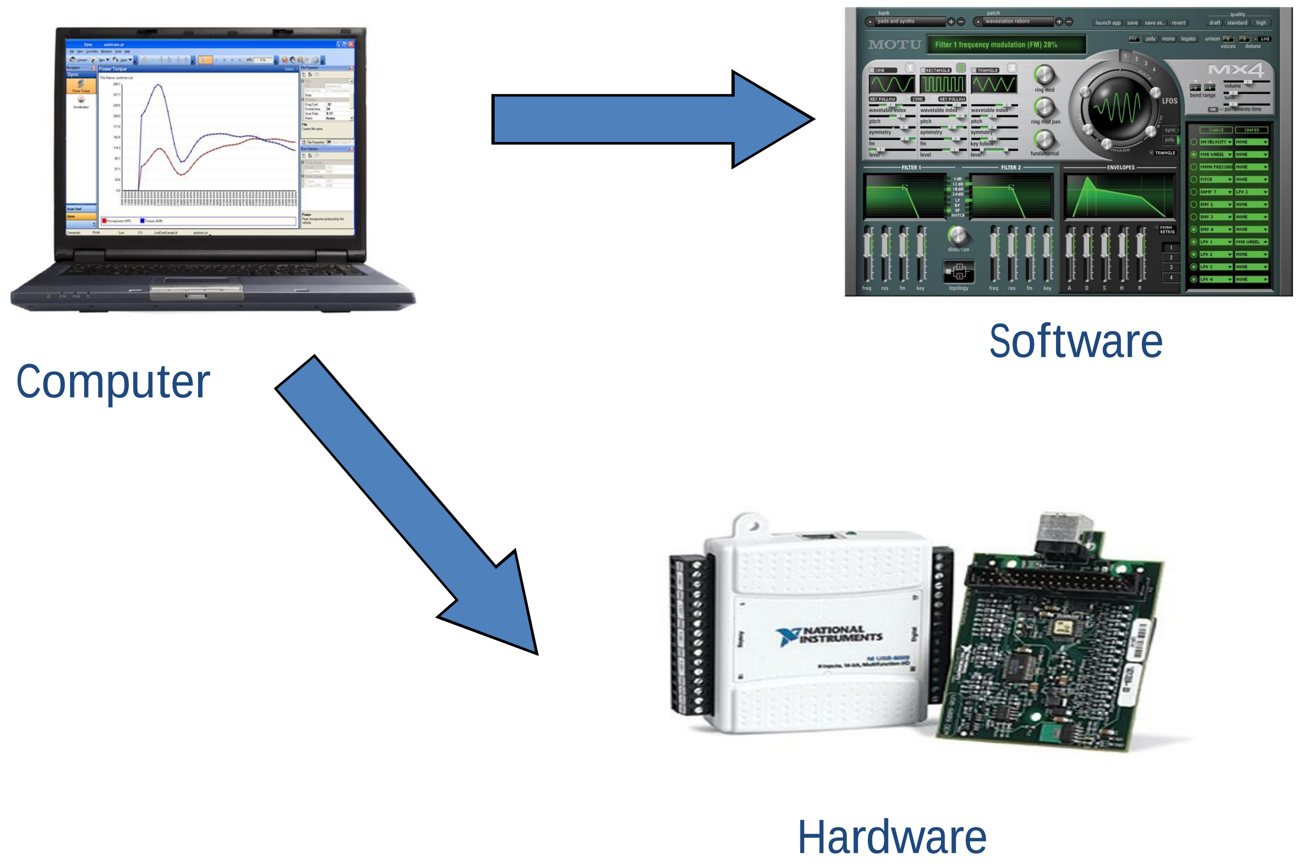
UNIT-I INTRODUCTION

Virtual Instrumenation-Introduction

Virtual Instrumentation is the use of customizable software and modular measurement hardware to create user-defined measurement systems, called virtual instruments.



The technology of Virtual Instruments (cont.)

- **Advantages of Virtual Instruments versus Traditional Instruments**

  Flexibility
  You can easily add additional functions such as a filter routine or a new data view to a virtual instrument.

  Storage
  Today's personal computers have hard disks that can store dozens of gigabytes which is an absolute plus if you want to process mass data like audio or video.

  Display
  Computer monitors usually have better color depth and pixel resolution than traditional instruments. Also you can switch easily between different views of the data (graphical, numerical).

What is Virtual Instrumentation?

Virtual instrumentation combines mainstream commercial technologies. such as C, with flexible software and a wide variety of measurement and control hardware, so engineers and scientists can create user-defined systems that meet their exact application needs. With virtual instrumentation, engineers and scientists reduce development time, design higher quality products, and lower their design costs.

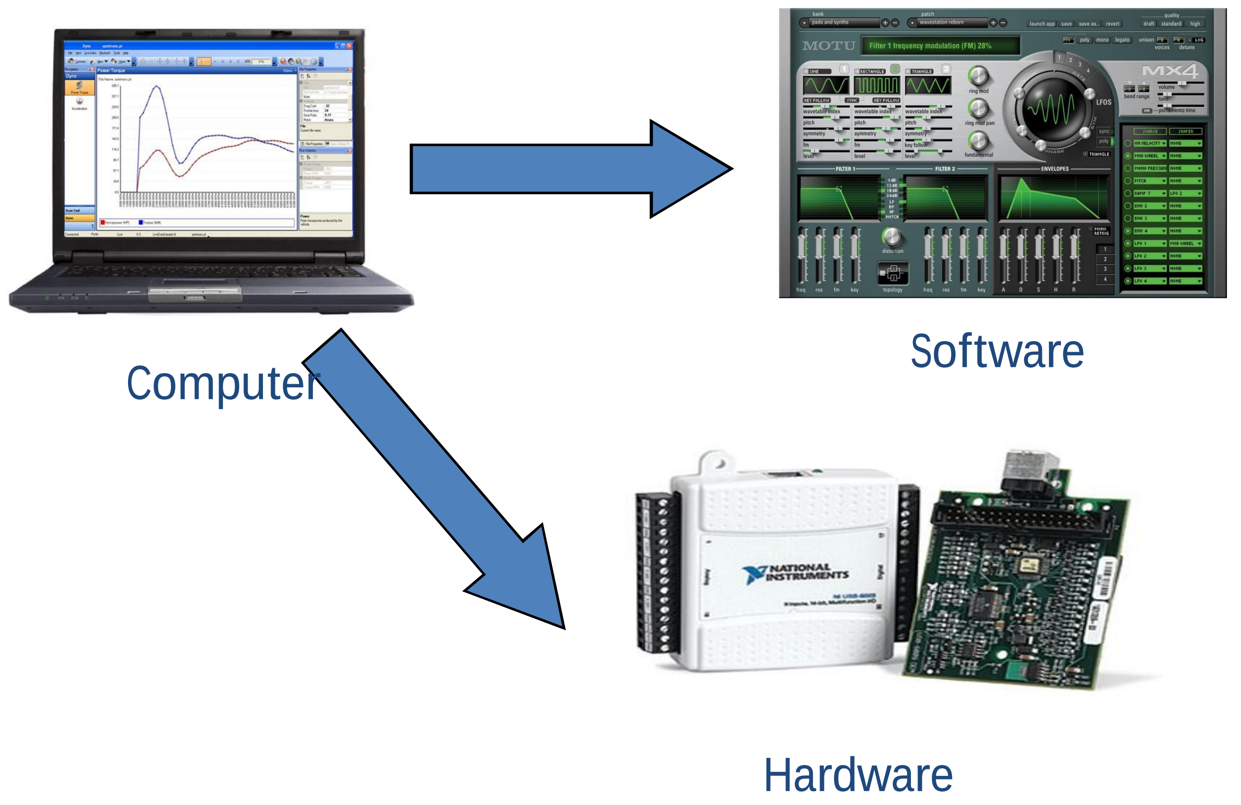
Why is Virtual Instrumentation necessary?

Virtual instrumentation is necessary because it delivers instrumentation with the rapid adaptability required for today’s concept, product, and process design, development, and delivery. Only with virtual instrumentation can engineers and scientists create the user-defined instruments required to keep up with the world’s demands.

The only way to meet these demands is to use test and control architectures that are also software centric. Because virtual instrumentation uses highly productive software, modular I/O, and commercial platforms, it is uniquely positioned to keep pace with the required new idea and product development rate

The technology of Virtual Instruments

Virtual Instrumentation is the use of customizable software and modular measurement hardware to create user-defined measurement systems, called virtual instruments.

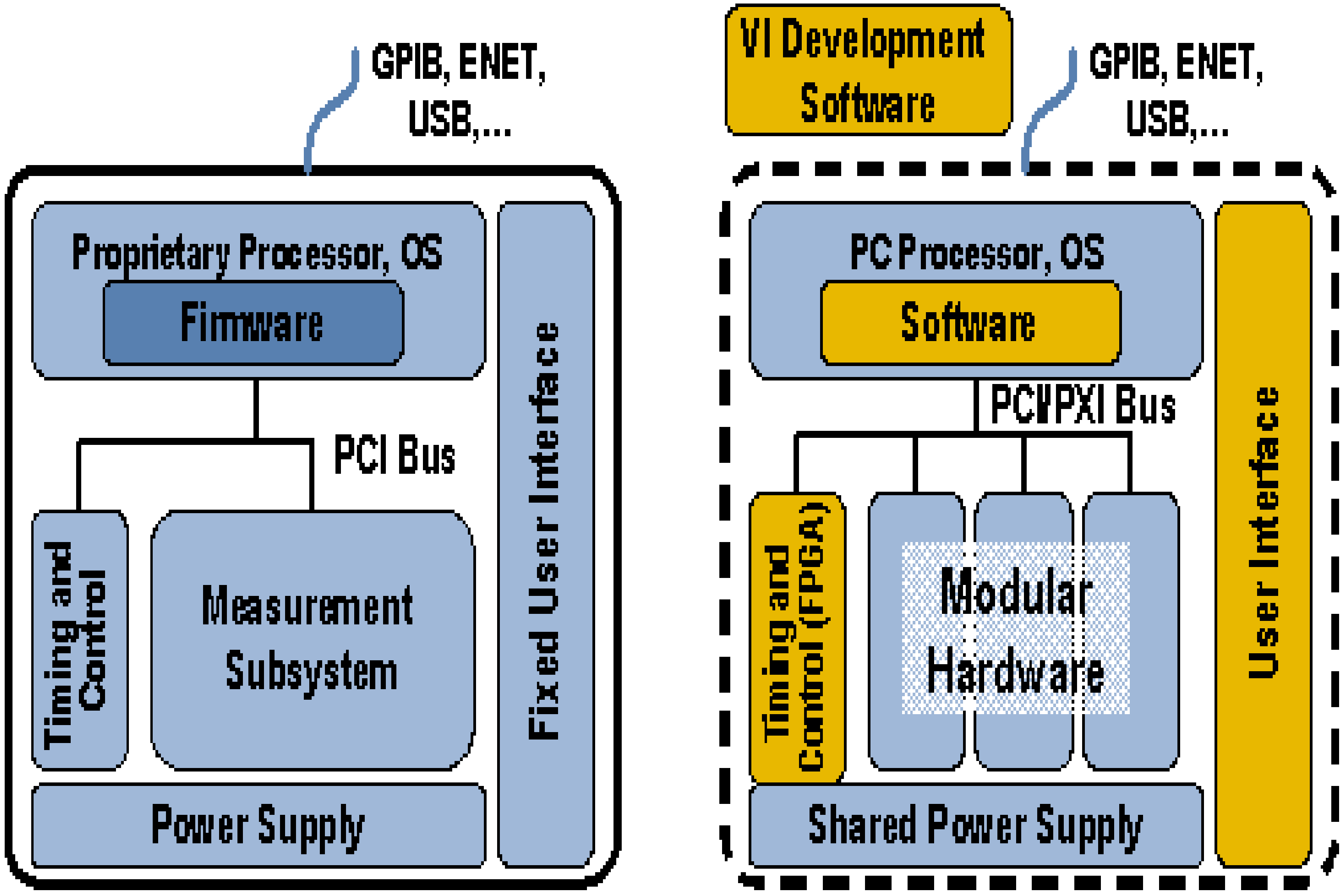


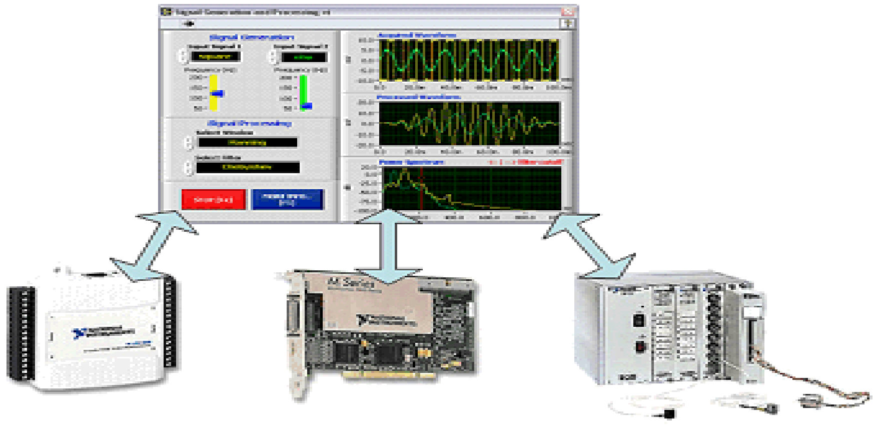
What is a virtual instrument and how is it different from a traditional instrument?

Every virtual instrument consists of two parts – software and hardware. A virtual instrument typically has a sticker price comparable to and many times less than a similar traditional instrument for the current measurement task.

A traditional instrument provides them with all software and measurement circuitry packaged into a product with a finite list of fixed-functionality using the instrument front panel. A virtual instrument provides all the software and hardware needed to accomplish the measurement or control task. In addition, with a virtual instrument, engineers and scientists can customize the acquisition, analysis, storage, sharing, and presentation functionality using productive, powerful software.

Traditional instruments (left) and software based virtual instruments (right) largely share the same architectural components, but radically different philosophies





One Application -- Different Devices

For this particular example, an engineer is developing an application using LabVIEW and an M Series DAQ board on a desktop computer PCI bus in his lab to create a DC voltage and temperature measurement application. After completing the system, he needs to deploy the application to a PXI system on the manufacturing floor to perform the test on new product.

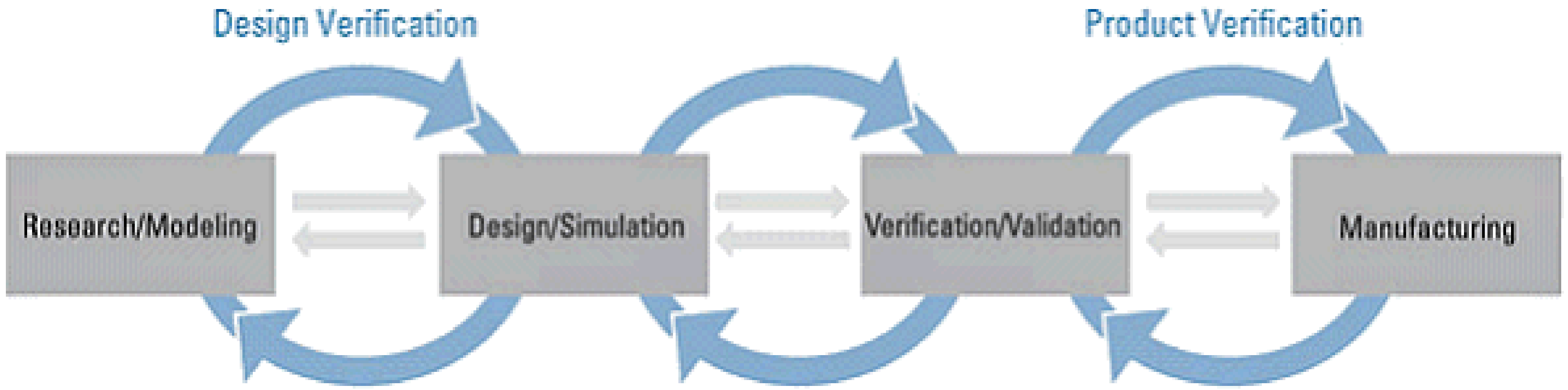
Alternatively, he may need the application to be portable, and so he selects NI USB DAQ products for the task. In this example, regardless of the choice, he can use virtual instrumentation in a single program in all three cases with no code change needed.

Virtual Instrumentation for Industrial I/O and Control

- PCs and PLCs both play an important role in control and industrial applications. PCs bring greater software flexibility and capability, while PLCs deliver outstanding ruggedness and reliability. But as control needs become more complex, there is a recognized need to accelerate the capabilities while retaining the ruggedness and reliabilities.
- Multi domain functionality (logic, motion, drives, and process) – the concept supports multiple I/O types. Logic, motion, and other function integration is a requirement for increasingly complex control approaches.
- Software tools for designing applications across several machines or process units – the software tools must scale to distributed operation.

Virtual Instrumentation for Design

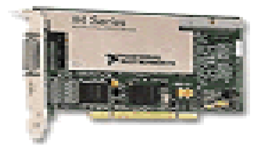
- The same design engineers that use a wide variety of software design tools must use hardware to test prototypes. Commonly, there is no good interface between the design phase and testing/validation phase, which means that the design usually



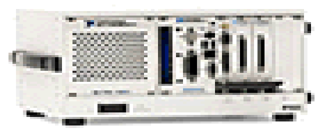
On which hardware I/O and platforms does virtual instrumentation software run?

- Standard hardware platforms that house the I/O are important to I/O modularity. Laptop and desktop computers provide an excellent platform where virtual instrumentation can make the most of existing standards such as the USB, PCI, Ethernet, and PCMCIA buses.

USB data acquisition starting at $145
• Pocket-sized for ultimate portability
• Powered by the USB connection
• Onboard precision voltage reference



M Series Plug-In Measurement Hardware
• 24 new devices
• Up to 18-bit resolution and 1.25 MS/s
• Starting at less than $7/channel



PXI Modular Instrumentation
• Open, multivendor standard
• Rackmount, portable and benchtop options
• 10X performance increase over ordinary instrumentation

- for example, USB 2.0 and PCI Express

Layers of Virtual Instrumentation

- Application Software: Most people think immediately of the application software layer. This is the primary development environment for building an application.
- Test and Data Management Software: Above the application software layer the test executive and data management software layer. This layer of software incorporates all of the functionality developed by the application layer and provides system-wide data management.
- Measurement and Control Services Software: The last layer is often overlooked, yet critical to maintaining software development productivity.

Advantages of Virtual Instruments versus Traditional Instruments

Flexibility
You can easily add additional functions such as a filter routine or a new data view to a virtual instrument.

Storage
Today's personal computers have hard disks that can store dozens of gigabytes which is an absolute plus if you want to process mass data like audio or video.

Display
Computer monitors usually have better color depth and pixel resolution than traditional instruments. Also you can switch easily between different views of the data (graphical, numerical).

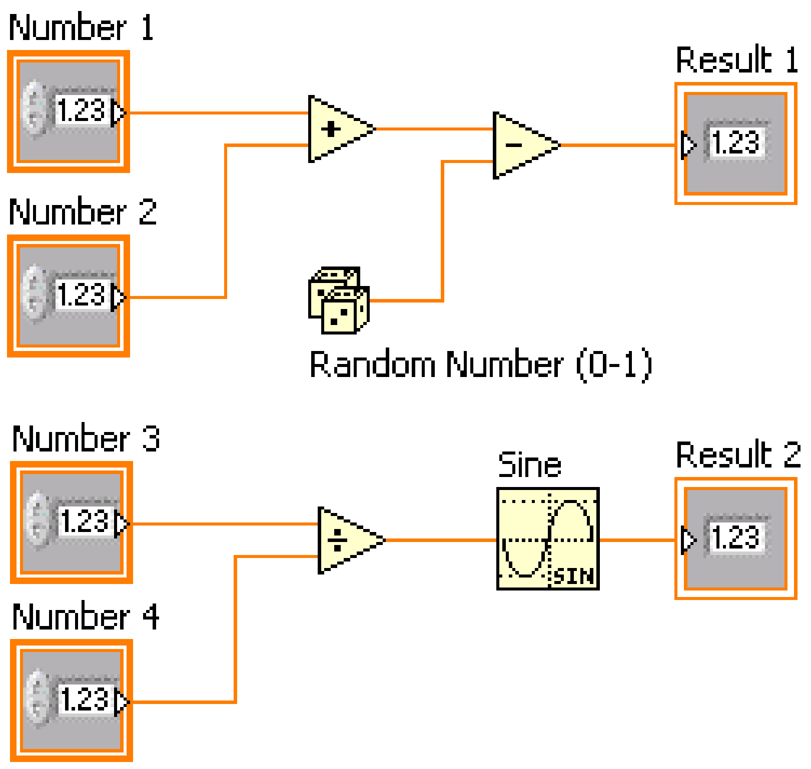
Costs
PC add-in boards for signal acquisition and software mostly cost a fraction of the traditional hardware they emulate.

Dataflow Programming

- Block diagram executes dependent on the flow of data; block diagram does NOT execute left to right

- Node executes when data is available to ALL input terminals

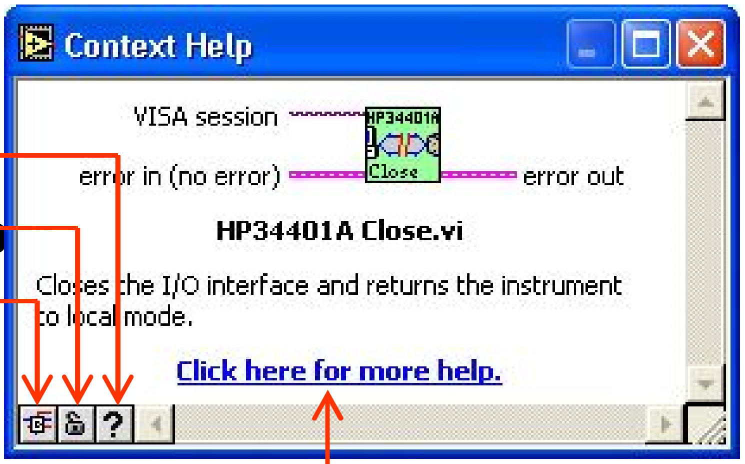
- Nodes supply data to all output terminals when done



Help Options

Context Help

- **Online help**
- **Lock help**
- **Simple/Complex Diagram help**
- **Ctrl + H**



HP34401A Close.vi

Closes the I/O interface and returns the instrument to local mode.

Click here for more help.

Online reference

- **All menus online**
- **Pop up on functions in diagram to access online info directly**

Graphical programming in data flow

LabVIEW

LabVIEW is a graphical programming language that uses icons instead of lines of text to create applications. In contrast to text-based programming languages, where instructions determine program execution, LabVIEW uses dataflow programming, where the flow of data determines execution order.

You can purchase several add-on software toolkits for developing specialized applications. All the toolkits integrate seamlessly in LabVIEW. Refer to the National Instruments Web site for more information about these toolkits.

LabVIEW also includes several wizards to help you quickly configure your DAQ devices and computer-based instruments and build applications

LabVIEW programs are called virtual instruments (VIs).

Controls are inputs and indicators are outputs.

Each VI contains three main parts:

•Front Panel – How the user interacts with the VI.

•Block Diagram – The code that controls the program.

•Icon/Connector – Means of connecting a VI to other VIs.

In LabVIEW, you build a user interface by using a set of tools and objects. The user interface is known as the front panel. You then add code using graphical representations of functions to control the front panel objects. The block diagram contains this code. In some ways, the block diagram resembles a flowchart

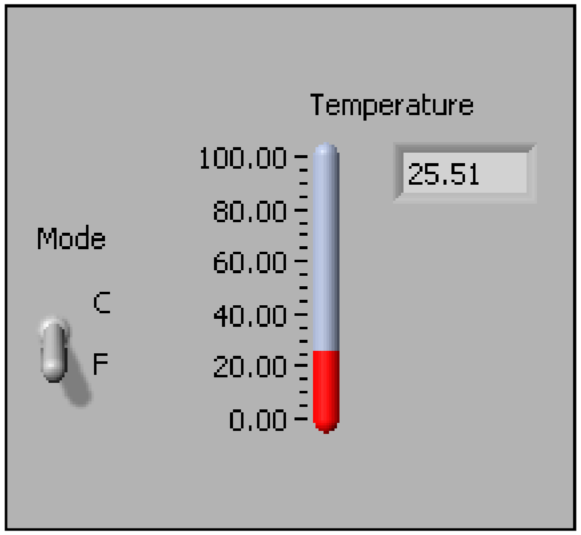
Users interact with the Front Panel when the program is running. Users can control the program, change inputs, and see data updated in real time. Controls are used for inputs such as, adjusting a slide control to set an alarm value, turning a switch on or off, or to stop a program. Indicators are used as outputs. Thermometers, lights, and other indicators display output values from the program. These may include data, program states, and other information.

Every front panel control or indicator has a corresponding terminal on the block diagram. When a VI is run, values from controls flow through the block diagram, where they are used in the functions on the diagram, and the results are passed into other functions or indicators through wires.

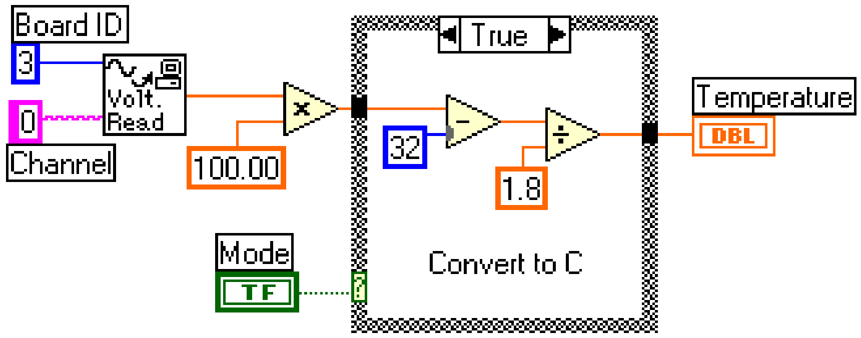
UNIT-II- VI PROGRAMMING TECHNIQUES

LabVIEW Programs Are Called Virtual Instruments (VIs)

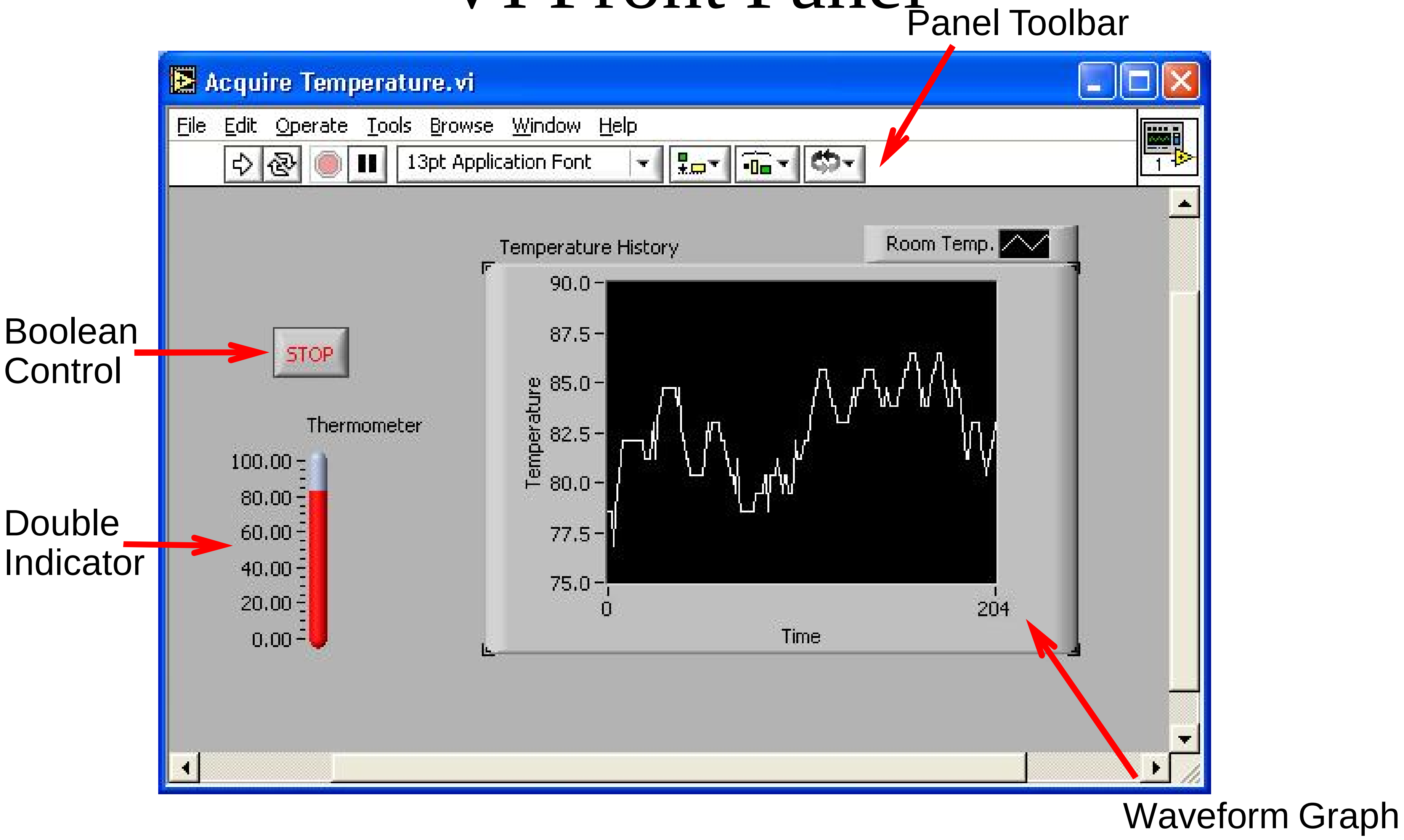
Front Panel
- Controls = Inputs
- Indicators = Outputs



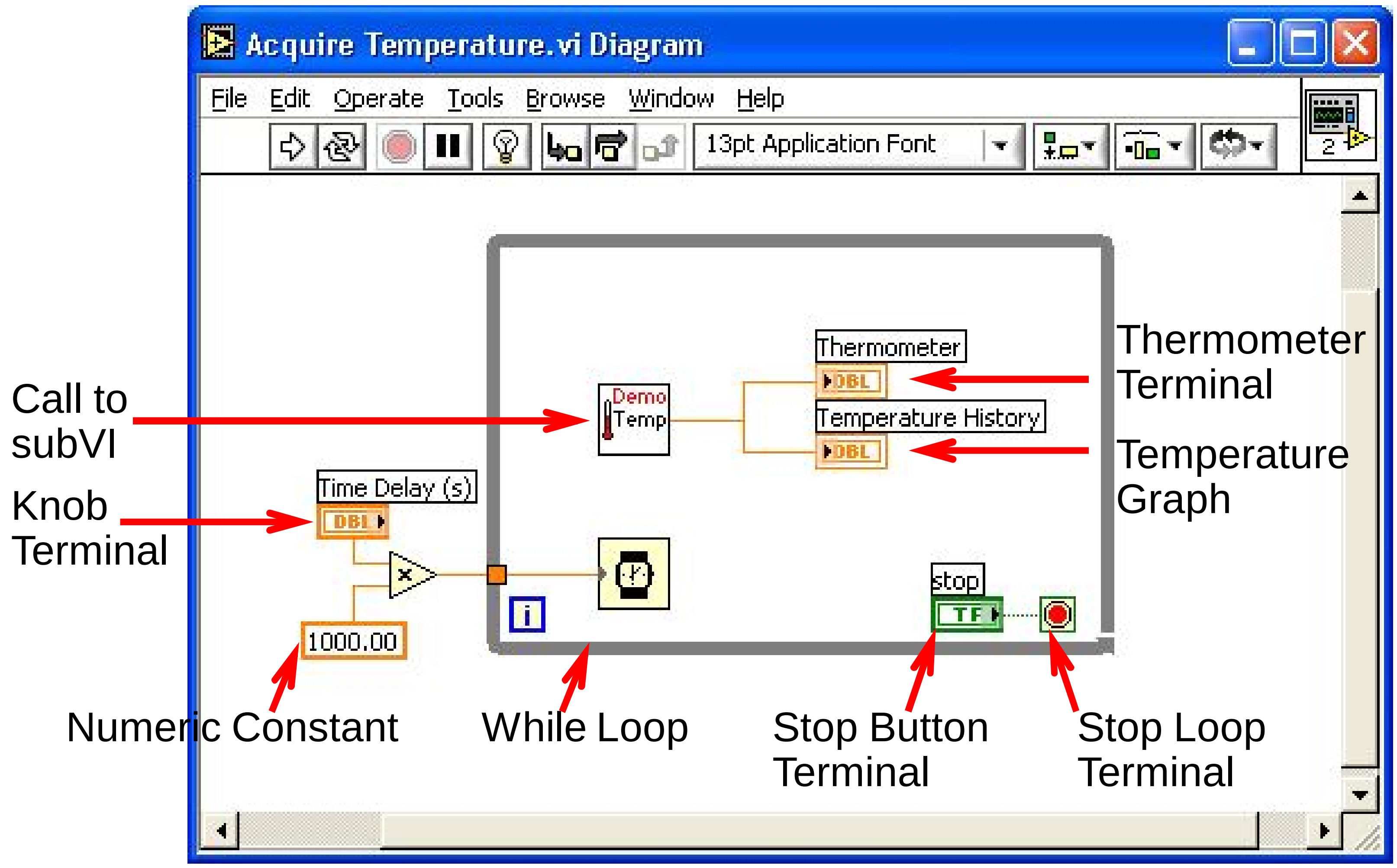
Block Diagram
- Accompanying “program” for front panel
- Components “wired” together



VI Front Panel

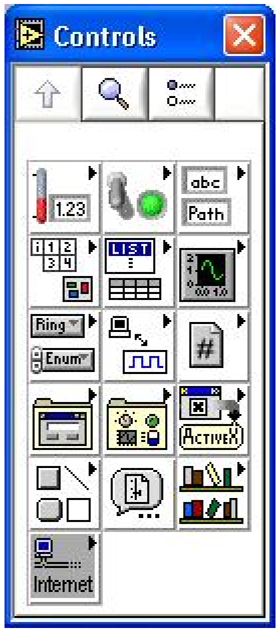


VI Block Diagram

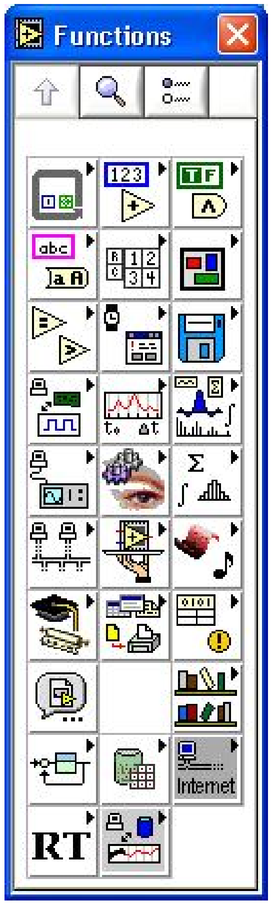


Controls and Functions Palettes

Controls Palette
(Panel Window)



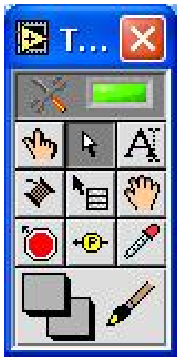
Functions Palette
(Diagram Window)



Graphical, floating palettes

Used to place controls & indicators on the front panel, or to build the block diagram

Tools Palette



- Floating Palette
- Used to operate and modify front panel and block diagram objects.

