEW DAQmx ?

- If the acquisition is **continuous** (sample mode in DAQmx Timing.vi set to Continuous Samples), NI-DAQmx allocates a PC buffer equal in size to the value of the **samples per channel** (gives the number of samples to acquire) property, unless that value is less than the value listed in the following table. If the value of the **samples per channel** property is less than the value in the table below, NI-DAQmx uses the value in the table.

Sample Rate	PC Buffer Size
No rate specified	10 kS
0–100 S/s	1 kS
101–10,000 S/s	10 kS
10,001–1,000,000 S/s	100 kS
>1,000,000 S/s	1 MS

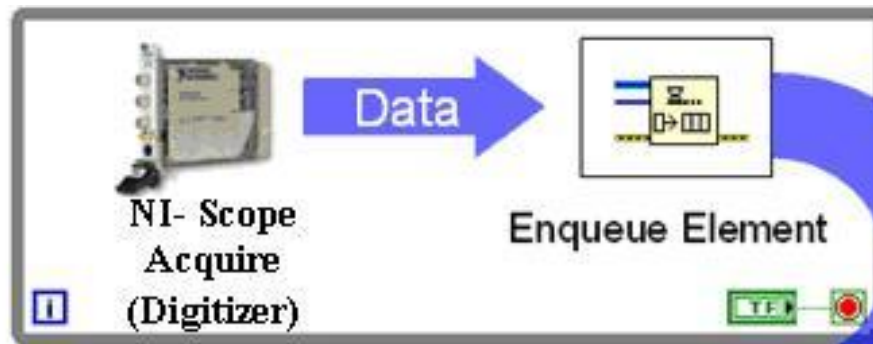
- You can override the default buffer size using the function *DAQmx Configure Input Buffer.vi*

High-speed DAQ

Solve the DAQ related problems listed on the next slide!

- Based on the producer-consumer architecture
- See the high-speed streaming lecture for more information!

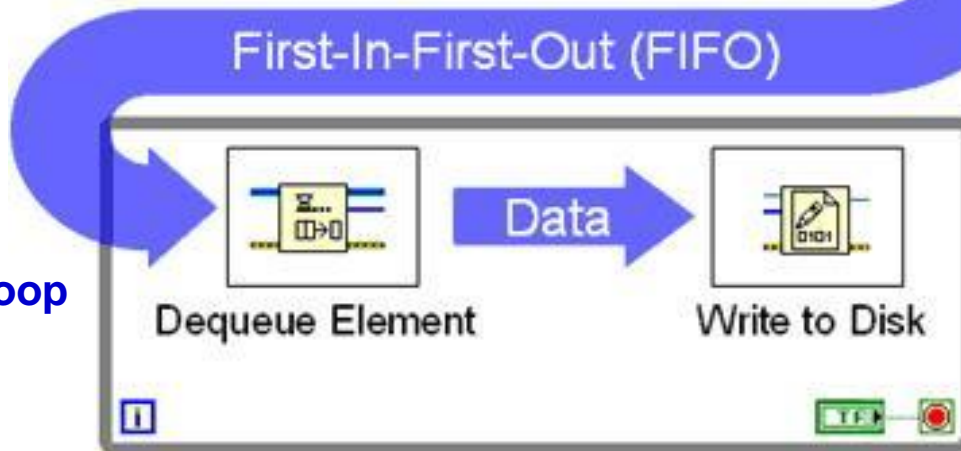
Producer loop



Hardware timing, since no Wait function is used in the producer loop. The producer loop rate is given by the DAQ-card setup

$$f_{producer} = \frac{sample\ rate\ (Hz)}{DAQ\ card\ buffer\ size}$$

Consumer loop



Common Pitfalls of Data Communication

Race conditions- two requests made to the same shared resource

Deadlock- two or more depended processes are waiting for each other to release the same resource

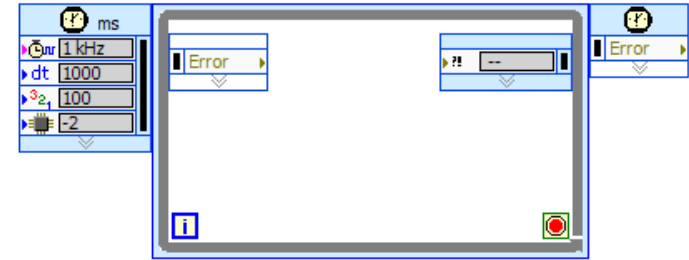
Data loss- gaps or discontinuities when transferring data

Performance degradation- poor processing speed due to dependencies on shared resources

