Instrumentation and data acquisition

Spring 2011 – 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:
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)
- To 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

![RSE Diagram](image1)

![NRSE Diagram](image2)
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

**BETTER**
- Rejects Common-Mode Voltage
- Cuts Channel Count in Half

**NOT RECOMMENDED**
- Voltage difference (Vg) between the two grounds makes a ground loop that could damage the device

**GOOD**
- Allows use of entire channel count
- Doesn’t reject Common-Mode Voltage

Differential

RSE

NRSE
Options for Floating Signal Sources

**BEST**
- + Rejects Common-Mode Voltage
- - Cuts Channel Count in Half
- - Need bias resistors

**BETTER**
- + Allows use of entire channel count
- + Don’t need bias resistors
- - Doesn’t reject Common-Mode Voltage

**GOOD**
- + 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 $R_f$
- A good Opamp for current measurements:
  - LT1793

Input noise current density: $0.8 \text{ fA}/\sqrt{\text{Hz}}$

$V_O = -I_{IN} R_F$

Noise equivalent circuit:

$e_{nb} = \text{input current noise} \times R_f$
$e_{nv} = \text{input voltage noise}$
$e_{nj} = \text{thermal noise (voltage)}$
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)

$$SNR = \frac{V_{signal}}{V_{noise}} = \frac{(10 \text{ mV} \times 1000)}{1 \text{ mV}} = 10000$$

Example:
-5 V to +5 V

$$SNR = 20 \log \left( \frac{V_{signal}}{V_{noise}} \right)$$
Op-amp forsterkere

- Inverterende op-amp forsterker
  - \( V_o = -\frac{R_2}{R_1} \cdot V_i \)

- Ikke-inverterende op-amp forsterker
  - \( V_o = (1+\frac{R_2}{R_1}) \cdot V_i \)

- Ikke-inverterende opamp-kobling nyttig når man trenger høy input-impedans
- Inverterende opamp-kobling nyttig når man ønsker lav input-impedans
- Ikke-inverterende opamp-kobling gir mindre støy (pga \( G = 1+\frac{R_2}{R_1} \), i stedet for \( G = -\frac{R_2}{R_1} \))
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
- Separate ground planes of data acquisition device and sensor
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:

Want a small transition band!
Analoge filtre

- Filtertyper: LP, HP, BP, BS, Notch

- Passive filtre:
  - RC, LCR
  - (ønsker vanligvis å unngå L, men trengs for å få liten demping/høy Q)

- Aktive filtre
  - opamp + R og C
  - (kan unngå L)

- Noen vanlige filterkarakteristikker
  - Butterworth
  - Chebyshev
  - Bessel (konstant gruppeforsinkelse in PB)
  - Elliptisk
Sallen-Key - Active analog filter

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

![Sallen-Key Filter Diagrams](image-url)
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

$$f_c = f_{CLK} / 100$$
Importance of LP-filter selection for DAQ bandwidth

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

\begin{align*}
M (\text{dB}) \\
\begin{array}{c}
\text{BW} \\
0 \\
-3 \\
M_{\text{stop}} \\
\end{array} \\
\begin{array}{c}
f_c' \\
f_c \\
f_{\text{stop}} \\
f_s = 2f_c \\
f_s = 2f_{\text{stop}} \\
fs = 5f_c \text{ (in this example)}
\end{array}
\end{align*}
Sampling concepts

- Sampling
- Oversampling
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

Signal

Aliased Signal
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

<table>
<thead>
<tr>
<th>100Hz Sine Wave</th>
<th>Sampled at 100Hz</th>
<th>Aliased Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>100Hz Sine Wave</td>
<td>Sampled at 200Hz</td>
<td>Adequately Sampled for Frequency Only (Same # of cycles)</td>
</tr>
<tr>
<td></td>
<td>Sampled at 1kHz</td>
<td>Adequately Sampled for Frequency and Shape</td>
</tr>
</tbody>
</table>
ADC architectures

- Multiplexed
- Simultaneous sampling
ADC resolution

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

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

Example:

\[
\frac{\text{device range}}{2^{\text{resolution}}} = \frac{10}{2^{16}} = .15 \text{ mV}
\]
ADC oversampling

- The SNR of an ideal N-bit ADC (due to quantization effects) is:
  $$\text{SNR(dB)} = 6.02N + 1.76$$
- If the sampling rate is increased, we get the following SNR:
  - SNR(dB) = 6.02*N + 1.76 + 10* log_{10}(OSR)
  - OSR = f_s/f_{nyquist}
- Nyquist sampling theorem: $f_s \geq 2 \Delta f_{signal}$
- 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 $= N + 1/2 \log_2 (f_s/f_{nyquist})$, where N is the resolution of an ideal N-bit ADC at the Nyquist rate
  - If OSR = $f_s/f_{nyquist} = 1024$, an 8-bit ADC gets and effective resolution equal to that of a 13-bit ACD at the Nyquist rate (which is $2 \Delta f_{signal}$)
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 = more precise representation of your signal
- Unipolar signals are signals that range from 0 value to a positive value (e.g. 0 – 5 V)
- Bipolar signals are signals that range from a negative to a positive value (e.g. -5 to +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 devices
- You don’t choose the gain directly
  - Choose the input limits of your signal in LabVIEW
  - 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)
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
Data Acquisition Hardware