Automatic Selection Tool

Operating Tool

Positioning/Resizing Tool

Labeling Tool

Wiring Tool

Shortcut Menu Tool

Scrolling Tool

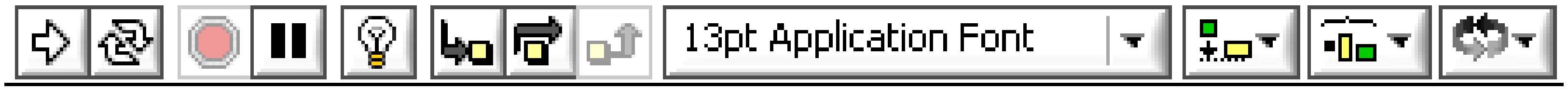
Breakpoint Tool

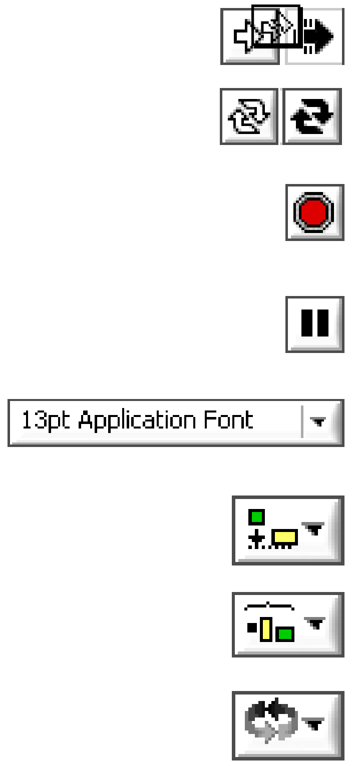
Probe Tool

Color Copy Tool

Coloring Tool

Status Toolbar





Run Button

Continuous Run Button

Abort Execution

Pause/Continue Button

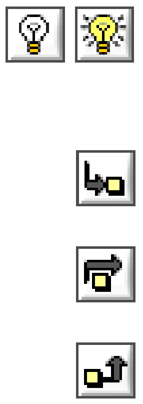
Text Settings

Align Objects

Distribute Objects

Reorder

Additional Buttons on the Diagram Toolbar



Execution Highlighting Button

Step Into Button

Step Over Button

Step Out Button

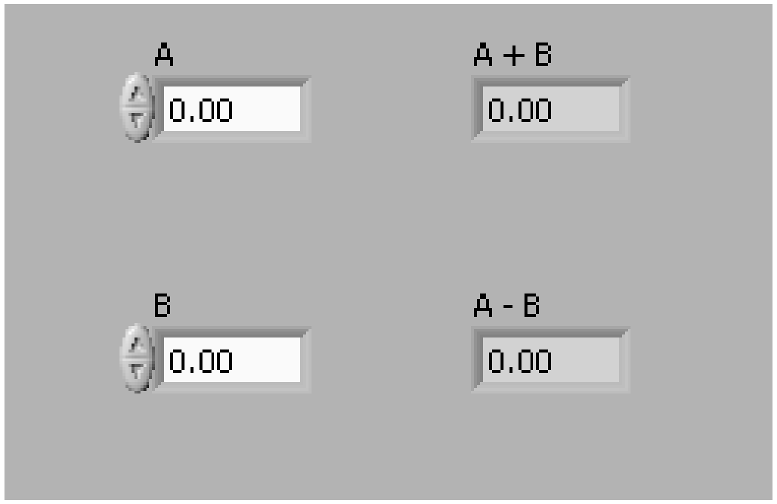
Open and Run a Virtual Instrument

Signal Generation
and Processing.vi

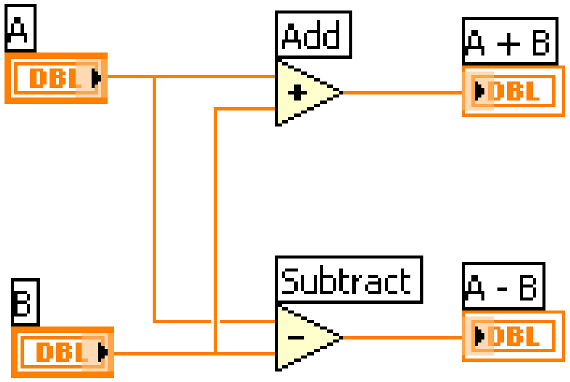
- Help » Find Examples…
- Browse According to: Task
  - » Analyzing and Processing Signals
    - » Signal Processing
      - » Signal Generation and Processing.vi

Creating a VI

Front Panel Window



Block Diagram Window



Control Terminals

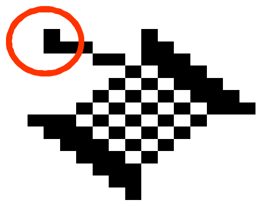
Indicator Terminals

Creating a VI – Block Diagram

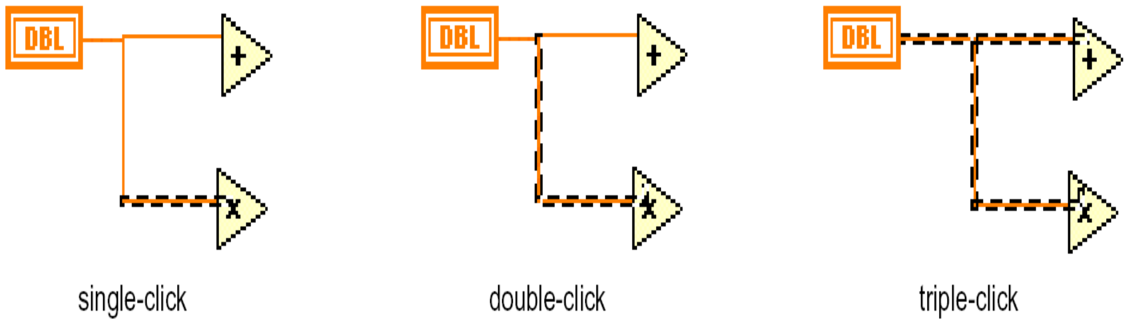
- After Creating Front Panel Controls and Indicators, Switch to Block Diagram <Ctrl-E>
- Move Front Panel Objects to Desired Locations Using the Position/Size/Select Tool
- Place Functions On Diagram
- Wire Appropriate Terminals Together to Complete the Diagram

Wiring Tips – Block Diagram

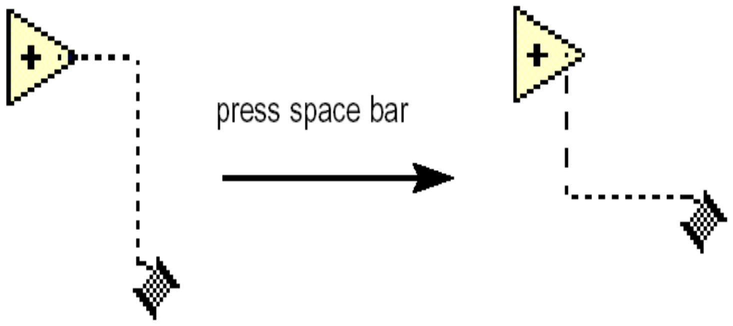
Wiring "Hot Spot"



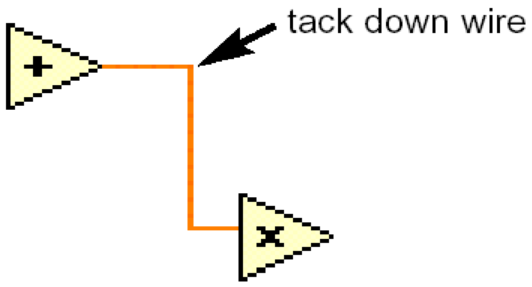
Click To Select Wires



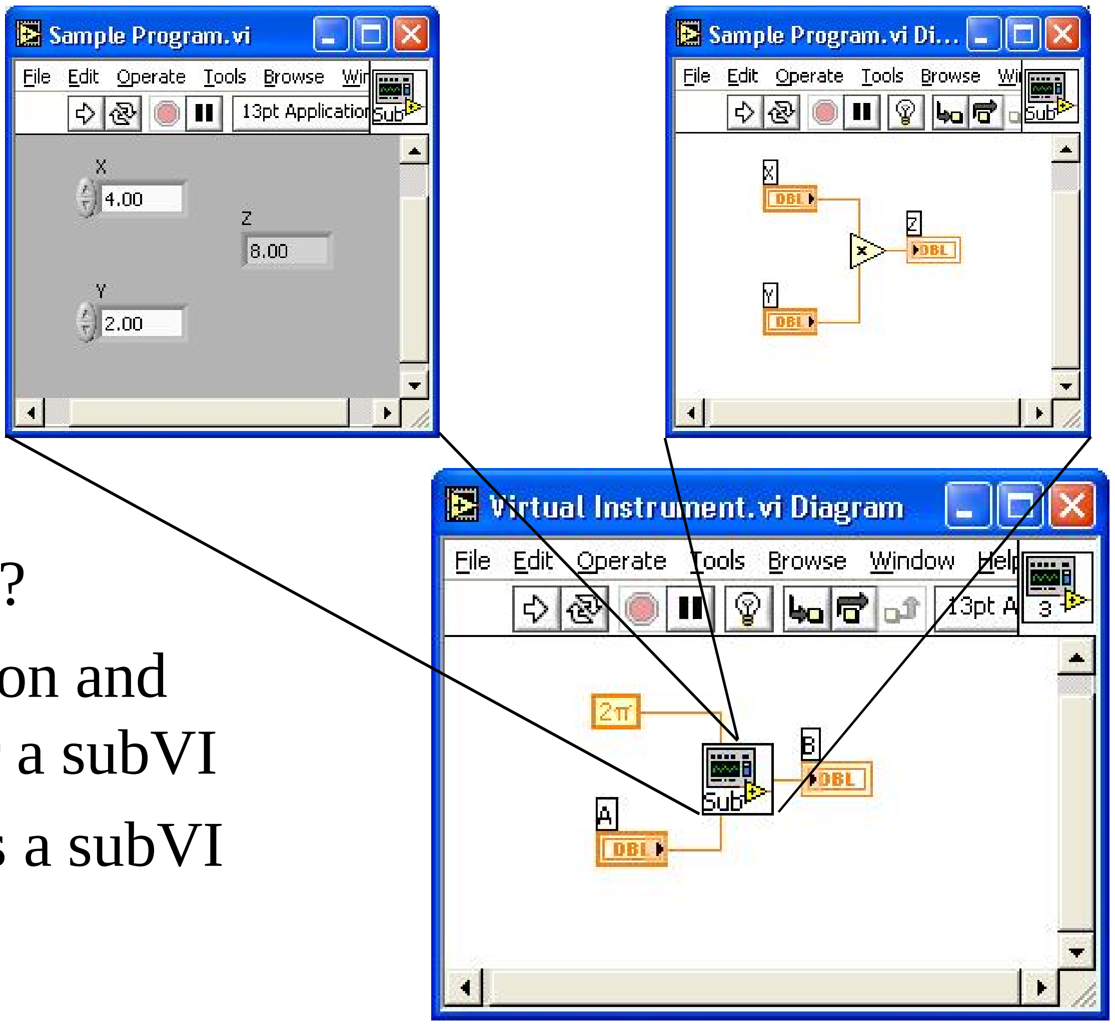
Spacebar Flips Wire Orientation



Click While Wiring To Tack Wires Down



SubVirtual Instruments


Sample Program.vi
File Edit Operate Tools Browse Wi
13pt Application
X
4.00
Z
8.00
Y
2.00
Sample Program.vi Di...
DBL
Virtual Instrument.vi Diagram
File Edit Operate Tools Browse Window Help
13pt A
2π
B
A
Sub

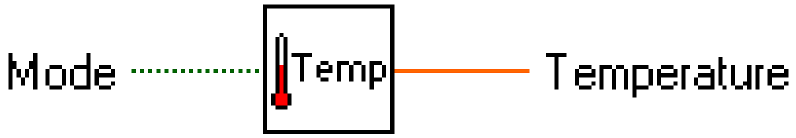
What is a subVI?

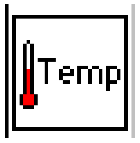
- Making an icon and connector for a subVI
- Using a VI as a subVI

SubVIs

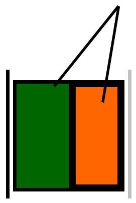
- A SubVI is a VI that can be used within another VI
- Advantages
  - Modular
  - Easier to debug
  - Don’t have to recreate code
  - Require less memory

Icon and Connector



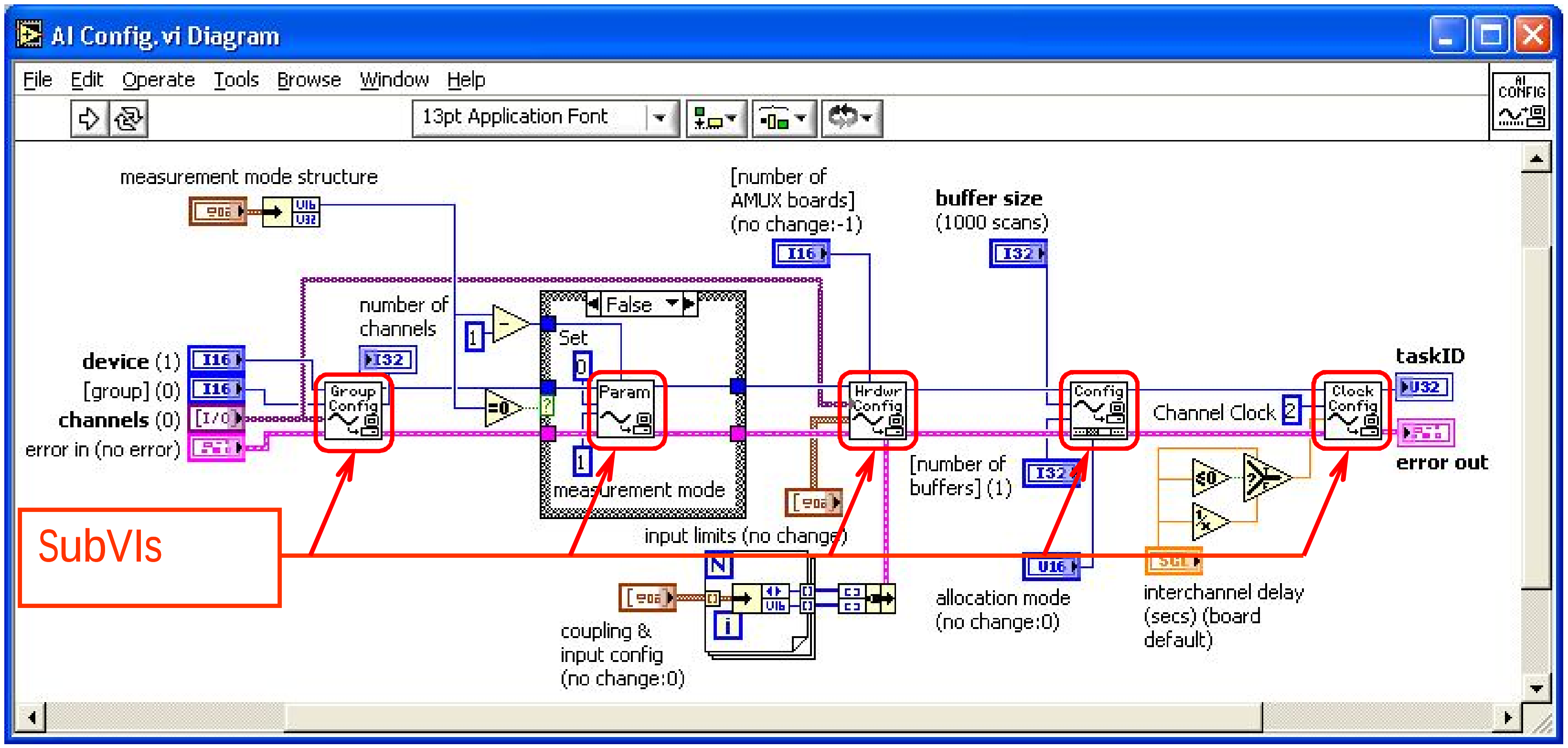


- **An icon represents a VI in other block diagrams**



- **A connector shows available terminals for data transfer**

SubVIs

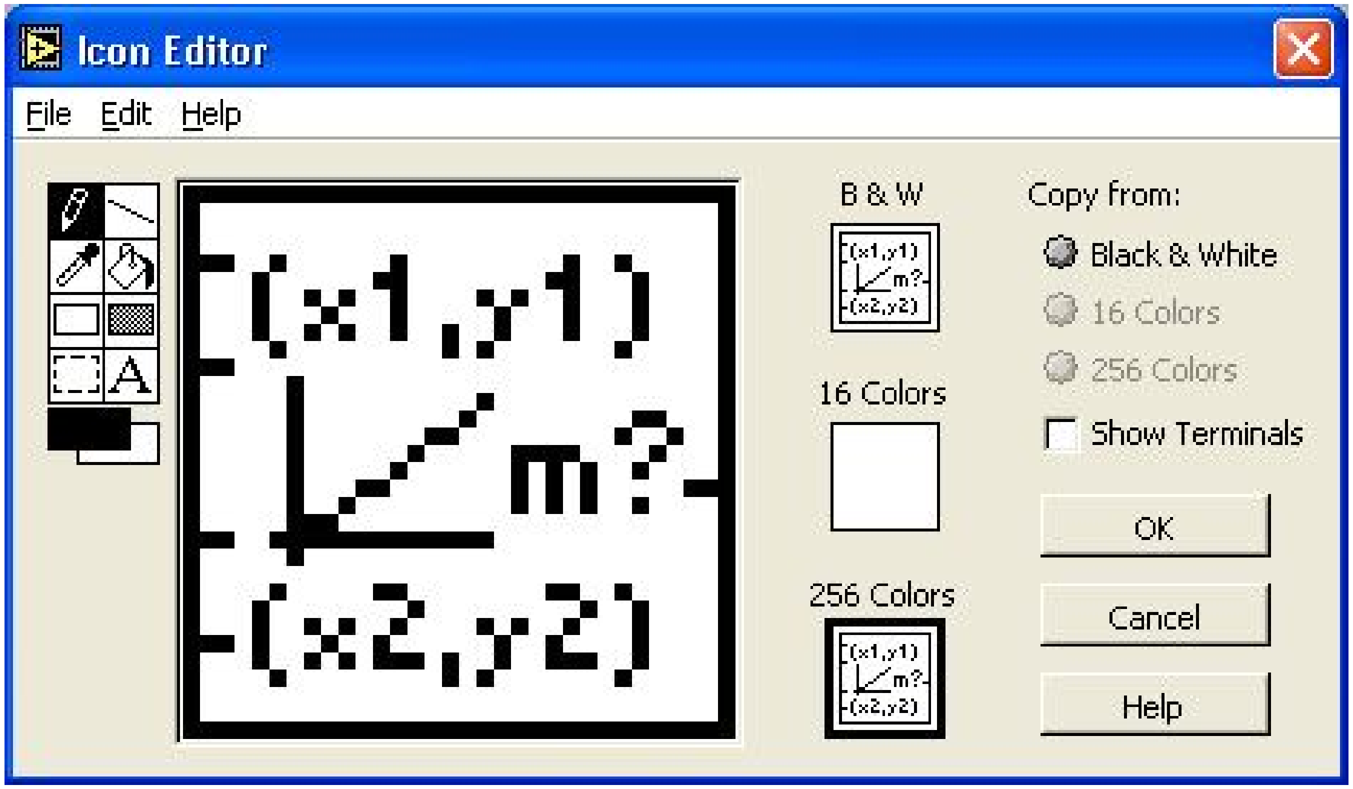

AI Config.vi Diagram
File Edit Operate Tools Browse Window Help
13pt Application Font
measurement mode structure
[number of AMUX boards] (no change:-1)
buffer size (1000 scans)
number of channels
False
Set
device (1)
[group] (0)
channels (0)
error in (no error)
Group Config
Param
Hrdwr Config
Config
Clock Config
taskID
Channel Clock
error out
measurement mode
[number of buffers] (1)
input limits (no change)
SubVIs
coupling & input config (no change:0)
allocation mode (no change:0)
interchannel delay (secs) (board default)

Steps to Create a SubVI

- Create the Icon
- Create the Connector
- Assign Terminals
- Save the VI
- Insert the VI into a Top Level VI

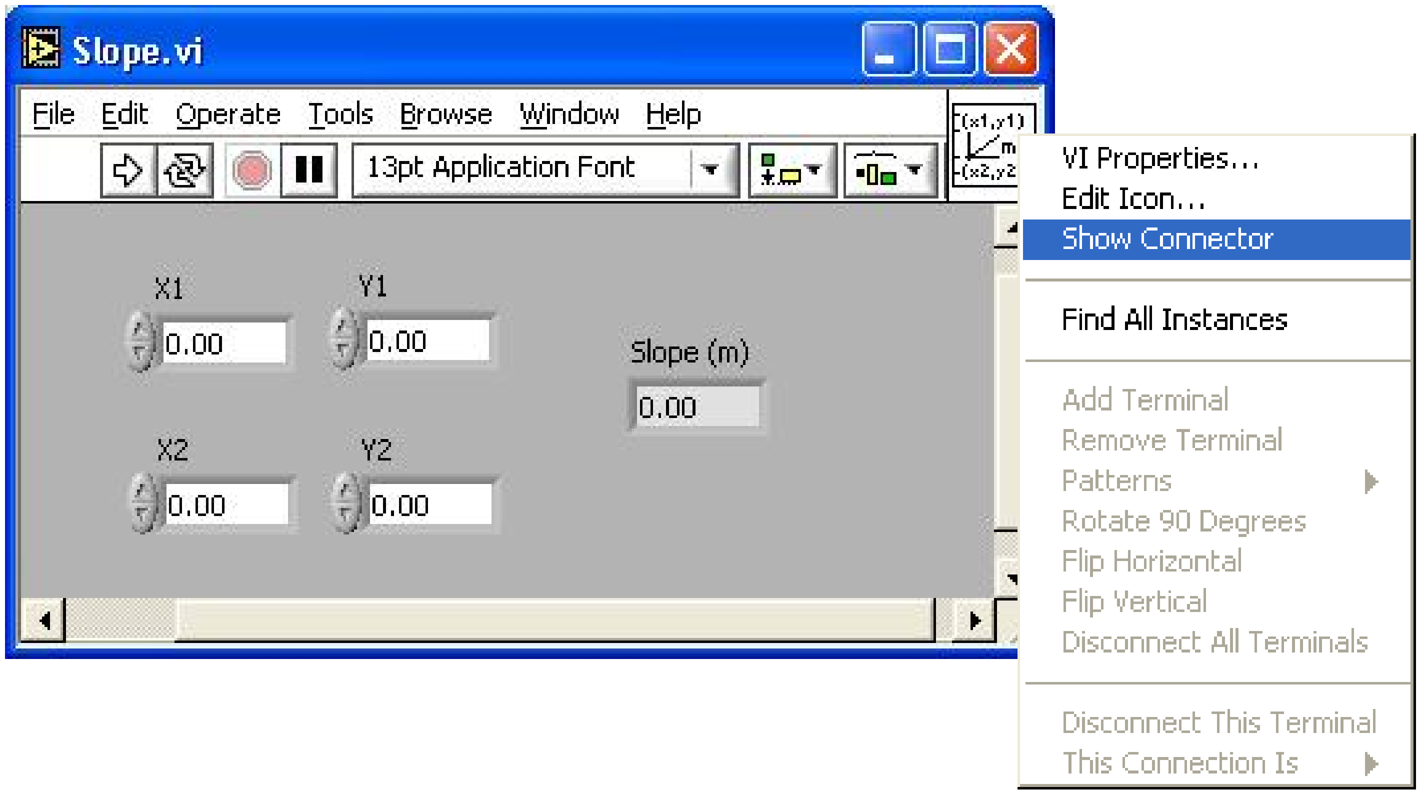
Create the Icon

- Right-click on the icon in the diagram or front panel

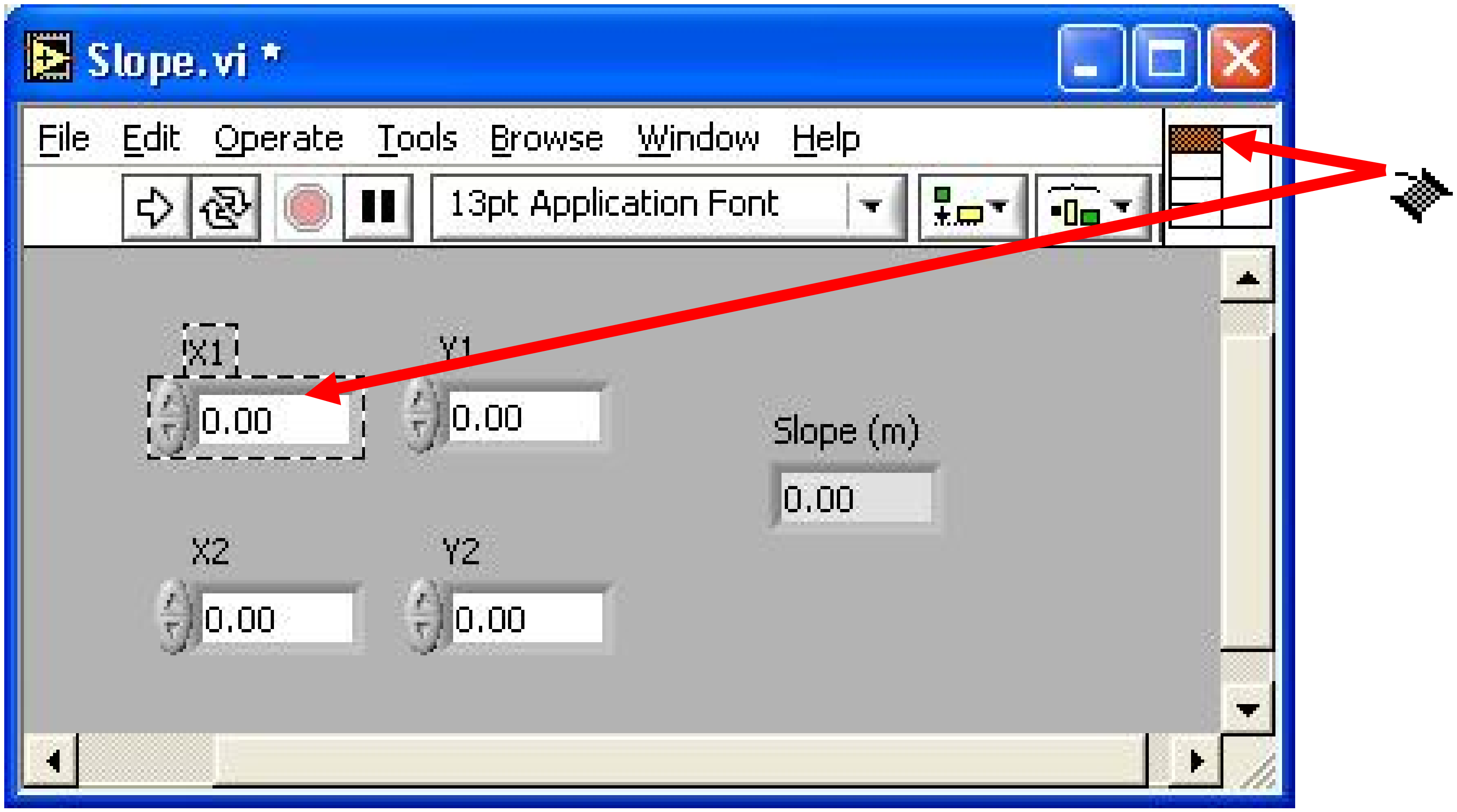


Create the Connector

Right click on the icon pane (front panel only)



Assign Terminals



Save The VI

- Choose an Easy to Remember Location
- Organize by Functionality
  - Save Similar VIs into one directory (e.g. Math Utilities)
- Organize by Application
  - Save all VIs Used for a Specific Application into one directory or library file (e.g. Lab 1 – Frequency Response)
    - Library Files (.llbs) combine many VI’s into a single file, ideal for transferring entire applications across computers

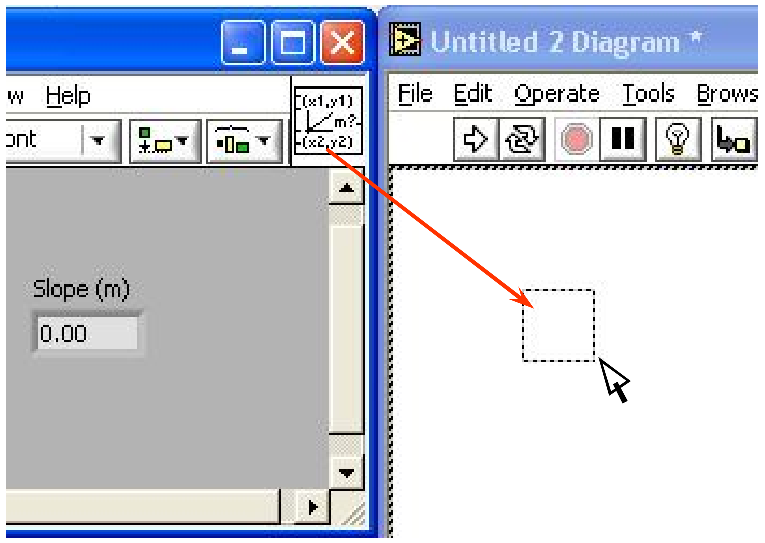
Insert the SubVI into a Top Level VI

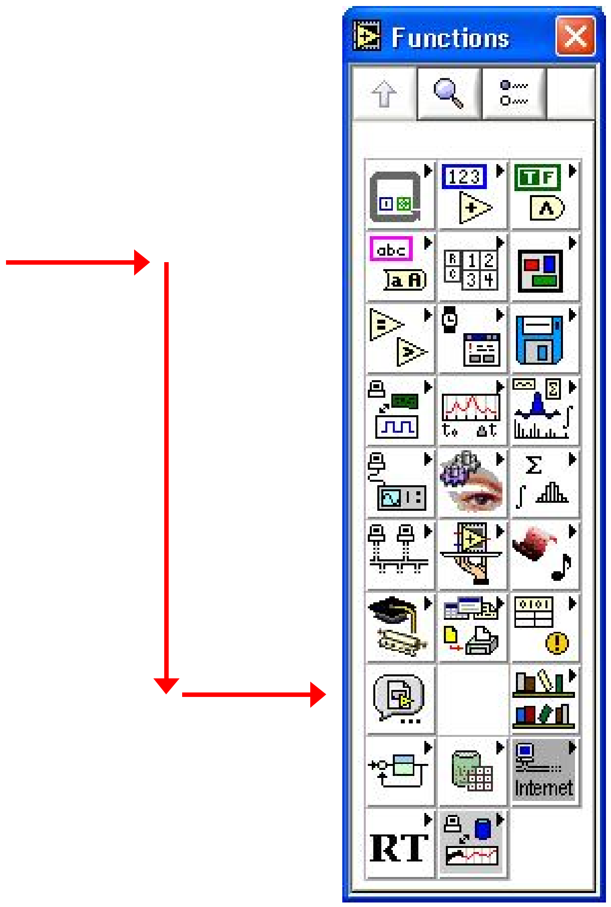
Accessing user-made subVIs

Functions >> Select a VI

Or

Drag icon onto target diagram





Tips for Working in LabVIEW

- Keystroke Shortcuts
  - <Ctrl-H> – Activate/Deactivate Context Help Window
  - <Ctrl-B> – Remove Broken Wires From Block Diagram
  - <Ctrl-E> – Toggle Between Front Panel and Block Diagram
  - <Ctrl-Z> – Undo (Also in Edit Menu)
- Tab Key – Toggle Through Tools on Toolbar
- Tools » Options… – Set Preferences in LabVIEW
- VI Properties – Configure VI Appearance, Documentation, etc.

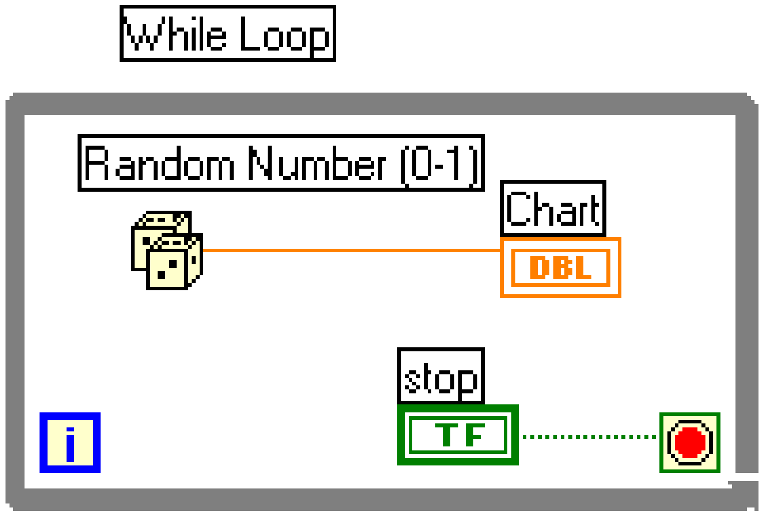
Loops and Charts

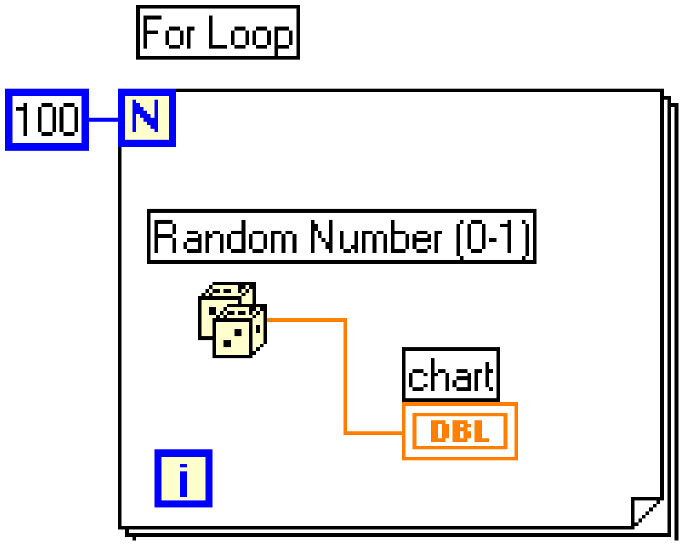
- For Loop
- While Loop
- Charts
- Multiplots

Loops

- While Loops
  - Have Iteration Terminal
  - Always Run Once
  - Run According to Continue Terminal

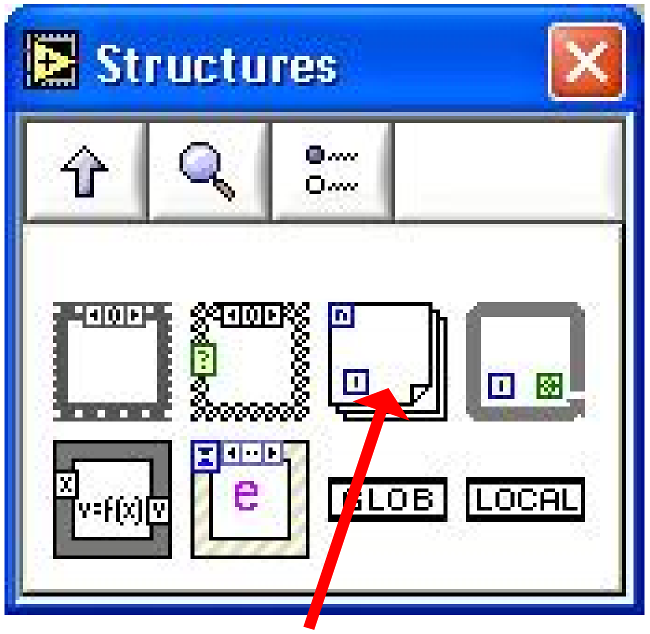
- For Loops
  - Have Iteration Terminal
  - Run According to input N



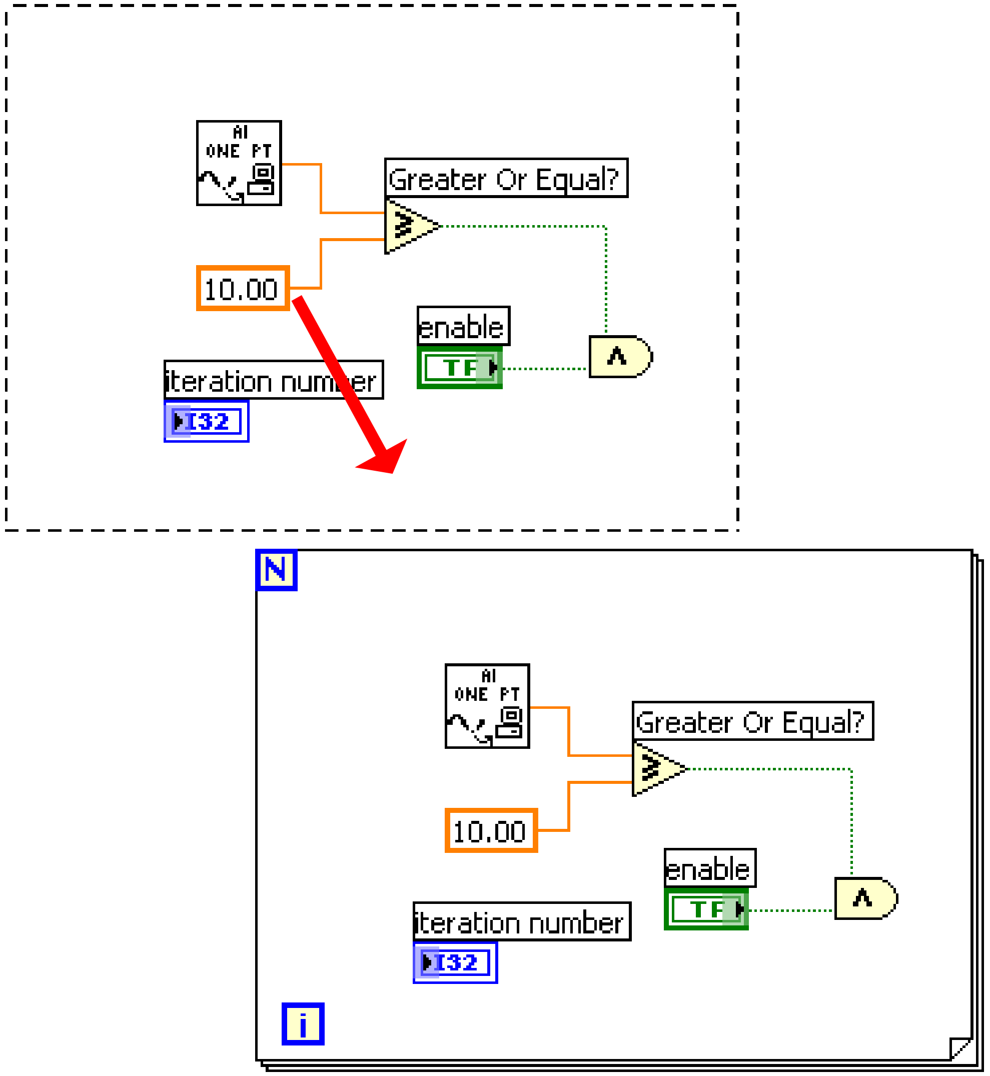


Loops (cont.)

1. Select the loop
2. Enclose Code to Repeat



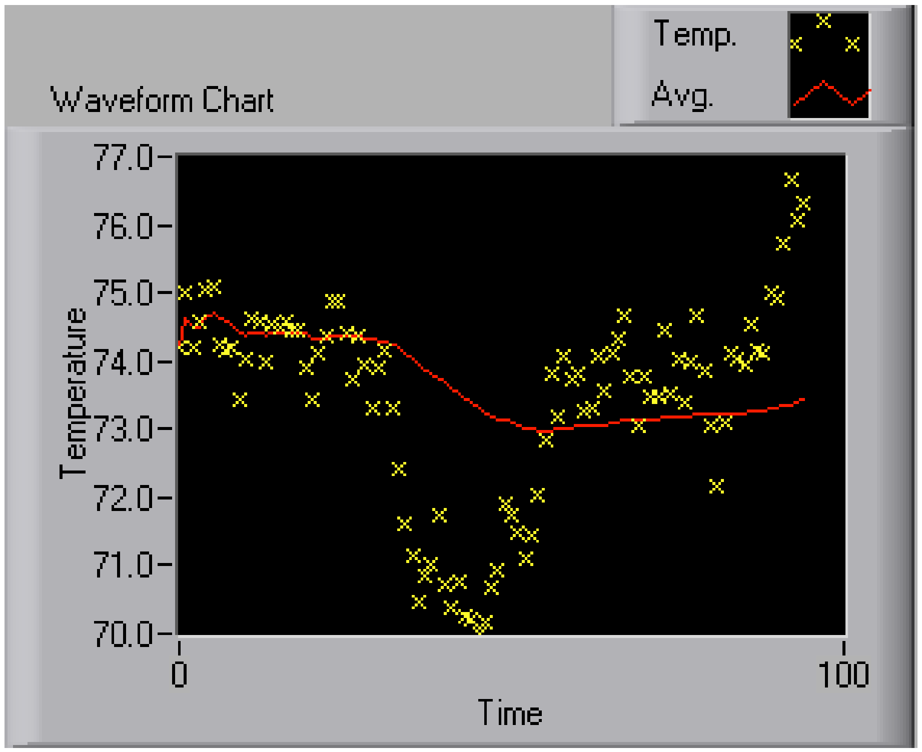
For Loop

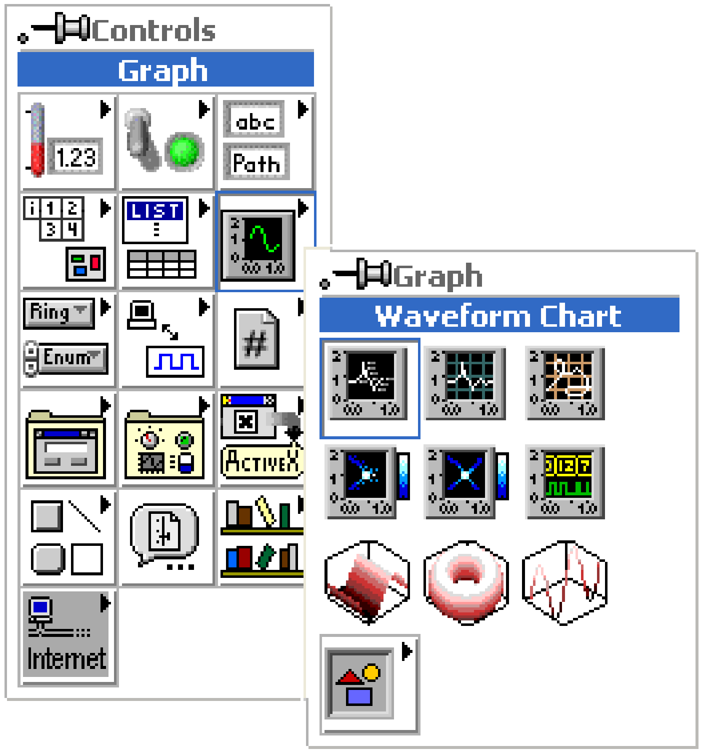


Charts

Waveform chart – special numeric indicator that can display a history of values

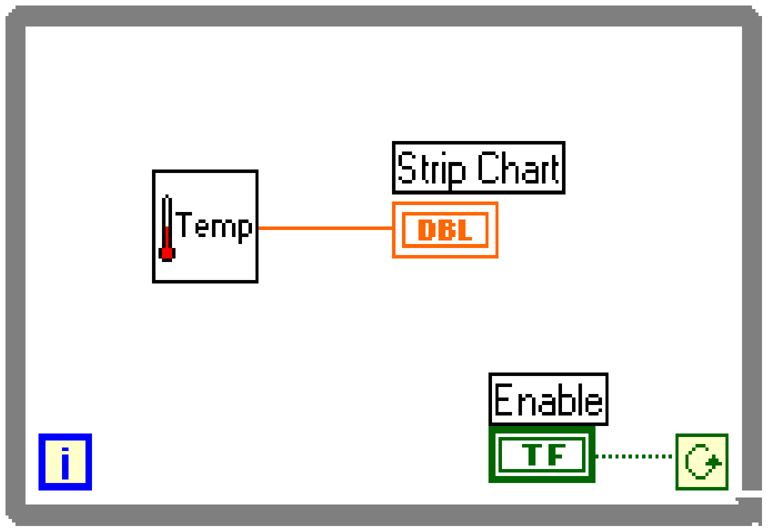
Controls >> Graphs >> Waveform Chart



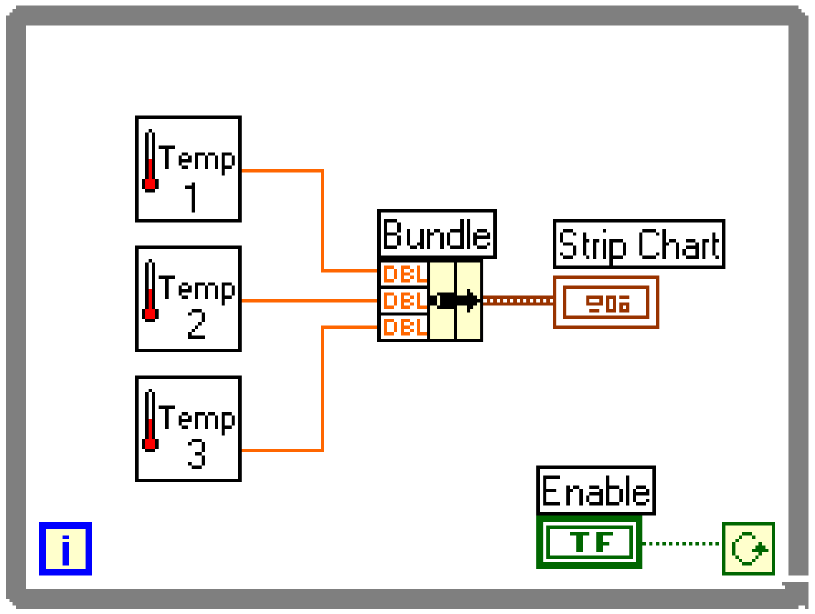


Wiring Data into Charts

Single Plot Charts

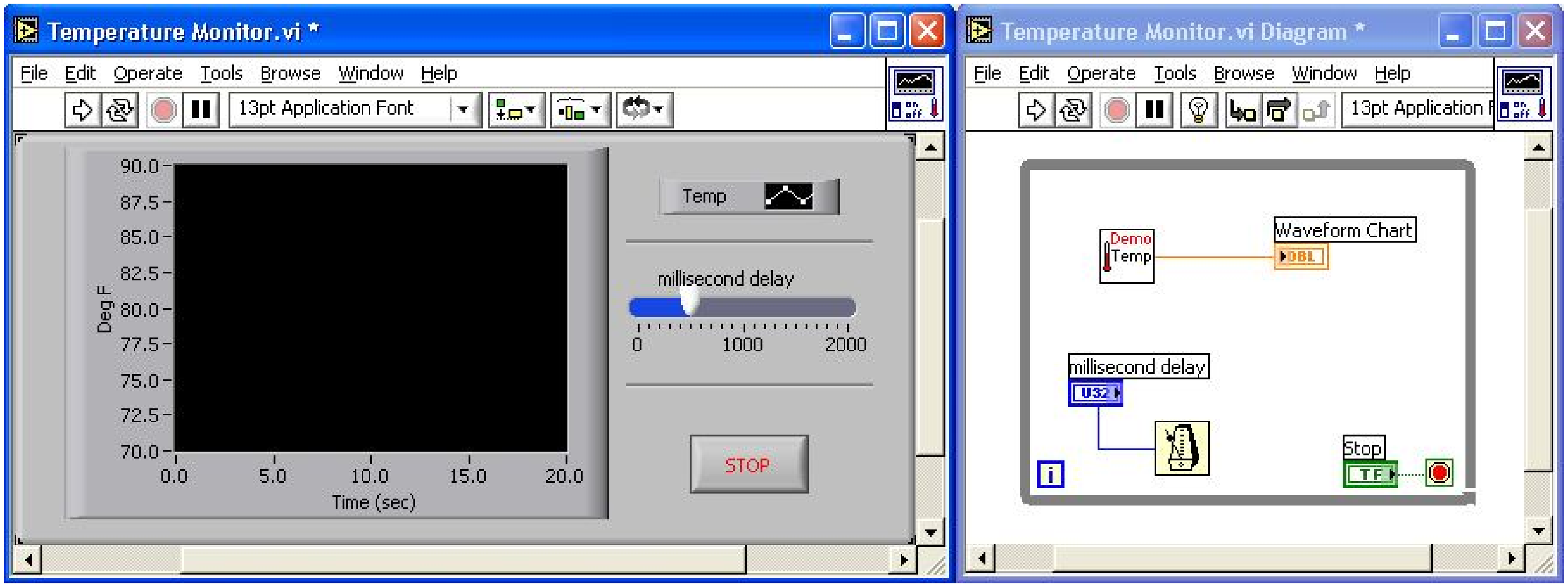


Multiplot Charts



Monitoring Temperature

Students build Temperature Monitor.vi.