Buffer overflows- writing to a buffer faster than it is read from the buffer

Stale data- reading the same data point more than once

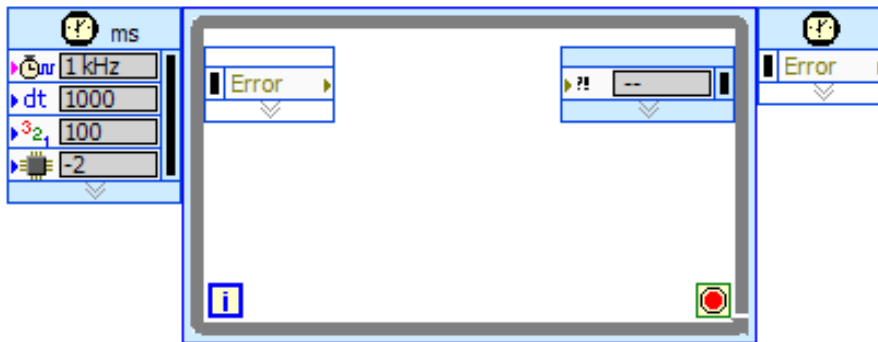
Timed Loop II



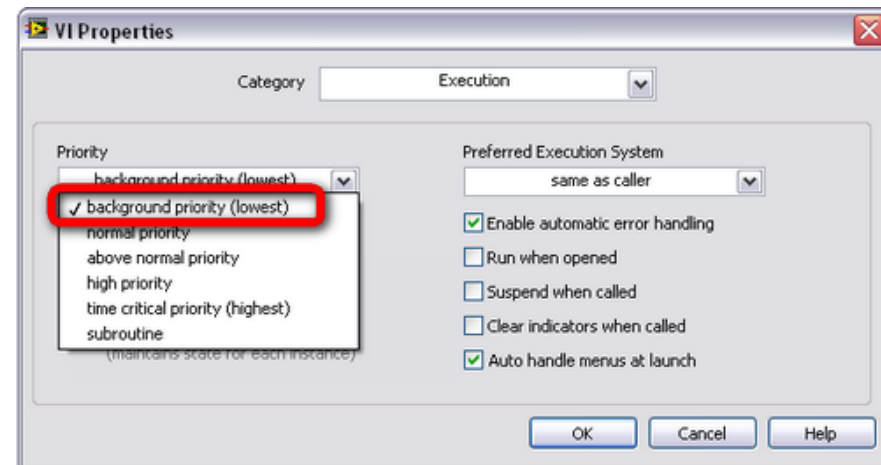
- 1 kHz internal clock (Windows)
- 1 MHz internal clock (for RT-targets)
- External timing source capability (using DAQmx Create Timing Source.vi)
- A Timed Loop gives you:
 - precise timing
 - feedback on loop execution
 - timing characteristics that can change dynamically
 - possibility to start the loop at a precise time each day (using a time stamp)
 - phase (offset) control
 - possibility to specifies the processor you want to handle execution
 - execution priority
 - precise determinism in a real-time operating system
- When Timed Loops as used on Windows (no RTOS) the OS can preempt your structure at any time to let a low-priority task run (based on “fairness”)

Execution priority

- While loops run at normal priority, and timed structures run between time-critical priority and above high priority.
- Therefore if you would like to have control of the priority of each aspect of your application, simply use timed structures, and set the priority between them using the priority input

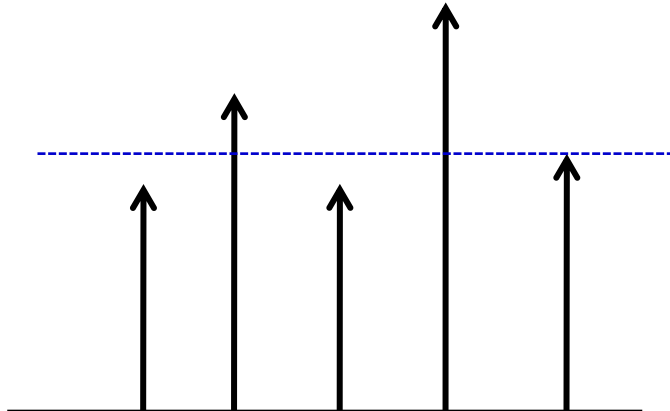


File - VI Properties»Execution:



Software Timing

- Attempts to resolve milliseconds on a PC within the limitations of the operating system (such as Windows)
- If you need better resolution or accuracy (determinism) you have to use a hardware solution (“software in the loop” degrades precision)



Hardware Timing

- Necessary for high-resolution and high-accuracy timing
 - e.g. for data correlation
- Hardware timing can give **ns** to **μs** accuracy (recall that software timing gives accuracy in the **ms** range)
- **Using GPS you can get a time uncertainty in the range of 10 – 100 ns**
- Ordinary DAQ-cards includes a stable crystal oscillator (for the ADC) that gives a resolution of μs or better, and this can be used for timing

Triggering

- A trigger is a signal that causes a device to perform an action, such as starting an acquisition. You can program your DAQ device to generate triggers on any of the following:
 - a software command
 - a condition on an external digital signal
 - a condition on an external analog signal
 - E.g. level triggering