- DAQ Hardware turns your PC into a measurement and automation system
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
20 Years | NI Data Acquisition

- NuBus (1988)
- PCI (1989)
- ISA (1995)
- PXI (1996)
- R Series (1998)
- USB (2003)
- PCI Express (2004)
- PXI Express (2005)
- NI Wi-Fi DAQ (2008)
- Ethernet DAQ (2009)
- X Series (2010)
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.
Major PXI Vendors

- National Instruments
- Agilent
- Aeroflex
- EADS (Racal)
- Advantest
- Adlink
- Keithley
- Pickering

- Teradyne
- Acqiris
- Rohde & Schwarz
- JTAG Technologies
- PXIT
- Virginia Panel Corp
- Advantech
- Conduant Corporation

Over 1200 Products From More Than 70 Vendors

For a complete list of vendors, members and products: see www.pxisa.org

ni.com
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
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 rather than spend time looking up the correct command, formatting command strings, and parsing returned data.
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.
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
LabVIEW Express: DAQ assistant
Select the measurement type for the task.

A task is a collection of one or more virtual channels with timing, triggering, and other properties.

To have multiple measurement types within a single task, you must first create the task with one measurement type. After you create the task, click the Add Channels button to add a new measurement type to the task.

Select the physical channel(s) to add to the task.

If you have previously configured global virtual channels of the same measurement type as the task, click the Virtual tab to add or copy global virtual channels to the task. When you copy the global virtual channel to the task, it becomes a local virtual channel. When you add a global virtual channel to the task, the task uses the actual global virtual channel, and any changes to that global virtual channel are reflected in the task.

If you have TEDS configured, click the TEDS tab to add TEDS channels to the task.

For hardware that supports multiple channels in a task, you can select multiple channels to add to a task at the same time.
Click the Add Channels button (+) to add more channels to the task.
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).

• These steps happen automatically. All that's required from you is some initial configuration of the hardware.
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

• Direct memory access (DMA) is a system whereby samples are automatically stored in system memory while the processor (CPU) does something else. The process of transferring data via DMA is given below:
  – When data is ready for transfer, the DAQ-card directs the system DMA controller to put data into system memory as soon as possible.
  – As soon as the CPU is able (which is usually very quickly), it stops interacting with the data acquisition hardware and the DMA controller moves the data directly into memory.
  – The DMA controller gets ready for the next sample by pointing to the next open memory location.
  – The previous steps are repeated indefinitely, with data going to each open memory location in a continuously circulating buffer. **No interaction between the CPU and the board is needed.**

• A computer usually supports several different DMA channels.
LabVIEW DAQ - hardware setup

- When the sample clock is configured, DAQmx configures the board for hardwared-timed I/O

- By enabling continuous sampling DAQmx automatically sets up a circular buffer

- DMA is the default method of data transfer for DAQ devices that support it
High-speed DAQ

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

Timed Loop Timing Sources

Internal Timing Source Under the Hood

- RTC
- NI-TimeSync

Software
- IEEE 1588
- SNTP

LabVIEW Timed Loop
- SW Timing Source API
  - Create Software-Triggered Timing Source
  - Fire Software-Triggered Timing Source

LabVIEW DAQmx
- Timing Source API

LabVIEW Timed Loop
- Software-Triggered Timing Source
- External Timing Source
Timed Loop

DAQmx Create Timing Source.vi
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”
Timed Loop external synch using DAQmx

Steps:
1. Create an Analog Input voltage channel.
2. Set the timing type of the analog input task to Sample Clock.
3. Call the DAQmx Create Timing Source (Digital Edge Counter) VI and specify the edge count to be the Samples to Read. What this does is cause the timed loop to only fire once it has received “samples to read” number of edges from the AI sample clock. This is an efficient use of CPU resources because the loop, and therefore the AI Read, will only execute once all of the samples are available.
4. Call the Start VI to start the acquisition.
5. Read the waveform once all the samples to read are available. In this approach the Read VI behaves in a "non-blocking" fashion, since it does not have to wait for samples to become available.
6. Call the Clear Task VI to Clear the Task.
7. Use the popup dialog box to display an error if any.
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:
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
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.
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
• By connecting a GPS with NTP-server to a LAN computers can synchronize their clocks within ms using standard Ethernet connections

GPS with NTP-server and IRIG-B output

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.
Time-based synchronization hardware

Examples from NI:

- **PXI-6682**
  - GPS, IEEE 1588, IRIG-B.
  - Can function as IEEE 1588 grandmasters to achieve synchronization of external 1588 devices.
  - Can timestamp rising/falling edges.

- **PCI-1588**
  - Synchronization of distributed PC-based data acquisition or control systems.
  - Operating as either an IEEE 1588 grandmaster or slave clock.
Time-based synchronization in LabVIEW

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

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

![Tracking and Airbag deployment test images](image-url)
Example – DAQ reference design for LabVIEW

http://zone.ni.com/devzone/cda/tut/p/id/11805