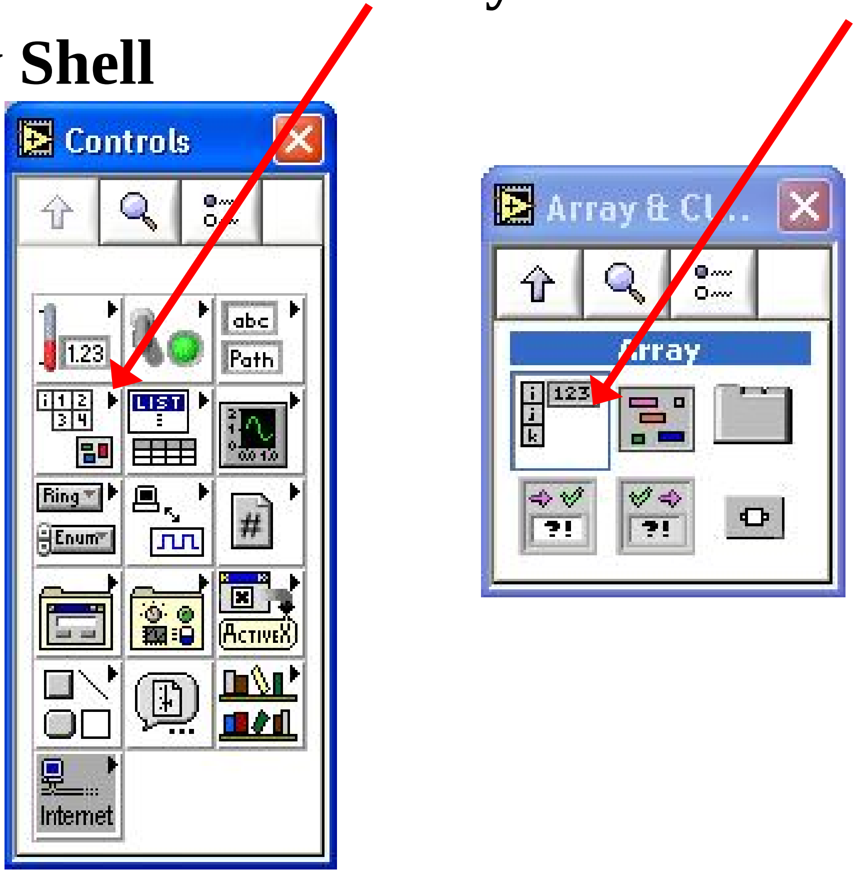


Arrays & File I/O

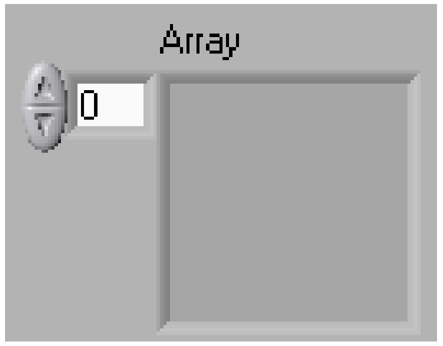
- Build arrays manually
- Have LabVIEW build arrays automatically
- Write to a spreadsheet file
- Read from a spreadsheet file

Adding an Array to the Front Panel

From the **Controls >> Array and Cluster** subpalette, select the **Array Shell**

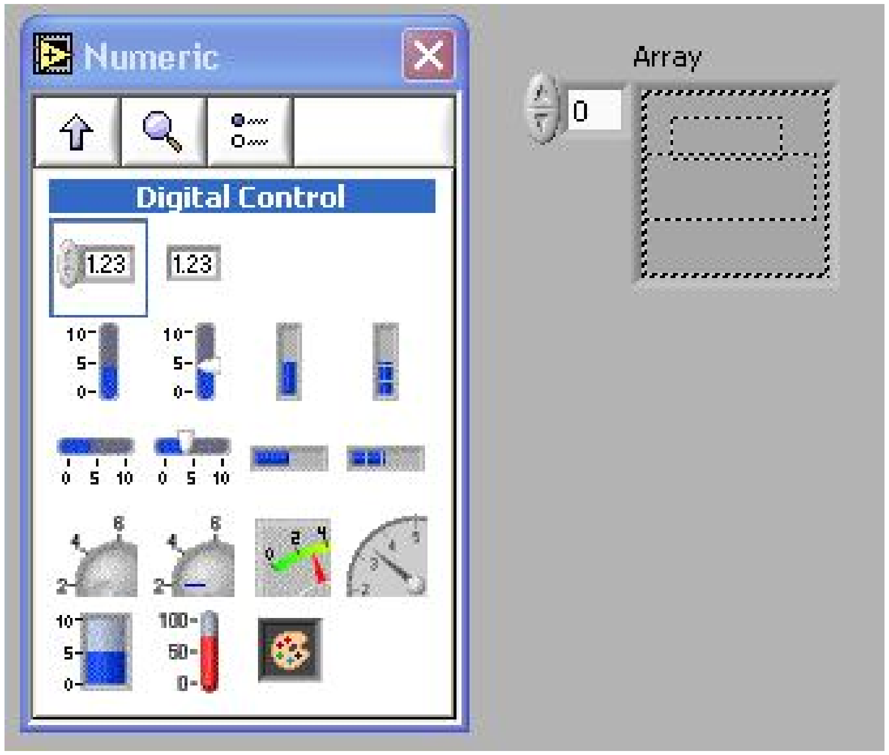


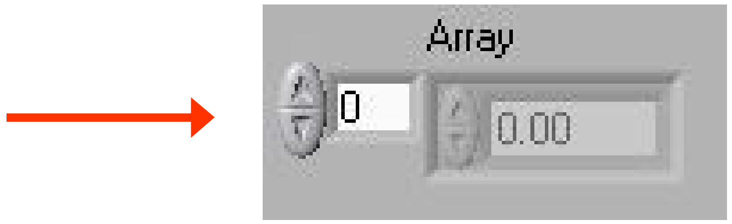
Drop it on the screen.



Adding an Array (cont.)

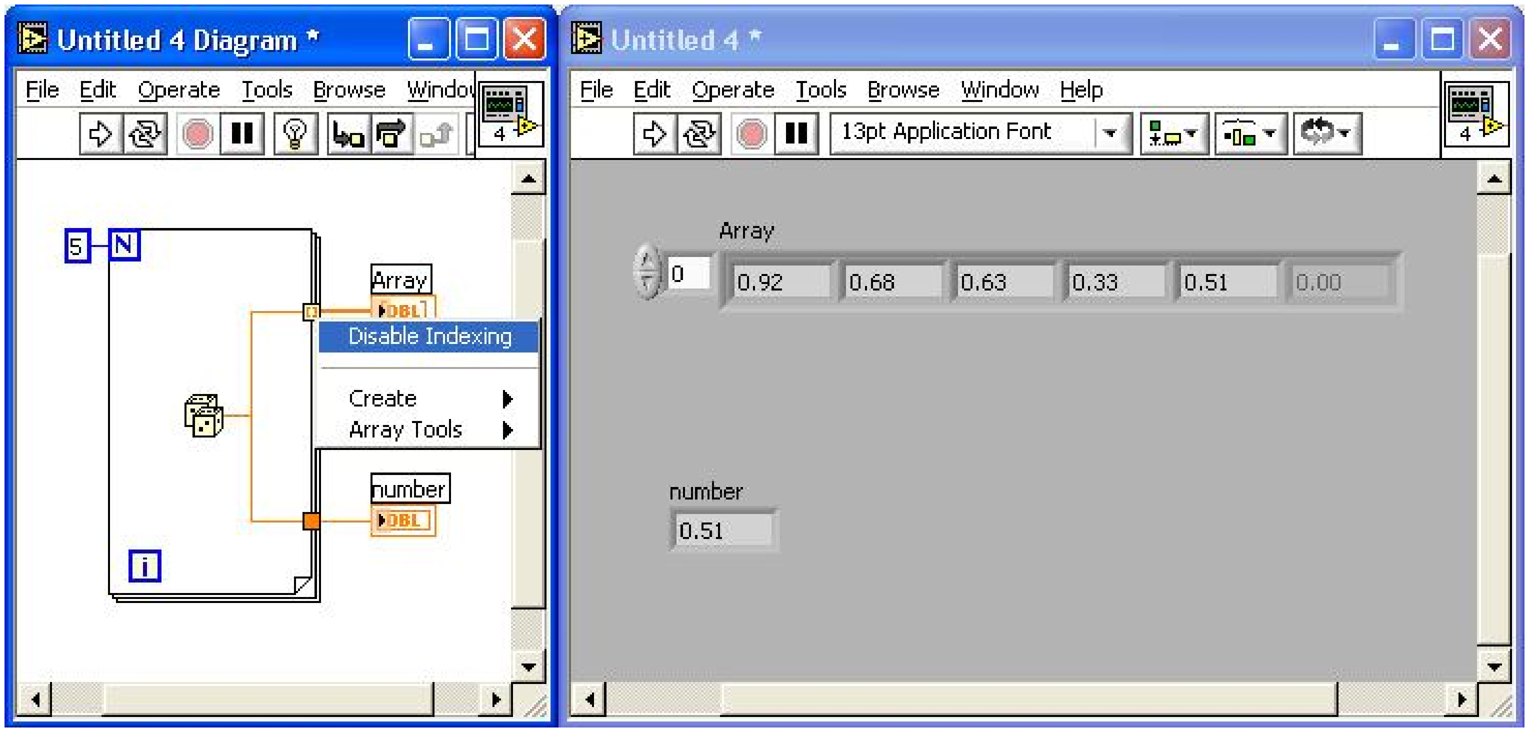
- Place data object into shell (e.g. digital control).



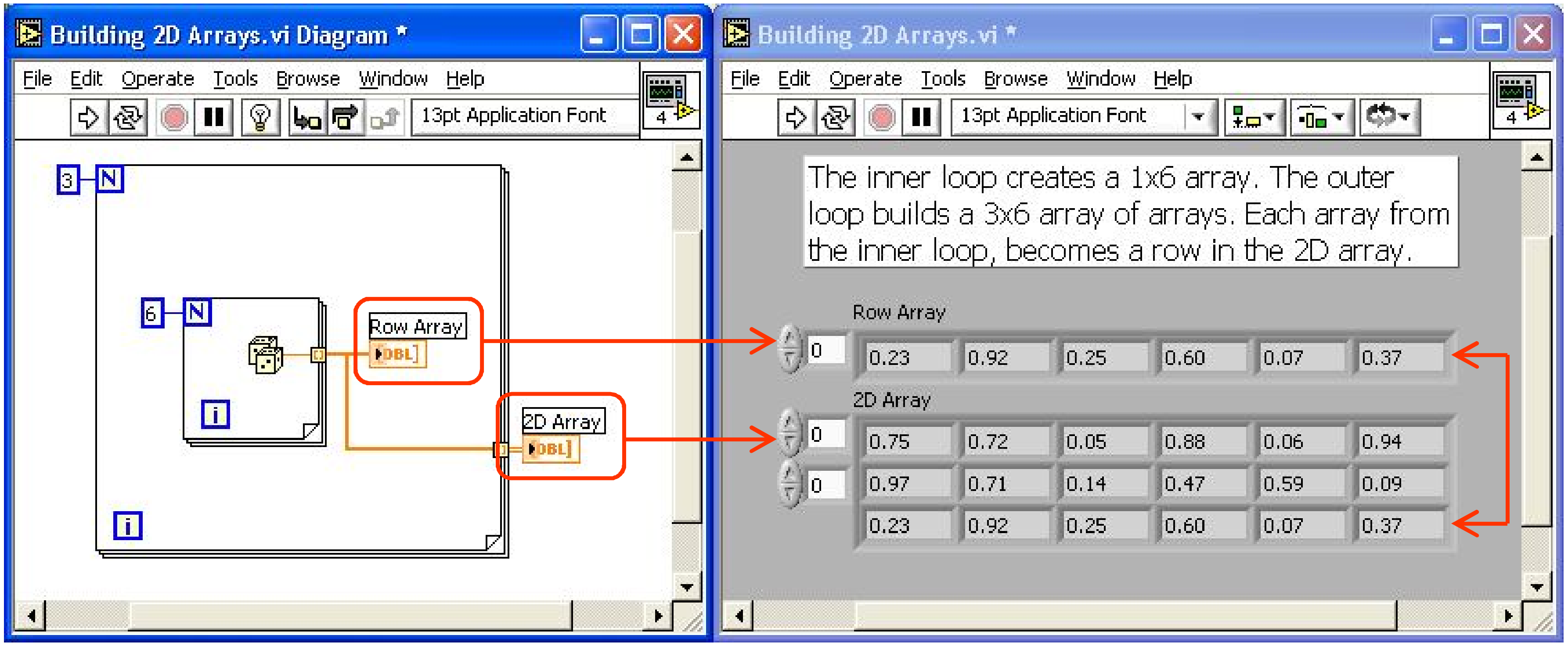


Creating an Array with a Loop

- Loops accumulate arrays at their boundaries



Creating 2D Arrays



The inner loop creates a 1x6 array. The outer loop builds a 3x6 array of arrays. Each array from the inner loop, becomes a row in the 2D array.

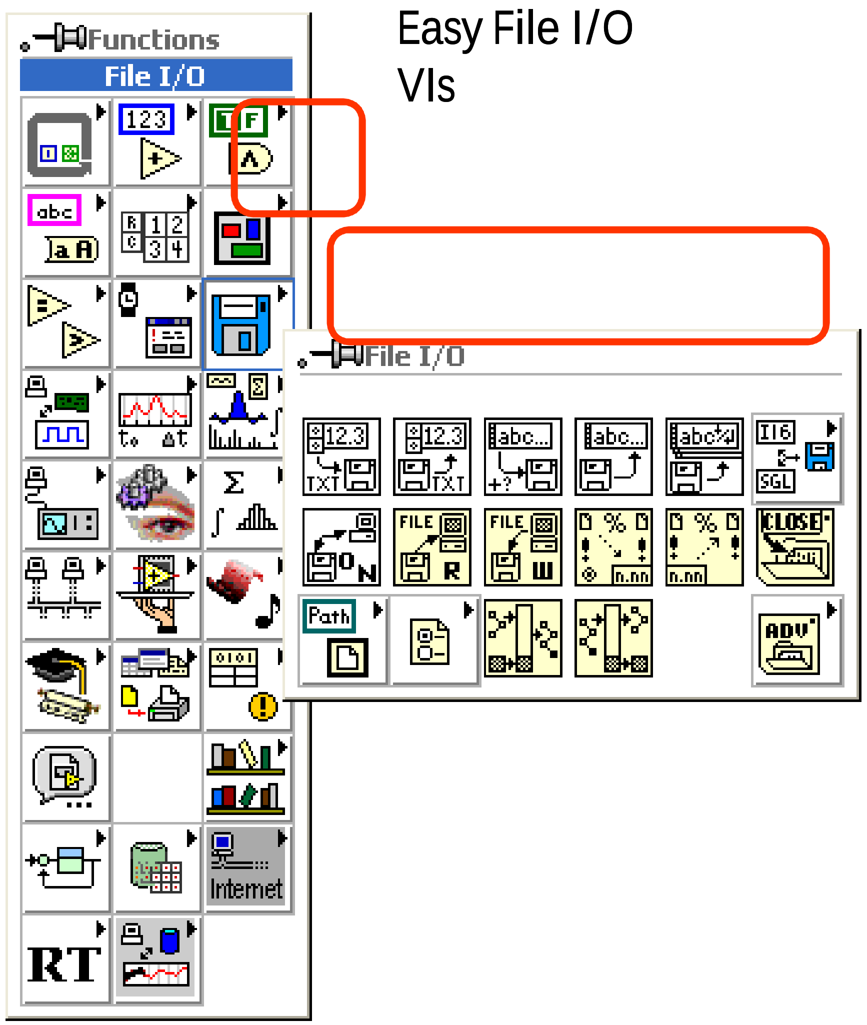
Row Array

0.23	0.92	0.25	0.60	0.07	0.37

2D Array

0.75	0.72	0.05	0.88	0.06	0.94
0.97	0.71	0.14	0.47	0.59	0.09
0.23	0.92	0.25	0.60	0.07	0.37

File I/O



•Read/write to spreadsheet file

•Read/write characters to file (ASCII)

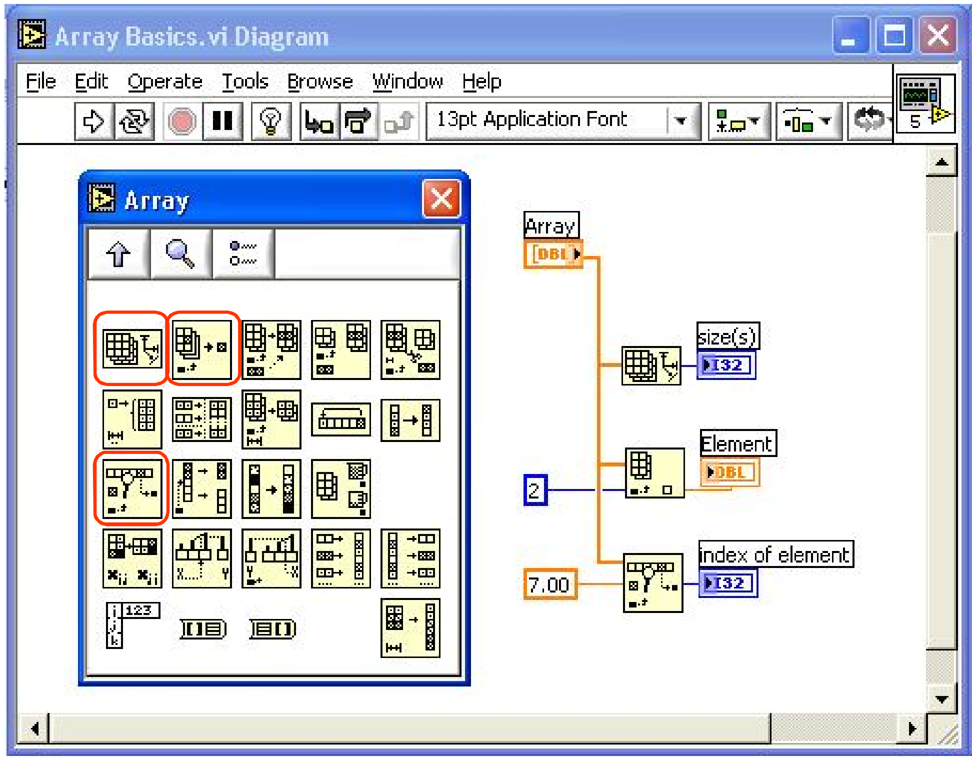
•Read lines from file

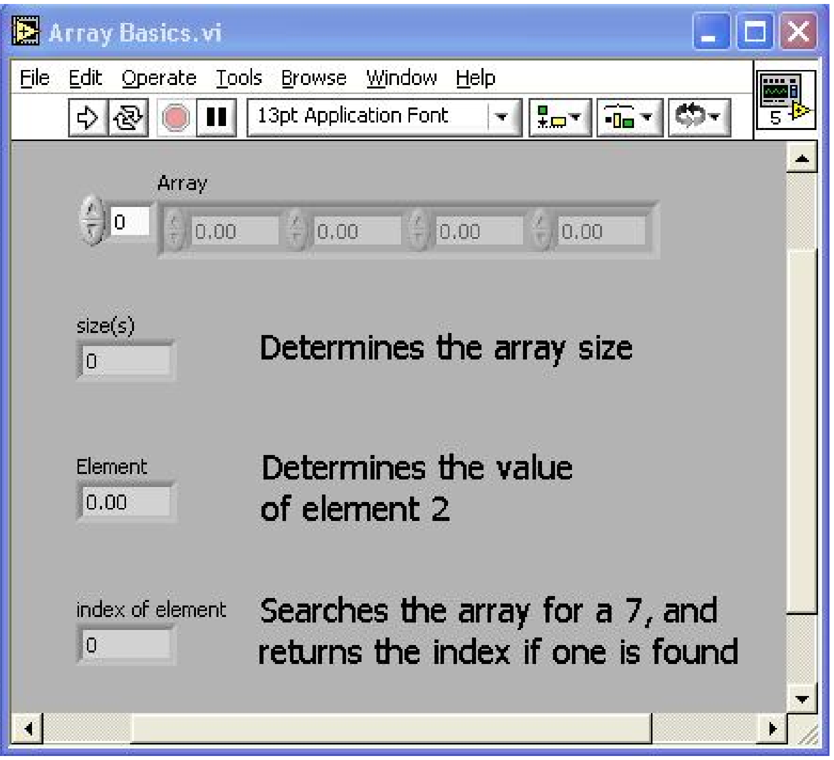
•Read/write binary file

Array Functions & Graphs

- Basic Array Functions
- Use graphs
- Create multiplots with graphs

Array Functions – Basics



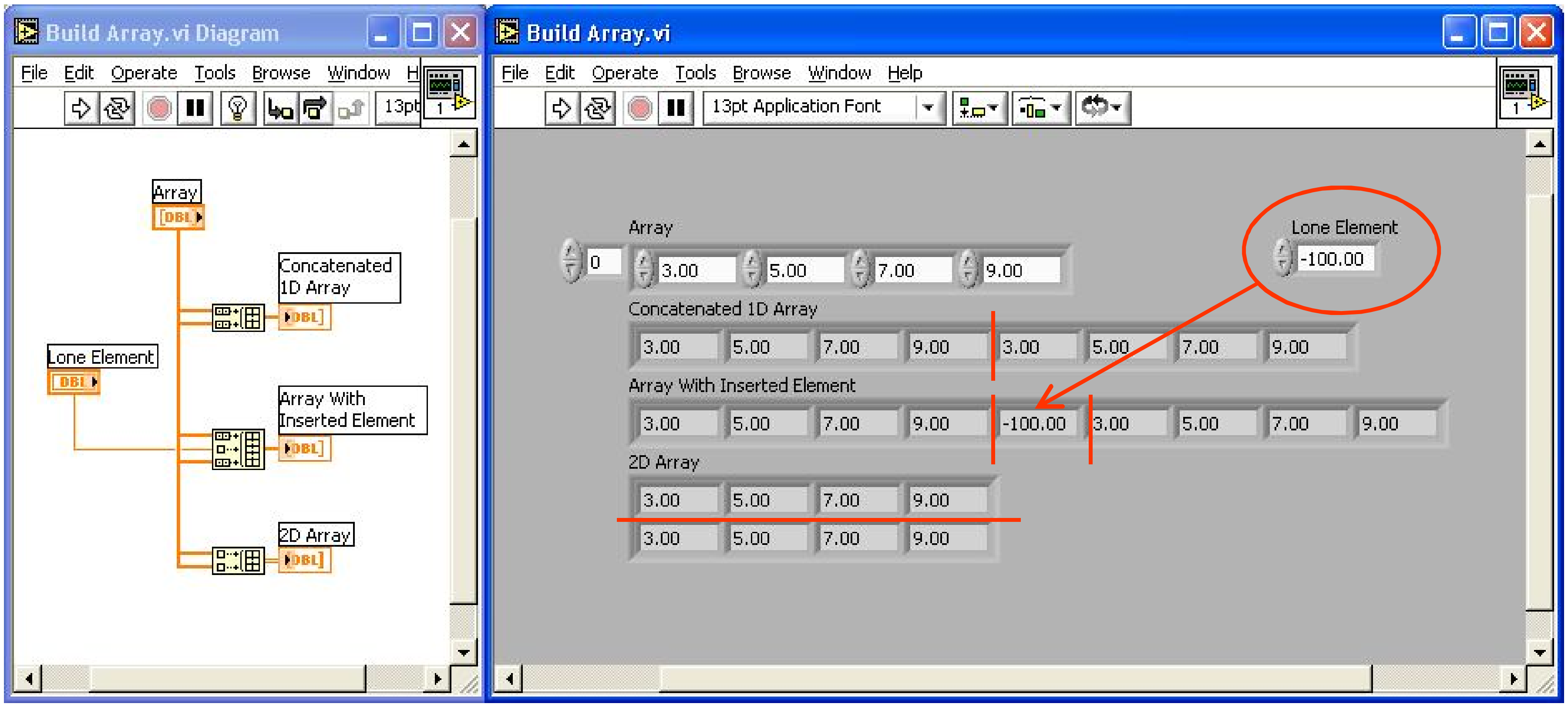


Determines the array size

Determines the value of element 2

Searches the array for a 7, and returns the index if one is found

Array Functions – Build Array



Array

0	3.00	5.00	7.00	9.00

Lone Element

-100.00

Concatenated 1D Array

3.00	5.00	7.00	9.00	3.00	5.00	7.00	9.00

Array With Inserted Element

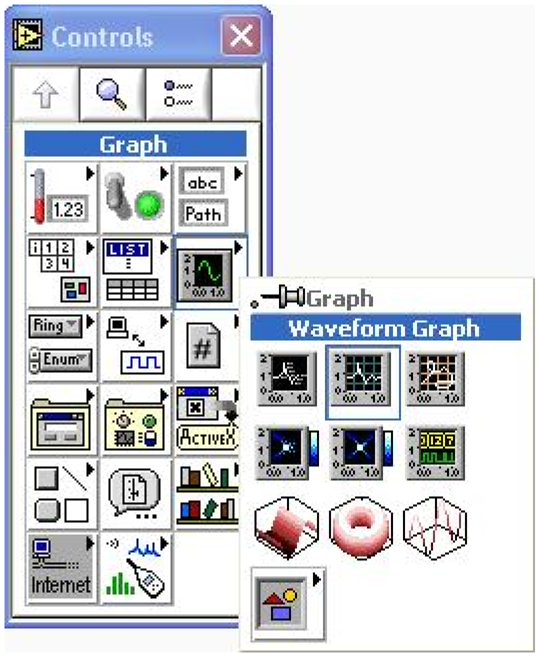
3.00	5.00	7.00	9.00	-100.00	3.00	5.00	7.00	9.00

2D Array

3.00	5.00	7.00	9.00
3.00	5.00	7.00	9.00

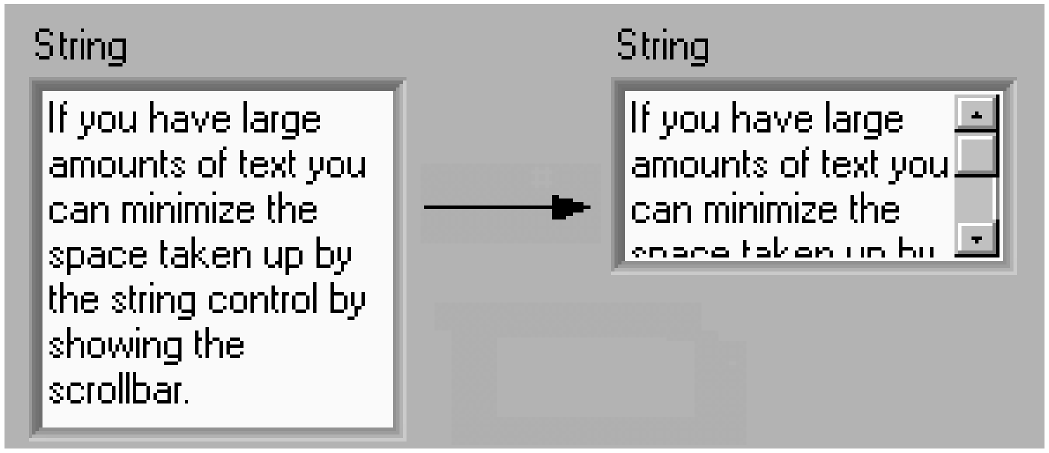
Graphs

- Selected from the Graph palette of Controls menu
  - –Waveform Graph – Plot an array of numbers against their indices
  - –XY Graph – Plot one array against another
  - –Digital Waveform Graph – Plot bits from binary data



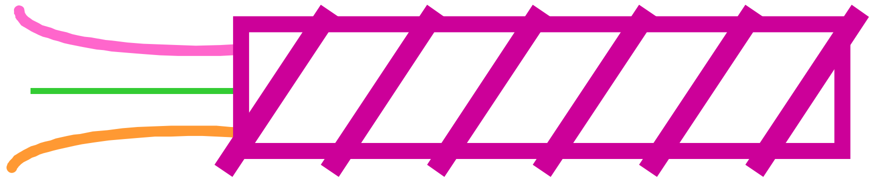
Strings

- A string is a sequence of displayable or nondisplayable characters (ASCII)
- Many uses – displaying messages, instrument control, file I/O
- String control/indicator is in the Controls»String subpalette


String
If you have large amounts of text you can minimize the space taken up by the string control by showing the scrollbar.
String
If you have large amounts of text you can minimize the

Clusters

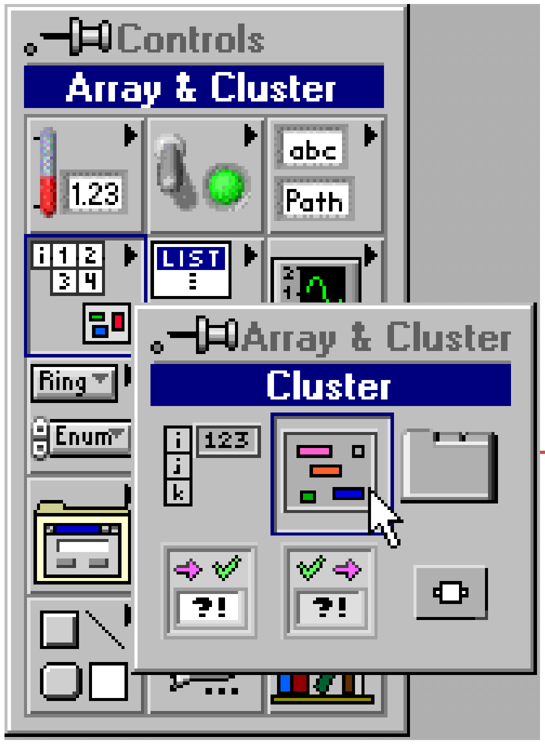
- Data structure that groups data together
- Data may be of different types
- Analogous to *struct* in C
- Elements must be either all controls or all indicators
- Thought of as wires bundled into a cable

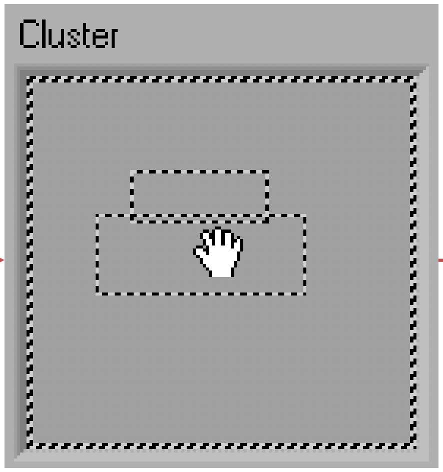


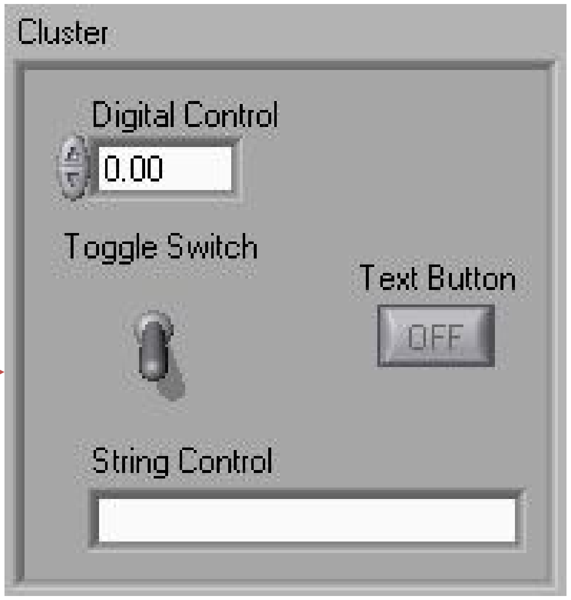
Creating a Cluster

1. Select a **Cluster** shell from the **Array & Cluster** subpalette

2. Place objects inside the shell

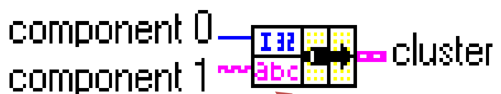






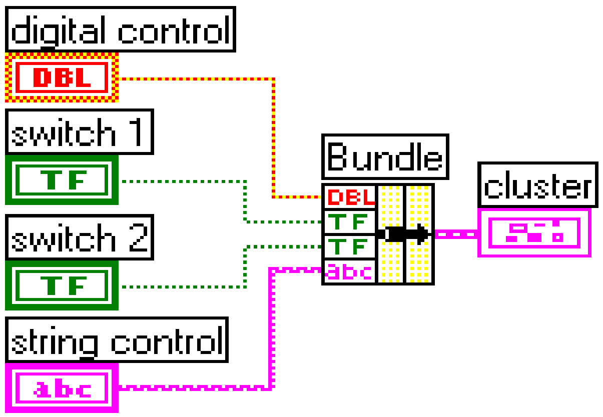
Cluster Functions

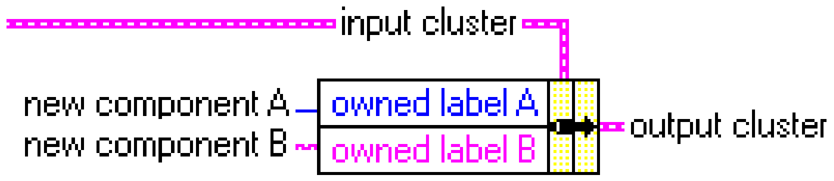
- In the Cluster subpalette of the Functions palette
- Can also be accessed by right-clicking on the cluster terminal



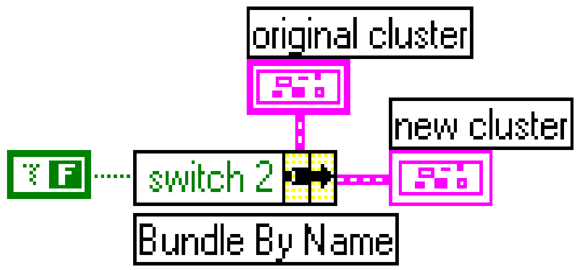
(Terminal labels reflect data type)

Bundle

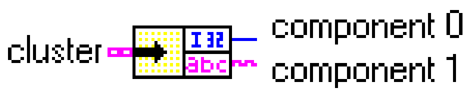




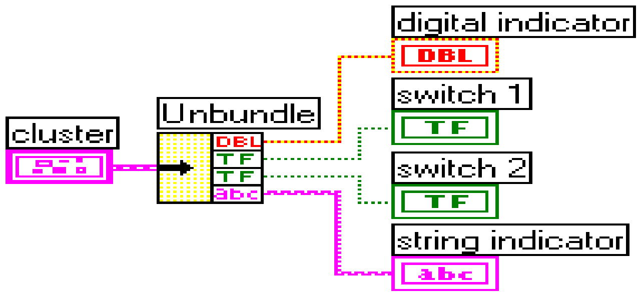
Bundle By Name

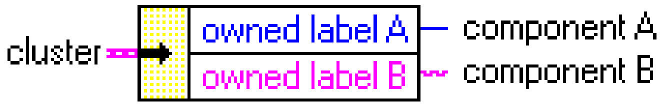


Cluster Functions

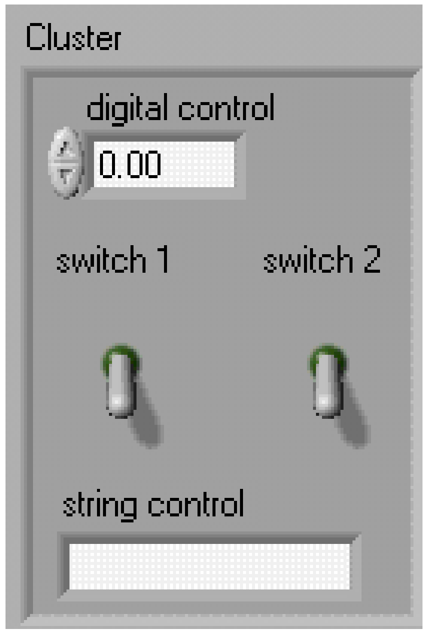


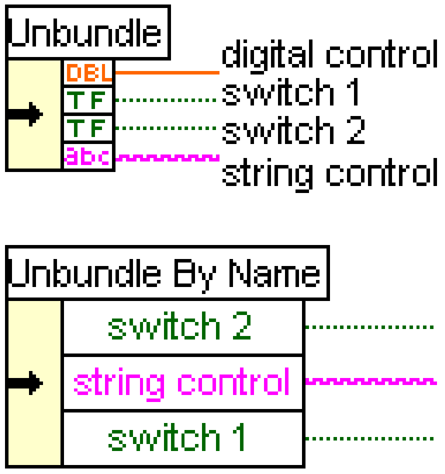
Unbundle





Unbundle By Name

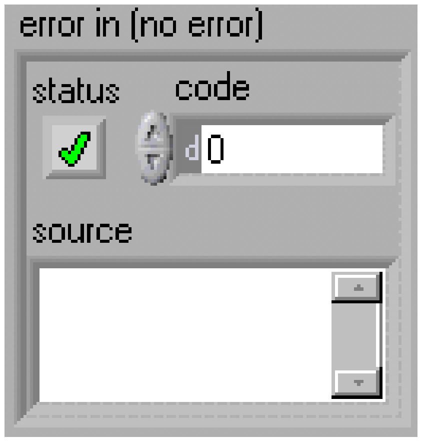


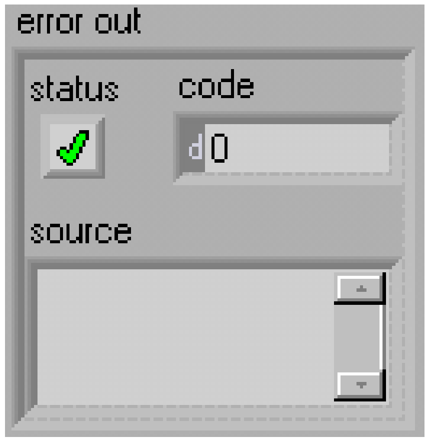


Unbundled cluster in the diagram

Error Clusters

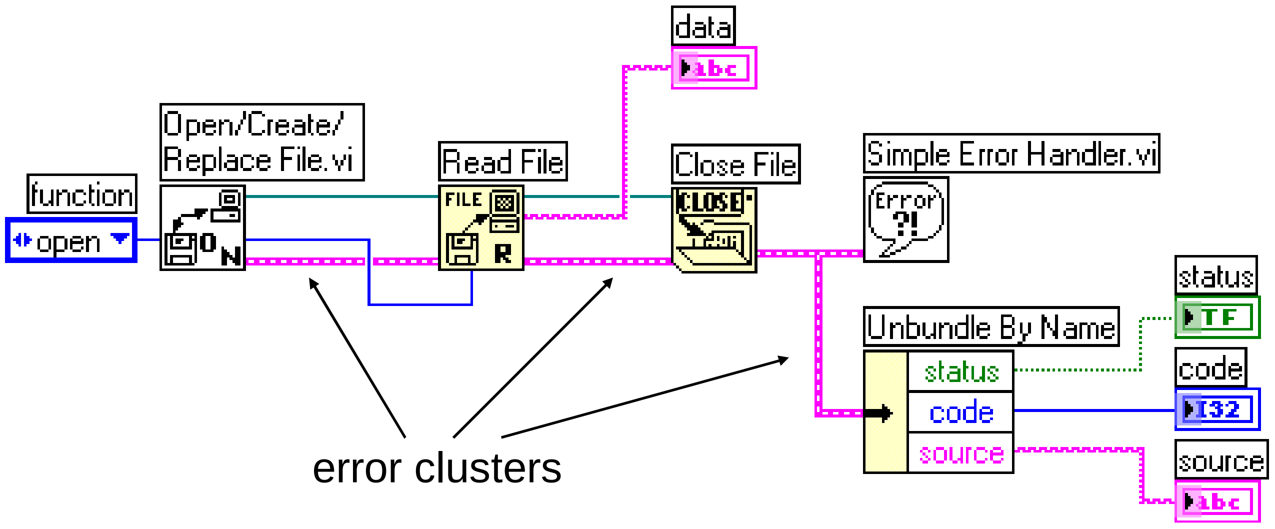
- Error cluster contains the following information:
  - Boolean to report whether error occurred
  - Integer to report a specific error code
  - String to give information about the error