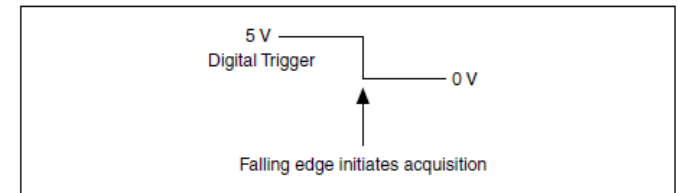


Figure 13-1. Falling-Edge Trigger

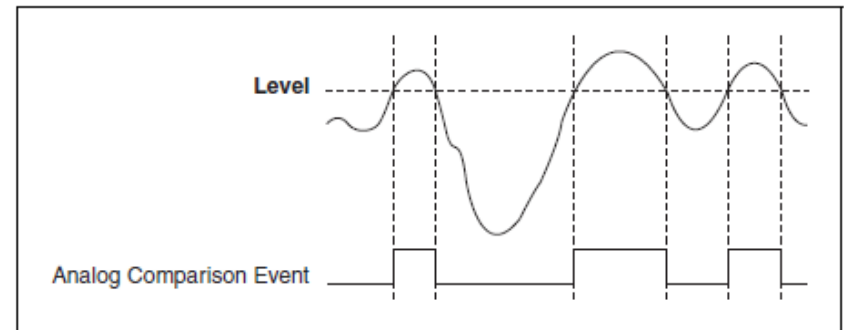


Figure 13-4. Above-Level Analog Triggering Mode

Time stamping of data

- Often need to timestamp an image in a video stream or a block of data from a DAQ-card to GPS (UTC) time; e.g. for use in data fusion in post-analysis.
- If the data samples has a deterministic (regular) interval, such as samples from a DAQ-card, it is sufficient to time stamp the first sample/sample batch.
 - The time of the remaining samples are found from

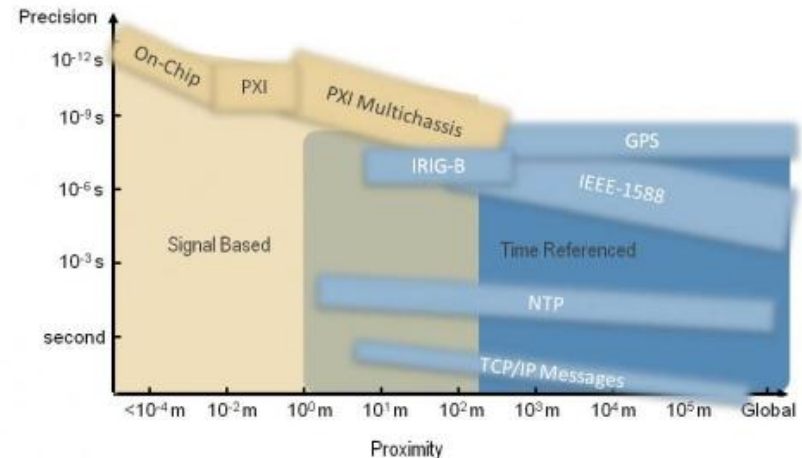
$$t = t_0 + \Delta t$$
$$\Delta t = \frac{1}{f_s}$$

- If the data samples are not deterministic (regular), e.g. video frames from a camera, each data point/video frame must include a timestamp

Signal based vs. time-based synchronization

- **Signal-based synchronization** involves sharing signals such as clocks and triggers directly (wires) between nodes that need to be synchronized.
- **Time-based synchronization** involves nodes independently synchronizing their individual clocks based on some time source, or time reference.
- There are advantages and disadvantages to both methods of device synchronization.

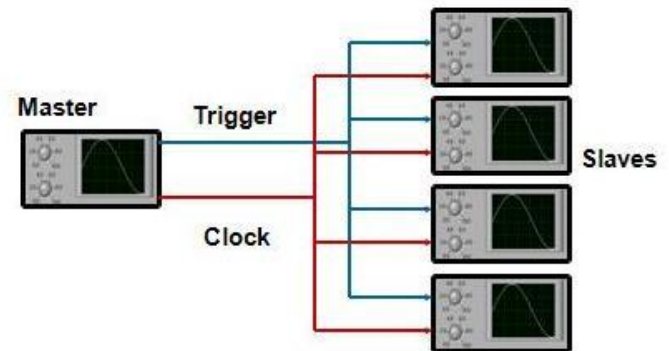
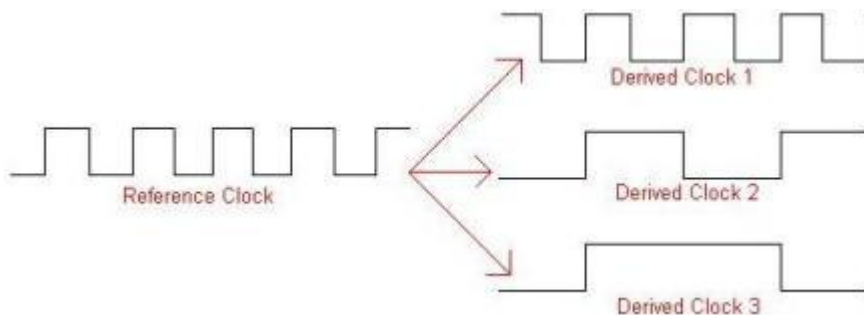
Synchronization Technologies



Synchronization Basics

Signal-based synchronization

- In systems where the devices are near each other, sharing a common timing signal is generally the easiest and most accurate method of synchronization.
- For example, modular instruments in a PXI chassis all share a common 10 MHz clock signal from the PXI backplane, enabling synchronization to less than 1 ns.
- To accurately use a common timing signal, a device must be calibrated to account for the signal propagation delay from the timing source to the device