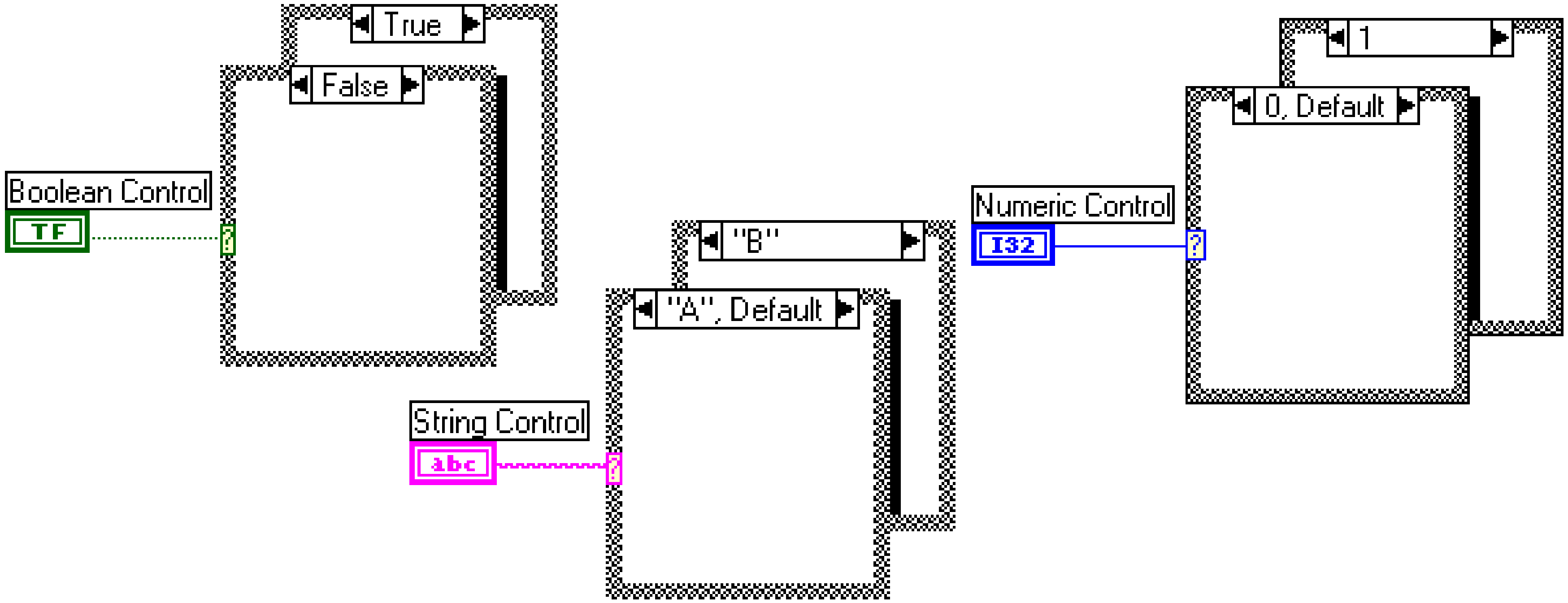
Error Handling Techniques

- Error information is passed from one subVI to the next
- If an error occurs in one subVI, all subsequent subVIs are not executed in the usual manner
- Error Clusters contain all error conditions

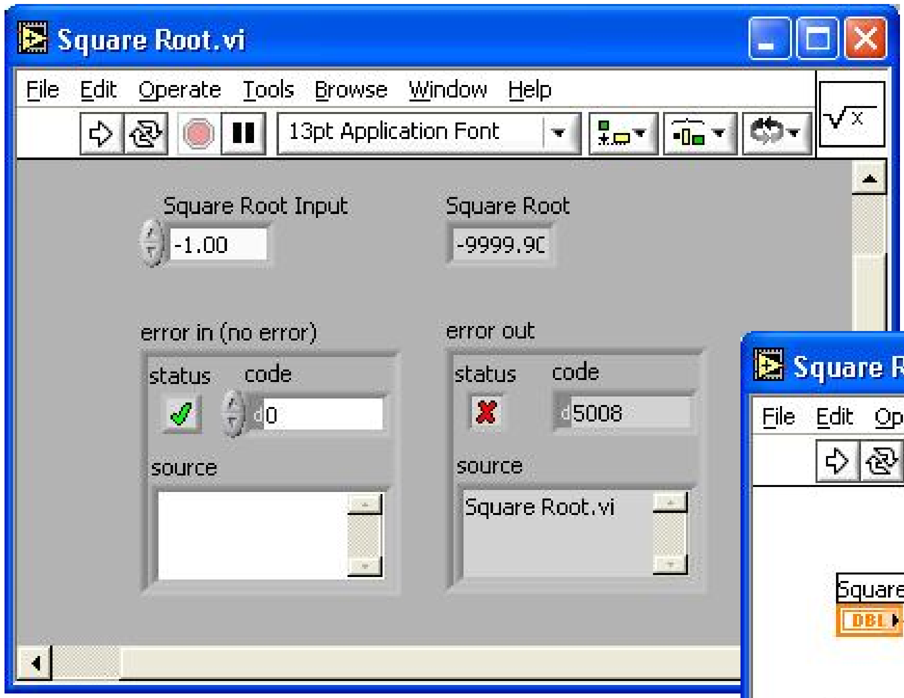


Case Structures

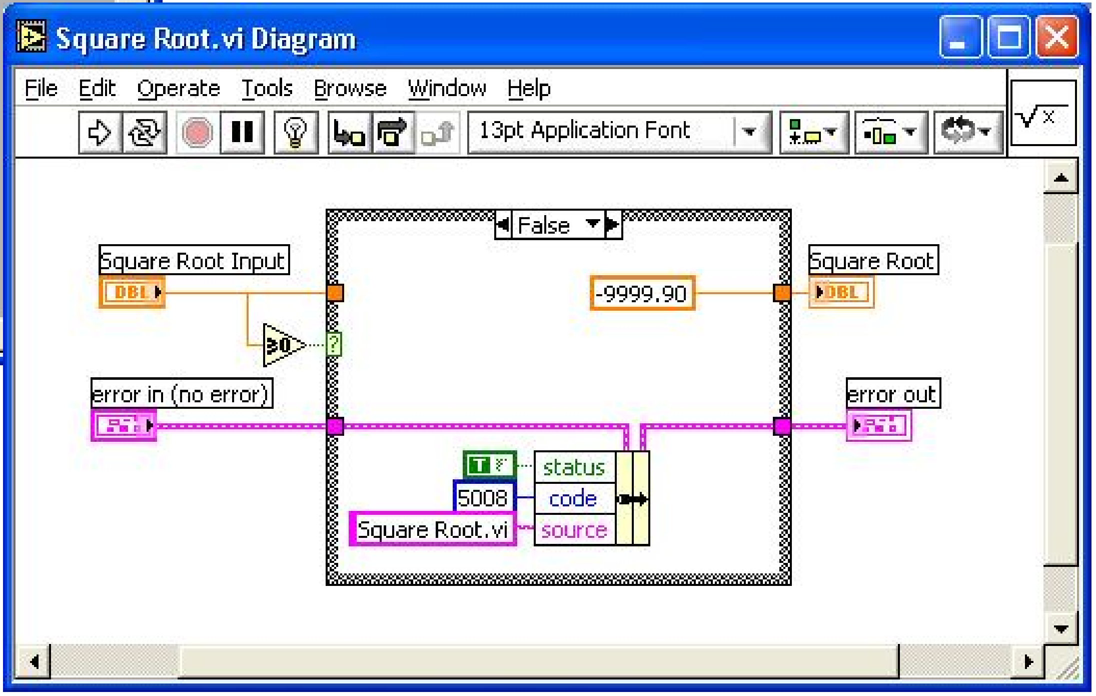
- In the Structures sub palette of Functions palette
- Enclose nodes or drag them inside the structure
- Stacked like a deck of cards, only one case visible



Error Clusters & Handling

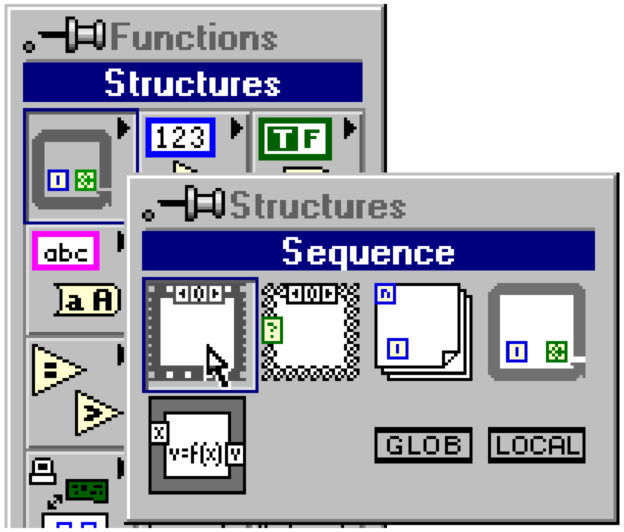


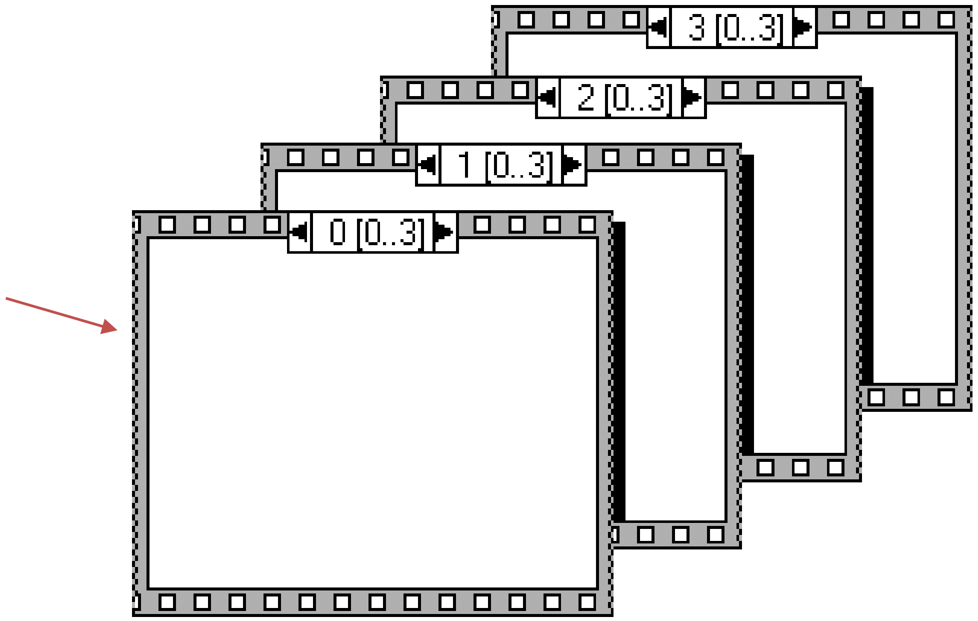
Square Root Input: -1.00
Square Root: -9999.90
error in (no error): status ✓, code 0, source
error out: status ✗, code 5008, source Square Root.vi



Sequence Structures

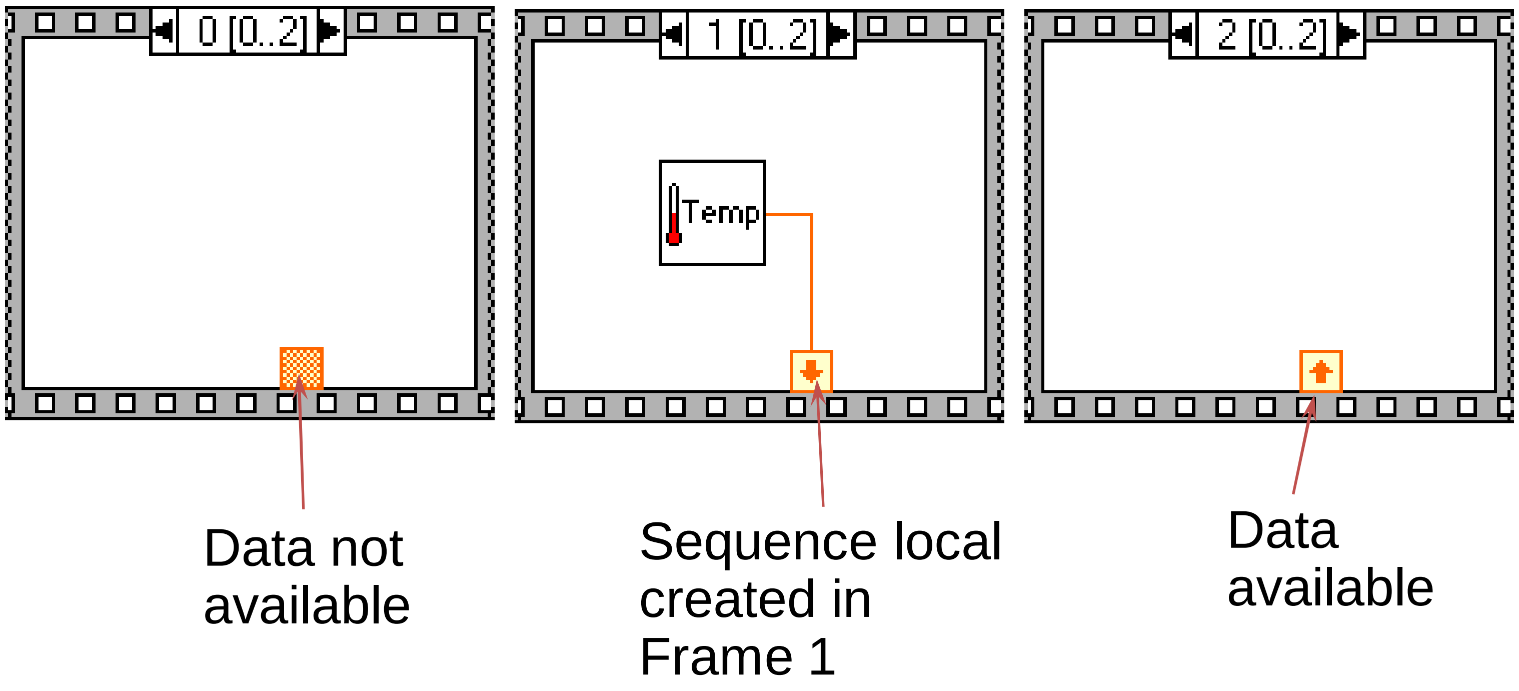
- In the Structures subpalette of Functions palette
- Executes diagrams sequentially, Frame 0 (0..x), where x is the total number of frames
- Stacked like a deck of cards, only one frame visible





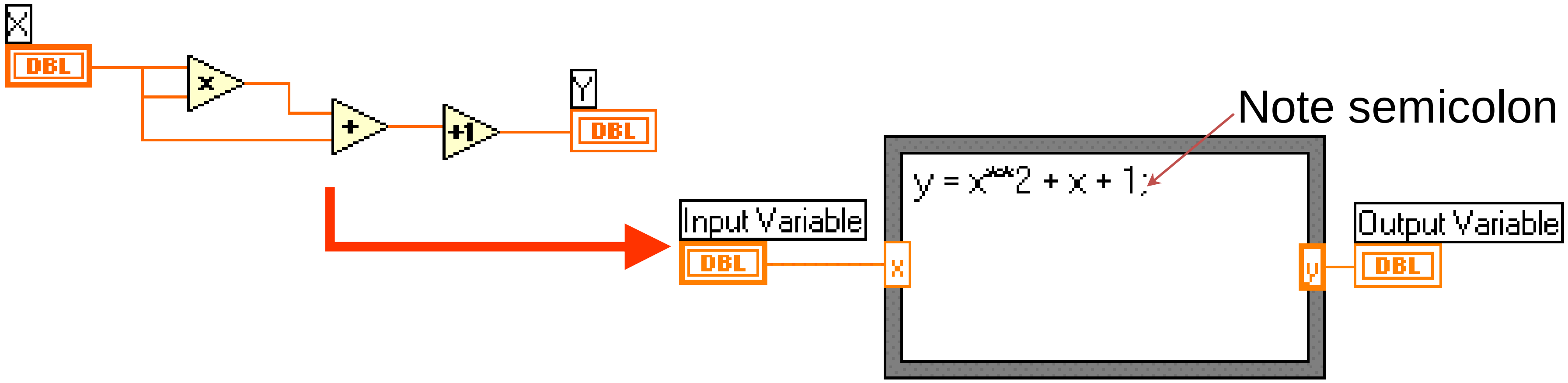
Sequence Locals

- Pass data from one frame to future frames
- Created at the border of the Sequence structure



Formula Nodes

- In the Structures subpalette
- Implement complicated equations
- Variables created at border
- Variable names are case sensitive
- Each statement must terminate with a semicolon (;)
- Context Help Window shows available functions



Printing & Documentation

- Print From File Menu to Printer, HTML, Rich Text File
- Programmatically Print Graphs or Front Panel Images
- Document VIs in VI Properties » Documentation Dialog
- Add Comments Using Free Labels on Front Panel & Block Diagram

Printing

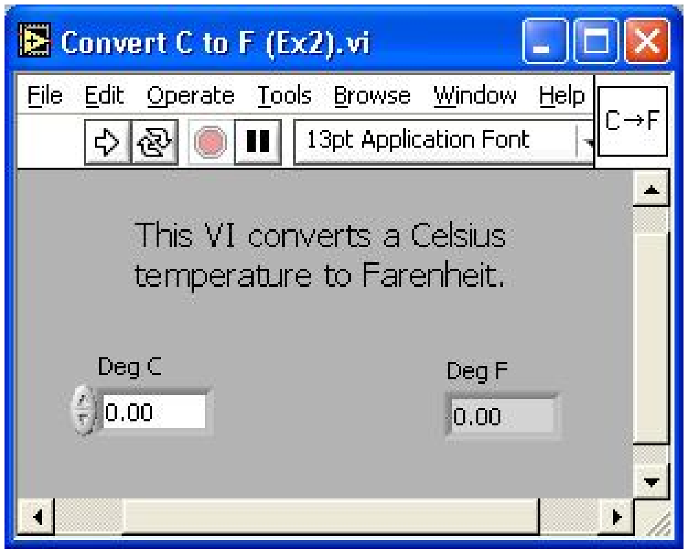
- File » Print… Gives Many Printing Options
  - Choose to Print Icon, Front Panel, Block Diagram, VI Hierarchy, Included SubVIs, VI History
- Print Panel.vi (Functions » Application Control) Programmatically Prints a Front Panel
- Generate & Print Reports (Functions » Report Generation)
  - Search in Find Examples for Report Generation

Documenting VIs

- VI Properties » Documentation
  - Provide a Description and Help Information for a VI
- VI Properties » Revision History
  - Track Changes Between Versions of a VI
- Individual Controls » Description and Tip…
  - Right Click to Provide Description and Tip Strip
- Use Labeling Tool to Document Front Panels & Block Diagrams

Simple VI Architecture

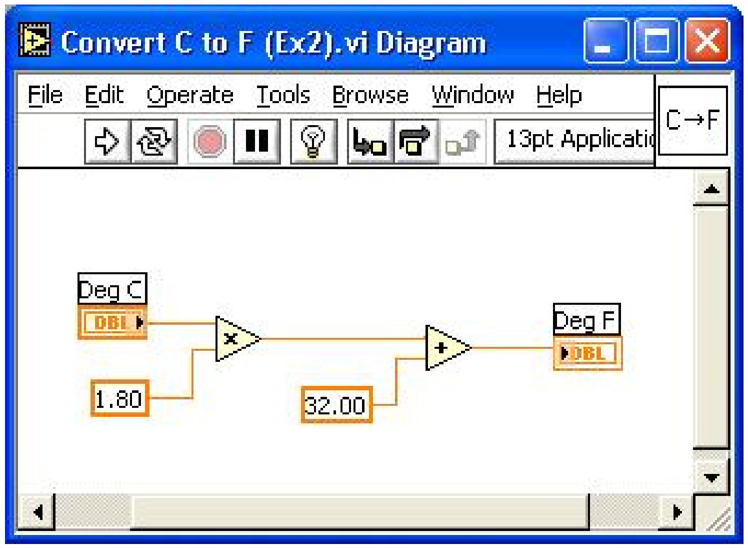
- Functional VI that produces results when run
  - No “start” or “stop” options
  - Suitable for lab tests, calculations
- Example: Convert C to F.vi



This VI converts a Celsius temperature to Farenheit.

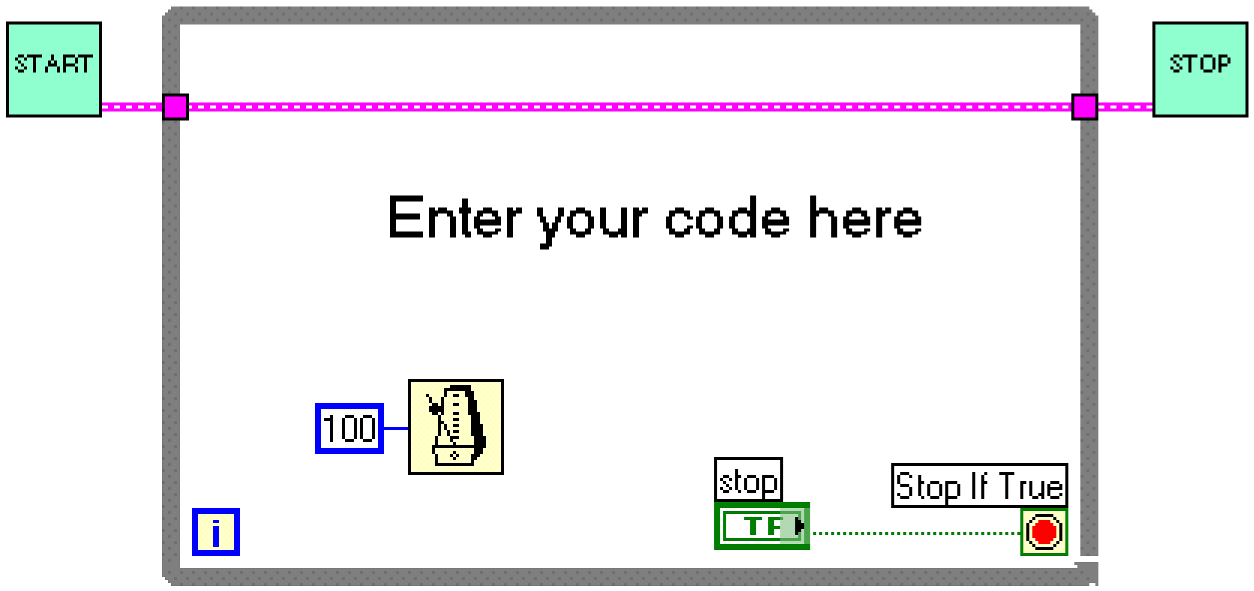
Deg C: 0.00

Deg F: 0.00


Convert C to F (Ex2).vi Diagram
File Edit Operate Tools Browse Window Help
13pt Applicati
C→F
Deg C
DBL
1.80
32.00
Deg F
DBL

General VI Architecture

- Three Main Steps
  - Startup
  - Main Application
  - Shutdown



State Machine Architecture

- Advantages
  - Can go from any state from any other
  - Easy to modify and debug
- Disadvantages
  - Can lose events if two occur at the same time

States:

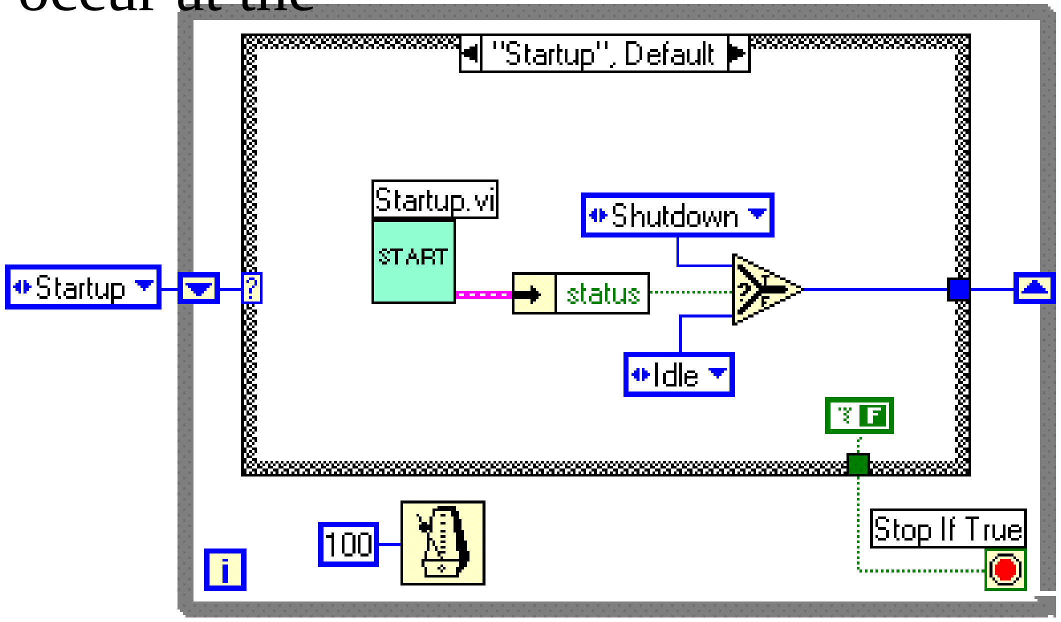
0: Startup

1: Idle

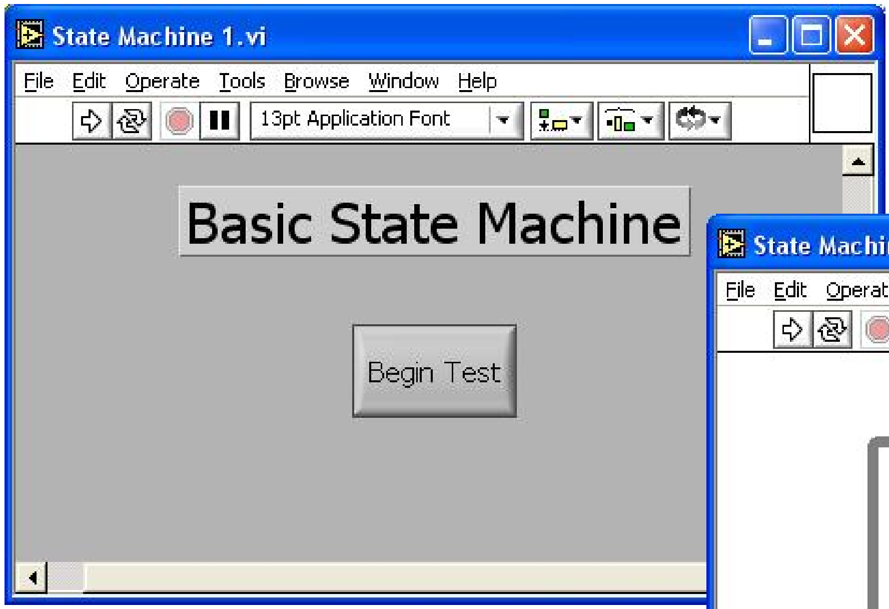
2: Event 1

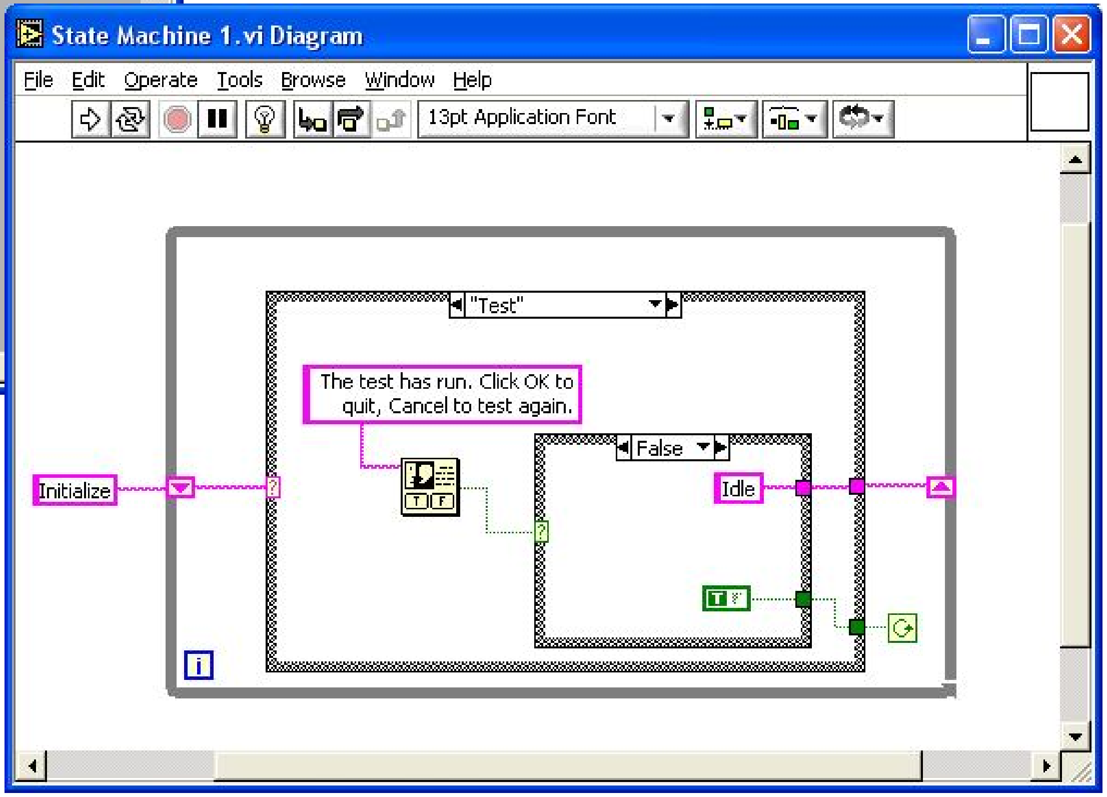
3: Event 2

4:Shutdown



Exercise 7 – Simple State Machine



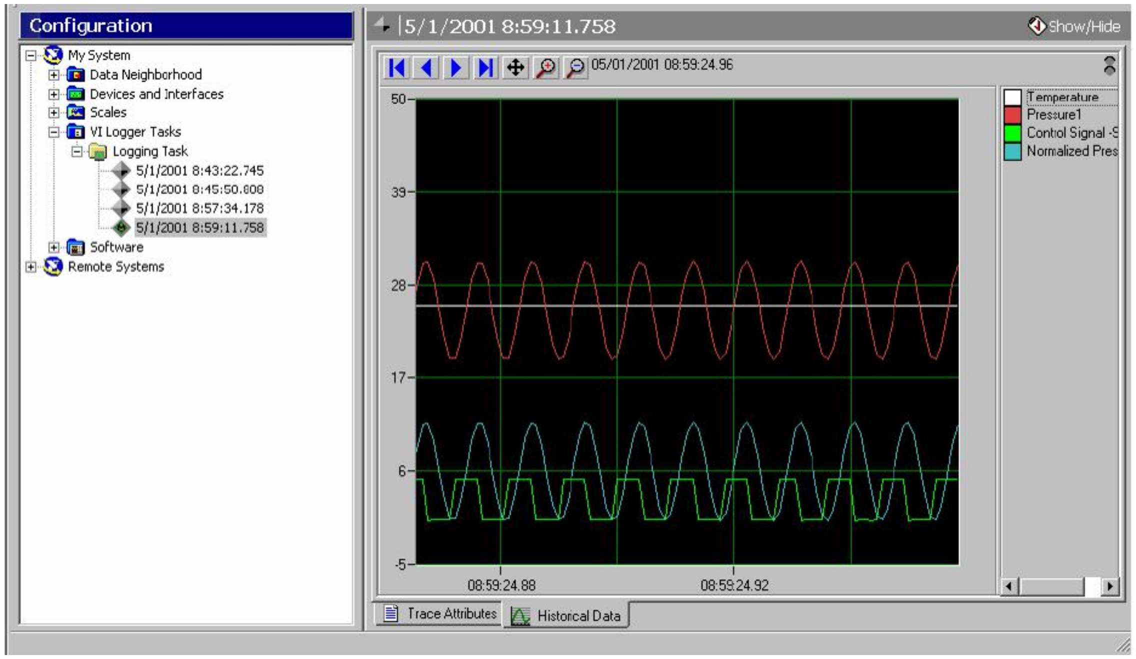


State Machine 1.vi Diagram
File Edit Operate Tools Browse Window Help
13pt Application Font
"Test"
The test has run. Click OK to quit, Cancel to test again.
False
Initialize
Idle

UNIT-III DATA ACQUISITION

Introduction to DAQ

- Data Acquisition – “Sampling of the real world to generate data that can be analyzed and presented by a computer.”



PC Based Data Acquisition System Overview:

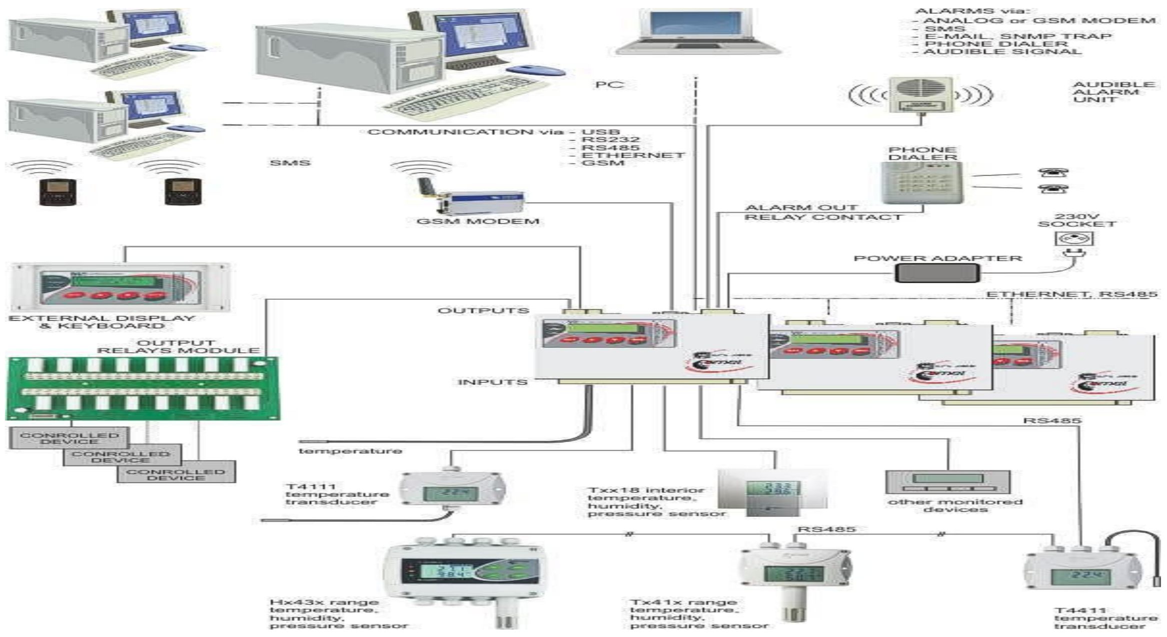
In the last few years, industrial PC I/O interface products have become increasingly reliable, ccurate and affordable. PC-based data acquisition and control systems are widely used in industrial and laboratory applications like monitoring, control, data acquisition and automated testing.

Selecting and building a DA&C (Data Acquisition and Control) system that actually does what you want it to do requires some knowledge of electrical and computer engineering.

- Transducers and actuators
- Signal conditioning
- Data acquisition and control hardware
- Computer systems software

Data acquisition System Introduction I:

Data acquisition involves gathering signals from measurement sources and digitizing the signals for storage, analysis, and presentation on a PC. Data acquisition systems come in many different PC technology forms to offer flexibility when choosing your system. You can choose from PCI, PXI, PCI Express, PXI Express, PCMCIA, USB, wireless, and Ethernet data acquisition for test, measurement, and automation applications.



Data acquisition System Introduction II:

All industrial processing systems, factories, machinery, test facilities, and vehicles consist of hardware components and computer software whose behavior follow the laws of physics as we understand them.

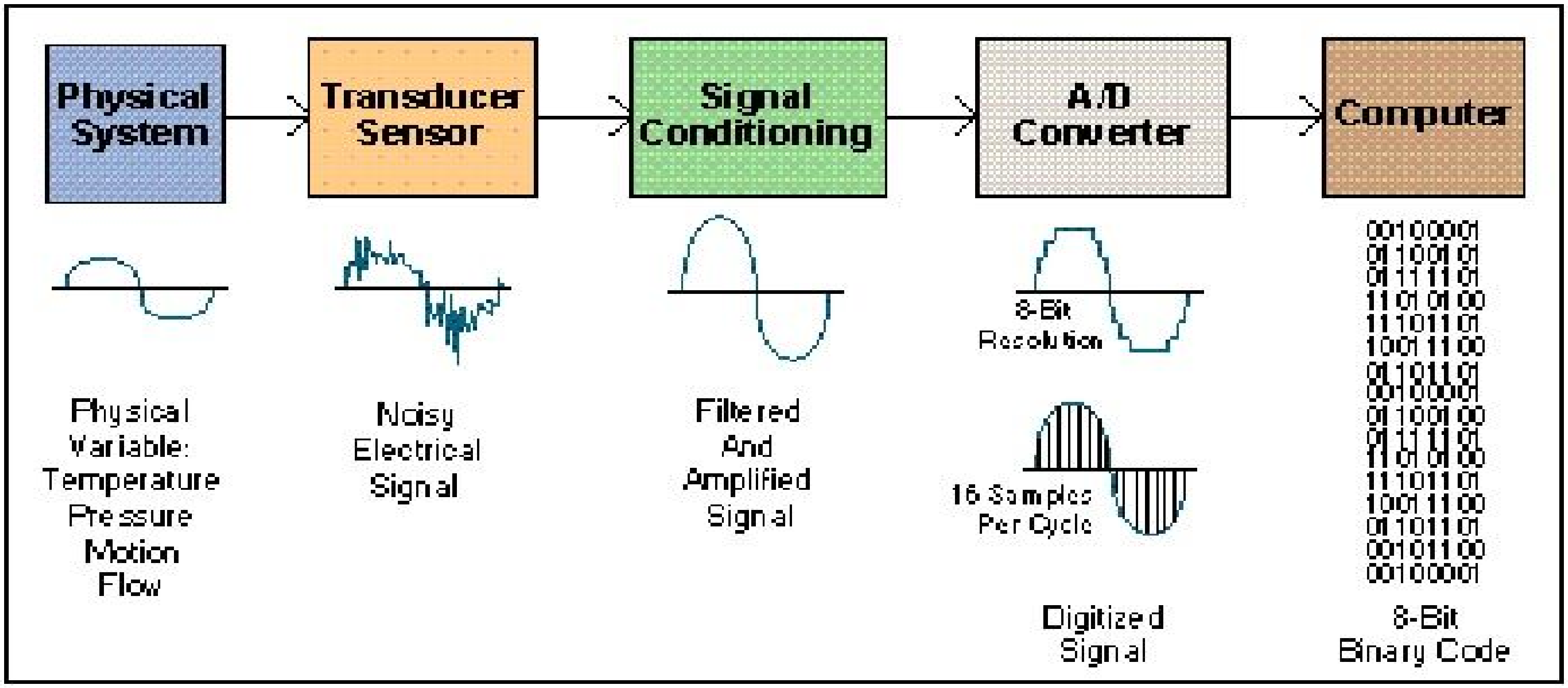
These systems contain thousands of mechanical and electrical phenomena that are continuously changing; they are not steady state.

The measurable quantities that represent the characteristics of all systems are called variables. The proper functioning of a particular system depends on certain events in time and the parameters of these variables.

Often, we are interested in the location, magnitude, and speed of the variables, and we use instruments to measure them.

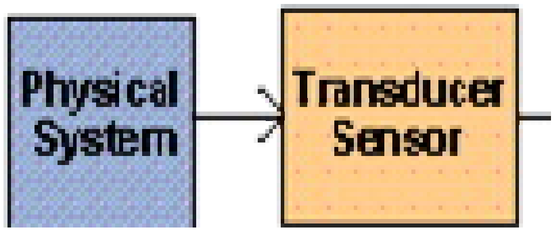
We assign the variables units of measure such as volts, pounds, and miles per hour, to name a few.

DAQ BLOCK DIAGRAM



Transducers:

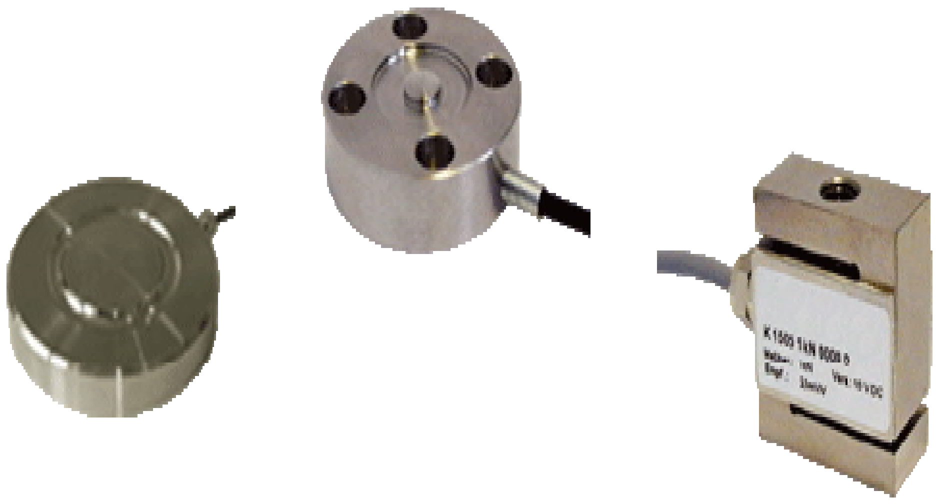
Data acquisition systems have multiple components that work together to gather and process information. They can be used to analyze information regarding physical phenomena, such as temperature, voltage, and pressure. However, because temperature, voltage, and pressure are all distinct different, they require different systems of measurement and representation. In data acquisition systems, a transducer serves as the component that translates raw data into a comprehensible electrical signal. When a data acquisition system uses DAQ (data acquisition hardware) the transducer also functions as a sensor, gathering the data from which it will then generate a signal. As a result of all the different variables data acquisition systems can measure, there are several kinds of transducers. A transducer must be capable of generating different signals depending on the particular phenomenon measured. Two general types of signals commonly are used: analog and digital.



Transducer and Acutuator:

A transducer converts temperature, pressure, level, length, position, etc. into voltage, current, frequency, pulses or other signals.

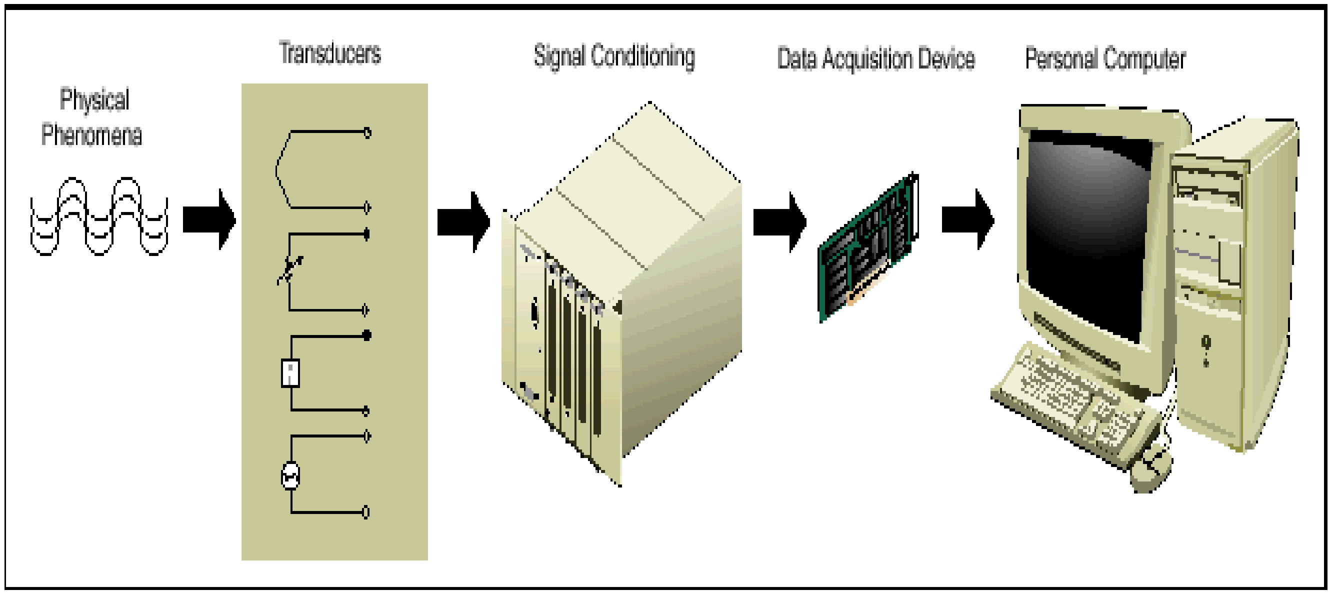
An actuator is a device that activates process control equipment by using pneumatic, hydraulic or electrical power. For example, a valve actuator opens and closes a valve to control fluid rate.



Signal conditioning :

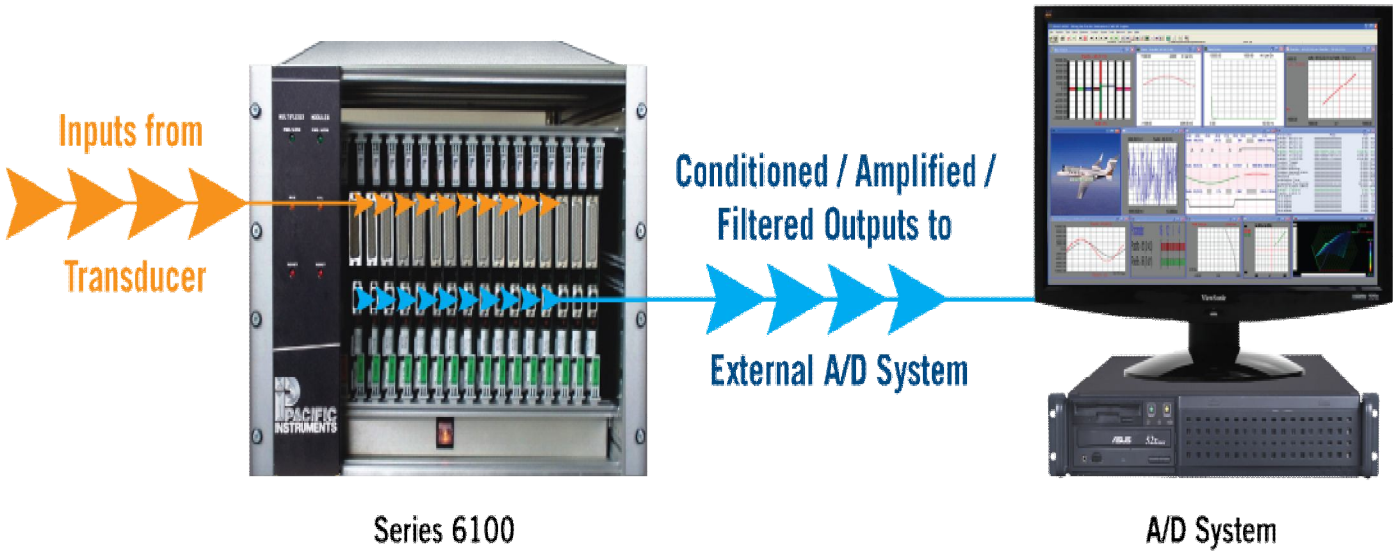
Signal conditioning circuits improve the quality of signals generated by transducers before they are converted into digital signals by the PC's data-acquisition hardware.

Examples of signal conditioning are signal scaling, amplification, linearization, cold-junction compensation, filtering, attenuation, excitation, common-mode rejection, and so on.

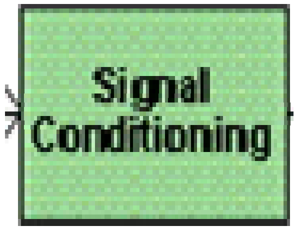


One of the most common signal conditioning functions is amplification. For maximum resolution the voltage range of the input signals should be approximately equal to the maximum input range of the A/D converter.

Amplification expands the range of the transducer signals so that they match the input range of the A/D converter. For example, a x 10 amplifier maps transducer signals which range from 0 to 1 V onto the range 0 to 10 V before they go into the A/D converter.



Signal Conditioning



Electrical signals are conditioned so they can be used by an analog input board.

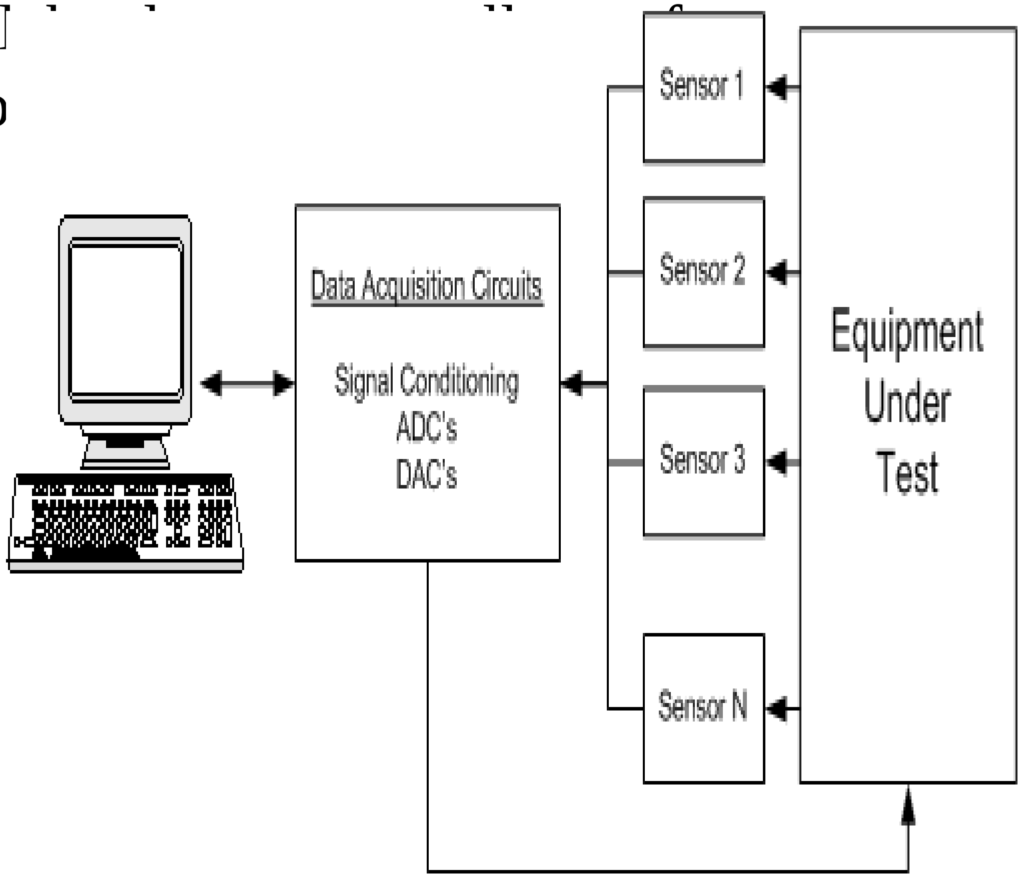
The following features may be available:

- Amplification
- Isolation
- Filtering
- Linearization

Data acquisition

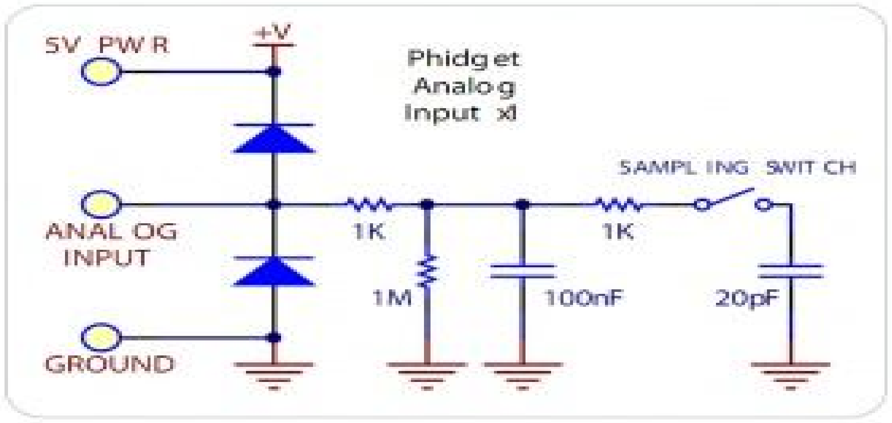
Data acquisition and control
more of the following functio

•analog input,
•analog output,
•digital input,
•digital output and
•counter/timer functions.



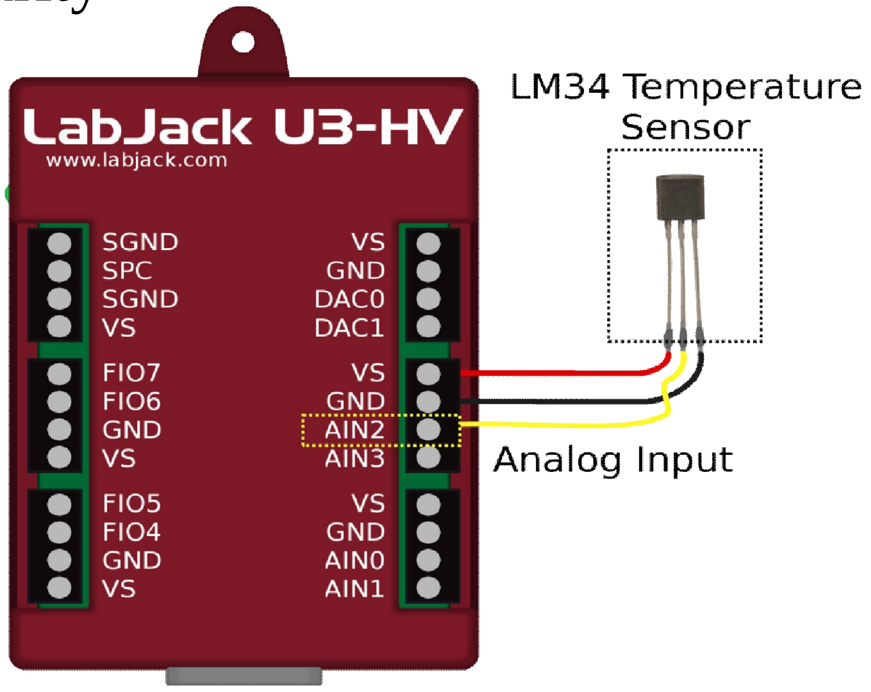
Analog input

An analog input converts a voltage level into a digital value that can be stored and processed in a computer. Why would you want to measure voltages? There are a multitude of sensors available which convert things like temperature, pressure, etc. into voltages. The voltages can then be easily measured by various kinds of hardware, such as a LabJack U3-HV, and then read into a computer. The computer can then convert the voltage value into it's original type (temperature, pressure, etc) and the value can then be stored in a file, emailed to someone, or used to control something else outside of the computer.

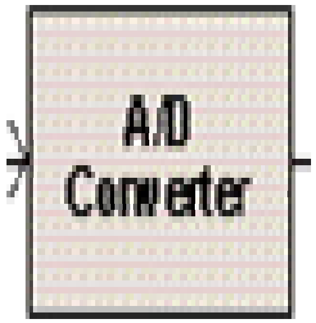


The most significant criteria when selecting A/D hardware are:

1. Number of input channels
2. Single-ended or differential input signals
3. Sampling rate (in samples per second)
4. Resolution (usually measured in bits of resolution)
5. Input range (specified in full-scale volts)
6. Noise and nonlinearity



Analog to Digital Converter



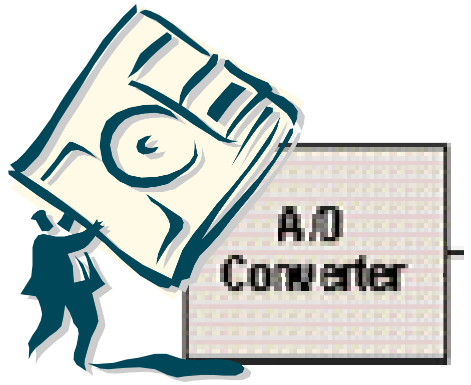
An Analog to Digital Converter (ADC) is a very useful feature that converts an analog voltage on a pin to a digital number. By converting from the analog world to the digital world, we can begin to use electronics to interface to the analog world around us.

Not every pin on a microcontroller has the ability to do analog to digital conversions. On the Arduino board, these pins have an ‘A’ in front of their label (A0 through A5) to indicate these pins can read analog voltages.

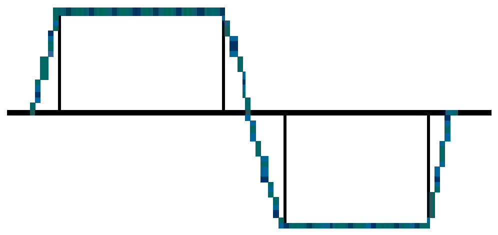
ADCs can vary greatly between microcontroller. The ADC on the Arduino is a 10-bit ADC meaning it has the ability to detect 1,024 (210) discrete analog levels.
Some microcontrollers have 8-bit ADCs (28 = 256 discrete levels) and some have 16-bit ADCs (216 = 65,535 discrete levels).

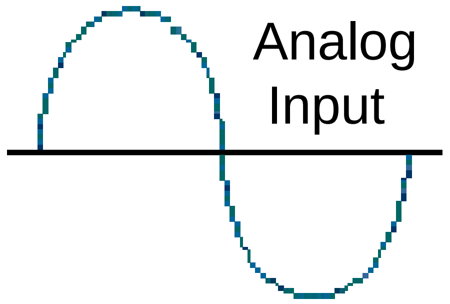
Sampling rate

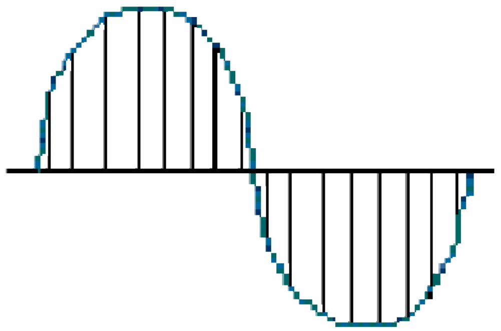
Sampling rate is the speed at which the digitizer's ADC converts the input signal, after the signal has passed through the analog input path, to digital values. Hence, the digitizer samples the signal after any attenuation, gain, and/or filtering has been applied by the analog input path, and converts the resulting waveform to digital representation. The sampling rate of a high-speed digitizer is based on the sample clock that controls when the ADC converts the instantaneous analog voltage to digital values



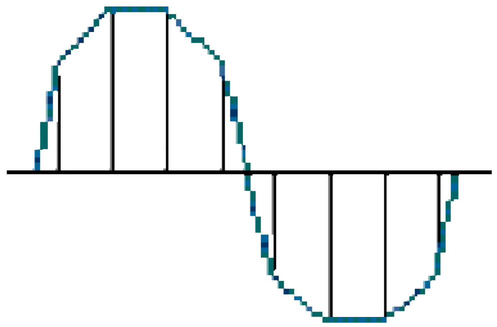
4 Samples/cycle



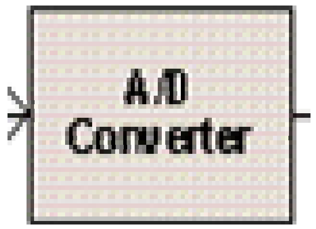




16 Samples/cycle



8 Samples/cycle



Effective rate of each individual channel is inversely proportional to the number of channels sampled.

<u>Example:</u>

100 KHz maximum.

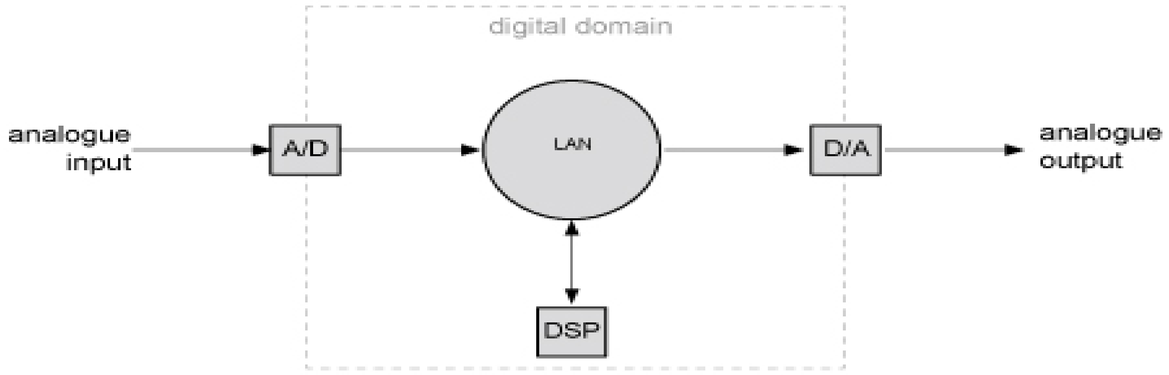
16 channels.

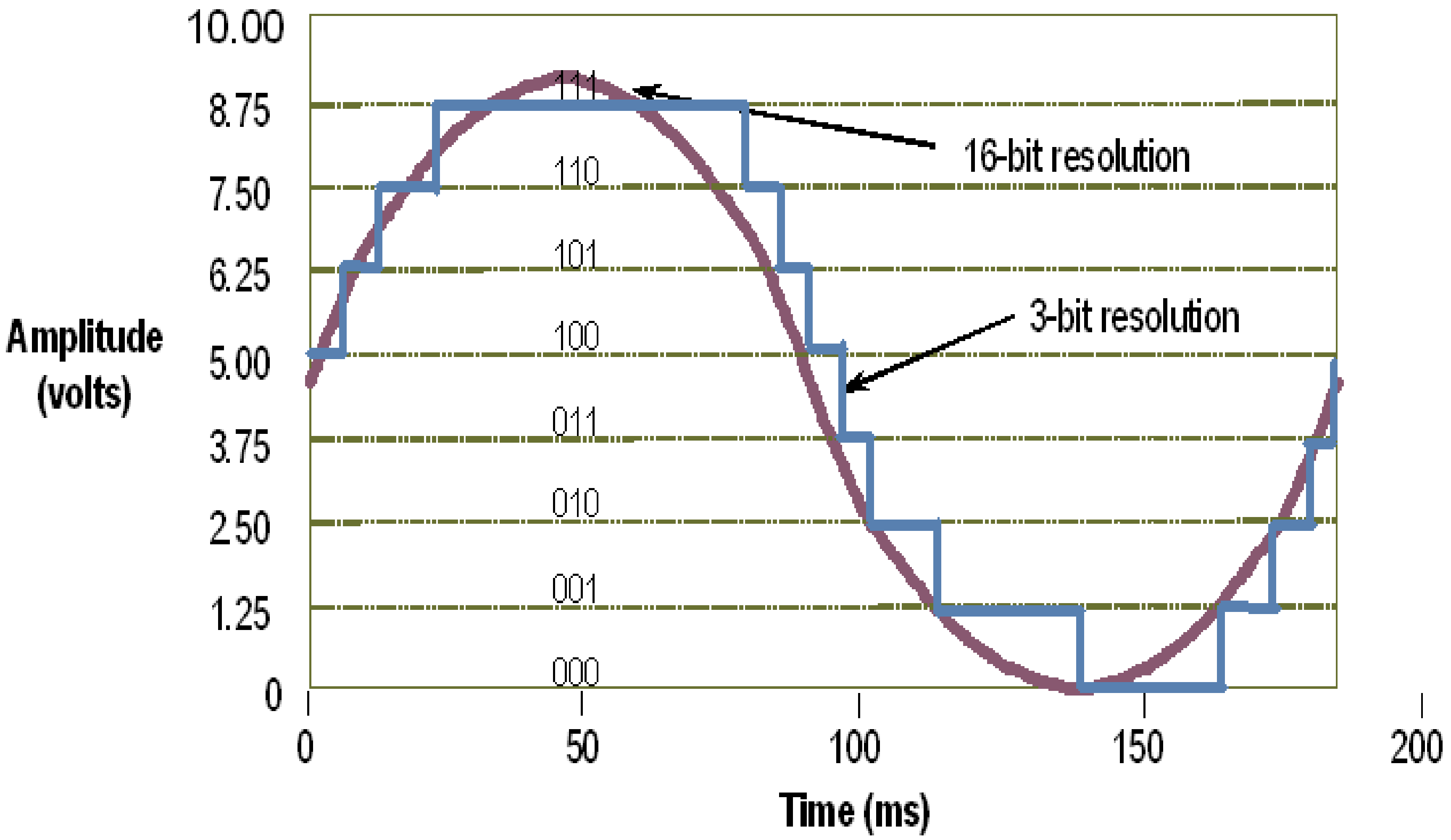
100 KHz/16 = 6.25 KHz per channel.

A/D converter Range

Dynamic range is often a key parameter within signal processing systems and a shortfall can limit the quality and range of signals that can be received. The technical progress made on improving this gateway between the analogue and digital world has not kept pace with Moore's law because the challenges are more fundamental than simply reducing transistor sizes. Methods to increase a/d converter dynamic range are therefore always of interest, although each solut

figure 501: basic signal chain in an audio network




10.00
8.75
7.50
6.25
5.00
3.75
2.50
1.25
0
Amplitude (volts)
111
110
101
100
011
010
001
000
16-bit resolution
3-bit resolution
0
50
100
150
200
Time (ms)

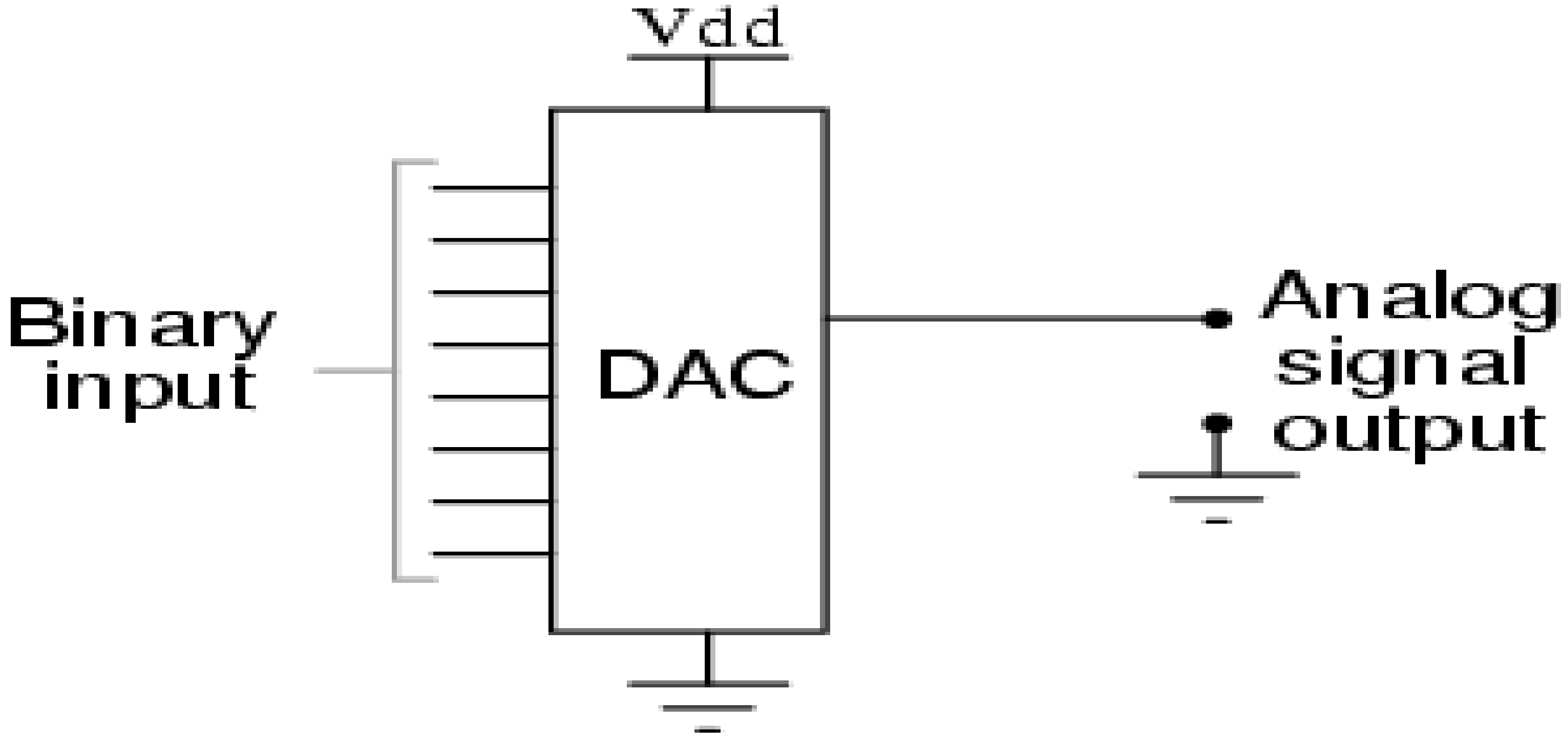
Analog output (D/A)

An analog output is a measurable electrical signal with a defined range that is generated by a controller and sent to a controlled device, such as a variable speed drive or actuator.
Changes in the analog output cause changes in the controlled device that result in changes in the controlled process.

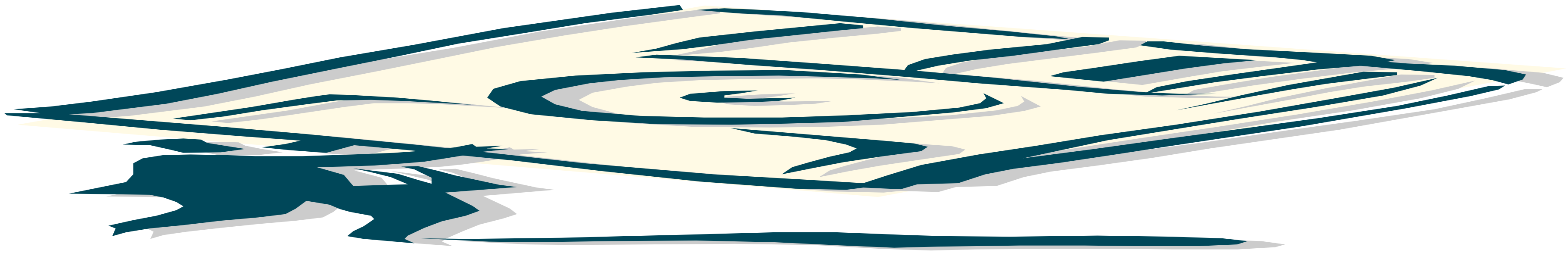
Controller output digital to analog circuitry is typically limited to a single range of voltage or current, such that output transducers are required to provide an output signal that is compatible with controlled devices using something other than the controller's standard signal.

Common Types:There are four common types of analog outputs; voltage, current, resistance and pneumatic.

Analog output (D/A)



Data Aquisition software



- It can be the most critical factor in obtaining reliable, high performance operation.
- Transforms the PC and DAQ hardware into a complete DAQ, analysis, and display system.
- Different alternatives:
  - Programmable software.
  - Data acquisition software packages.

Programmable software

- Involves the use of a programming language, such as:
  - C++, visual C++
  - BASIC, Visual Basic + Add-on tools (such as VisuaLab with VTX)
  - Fortran
  - Pascal

✓ <u>Advantage</u>: flexibility

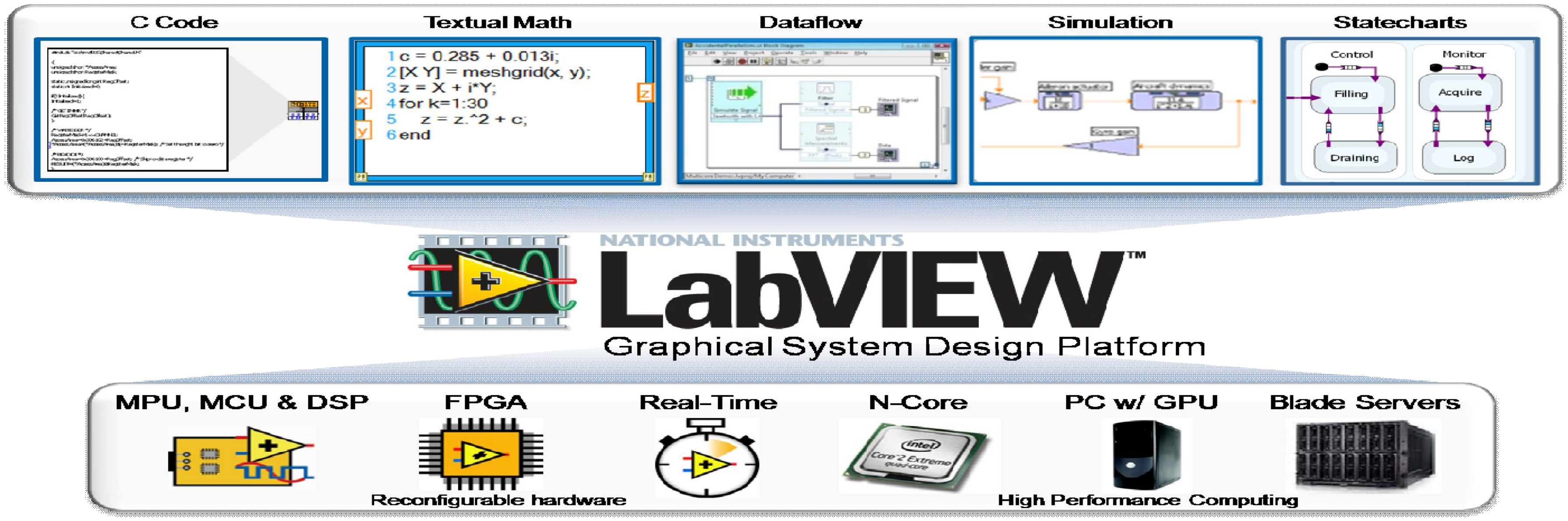
✓ <u>Disadvantages</u>: complexity and steep learning curve

Data acquisition software

- Does not require programming.
- Enables developers to design the custom instrument best suited to their application.

Examples: TestPoint, SnapMaster, LabView, DADISP, DASYLAB, etc.

Below is an image with LabView:



Small Computer System Interface [SCSI]

Small Computer System Interface (SCSI), an ANSI standard, is a parallel interface standard used by Apple Macintosh computers, PCs, and many UNIX systems for attaching peripheral devices to computers. SCSI interfaces provide for faster data transmission rates than standard serial and parallel ports. In addition, you can attach many devices to a single SCSI port. There are many variations of SCSI: SCSI-1, SCSI-2, SCSI-3 and the recently approved standard Serial Attached SCSI (SAS).

SCSI-1 : SCSI-1 is the original SCSI and it is obsolet so far. Basically, SCSI-1 uses an 8-bit bus, and supports data rates of 4 MBps.

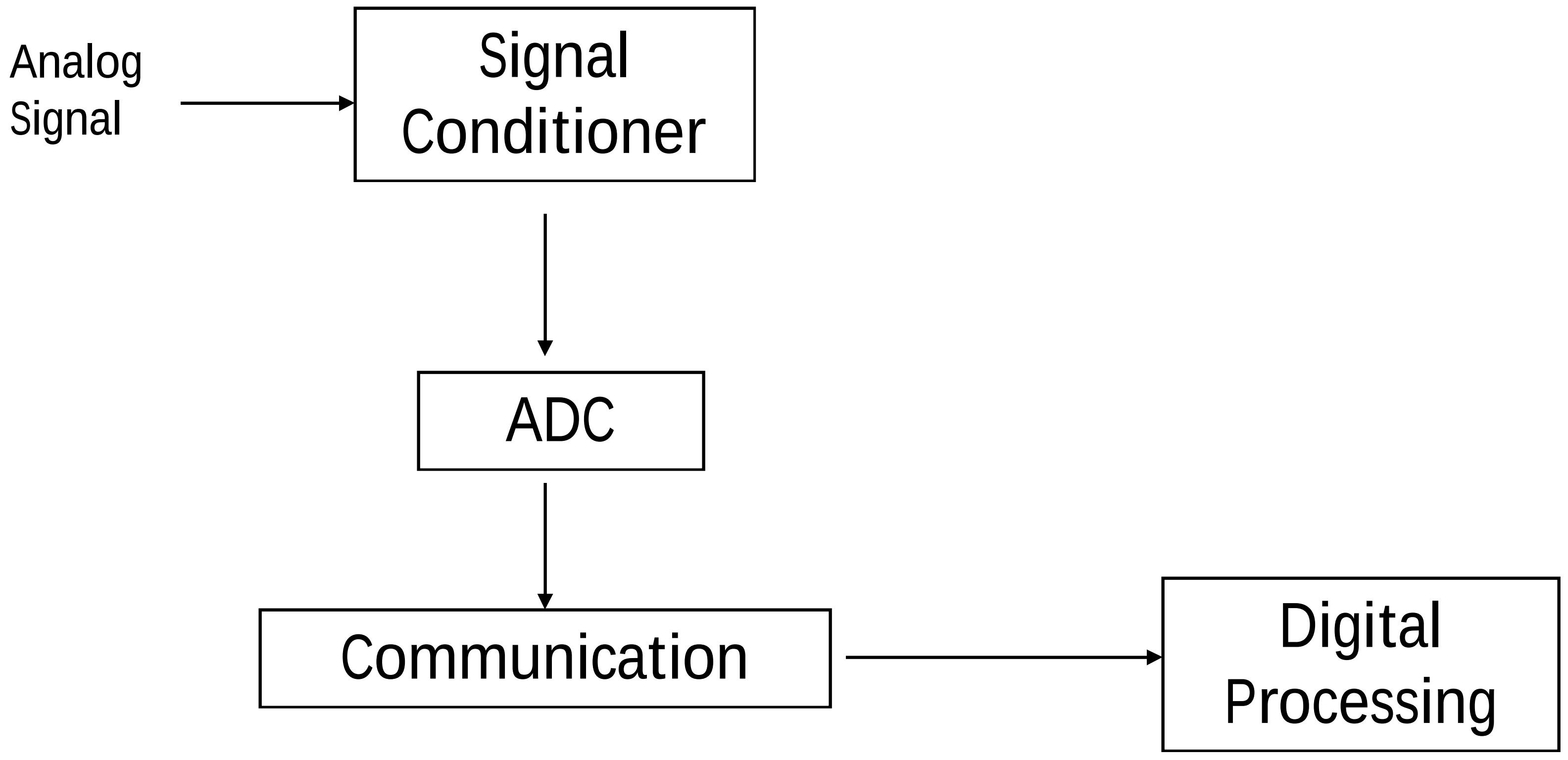
SCSI-2

SCSI-2 is an improved version of SCSI-1. SCSI-2 is based on CCS which is a minimum set of 18 basic commands all manufacture's hardware would work together. SCSI-2 also provided extra speed with options called Fast SCSI and a 16-bit version called Wide SCSI. A feature called command queuing gave the SCSI device the ability to execute command in an order that would be most efficient. Fast SCSI delivers a 10 MB/sec transfer rate. When combined with the 16-bit bus, this doubles to 20 MB/sec. This is called Fast-Wide SCSI.