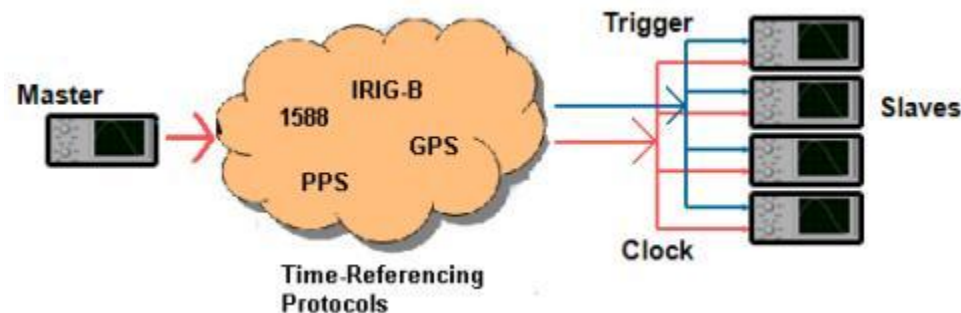
Time-based synchronization



- Necessary for long distances
- Because of the inherent instabilities in (crystal oscillator) clocks, distributed clocks must be synchronized continually to a time reference to match each other in frequency and phase.

- Time references:

- GPS
- IEEE 1588 masters
- IRIG-B sources



GPS with NTP-server and IRIG-B output



What is NTP?