SCSI-3

SCSI-3 has many advances over SCSI-2 such as Serial SCSI. This feature will allow data transfer up to 100MB/sec through a six-conductor coaxial cable. SCSI-3 solves many of the termination and delay problems of older SCSI versions. SCSI-3 eases SCSI installation woes by being more plug-and-play in nature, such as automatic SCSI ID assigning and termination. SCSI-3 also supports 32 devices while SCSI-2 supports only 8.

Data Acquisition System



Analog vs. Digital Signal

- Analog signals:
  - Continuous, expressed in decimal system
  - No limitation on the maximum/minimum value
  - Can not be processed by computer

- Digital signals: binary number system
  - All numbers are expressed by a combination of 1 & 0
  - The maximum value is limited by # of bits available

Signal Conditioning

Functions: modify the analog signal to match the performance of the ADC

- Pre-filtering: remove undesirable high frequency components
- Amplification: amplify the signal to match the dynamic range of the ADC

Analog-to-Digital Conversion (ADC)

Function: convert analog signals into digital signals

- Sample & hold
- Quantization
- Coding

$y(t)=f(t) \Rightarrow y_k=f(t_k)$

Quantization

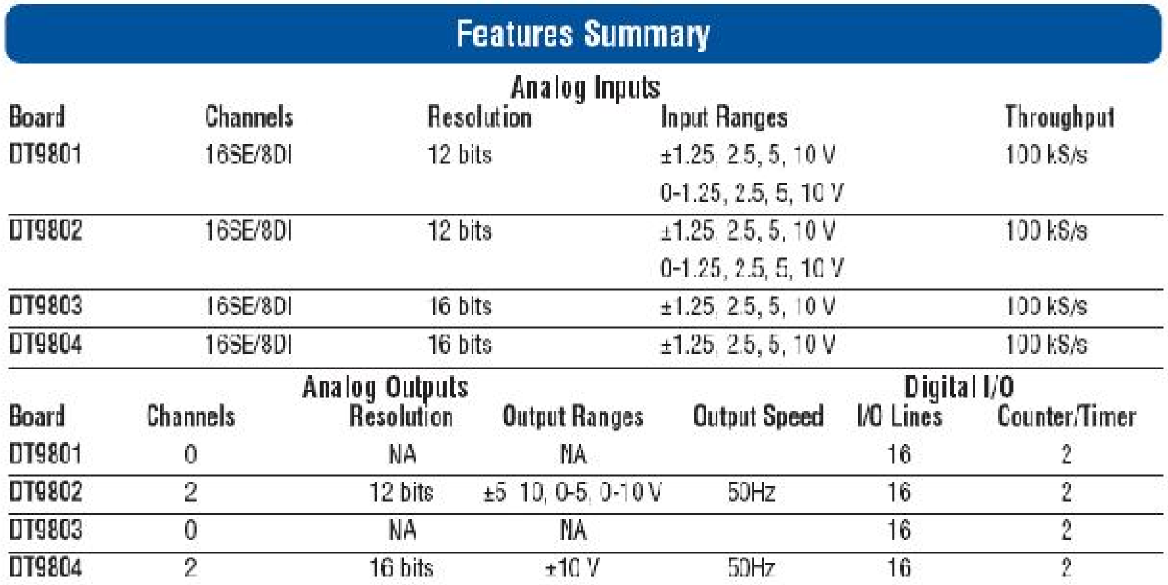
Definition: transformation of a continuous analog input into a set of discrete output state

- Coding: the assignment of a digital code word or number to each output states
- # of possible state: $N=2^n$, n is # of bits
- Quantization resolution: Q=(Vmax-Vmin)/N
- Quantization Error: $$\Delta = \sum_N \left| f(t_k) - f_k \right|$$

Select a Data Acquisition Card

- Functions: A/D, D/A, Digital I/O, signal conditioning (amplification, prefiltering), timer, trigger, buffer
- Features:
  - A/D resolution (# of bits used)
  - Maximum sampling rate
  - # of channels
  - Total throughput
  - Aperture time

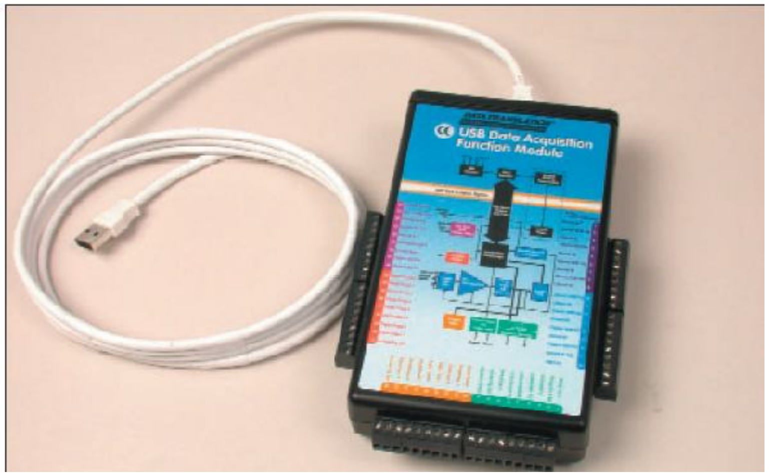
Example of Data Acquisition Card



Features Summary

		Analog Inputs		
Board	Channels	Resolution	Input Ranges	Throughput
DT9801	16SE/8DI	12 bits	±1.25, 2.5, 5, 10 V 0-1.25, 2.5, 5, 10 V	100 kS/s
DT9802	16SE/8DI	12 bits	±1.25, 2.5, 5, 10 V 0-1.25, 2.5, 5, 10 V	100 kS/s
DT9803	16SE/8DI	16 bits	±1.25, 2.5, 5, 10 V	100 kS/s
DT9804	16SE/8DI	16 bits	±1.25, 2.5, 5, 10 V	100 kS/s

	Analog Outputs				Digital I/O	
Board	Channels	Resolution	Output Ranges	Output Speed	I/O Lines	Counter/Timer
DT9801	0	NA	NA		16	2
DT9802	2	12 bits	±5, 10, 0-5, 0-10 V	50Hz	16	2
DT9803	0	NA	NA		16	2
DT9804	2	16 bits	±10 V	50Hz	16	2



What is PXI?

- PXI = **P**CI e**X**tensions for **I**nstrumentation
- Open specification governed by the PXI Systems Alliance (PXISA) and introduced in 1997
- PC-based platform optimized for test, measurement, and control
- PCI electrical-bus with the rugged, modular, Eurocard mechanical packaging of CompactPCI
- Advanced timing and synchronization features

PXI Systems Alliance (PXISA)

- Founded in 1998
- PXISA Goals:
  - Maintain the PXI specification
  - Ensure interoperability
  - Promote the PXI standard
- Currently 68+ Members Comprise PXISA
- PXISA Website (www.pxisa.org)
  - Specifications
  - Tutorials, Application Notes, and White Papers
  - Locate member companies and products

PXI Specification

Mechanical

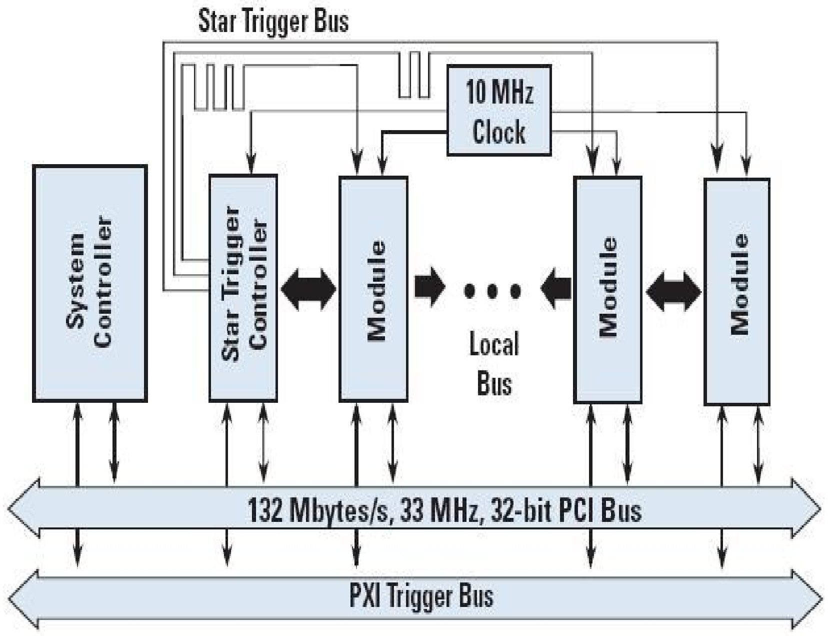
- High-performance connectors
- Eurocard mechanical packaging
- Forced-air cooling by chassis
- Environmental testing
- Electromagnetic testing

Electrical

- Industry-standard PC buses
- System reference clocks
- Star trigger buses
- PXI trigger bus

Software

- Microsoft Windows software frameworks
- Software components that define HW configuration and capabilities
- Virtual Instrument Software Architecture (VISA) implementation



PXI System Overview

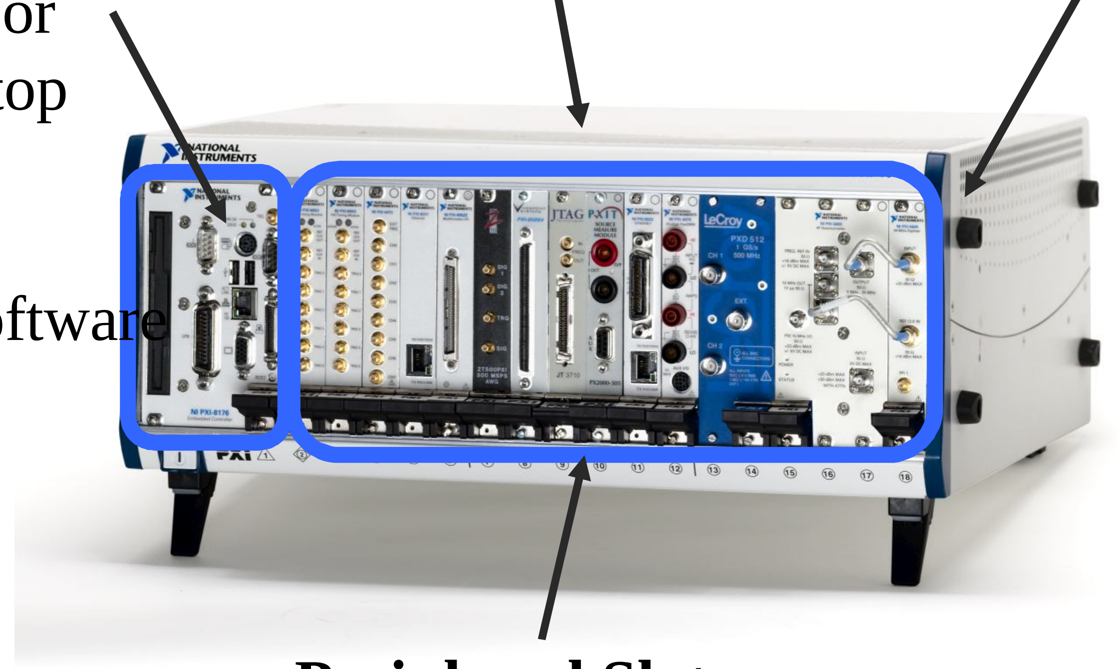
PXI Controller

- Embedded PC, remote PC or remote laptop interface
- Runs all standard software

PXI Backplane

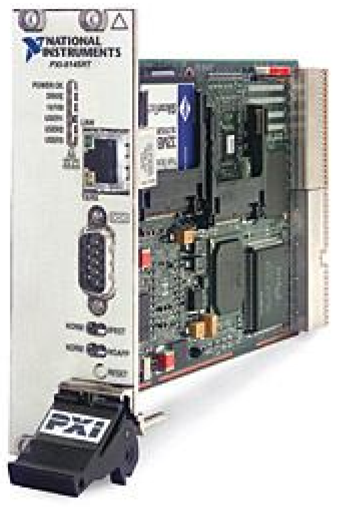
•PCI/PCI Express bus

•Synchronization



Embedded PXI System Controllers

Windows-Based Embedded Controllers
• High-performance
• Integrated peripherals
• Entire system in one chassis

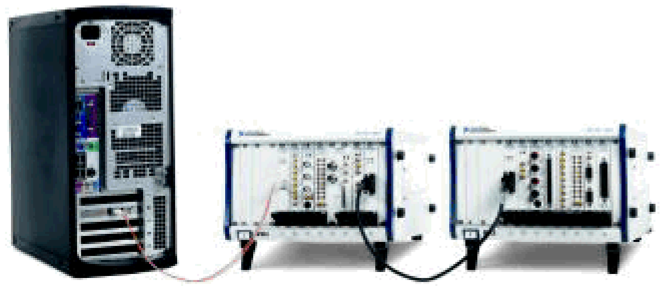


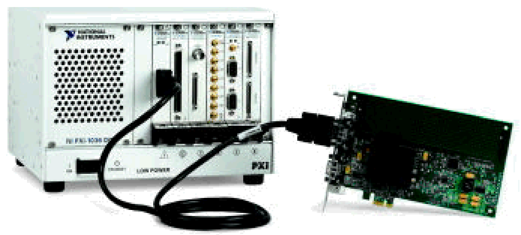
Real-Time Embedded Controllers
• Determinism and reliability with LabVIEW Real-Time
• Select high-performance or low-cost/low-power
• Headless operation

Remote PXI System Controllers

PC Control of PXI

- Use latest high-performance PCs
  - PCI Express with MXI-Express
  - PCI with MXI-4
- High-speed, software transparent links
  - Up to 110 MB/s sustained throughput
- Build multi-chassis PXI systems
- Copper and fiber-optic cabling options





Laptop Control of PXI

- Use latest high-performance laptop computers
  - ExpressCard with ExpressCard MXI
  - PCMCIA CardBus
- High-speed, software transparent links
  - Up to 110 MB/s sustained throughput
- PXI controllers for portable applications
- Use with DC-powered chassis for mobile systems

PXI Chassis

Chassis Offering

- 4, 6, 8, 14, and 18-slot
- Portable, benchtop, and rack-mount
- AC and DC options
- PXI/SCXI combination chassis

with integrated signal conditioning

Wide Range of PXI Modules

Data Acquisition and Control

Multifunction I/O

Analog Input/Output

Digital I/O

Counter/Timer

FPGA/Reconfigurable I/O

Machine Vision

Motion Control

Signal Conditioning

Temperature

Strain/Pressure/Force/Load

Synchro/Resolver

LVDT/RVDT

Many More. . .

Modular Instrumentation

Digital Waveform Generator

Digital Waveform Analyzer

Digital Multimeter

LCR Meter

Oscilloscope/Digitizer

Source/Signal Generator

Switching

RF Signal Generator

RF Signal Analyzer

RF Power Meter

Frequency Counter

Programmable Power Supply

Many More. . .

Bus Interfaces

Ethernet, USB, FireWire

SATA, ATA/IDE, SCSI

GPIB

CAN, DeviceNet

Serial RS-232, RS-485

VXI/VME

Boundary Scan/JTAG

MIL-STD-1553, ARINC

PCMCIA/CardBus

PMC

Profibus

LIN

Many More. . .

Others

IRIG-B, GPS

Direct-to-Disk

Reflective Memory

DSP

Optical

Resistance Simulator

Fault Insertion

Prototyping/Breadboard

Graphics

Audio

Many More. . .

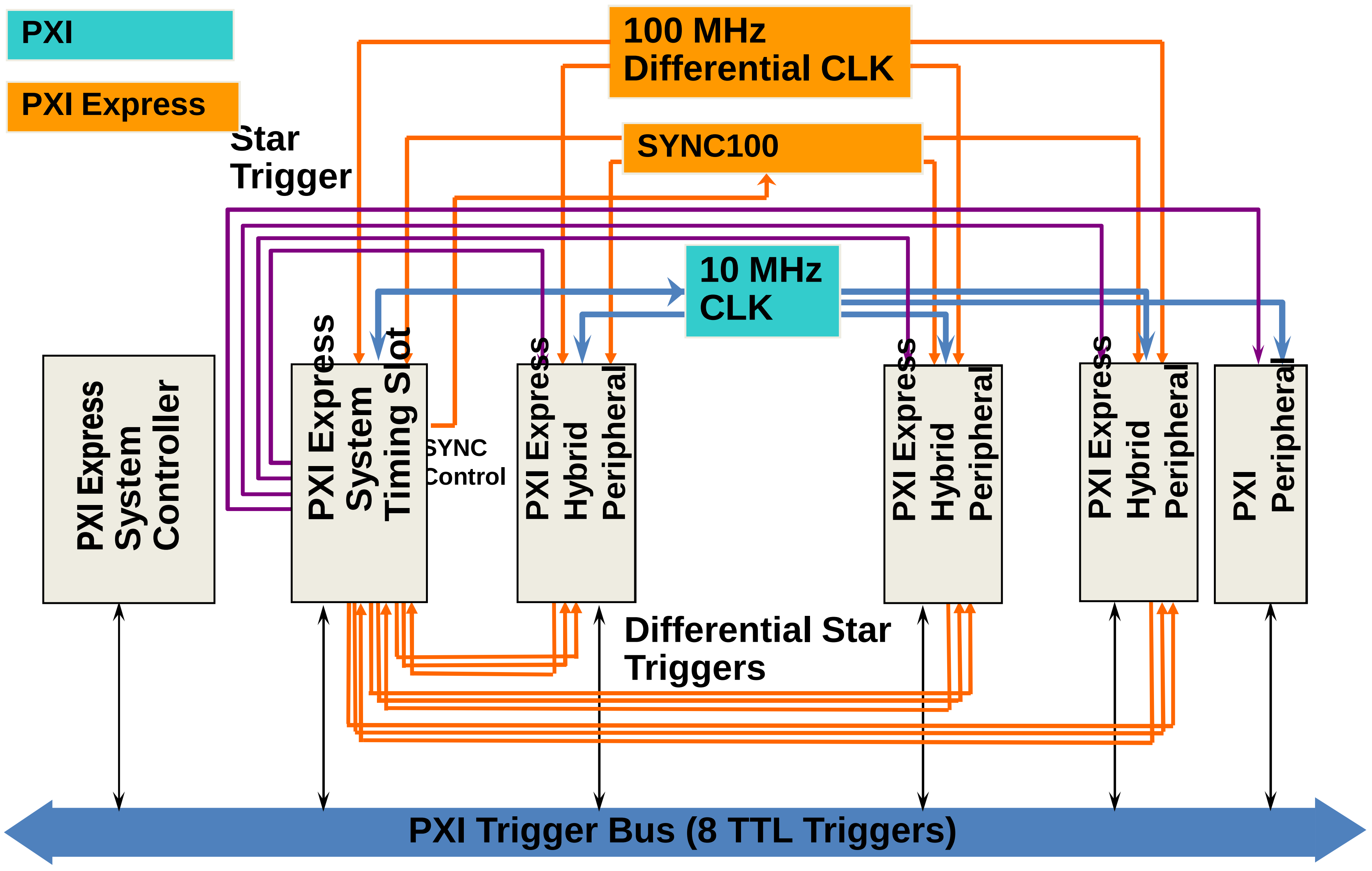
What’s New in PXI?

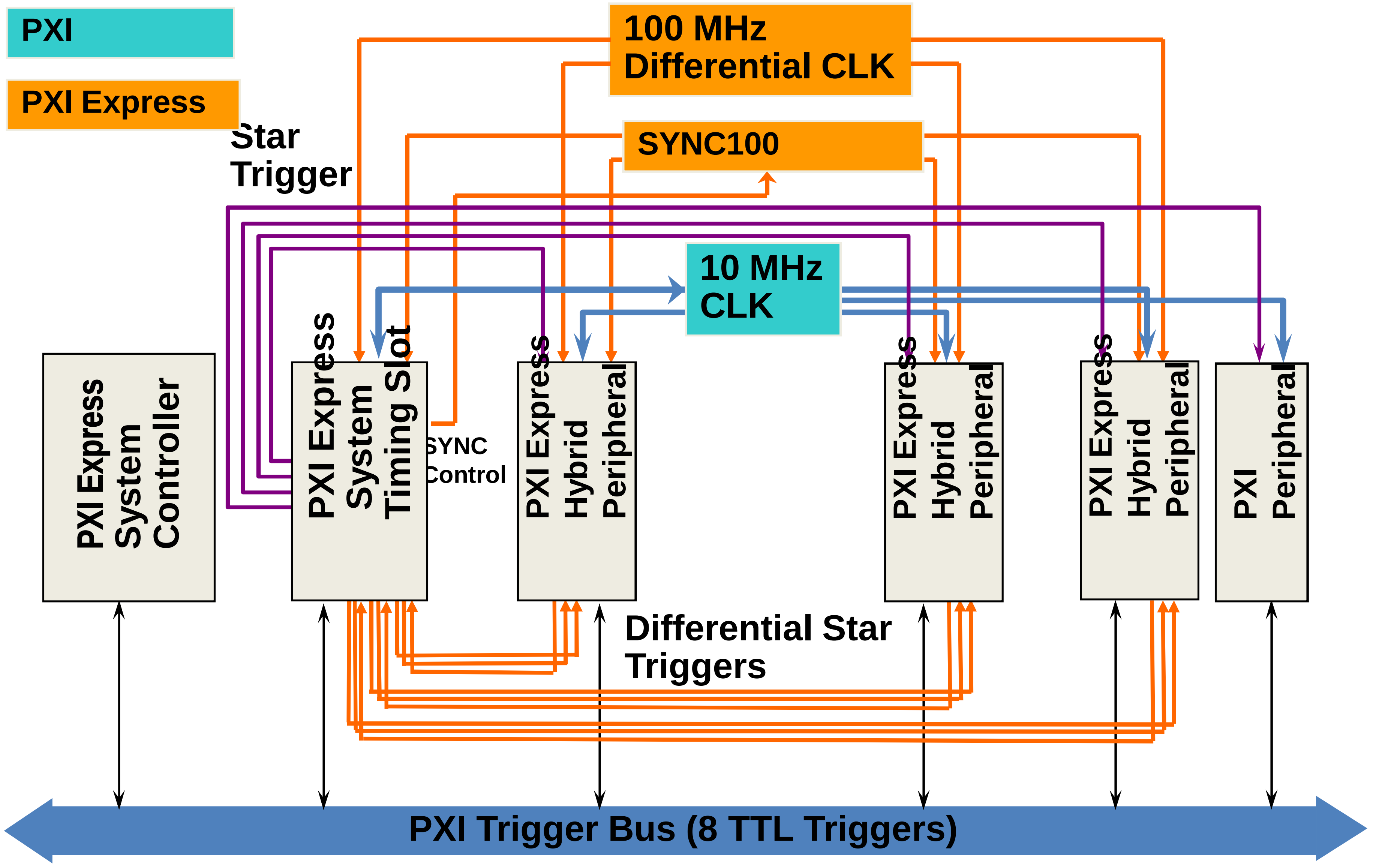
PXI Express

- Increases throughput with 2.0 GB/s per slot dedicated bandwidth
- Industry’s best synchronization and latency specification
- Ensures compatibility with your existing software and all 1000+ PXI modules

Increased BW Enables New Applications

- **PXI applications requiring PCI bandwidth**
  - General purpose automated test (DMMs, switching, baseband instruments, etc)
  - General purpose data acquisition (AI, AO, DIO, etc)
  - Bus interfaces (CAN, 1553, ARINC, etc)
  - Motion control
- **PXI applications requiring PCI Express bandwidth**
  - High frequency, resolution IF / RF systems
  - High speed digital interfaces
  - High channel count data acquisition
  - High speed imaging

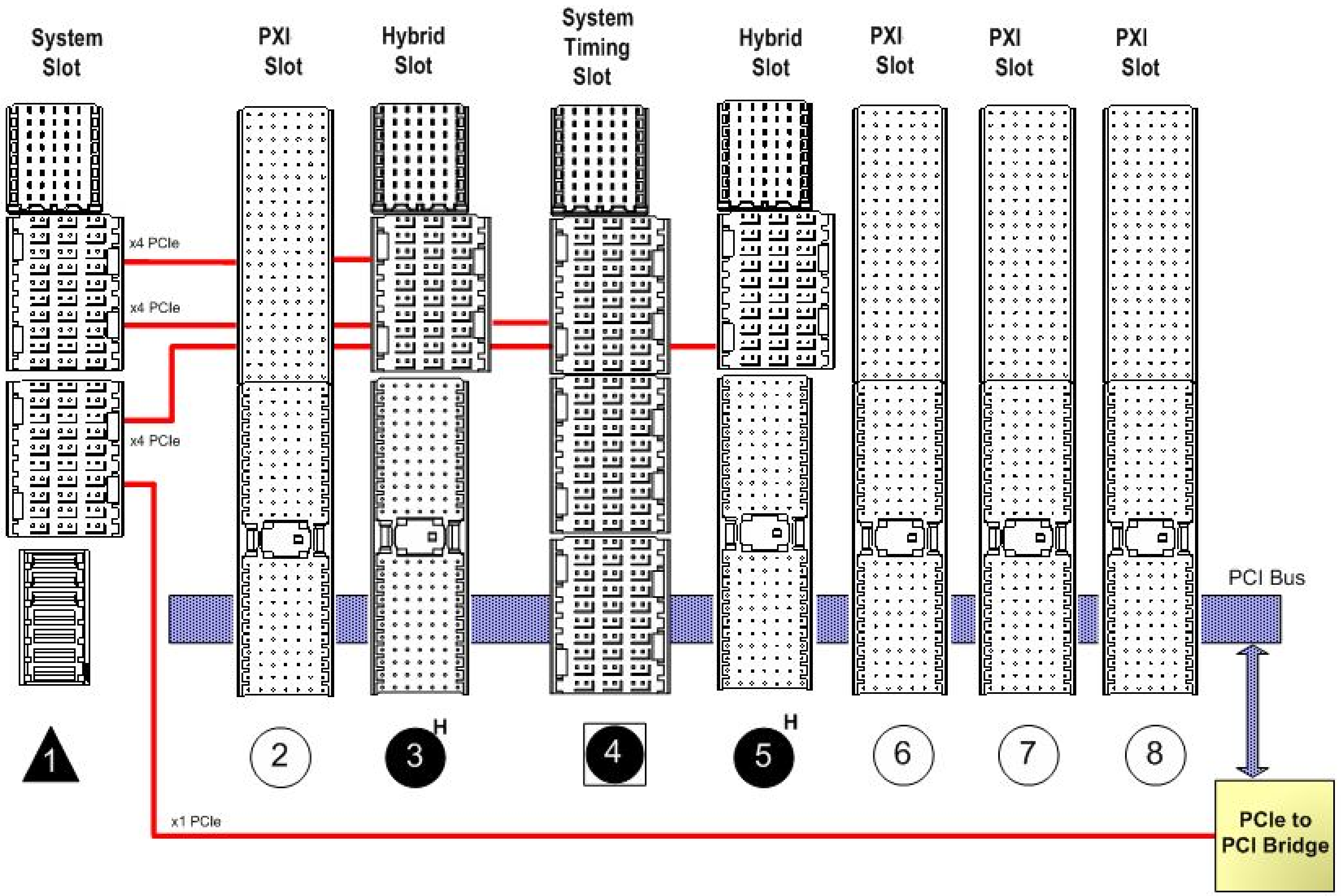

PXI
PXI Express
Star Trigger
100 MHz Differential CLK
SYNC100
10 MHz CLK
PXI Express System Controller
PXI Express System Timing Slot
SYNC Control
PXI Express Hybrid Peripheral
PXI Express Hybrid Peripheral
PXI Express Hybrid Peripheral
PXI Peripheral
Differential Star Triggers
PXI Trigger Bus (8 TTL Triggers)


PXI
PXI Express
Star Trigger
100 MHz Differential CLK
SYNC100
10 MHz CLK
PXI Express System Controller
PXI Express System Timing Slot
SYNC Control
PXI Express Hybrid Peripheral
PXI Express Hybrid Peripheral
PXI Express Hybrid Peripheral
PXI Peripheral
Differential Star Triggers
PXI Trigger Bus (8 TTL Triggers)

Benefits of PXI Express Timing and Synch. Features

- **Higher performance**
  - 100 MHz differential system reference clock
    - LVPECL lower jitter clock distribution
      - Clock 10 < 1 ns skew
      - Clock 100 < 200 ps skew
    - Tighter synchronization specifications
  - Multichassis synchronization with PXIe_SYNC100
  - Differential star triggers
    - Available to all slots
    - LVPECL low jitter clock, LVDS clocks/triggers for compatibility

PXI and Hybrid Slots Ensure Compatibility



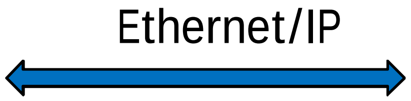
Ethernet-based Industrial Protocols

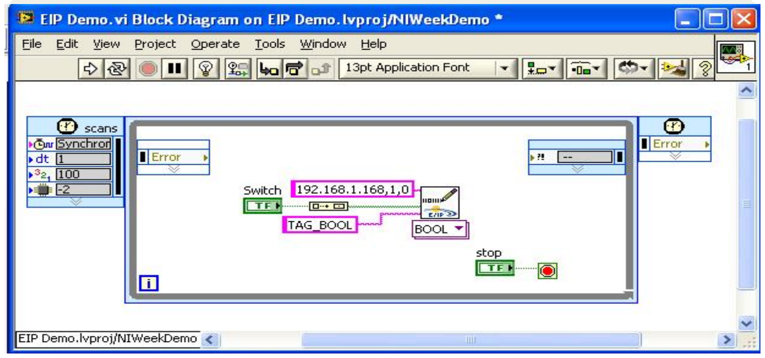
- Modbus TCP/IP
- Ethernet/IP
- EtherCAT
- Profinet

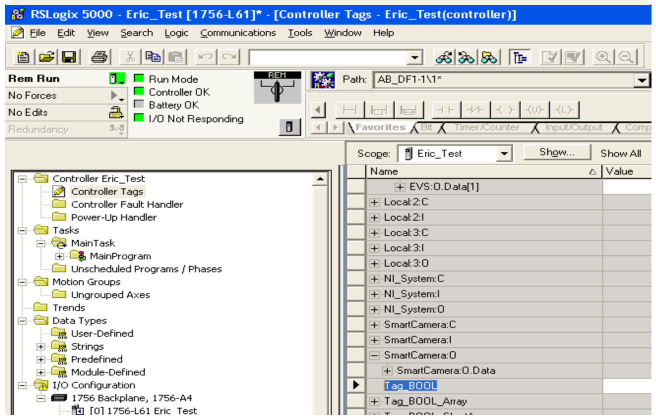
EtherNet/IP Overview

- Dominant bus for Rockwell Automation
  - Managed by Open Device Vendors Association (ODVA)
  - Extends DeviceNET concepts to Ethernet
- Advantages
  - Uses Ethernet transport layer (TCP and UDP)
- Disadvantages
  - Can overload networks with UDP messages if not correctly configured, recommend managed switches with IGMP snooping

Communication from NI PAC to ControlLogix and ComapctLogix PLCs




EIP Demo.vi Block Diagram on EIP Demo.lvproj/NIWeekDemo *
File Edit View Project Operate Tools Window Help
13pt Application Font
scans
Synchron
dt
1
100
-2
Error
Switch
192.168.1.168,1,0
TAG_BOOL
BOOL
stop
Error
EIP Demo.lvproj/NIWeekDemo


RSLogix 5000 - Eric_Test [1756-L61]* - [Controller Tags - Eric_Test(controller)]

File Edit View Search Logic Communications Tools Window Help

Rem Run | Run Mode
No Forces | Controller OK
No Edits | Battery OK
Redundancy | I/O Not Responding

REM

Path: AB_DF1-1\1*

Favorites | Bit | Timer/Counter | Input/Output | Comp

Scope: Eric_Test | Show... | Show All

- Controller Eric_Test
  - Controller Tags
  - Controller Fault Handler
  - Power-Up Handler
- Tasks
  - MainTask
    - MainProgram
  - Unscheduled Programs / Phases
- Motion Groups
  - Ungrouped Axes
- Trends
- Data Types
  - User-Defined
  - Strings
  - Predefined
  - Module-Defined
- I/O Configuration
  - 1756 Backplane, 1756-A4
    - [0] 1756-L61 Eric_Test

Name	Value
EVS:O.Data[1]	
Local:2:C	
Local:2:I	
Local:3:C	
Local:3:I	
Local:3:O	
NI_System:C	
NI_System:I	
NI_System:O	
SmartCamera:C	
SmartCamera:I	
SmartCamera:O	
SmartCamera:O.Data	
Tag_BOOL	
Tag_BOOL_Array	

Uses explicit messaging

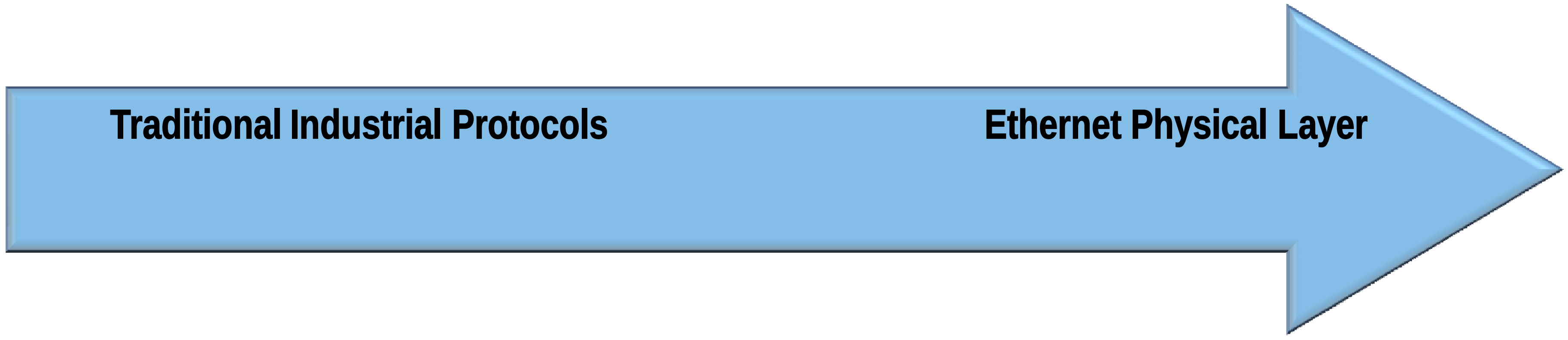
Available for download from NI Labs

EtherNet/IP VIs for LabVIEW

- Provides VIs for communication to “Logix” PLC Tags
  - Directly read and write tags on Allen Bradley ControlLogix and CompactLogix PLCs
- Runs on LabVIEW for Windows and LabVIEW Real-Time (Pharlap and VxWorks)
- Explicit Messaging
- Good for low numbers of tags

Why Ethernet: Protocols

$$\begin{tikzcd}
\text{Modbus} \arrow[r] & \text{Modbus TCP/IP} \\
\text{DeviceNet} \arrow[r] & \text{EtherNet/IP} \\
\text{CanOPEN} \arrow[r] & \text{EtherCAT} \\
\text{PROFIBUS} \arrow[r] & \text{PROFInet}
\end{tikzcd}$$



Motivations for IVI.NET

- Present an API more suited to .NET developers
- Simplify source code
  - Allow end users to understand instrument behavior by examining driver source
  - Allow end users to fix bugs on their own
  - Allow end users to add features to drivers on their own
- Richer, more expressive APIs
  - More flexibility with API data types
  - Clean handling of asynchronous notifications (aka "events")
- Side-by-side deployment of drivers
  - Only one version of an IVI-COM or IVI-C driver can be installed at a time
  - IVI.NET allows multiple versions of a driver to be installed

IVI-COM and IVI-C Driver Source

- IVI-COM and IVI-C drivers are both quite complicated internally
- Supporting IVI driver features requires a lot of code
  - Multi-thread safety
  - Simulation
  - Range-checking
  - State-caching
- “Basic” COM plumbing is complex and not well understood by many
- Multi-model driver support can be complicated
- Driver development tools are required, but can only do so much
  - Nimbus Driver Studio and LabWindows both work hard to factor as much code “out of the way”

Advanced Tooling for IVI.NET

- Many IVI-COM and IVI-C complaints tied to complex source code
- Tools have even more difficulty dealing with C/C++ source than humans
  - Includes files and macros are particularly problematic
  - Few really good C++ refactoring exist in any domain
- A prime motivator for .NET itself is the improved ability to create tooling
- Simpler source possible because .NET code can more easily be roundtripped
- Static analysis tools highlight issues at compile time that previously could only be detected at runtime
- Browsers can easily interrogate an IVI.NET driver and understand its features
- Declarative attributes can be used where procedural code was previously required
  - Achieved via “extending” the compiler (aka “code-weaving”)

Shared IVI.NET Data Types

- IVI Foundation felt it would be useful to offer commonly used data types as part of the IVI.NET Shared Components
  - Increase consistency amongst IVI.NET drivers
  - Facilitate data interchange between drivers
- Standardized IWaveform and ISpectrum interfaces
  - Digitizers and scopes and RF spectrum analyzers all read waveforms
  - Function generators and RF signal generators source waveforms
  - Without a common definition of a “waveform”, client applications would need to write the tedious code to translate between each class’s notion of a waveform
- Time-based parameters can use PrecisionDateTime and PrecisionTimeSpan
  - No confusion about ms vs sec, absolute vs relative time, UTC time, etc
  - Precision adequate for IEEE 1588 devices
- Common trigger source data type
  - Useful in “stitching” together devices in triggered source-measure operations

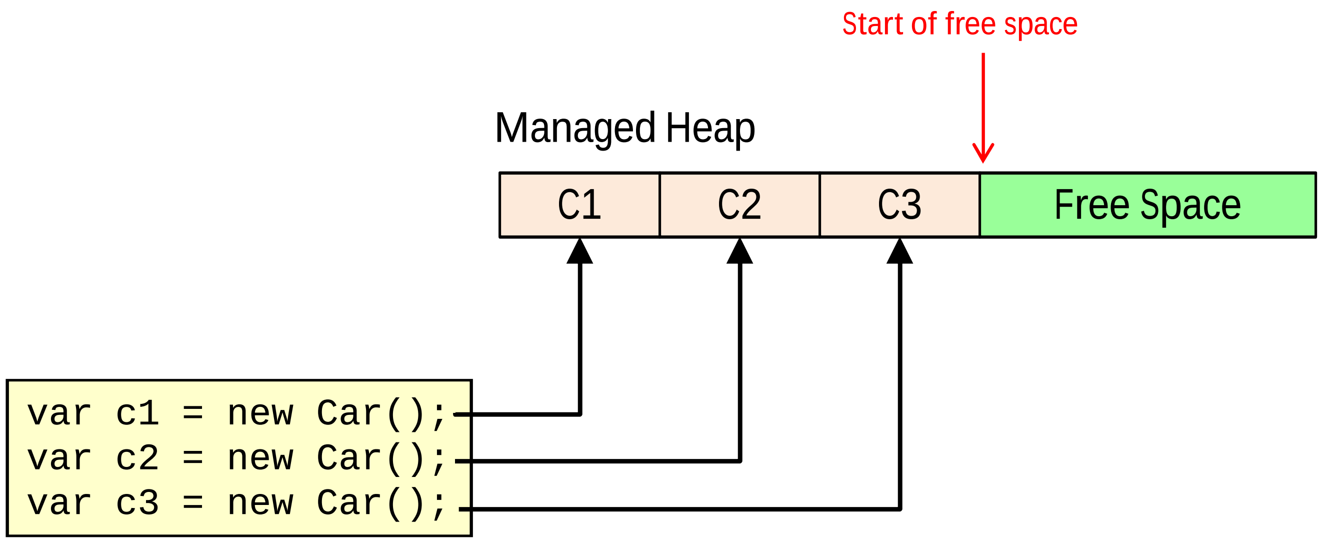
Error Handling in IVI.NET

- IVI-C drivers rely solely on return codes
  - Errors can easily be ignored by the client application
  - After getting the error code, a second function call is required to get the message
  - Special handling of warning codes required
- IVI-COM error code handling depends upon the client environment
  - Return codes in raw C++
  - Special exception classes in C++
  - COMException class in .NET interop scenarios
  - *.NET clients can't see warnings at all from IVI-COM drivers*
- IVI.NET drivers always use exceptions
  - User can always see the full context of the error
  - Error content less dependent upon specific driver implementation
  - Natural mechanism