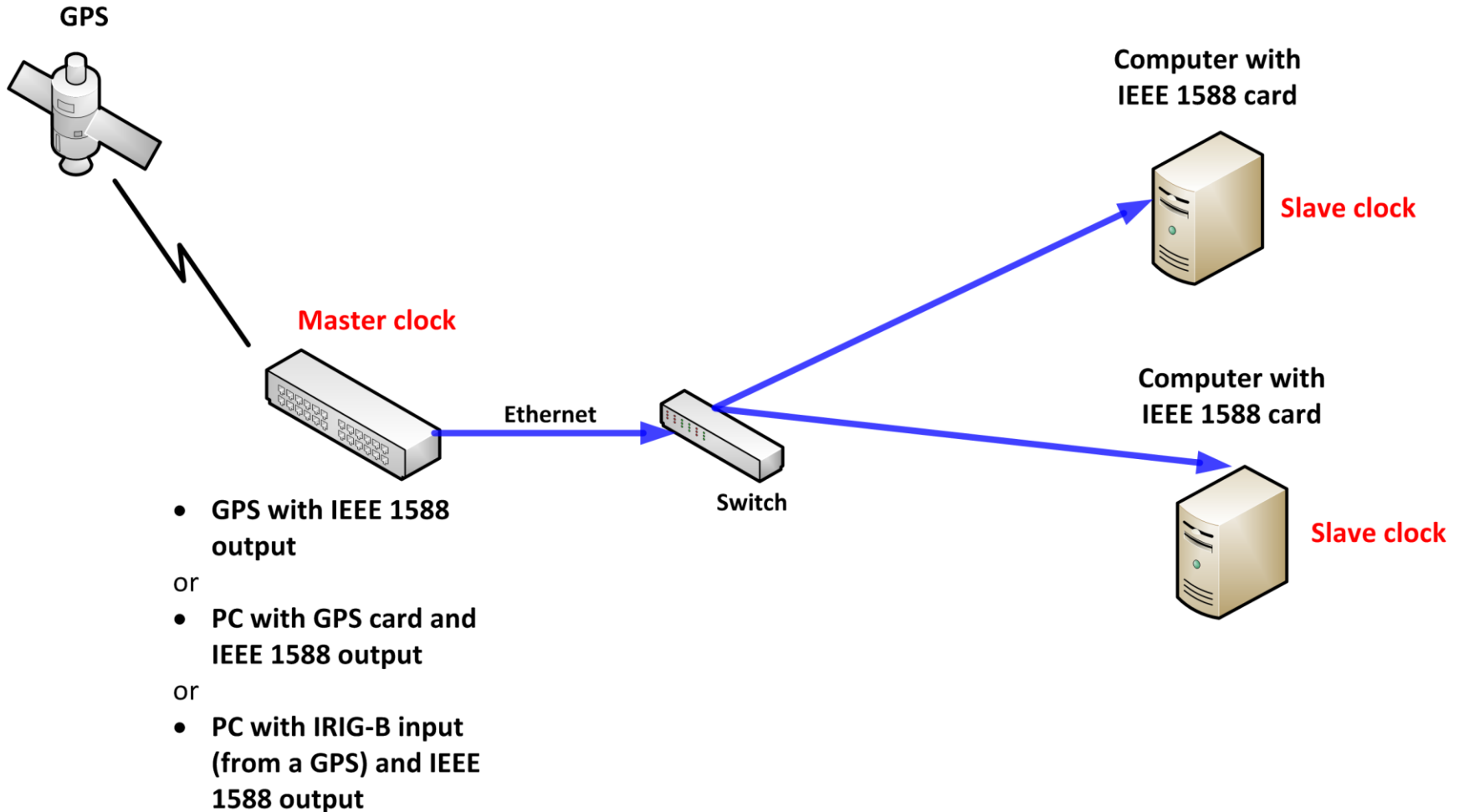
Timing and Synchronization Systems

IRIG = Inter Range Instrumentation Group
NTP = Network Time Protocol

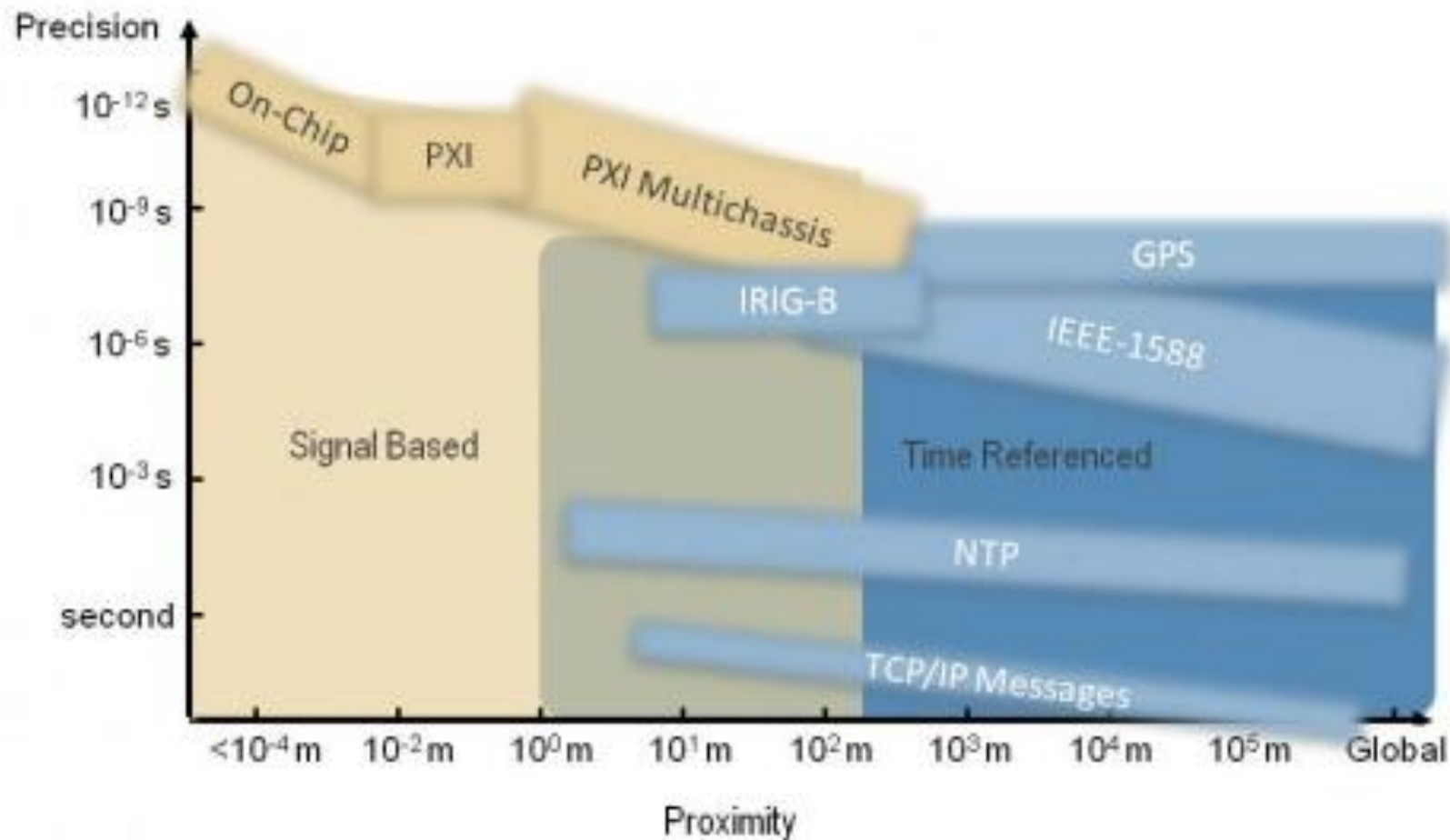
IEEE 1588 Protocol

- Gives sub-microsecond synchronization in distributed systems
- IEEE 1588 provides a standard protocol for synchronizing clocks connected via a multicast capable network, such as Ethernet.
 - uses a protocol known as the precision time protocol (PTP).
- All participating clocks in the network are synchronized to the highest quality clock in the network.
- The highest ranking clock is called the ***grandmaster clock***, and synchronizes all other ***slave clocks***.
- The level of precision achievable using the PTP protocol depends heavily on the jitter (the variation in latency) present in the underlying network topology.
 - Point-to-point connections provide the highest precision.

GPS & IEEE 1588 used for time synchronization of multiple PCs

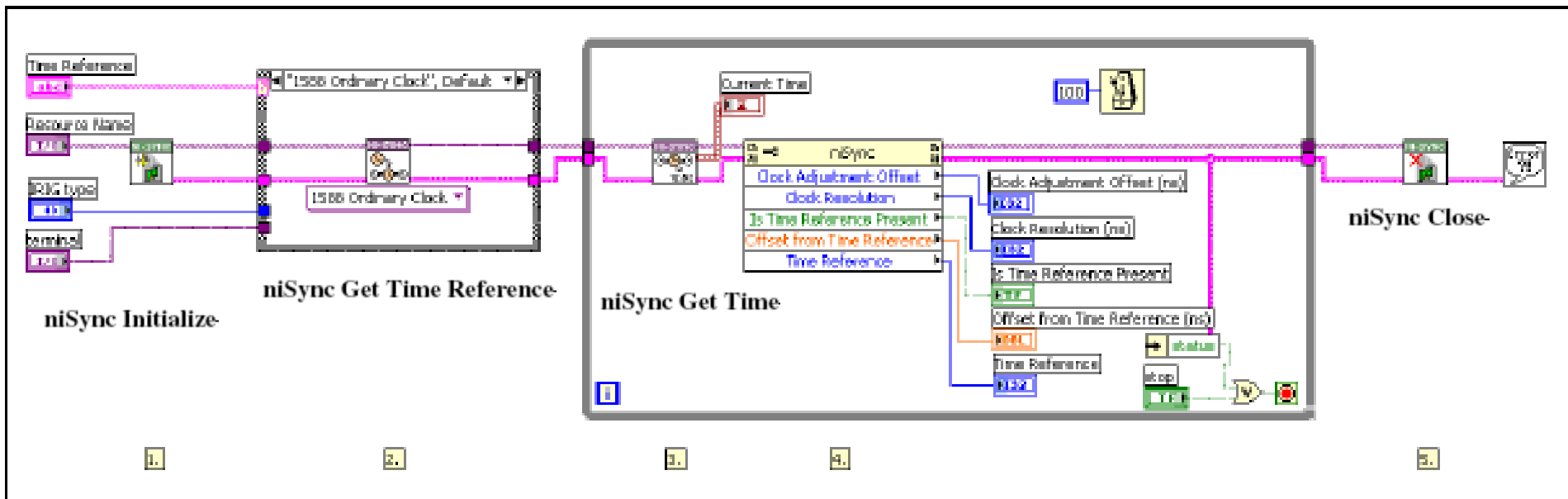


Synchronization Technologies

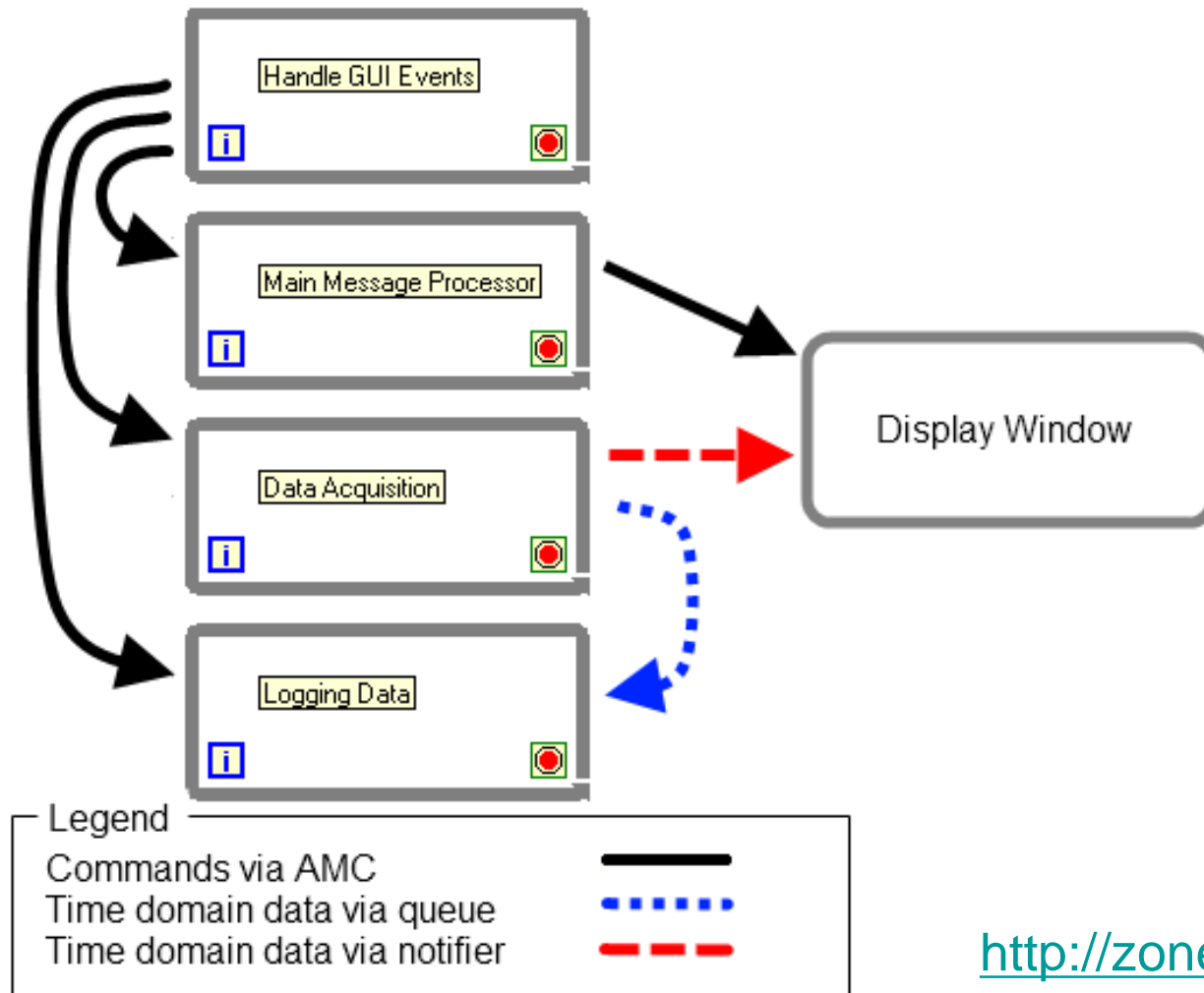


Time-based synchronization in LabVIEW

- Install the **NI-Sync** drivers
- Example: Time Reference (e.g. IEEE 1588, GPS or IRIG-B)



Example – DAQ reference design for LabVIEW



Postrun analysis / Quicklook tool

- Usually not sufficient to only look at the data in “real-time” (live)
- Usually very important to look at the data (perform a playback) as soon as possible after a data acquisition
 - E.g. to check the quality of the data
- It is very useful to have the Quicklook (playback) and analysis functions integrated in the data acquisition program!
- It should also be possible to read the measurement data by other common post-analysis tools (like Matlab)