Performance of IVI.NET

- Fewer memory leaks
- Reference counting has a cost
  - Reference count field per-object
  - Increment and decrement called much more frequently than one might think
  - Reference count field must be thread-safe
    - Even more per-object overhead
    - Frequently lock/unlock operations
- Conventional memory-managed systems (such as C-runtime library) produce highly fragmented memory
  - Allocation of objects can be expensive
  - Objects spread out in memory => poor locality of reference
- .NET memory allocation produces very good locality of reference
  - Object allocation extremely fast
  - Objects allocated close together in time live close together in memory
  - Fewer cache misses and better virtual paging performance

Dynamic Memory Allocation in .NET



```
var c1 = new Car();
var c2 = new Car();
var c3 = new Car();
```

UNIT-IV VI TOOLSETS

Definition

- All Periodic Waves Can be Generated by Combining Sin and Cos Waves of Different Frequencies
- Number of Frequencies may not be finite
- Fourier Transform Decomposes a Periodic Wave into its Component Frequencies

DFT Definition

- Sample consists of n points, wave amplitude at fixed intervals of time:
  $(p_0, p_1, p_2, \ldots, p_{n-1})$ (n is a power of 2)
- Result is a set of complex numbers giving frequency amplitudes for sin and cos components
- Points are computed by polynomial:
  $P(x) = p_0 + p_1 x + p_2 x^2 + \ldots + p_{n-1} x^{n-1}$

DFT Definition, continued

- The complete DFT is given by
  $P(1), P(w), P(w^2), \dots, P(w^{n-1})$
- w Must be a Primitive nth Root of Unity
- $w^n=1$, if $0<i<n$ then w^i ¹ 1

Primitive Roots of Unity

- w^i is an nth root of unity (not primitive)
- $w^{n/2} = -1$
- if 0£j£n/2-1 then $w^{(n/2)+j} = -w^j$
- if n is even and w is a primitive nth root of unity, then w^2 is a primitive n/2 root of unity
- Example: w = cos(2p/*n*) + *i*sin(2p/*n*)

$$\sum_{i=0}^{n-1} \omega^i = 0$$

Divide and Conquer

- Compute an n-point DFT using one or more n/2-point DFTs
- Need to find Terms involving w^2 in following polynomial
- $P(w)=p_0+p_1w+p_2w^2+p_3w^3+p_4w^4+ \dots +p_{n-1}\omega^{n-1}$

Here They Are

Windowing and Filtering

- Simplest way of designing FIR filters
- Method is all discrete-time no continuous-time involved
- Start with ideal frequency response

$$H_d(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h_d[n] e^{-j\omega n} \qquad h_d[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega}) e^{j\omega n} d\omega$$

- Choose ideal frequency response as desired response
- Most ideal impulse responses are of infinite length
- The easiest way to obtain a causal FIR filter from ideal is

$$h[n] = \begin{cases} h_d[n] & 0 \le n \le M \\ 0 & \text{else} \end{cases}$$

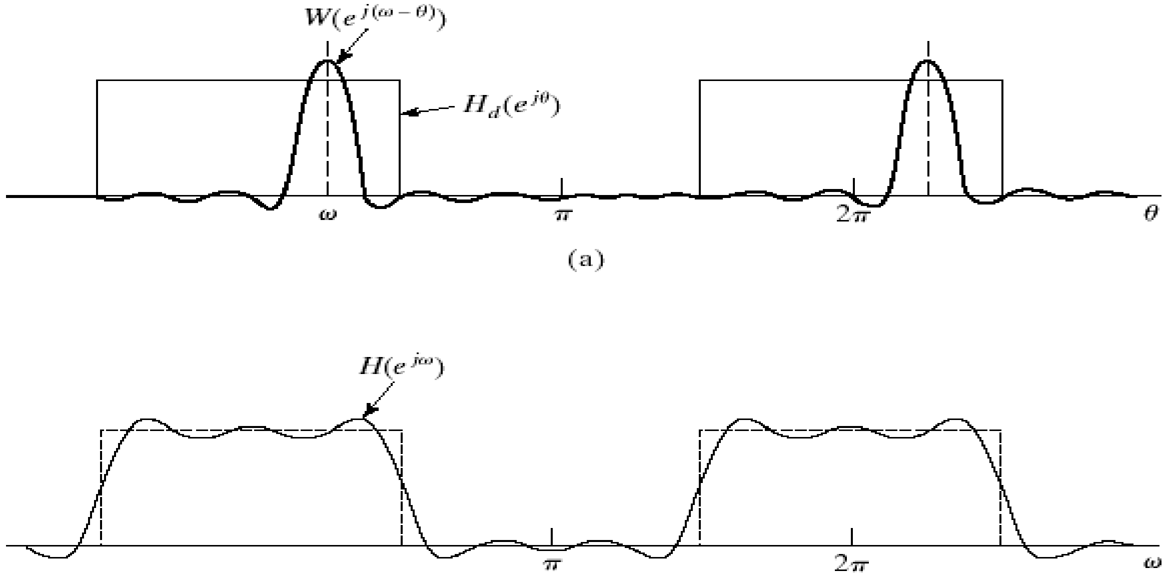
- More generally

$$h[n] = h_d[n]w[n] \quad \text{where} \quad w[n] = \begin{cases} 1 & 0 \le n \le M \\ 0 & \text{else} \end{cases}$$

Windowing in Frequency Domain

- Windowed frequency response $H(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega})W(e^{j(\omega-\theta)})d\theta$

- The windowed version is smeared version of desired response



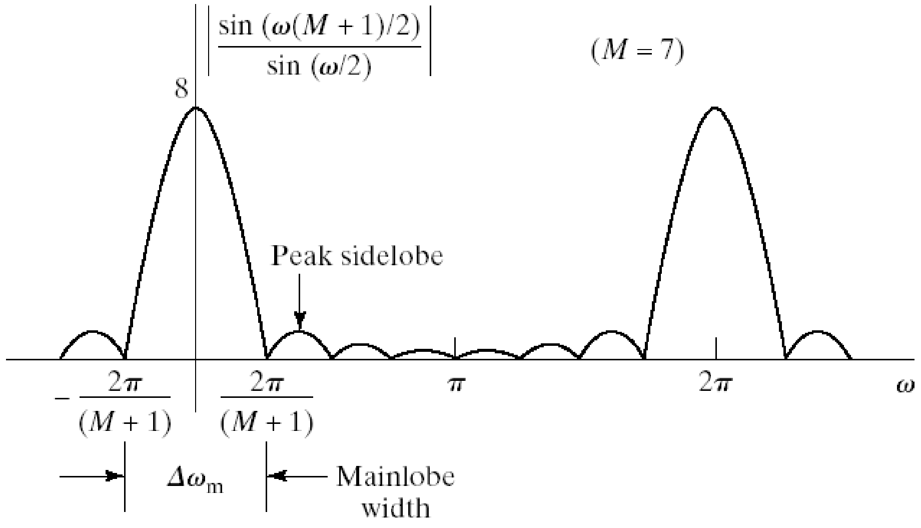
- If w[n]=1 for all n, then $W(e^{j\omega})$ is pulse train with 2π period

Properties of Windows

- Prefer windows that concentrate around DC in frequency
  - Less smearing, closer approximation
- Prefer window that has minimal span in time
  - Less coefficient in designed filter, computationally efficient
- So we want concentration in time and in frequency
  - Contradictory requirements
- Example: Rectangular window

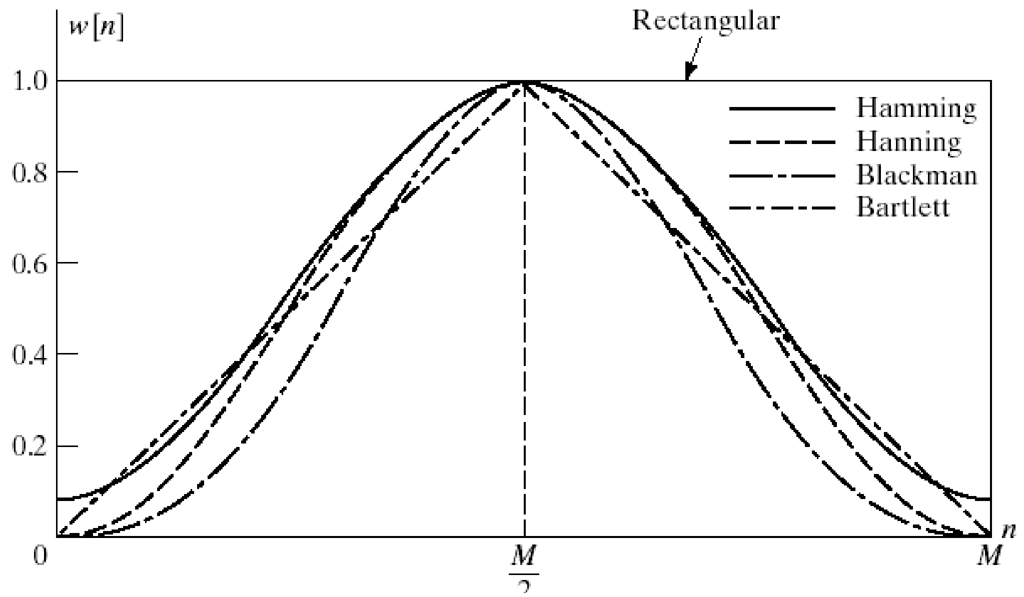
$$W\left(e^{j\omega}\right) = \sum_{n=0}^{M} e^{-j\omega n} = \frac{1 - e^{-j\omega(M+1)}}{1 - e^{-j\omega}} = e^{-j\omega M/2} \frac{\sin[\omega(M+1)/2]}{\sin[\omega/2]}$$

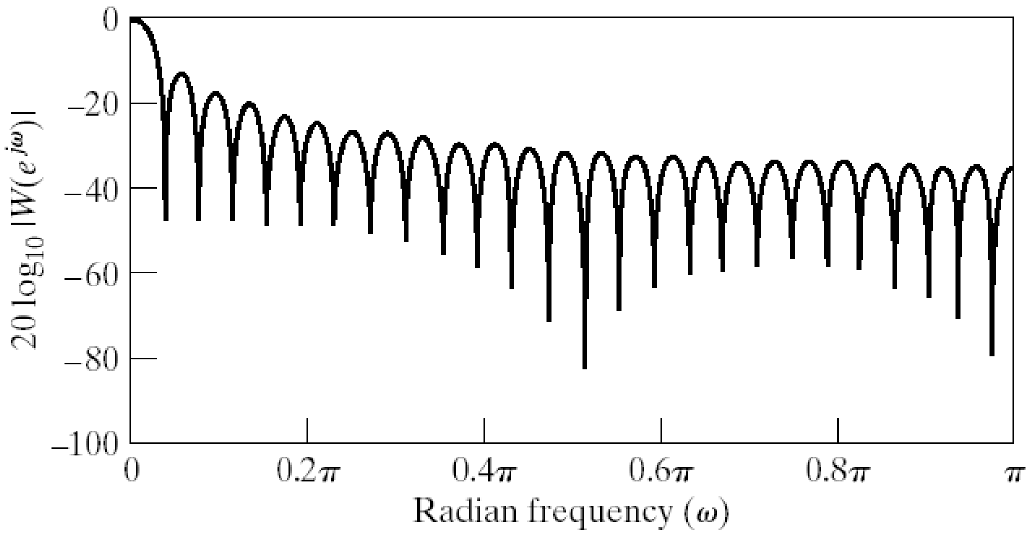
- Demo



Rectangular Window

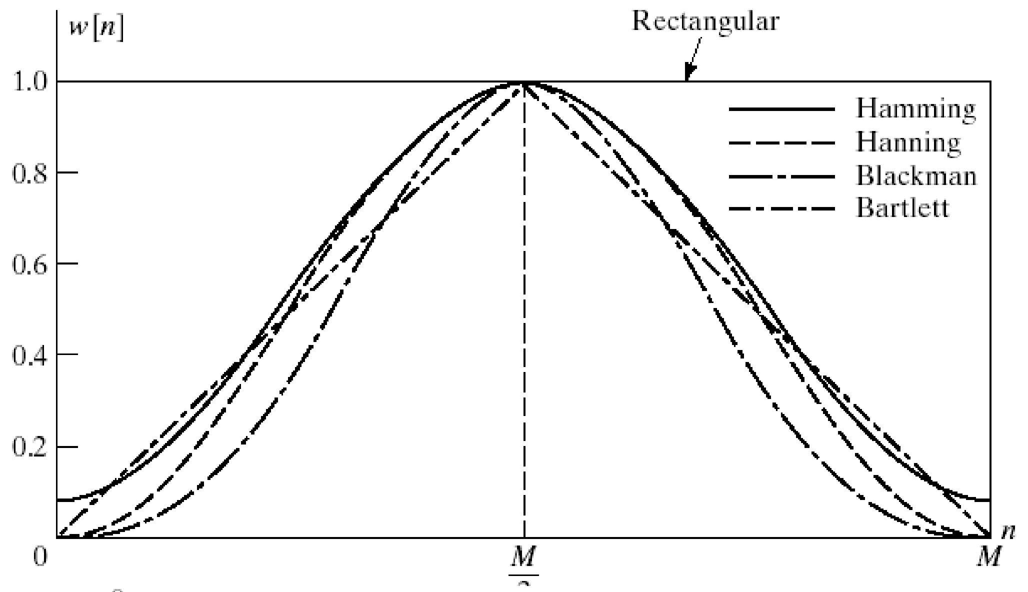
- Narrowest main lob
  - 4π/(M+1)
  - Sharpest transitions at discontinuities in frequency
- Large side lobs
  - -13 dB
  - Large oscillation around discontinuities
- Simplest window possible
- $$w[n] = \begin{cases} 1 & 0 \le n \le M \\ 0 & \text{else} \end{cases}$$

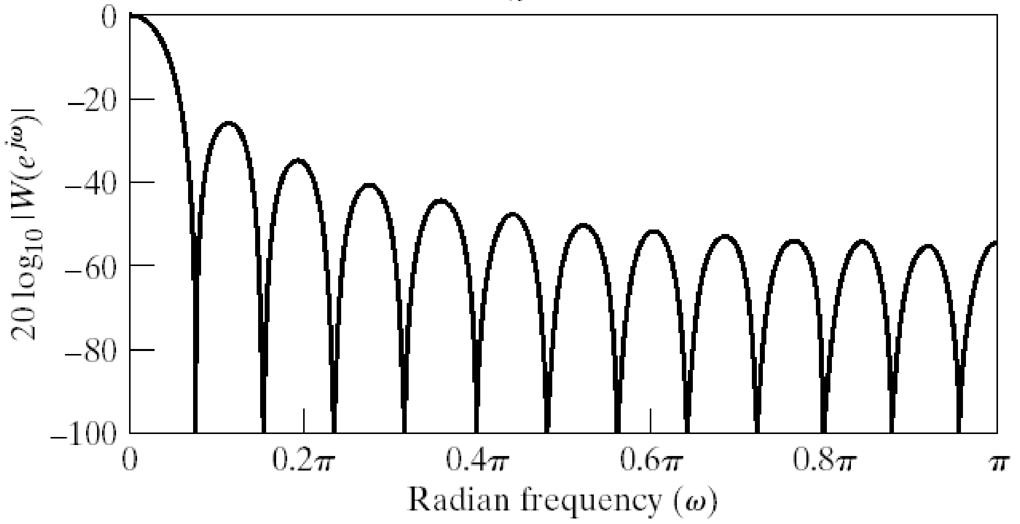




Bartlett (Triangular) Window

- Medium main lob
  - 8π/M
- Side lobs
  - -25 dB
- Hamming window performs better
- Simple equation

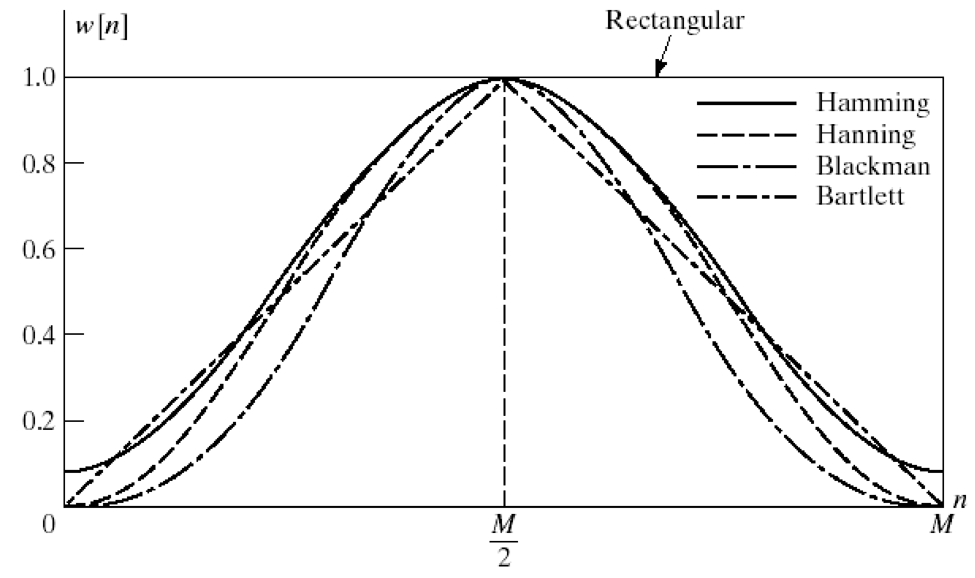


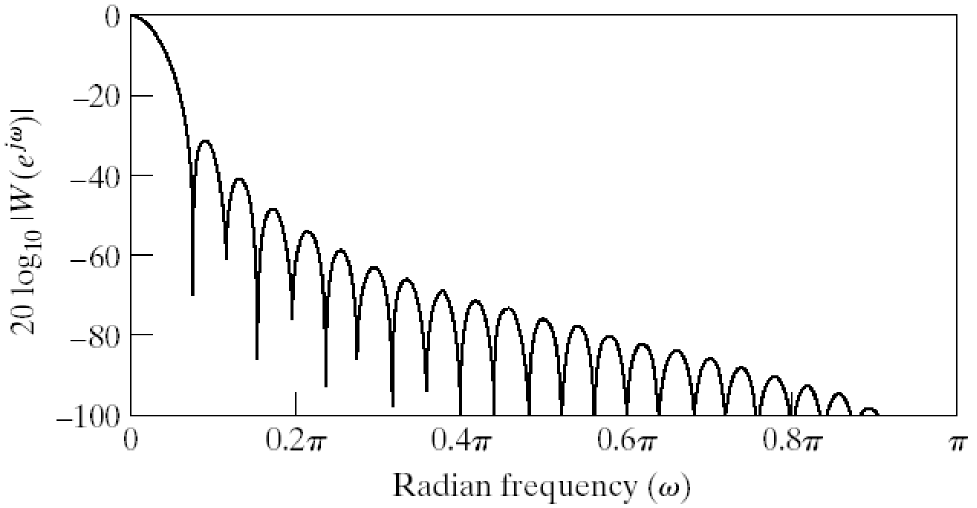


$$w[n] = \begin{cases} 2n/M & 0 \le n \le M/2 \\ 2 - 2n/M & M/2 \le n \le M \\ 0 & \text{else} \end{cases}$$

Hamming Window

- Medium main lob
  - 8π/M
- Side lobs
  - -31 dB
- Hamming window performs better
- Same complexity as Hamming

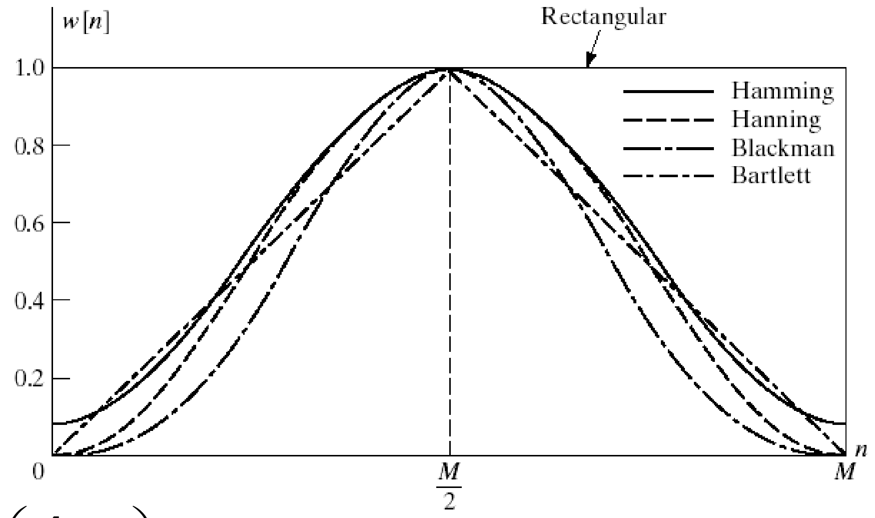




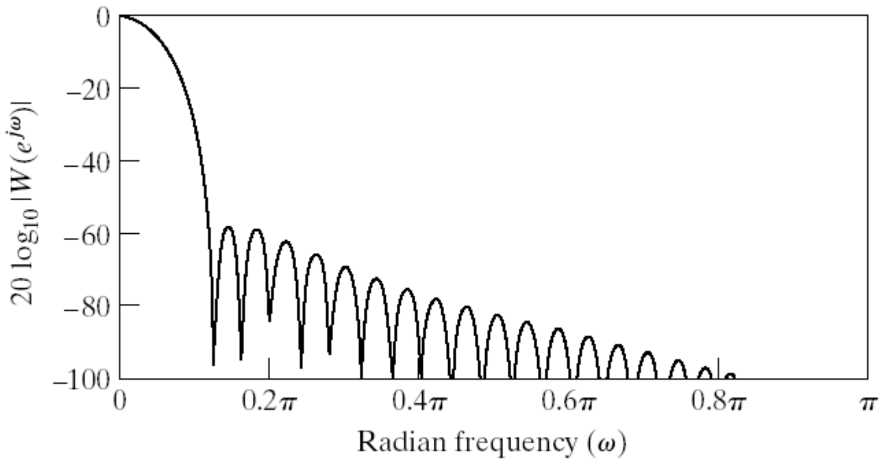
$$w[n] = \begin{cases} \frac{1}{2}\left[1 - \cos\left(\frac{2\pi n}{M}\right)\right] & 0 \le n \le M \\ 0 & \text{else} \end{cases}$$

Blackman Window

- Large main lob
  - 12π/M
- Very good side lobs
  - -57 dB
- Complex equation



$$w[n] = \begin{cases} 0.42 - 0.5\cos\left(\frac{2\pi n}{M}\right) + 0.08\cos\left(\frac{4\pi n}{M}\right) & 0 \le n \le M \\ 0 & \text{else} \end{cases}$$



Incorporation of Generalized Linear Phase

- Windows are designed with linear phase in mind
  - Symmetric around M/2

$$w[n] = \begin{cases} w[M-n] & 0 \le n \le M \\ 0 & \text{else} \end{cases}$$

$$W\left(e^{j\omega}\right) = W_e\left(e^{j\omega}\right)e^{-j\omega M/2} \qquad \text{where } W_e\left(e^{j\omega}\right) \text{is a real and even}$$

- So their Fourier transform are of the form

$$H_d\left(e^{j\omega}\right) = H_e\left(e^{j\omega}\right)e^{-j\omega M/2}$$

- Will keep symmetry properties of the desired impulse response
- Assume symmetric desired response

$$A_e\left(e^{j\omega}\right) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_e\left(e^{j\theta}\right)W\left(e^{j(\omega-\theta)}\right)d\theta$$

- With symmetric window

Linear-Phase Low pass filter

- Desired frequency response

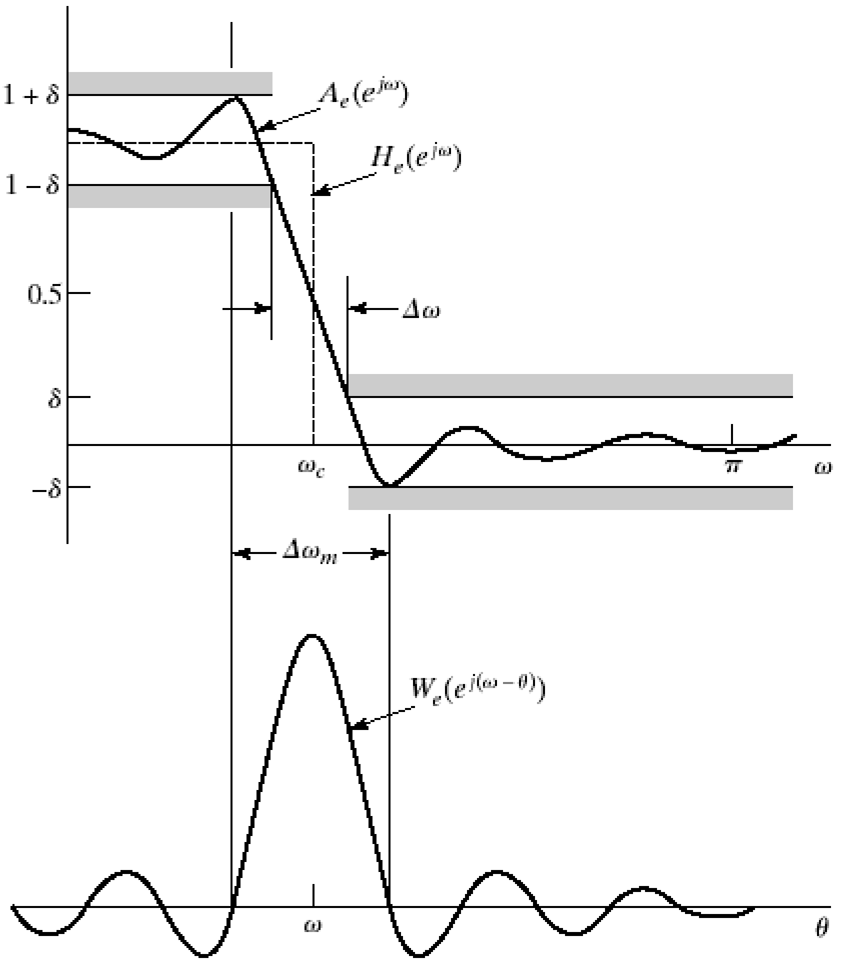
$$H_{lp}\left(e^{j\omega}\right) = \begin{cases} e^{-j\omega M/2} & |\omega| < \omega_c \\ 0 & \omega_c < |\omega| \le \pi \end{cases}$$

- Corresponding impulse response

$$h_{lp}[n] = \frac{\sin[\omega_c(n - M/2)]}{\pi(n - M/2)}$$

- Desired response is even symmetric, use symmetric window

$$h[n] = \frac{\sin[\omega_c(n - M/2)]}{\pi(n - M/2)} w[n]$$

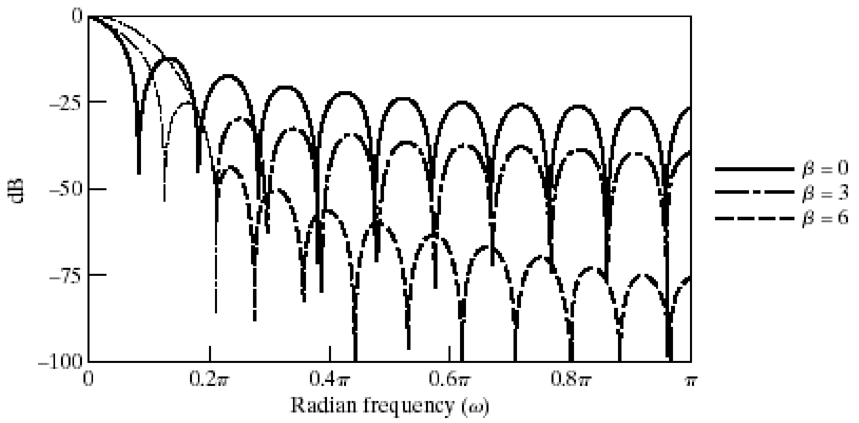


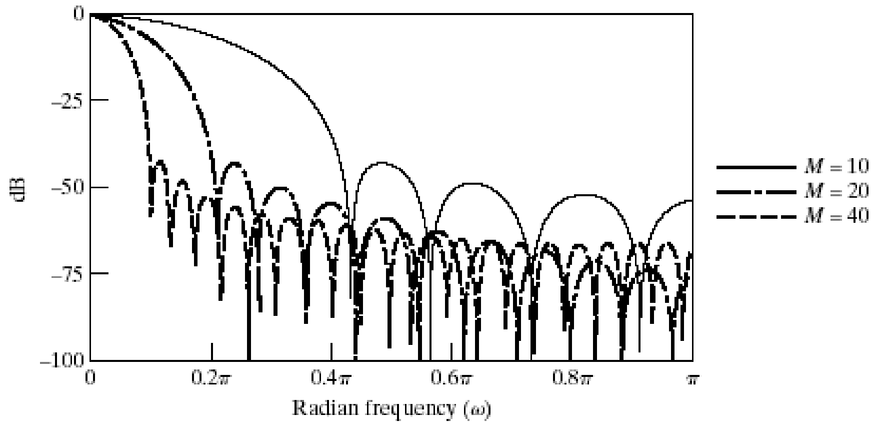
Kaiser Window Filter Design Method

- Parameterized equation forming a set of windows
  - Parameter to change main-lob width and side-lob area trade-off

$$w[n] = \begin{cases} \dfrac{I_0\left[\beta\sqrt{1-\left(\dfrac{n-M/2}{M/2}\right)^2}\right]}{I_0(\beta)} & 0 \le n \le M \\ 0 & \text{else} \end{cases}$$

  - $I_0(.)$ represents zeroth-order modified Bessel function of 1st kind





Determining Kaiser Window Parameters

- Given filter specifications Kaiser developed empirical equations

$$\Delta\omega = \omega_s - \omega_p$$

  - Given the peak approximation error δ or in dB as $A=-20\log_{10}\delta$
  - and transition band width
- The shape parameter β should be

$$\beta = \begin{cases} 0.1102(A-8.7) & A > 50 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21) & 21 \le A \le 50 \\ 0 & A < 21 \end{cases}$$

$$M = \frac{A-8}{2.285\Delta\omega}$$

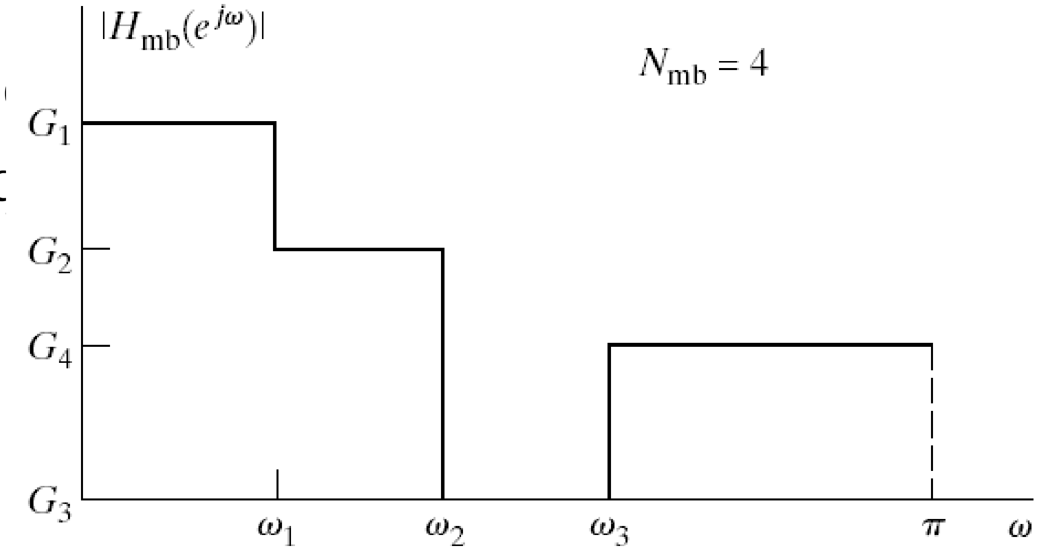
- The filter order M is determined approximately by

General Frequency Selective Filters

- A general multiband impulse response can be written as

$$h_{mb}[n] = \sum_{k=1}^{N_{mb}} \left(G_k - G_{k+1}\right) \frac{\sin \omega_k (n - M/2)}{\pi (n - M/2)}$$

- Window methods can b
- Example multiband frec
  - Special cases of
    - Bandpass
    - Highpass
    - Bandstop



Voltmeter is a measuring instrument, used to measure the voltage difference between two points in electrical network. Generally there are two types of voltmeters – one is analog voltmeter and the other one is digital voltmeter. In analog voltmeter a pointer moves on the scale to represent the voltage. Digital voltmeter directly displays the voltage in digits with the help of analog to digital converter. This article explains you how to design a digital voltmeter in two methods 1) using 8051 microcontroller and 2)the other using IC L7017.

Digital Voltmeter using 8051 Microcontroller

This project measures the input voltage from 0V to 5V. Here, the input voltage should be DC voltage to get the accurate output on LCD. If you apply AC voltage as input, then will see the continuous running numbers on LCD as AC varies continuously.

The major components in this project are 8051 microcontroller and ADC0804. In this project, we use analog to digital conversion process to display the voltage.

1. **Transducer:** Transducer or sensor is used to convert the physical quantity to electrical energy. Light dependent resistor, temperature sensor, humidity sensor, gas sensor etc. are examples of transducers.
2. **ADC (Analog to Digital Converter):** ADC converts the input electrical voltage to Digital value.
3. **Digital System:** this system reads input digital data and displays the physical quantity on LCD for understanding purpose.

re ADC IC generates the output digital value based on the input electrical voltage. The 8051 crocontroller reads this digital value and displays it on LCD.

How Digital Voltmeter Circuit Works using 8051 Microcontroller?

1. Initially burn the program to the at89c51 microcontroller.
2. Now give the connections as per the circuit diagram.
3. Connect variable resistor at the input of probes to vary the input analog input voltage.
4. make sure that maximum analog input voltage should be less than 5V DC
5. Connect a digital multi meter at the Pot to measure the input voltage.
6. Now switch on the board supply.
7. Now observe both LCD and digital multi meter, both displays the same voltage.
8. Slowly increase the analog input voltage by varying the pot, now you can observe that both multimeter and LCD displays the same voltages so that we can say that voltmeter is working properly.
9. Switch of the board supply.

Digital Voltmeter Circuit Applications

- This system is used to measure the voltage in low voltage applications.
- Used to measure the toy batteries.
- We can measure the physical quantities like temperature, humidity, gas etc. using this system with a little modification.

How to Operate Digital Voltmeter Circuit?

The IC is powered by a dual supply of +/- 5V. Once the circuit is powered, the reference signal is set by adjusting the reference resistor. The reference voltage needs to be about half of the input voltage. The oscillating components – resistor and capacitor determine the oscillating or clock frequency of the device. The reference capacitor is charged to the reference voltage. A feedback loop is then closed to charge the auto zero capacitor such that is compensates for any fluctuations in voltages. Later the converter integrates the differential voltage at the input for a fixed time such that output of the integrator is a ramp signal. A known reference voltage is then applied to the input of integrator and is allowed to ramp till the output of integrator becomes zero. The time taken for the output to return back to zero is proportional to the input signal and the digital reading is given as:

Display Count = (Vin/Vref)*1000.

The next process involves decoding the digital count to produce a seven segment compatible signal so as to drive the displays. The digital output is then displayed on the multiplexed 7-segment display.

Applications of Digital Voltmeter Circuit

1. This circuit can be used in digital multimeters to provide digital reading of measured voltage.
2. It can be used to measure AC and DC voltages.
3. It can be used to measure physical quantities like pressure, temperature, stress using transducer circuit and signal conditioning circuit.
4. It can be used in applications where high accuracy and high resolution is required.

Limitations of Digital Voltmeter Circuit

1. It can measure voltages only up to a low range.
2. The IC used is a CMOS device and is highly static.
3. Difference in reference voltage for negative and positive input voltage can cause rollover error, i.e. a common mode error.
4. Using a full scale negative input voltage of 2V can sometimes cause output of the integrator to saturate.
5. Internal heating from the LED drivers can cause degradation in performance.
6. Reference temperature coefficient, internal chip dissipation and package thermal resistance tend to increase the noise level.

UNIT-V APPLICATIONS

Digital

- Storing information often in a series of 1’s and 0’s or binary numbers. The process can be used to do calculations or sending pulses to regulate instrumentation or other electronic equipment turning it on/off or regulating the use of materials such as the flow of liquid through a valve. In instrumentation and process control this accomplished by concerting an analogue signal into a digital signal to control the process. A non-electronic example would be smoke signals or a beacon.

Analogue

- An analog or analogue signal is any continuously variable signal. It differs from a digital signal in that small fluctuations in the signal are meaningful within a given scale range from a small to large signal. Analog is usually thought of in an electrical context, however mechanical, pneumatic, hydraulic, and other systems may also use analog signals.
- A great example of an analogue device is a Wrist Watch with hands that move.

Binary

- Having the base of 2 for number system with two digits 0 and 1. Basis of electronic signal signals used in computers. Creates two states for the binary signal on or off, 0 being off and 1 being on. There is no state in between the device is either on or off. Often referred to as Boolean Logic

Microprocessor

- A silicon based processing chip or logic chip designed as the heart of the computer, contains all the necessary information to run a computer speed measured in megahertz (MHz) or gigahertz (GHz). These chips have areas for comparing numbers or doing calculations called registers.

- Good example: Digital Clocks and Wrist Watches

Fuzzy Logic

- The ability of a machine to answer questions that are not yes or no questions. Fuzzy logic use 0 and 1 as the extremes of yes and no but answers the degrees of maybe. Fuzzy logic works much closers to that of the human brain. It is subset of Boolean logic use to fill the concepts of a partial truth. An example of such is a half full glass of water is .50 of full.

Neural System

- In information technology, a neural network is a system of programs and data structures that approximates the operation of the human brain. A neural network usually involves a large number of processors operating in parallel, each with its own small sphere of knowledge and access to data in its local memory.

- Good example: Joystick for a computer game

Sensors

- Devices such as a photocell that respond to a signal or stimulus. A device that measures or detects a real-world condition, such as motion, heat or light and converts the condition into an analog or digital representation. These devices are use in manufacturing plants to tell how many items are in a package such as the example CD’s to fill a container for packaging or to the number of containers fill case for shipment.
- Good examples: motion detectors and burglar alarms

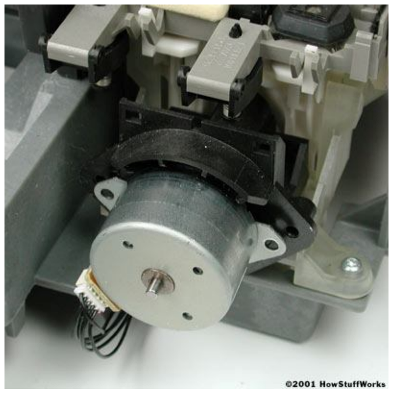
Actuators

- One that activates, especially a device responsible for actuating a mechanical device, such as one connected to a computer by a sensor link.
- An **actuator** is the mechanism by which an agent acts upon an environment. The agent can be either an artificial or any other autonomous being (human, other animal, etc).
- Examples: human hand, leg, arm, Part Picking Robot, Switches

Stepper Motors

- A mechanism that causes a device to be turned on or off, adjusted or moved. The motor and mechanism that moves the head assembly on a disk drive or an arm of a robot is called an actuator.

A good example printer motor moves the laser head cartridge across the paper.