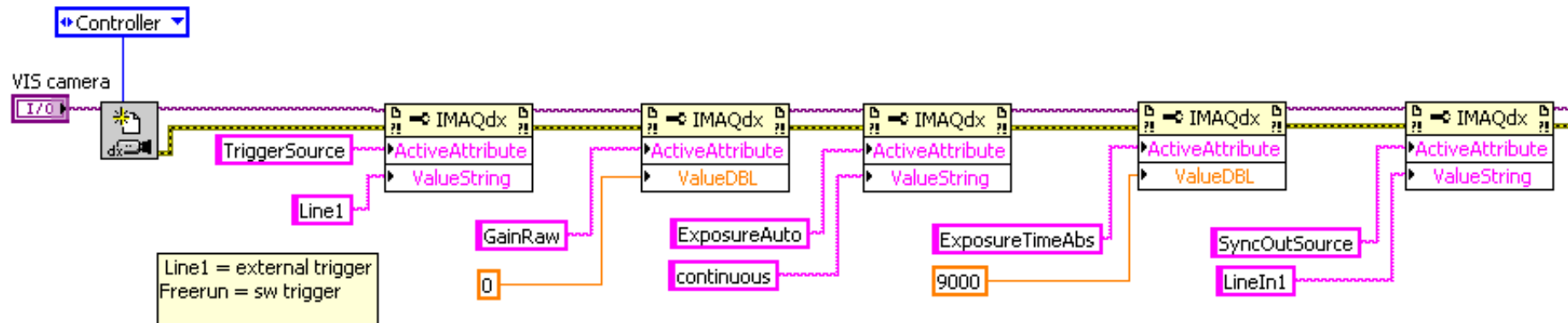
Example: VISION

- Example: Configuration of a GigE Vision camera
- Use MAX and the camera manual to find the commands to configure the camera

Tracking



Airbag deployment test



Vision time stamping example

GE680C – Visual camera (205 fps)



Question: How to accurately time stamp each image from a camera to GPS time, without any delay added from the software loop?

Possible configurations of **sync out**:

Exposing

Corresponds to when camera is integrating light.

Trigger Ready

Indicates when the camera will accept a trigger signal.

Trigger Input

A relay of the trigger input signal used to “daisy chain” the trigger signal for multiple cameras.

Readout

Valid when camera is reading out data.

Strobe

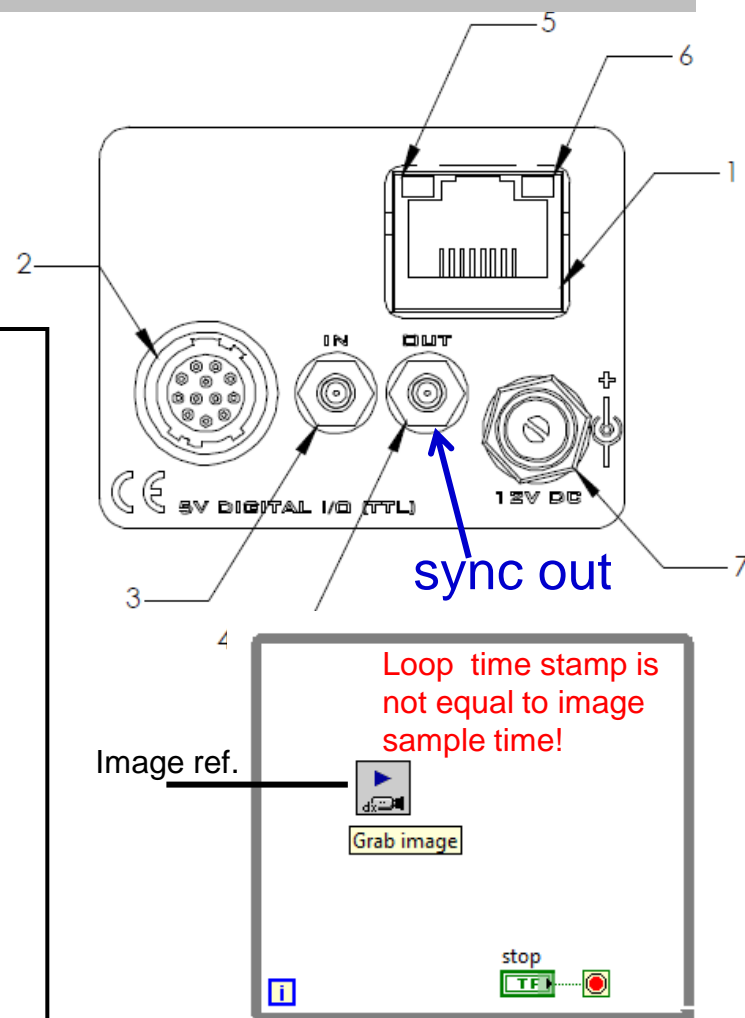
Programmable pulse based on one of the above events.

Imaging

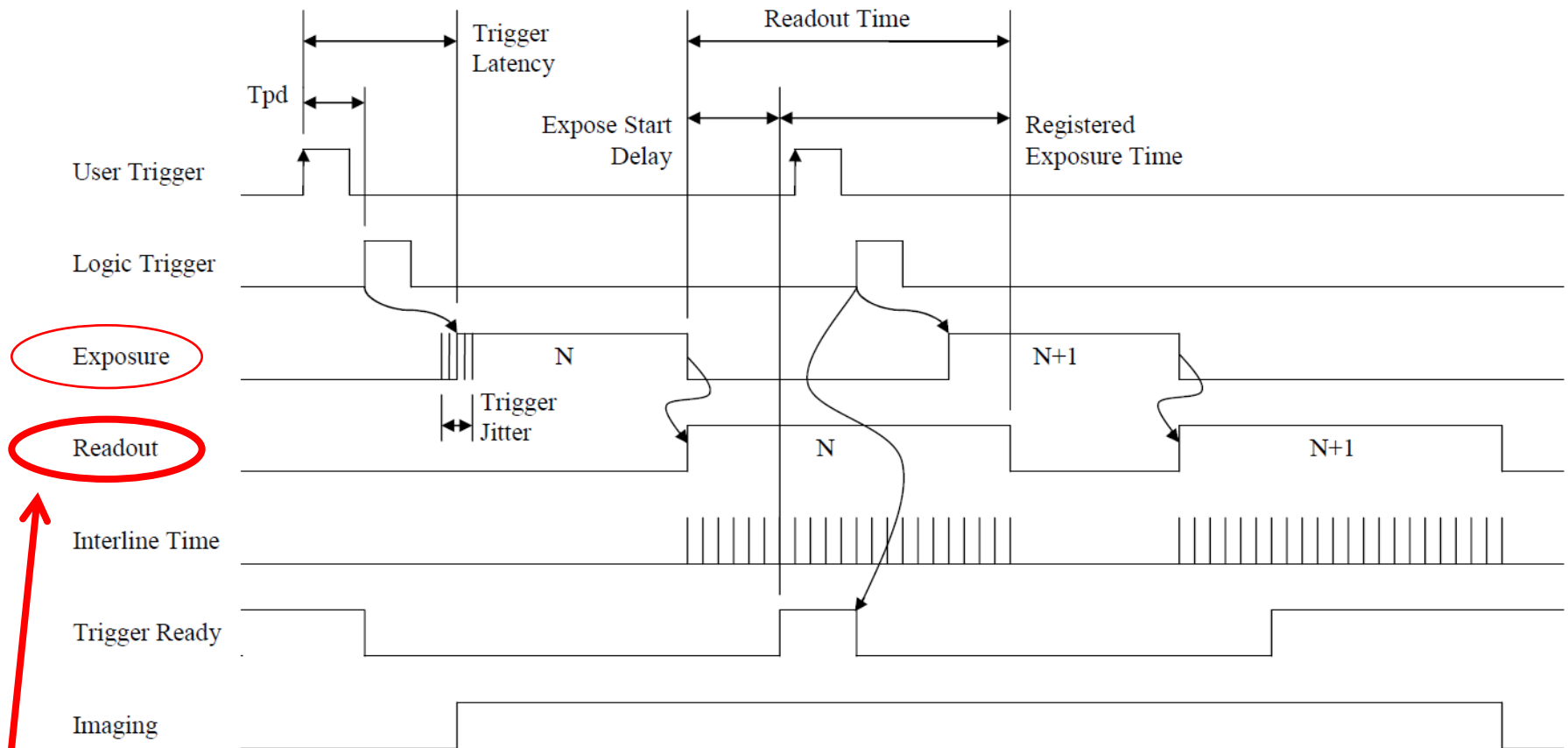
Valid when camera is exposing or reading out.

GPO

User programmable binary output.



Trigger Timing Diagram



Can timestamp the rising edge of the signal using e.g. the NI PXI 6682 card (GPS, IEEE 1588 and IRIG-B)

Trigger latency*

1.2 μ s

Trigger Jitter*

± 10 ns

Tpd*

90ns