Synchro Motor

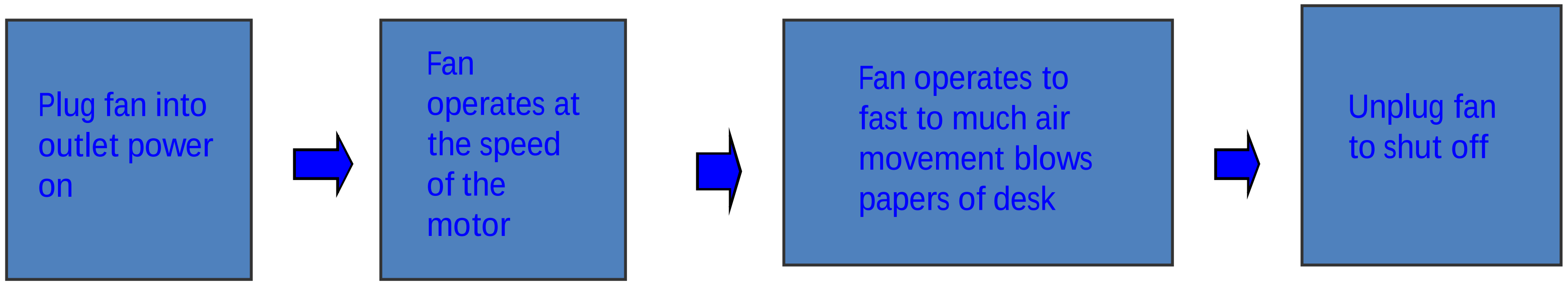
- A type of rotary transformer fixed to rotor which attached to a motor and can be adjusted. The current is adjusted to keep the rotor and motor operating at a synchronized speed. The result of this action causes the parts to work in unison.

- Good Examples: the gun turret on a naval destroyer and the film and sound of older movies before microelectronics.

Open-loop Control

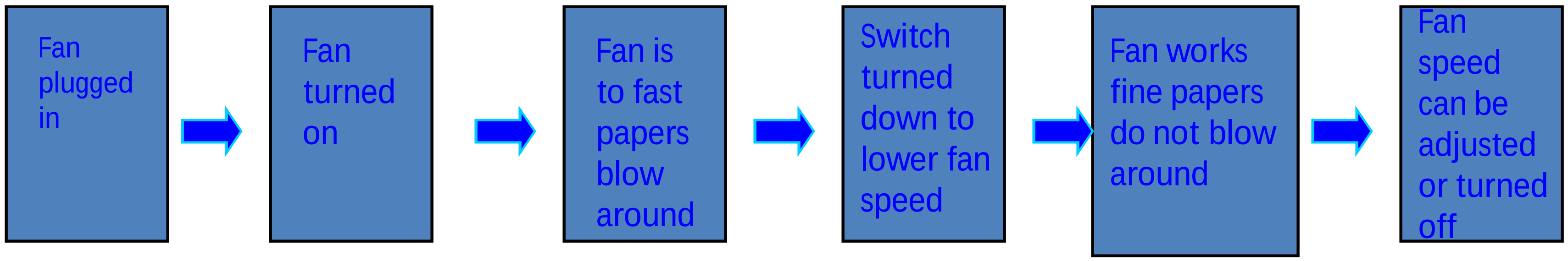
- A control loop operated by human intervention or does not have a feedback loop to self adjust.

Example A fan that plugs into the wall with no switch to turn on or off.


Plug fan into outlet power on
Fan operates at the speed of the motor
Fan operates to fast to much air movement blows papers of desk
Unplug fan to shut off

Closed-loop Control

- A control-loop operated by a feedback loop allowing self adjusting o the loop.” A mechanical, optical, or electronic system that is used t maintain a desired output.”
- Good example: Fan with a switch to allow the speed to be changed


Fan plugged in
Fan turned on
Fan is to fast papers blow around
Switch turned down to lower fan speed
Fan works fine papers do not blow around
Fan speed can be adjusted or turned off

Instrumentation

- Instrumentation is defined as "the art and science of measurement and control". Instrumentation is used to refer to the field in which Instrument technicians and engineers work. Instrumentation also can refer to the available methods of measurement and control

- Good example: the gauges that control the boilers for the school heating system

Development of process database management system

1. Basic Definitions

Database: A collection of related data.

Data: Known facts that can be recorded and have an implicit meaning.

Mini-world: Some part of the real world about which data is stored in a database. For example, consider student names, student grades and transcripts at a university.

Database Management System (DBMS): A software package/ system to facilitate the creation and maintenance of a computerized database.

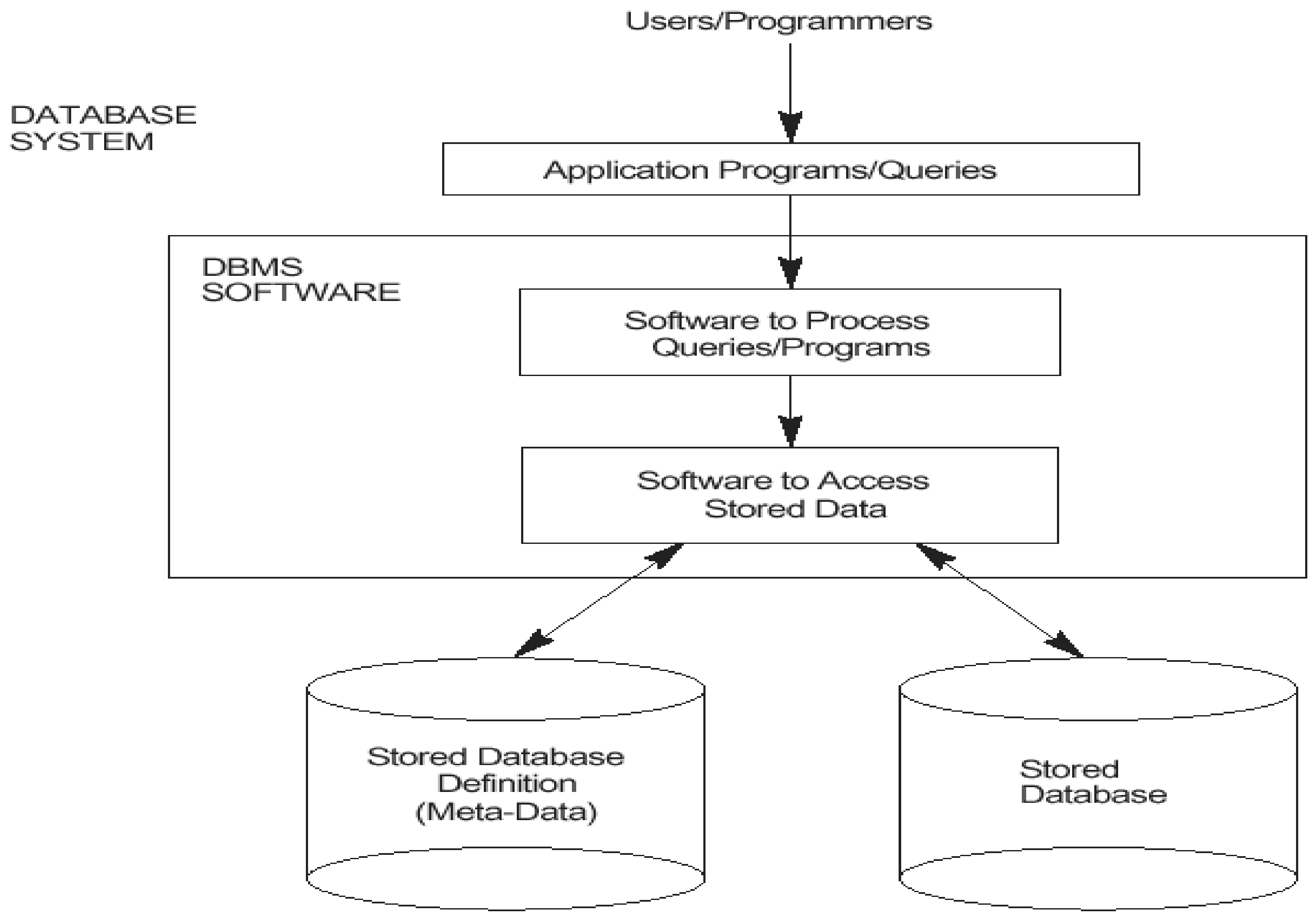
It

- defines (data types, structures, constraints)
- construct (storing data on some storage medium

controlled by DBMS)

- manipulate (querying, update, report generation) databases for various applications.

Database System: The DBMS software together with the data itself. Sometimes, the applications are also included.


Users/Programmers
DATABASE SYSTEM
Application Programs/Queries
DBMS SOFTWARE
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database

2.File Processing and DBMS

File Systems :

- Store data over long periods of time
- Store large amount of data

However :

- No guarantee that data is not lost if not backed up
- No support to query languages
- No efficient access to data items unless the location is known
- Application depends on the data definitions (structures)
- Change to data definition will affect the application programs
  - Single view of the data
  - Separate files for each application
  - Limited control to multiple accesses
  - Data viewed as physically stored

3. Main Characteristics of Database Technology

- Self-contained nature of a database system: A DBMS **catalog** stores the description (structure, type, storage format of each entities) of the database. The description is called **meta-data**). This allows the DBMS software to work with different databases.

- Insulation between programs and data: Called **program-data independence**. Allows changing data storage structures and operations without having to change the DBMS access programs.

- Data Abstraction: A **data model** is used to hide storage details and present the users with a conceptual view of the database; does not include how data is stored and how the operations are implemented.

-

- Support of multiple views of the data: Each user may see a different view of the database, which describes only the data of interest to that user.

- Sharing of Data and Multiple users

DBA – Database Administrator

- Responsible for authorizing access to the database, coordinating, monitoring its use, acquiring hardware, software needed.

Database designers

- Responsible for identifying the data to be stored, storage structure to represent and store data. This is done by a team of professionals in consultation with users, and applications needed.

4. Additional Benefits of Database Technology

- Controlling redundancy in data storage and in development and maintenance efforts.
- Sharing of data among multiple users.
- Restricting unauthorized access to data.
- Providing multiple interfaces to different classes of users.
- Representing complex relationships among data.
- Enforcing integrity constraints on the database.
- Providing backup and recovery services.
- Potential for enforcing standards.
- Flexibility to change data structures.
- Reduced application development time.
- Availability of up-to-date information.
- Economies of scale.

5 When <u>not</u> to use a DBMS

Main inhibitors (costs) of using a DBMS:

- High initial investment and possible need for additional hardware.
- Overhead for providing generality, security, recovery, integrity, and concurrency control.

When a DBMS may be unnecessary:

- If the database and applications are simple, well defined, and not expected to change.
- If there are stringent real-time requirements that may not be met because of DBMS overhead.
- If access to data by multiple users is not required.

When no DBMS may suffice:

- If the database system is not able to handle the complexity of data because of modeling limitations
- If the database users need special operations not supported by the DBMS.

6. **Data Models**

Data Model: A set of concepts to describe the structure (data types, relationships) of a database, and certain constraints that the database should obey.

Data Model Operations: Operations for specifying database retrievals and updates by referring to the concepts of the data model.

DBMS Languages

Data Definition Language (**DDL**): Used by the DBA and database designers to specify the **conceptual schema** of a database.

In many DBMSs, the DDL is also used to define internal and external schemas (views). In some DBMSs, separate **storage definition language** (**SDL**) and **view definition language** (**VDL**) are used to define internal and external schemas.

Data Manipulation Language (**DML**): Used to specify database retrievals and updates.

- DML commands (**data sublanguage**) can be embedded in a general-purpose programming language (**host language**), such as COBOL, PL/1 or PASCAL.
- Alternatively, stand-alone DML commands can be applied directly (**query language**).

High Level or non-Procedural DML – Describes what data to be retrieved rather than how to retrieve.

- Process many records at a time
- SQL

Low Level or Procedural DML – It needs constructs for both, what to retrieve and what to

retrieve

- Uses looping etc. like programming languages

Only access one record at a time

DBMS Interfaces

- Stand-alone query language interfaces.
- Programmer interfaces for embedding DML in programming languages:
  - Pre-compiler Approach
  - Procedure (Subroutine) Call Approach
- User-friendly interfaces:
  - Menu-based
  - Graphics-based (Point and Click, Drag and Drop etc.)
  - Forms-based
  - Natural language
  - Combinations of the above
  - Speech as Input (?) and Output
  - Web Browser as an interface
-

Classification of DBMSs

Based on the data model used:

- Traditional: Relational, Network, Hierarchical.
- Emerging: Object-oriented, Object-relational.

Other classifications:

- Single-user (typically used with micro- computers) vs. multi-user (most DBMSs).
- Centralized (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases)

Distributed Database Systems have now come to be known as <u>client server based database systems</u> because they do not support a totally distributed environment, but rather a set of database servers supporting a set of clients.

Simulation of systems using VI

SPICE
Introduction

- SPICE
  - Simulation Program with Integrated Circuit Emphasis
  - Developed at University of California at Berkeley
  - Three revisions, SPICE-3F5 is current

- Other circuit simulation technologies
  - XSPICE – behavioral SPICE – combines SPICE with component behavior in C
  - VHDL – Programmable Logic Design
  - IBIS – Used to model transfer function of sophisticated components (A/Ds, etc…)
  - PSpice®, HSPICE™ – commercial variations of the Berkeley SPICE.
  - RF with Electromagnetic Field Solvers (Agilent Advanced Design System™ or Ansoft Designer ®)

SPICE Primer

- SPICE Circuit
  - Built by creating a *netlist* of native SPICE primitive models.
  - Netlist is a text file that lists all connections and model information.
  - Schematic File
    - Vendor specific
    - May include package, footprint, and additional information
  - SPICE adds analysis commands on top of SPICE file allowing a SPICE simulation to extract information out of circuit (Transient, AC, Monte Carlo etc…)
- Variety of native SPICE components:
  - Resistors, Capacitors, Inductors, Sources, Transistors, etc…

Advantages to Using SPICE with Virtual Instrumentation

Mathematical capabilities of SPICE to accurately model complex circuits and devices

- AND –

Measurement capabilities of Virtual Instrumentation (such as data collection, automation, testing, etc)

SPICE

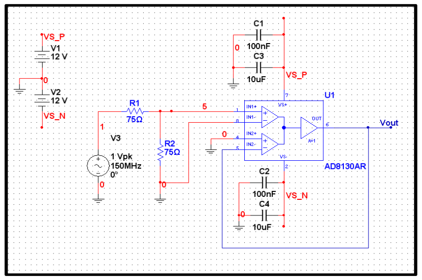
Schematic, Simulation, Analysis

Virtual Prototype

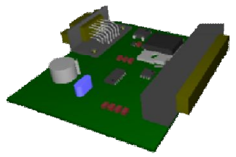
Testing

VI Software

Virtual Measurements



Comparison between simulation data and measurements is simplified

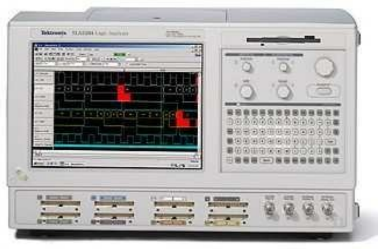


Measurements

Physical Measurements

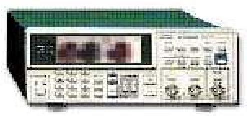
Simulation and Measurements for Design Engineers

- How do you effectively compare test bench data with simulation data?
- How can you bring in measurement data into simulation?
- Is there anyway to perform simulations, compare results and optimize the design automatically?



Logic Analyzer

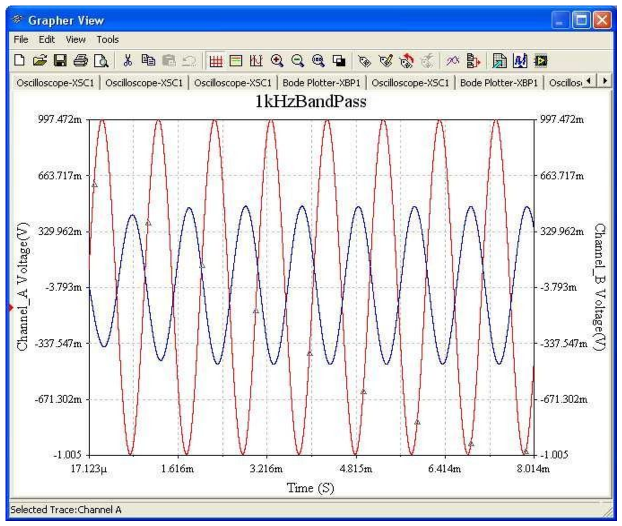
Scope



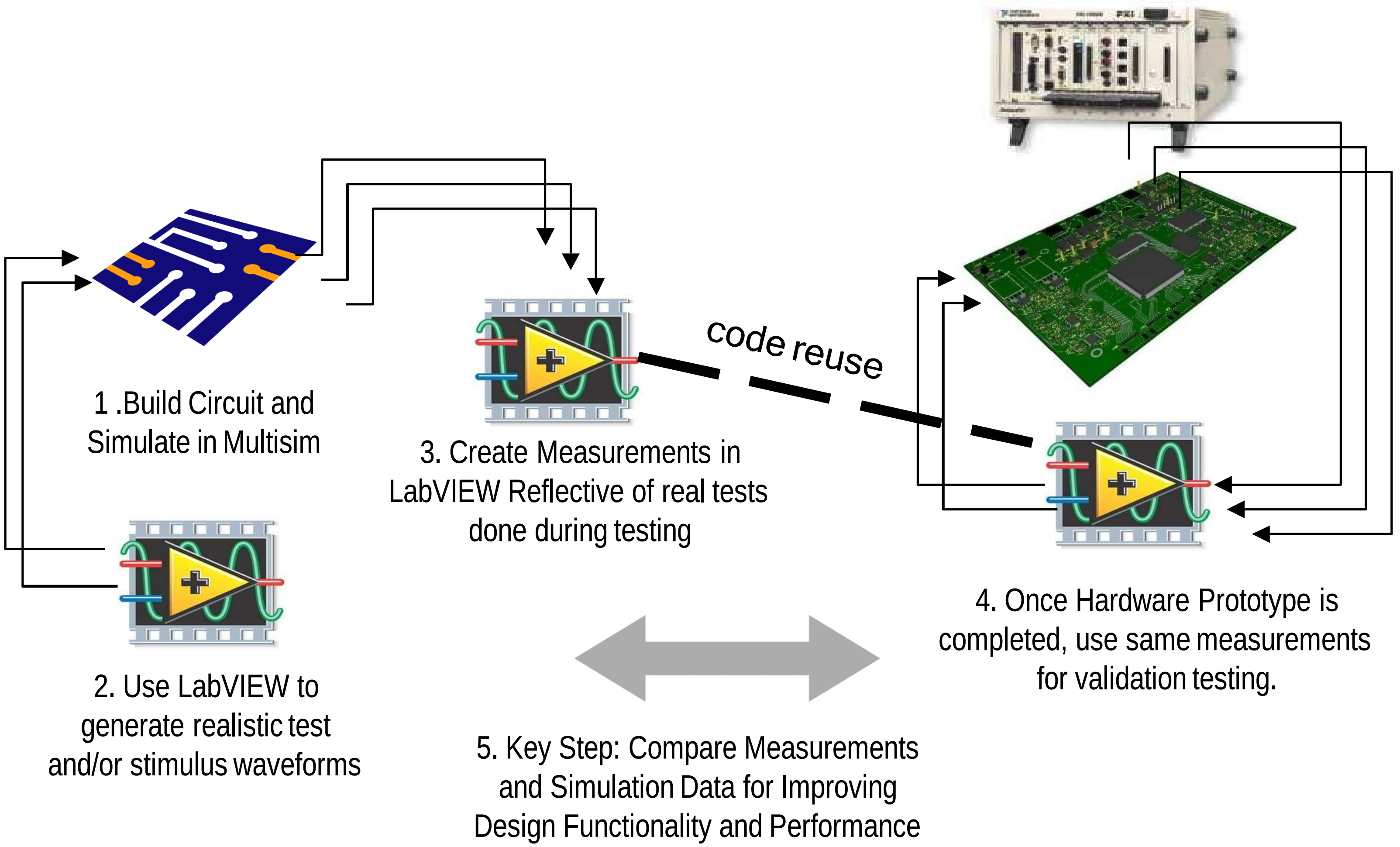
Power Supply

Function Generator

DMM



Multisim and LabVIEW Integration


code reuse
1 .Build Circuit and Simulate in Multisim
2. Use LabVIEW to generate realistic test and/or stimulus waveforms
3. Create Measurements in LabVIEW Reflective of real tests done during testing
4. Once Hardware Prototype is completed, use same measurements for validation testing.
5. Key Step: Compare Measurements and Simulation Data for Improving Design Functionality and Performance

Development of Control system

In this chapter we describe a general process for designing a control system.

A control system consisting of interconnected components is designed to achieve a desired purpose. To understand the purpose of a control system, it is useful to examine examples of control systems through the course of history. These early systems incorporated many of the same ideas of feedback that are in use today.

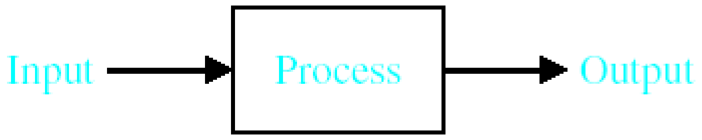
Modern control engineering practice includes the use of control design strategies for improving manufacturing processes, the efficiency of energy use, advanced automobile control, including rapid transit, among others.

We also discuss the notion of a design gap. The gap exists between the complex physical system under investigation and the model used in the control system synthesis.

System – An interconnection of elements and devices for a desired purpose.

Control System – An interconnection of components forming a system configuration that will provide a desired response.

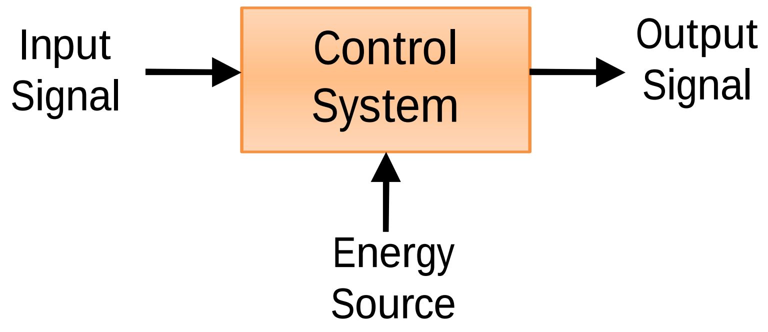
Process – The device, plant, or system under control. The input and output relationship represents the cause-and-effect relationship of the process.



Process to be controlled.

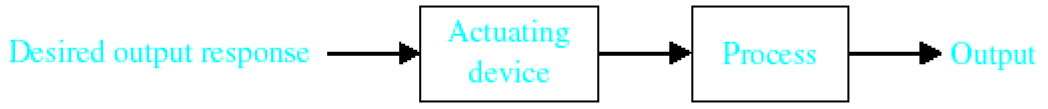
Control System

- Control is the process of causing a system variable to conform to some desired value.
- Manual control $\rightarrow$ Automatic control (involving machines only).
- A control system is an interconnection of components forming a system configuration that will provide a desired system response.

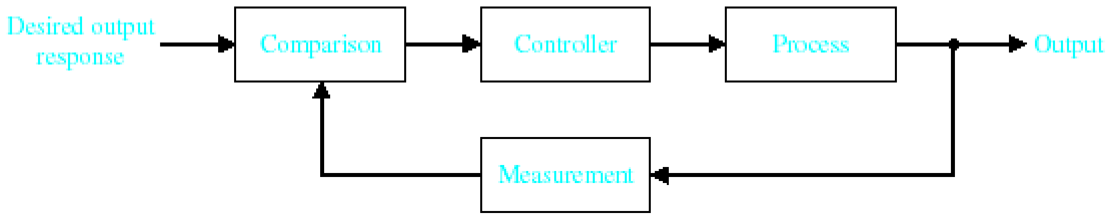


Open-Loop Control Systems utilize a controller or control actuator to obtain the desired response.

Closed-Loop Control Systems utilizes feedback to compare the actual output to the desired output response.

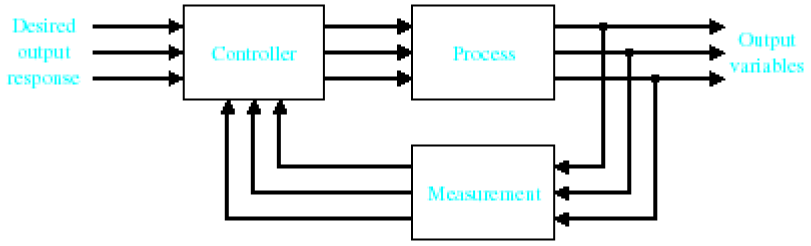


Open-loop control system (without feedback).



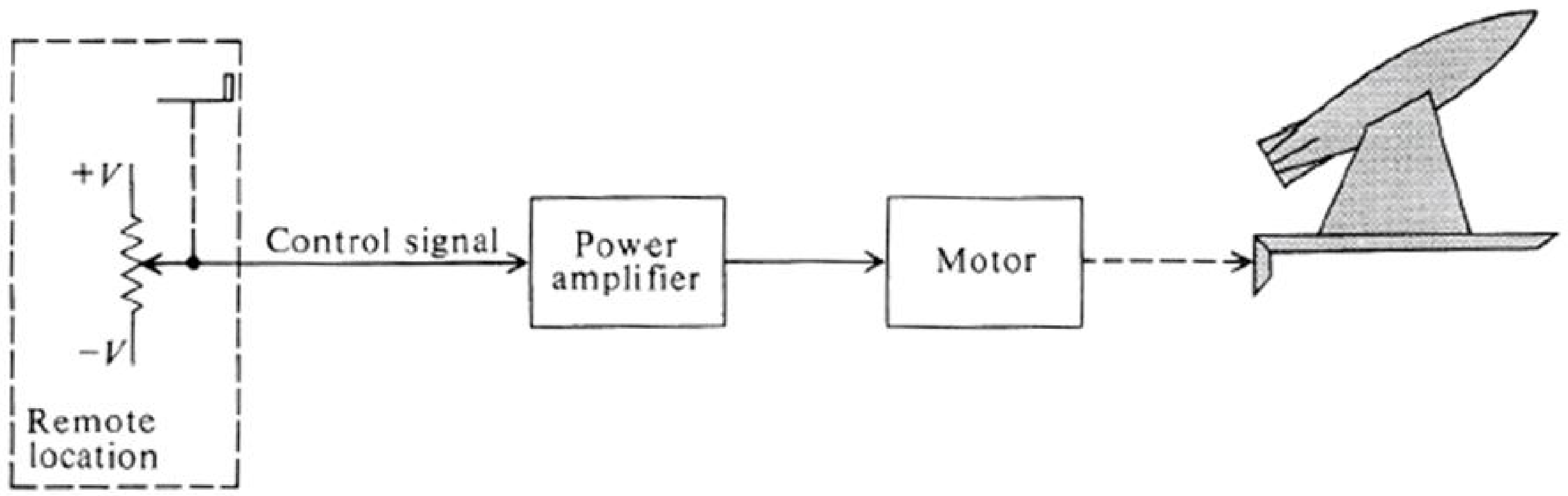
Closed-loop feedback control system (with feedback).

Multivariable Control System



Control System Classification

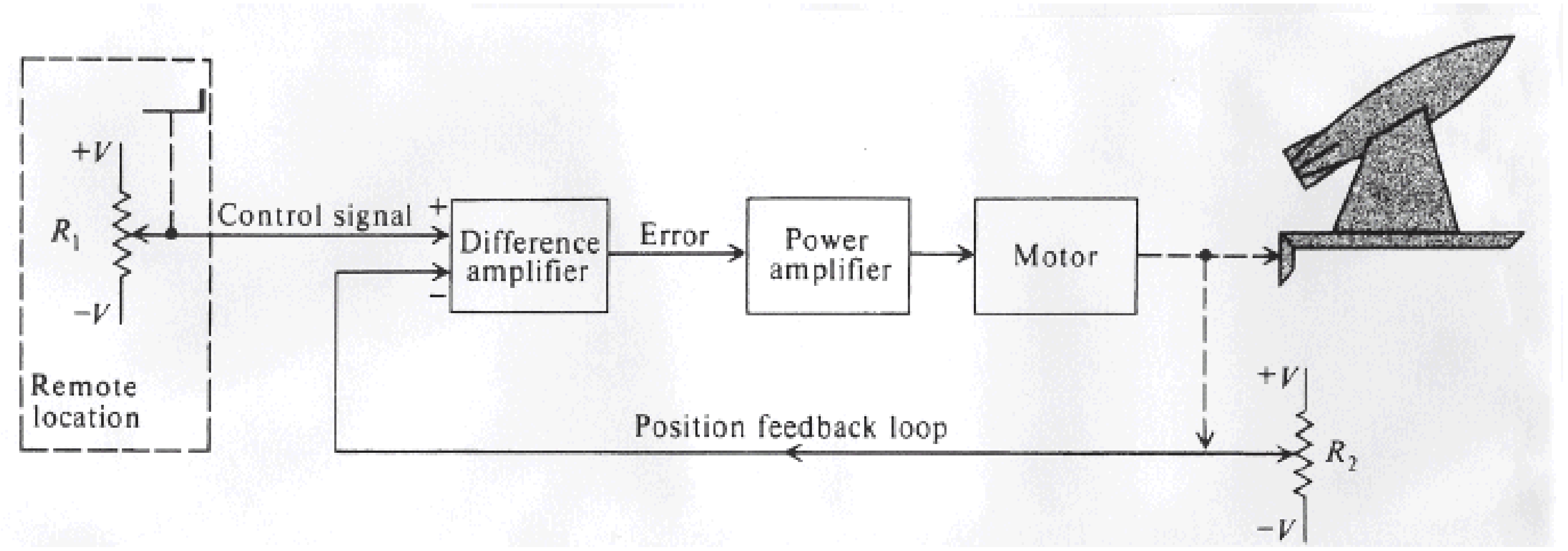
Missile Launcher System



Open-Loop Control System

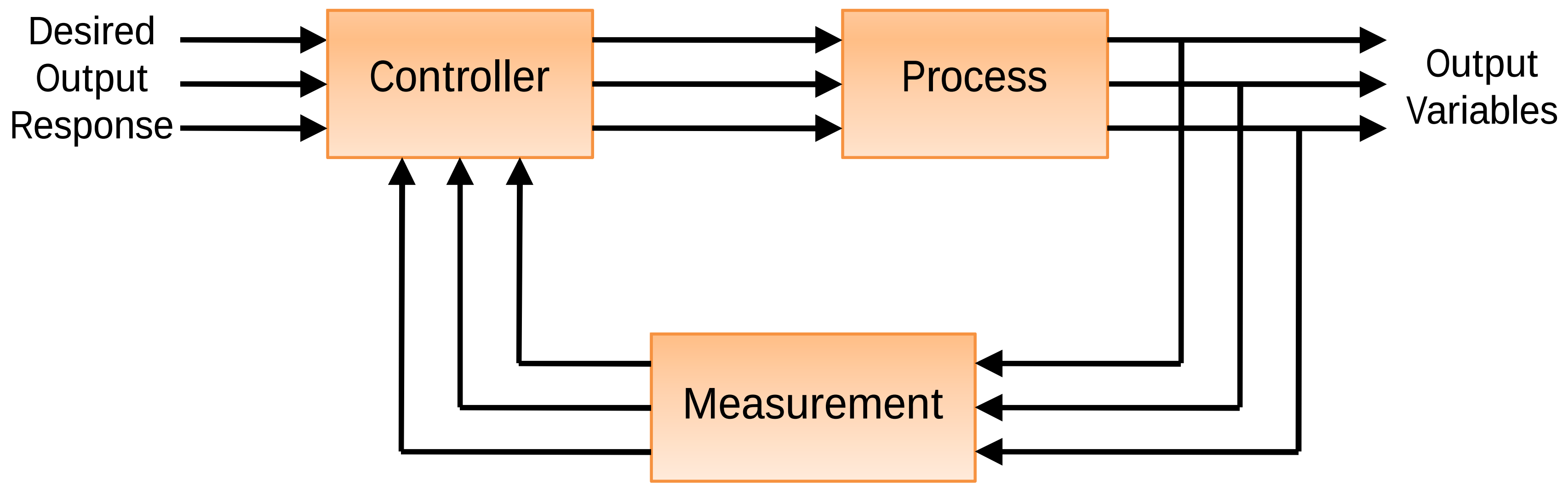
Control System Classification

Missile Launcher System



Closed-Loop Feedback Control System

Control System Classification



Multi Input Multi Output (MIMO) System

Purpose of Control Systems

i. Power Amplification (Gain)
   - Positioning of a large radar antenna by low-power rotation of a knob

ii. Remote Control
   - Robotic arm used to pick up radioactive materials

iii. Convenience of Input Form
   - Changing room temperature by thermostat position

iv. Compensation for Disturbances
   - Controlling antenna position in the presence of large wind disturbance torque

Historical Developments

i. Ancient Greece (1 to 300 BC)
   - Water float regulation, water clock, automatic oil lamp

ii. Cornellis Drebbel (17th century)
   - Temperature control

iii. James Watt (18th century)
   - Flyball governor

iv. Late 19th to mid 20th century
   - Modern control theory

Control System Components

i. System, plant or process
   - To be controlled

ii. Actuators
   - Converts the control signal to a power signal

iii. Sensors
   - Provides measurement of the system output

iv. Reference input
   - Represents the desired output

General Control System

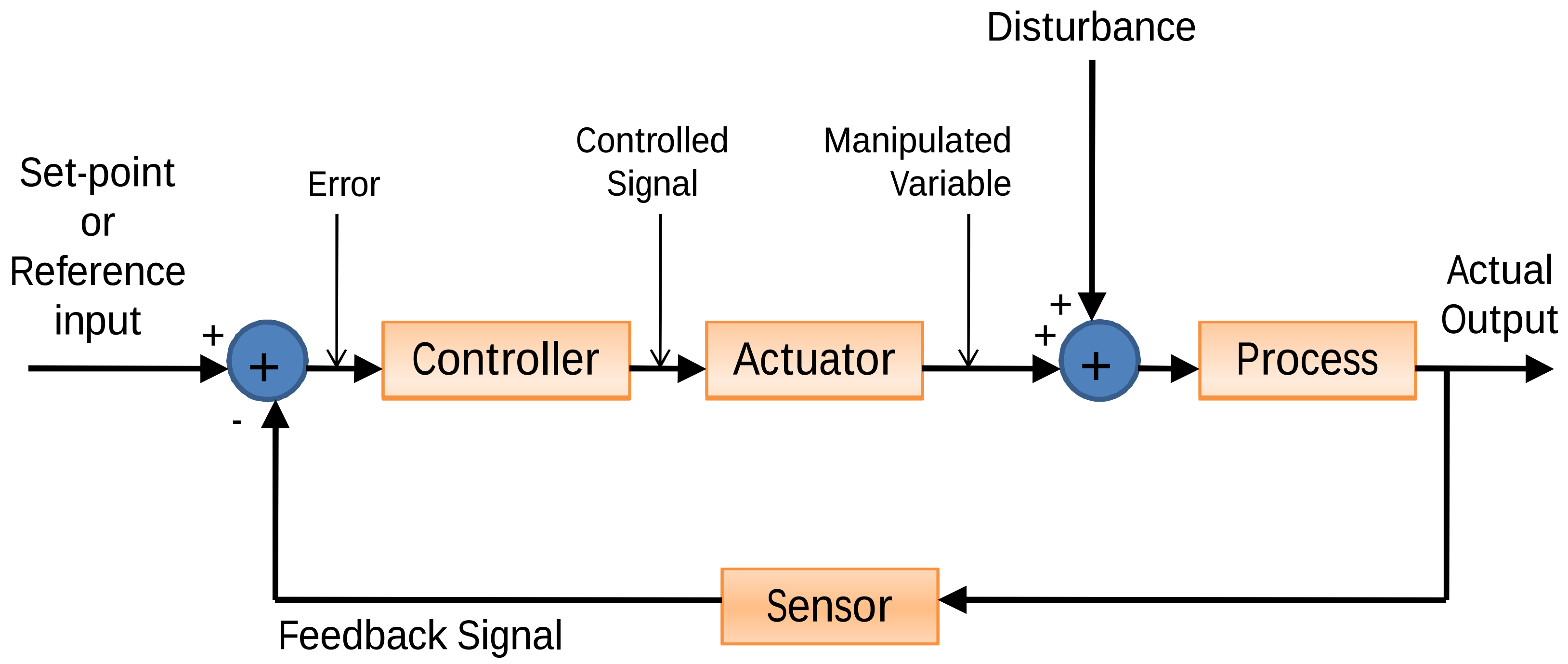


Image acquisition and processing- Image Processing Fields

- **Computer Graphics:** The creation of images
- **Image Processing:** Enhancement or other manipulation of the image
- **Computer Vision:** Analysis of the image content

Image Processing Fields

Input / Output	**Image**	**Description**
Image	Image Processing	Computer Vision
Description	Computer Graphics	AI

Sometimes, Image Processing is defined as “a discipline in which both the input and output of a process are images

But, according to this classification, trivial tasks of computing the average intensity of an image would not be considered an image processing operation

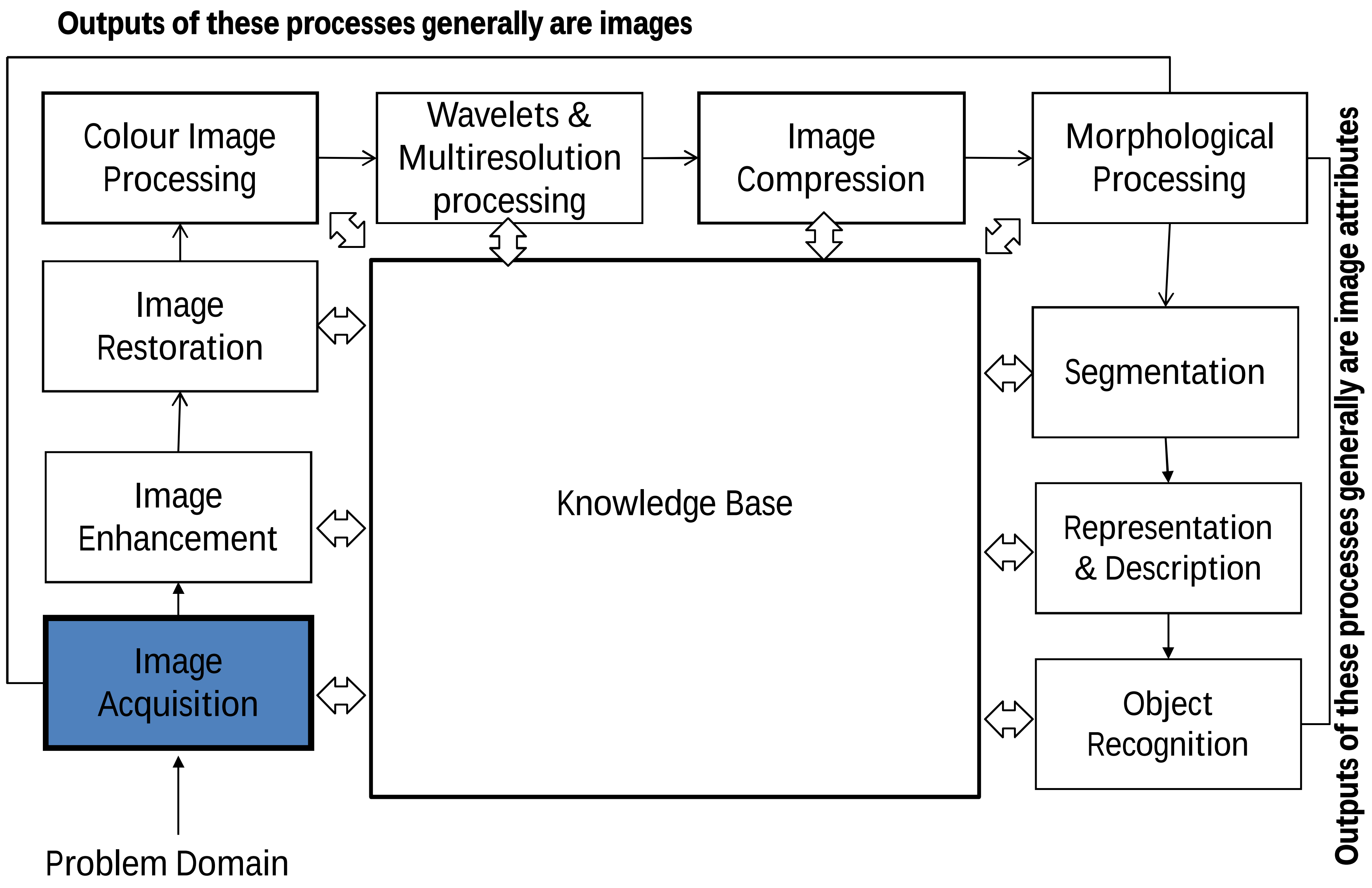
Computerized Processes Types

- **Mid-Level Processes:**
  - **Inputs**, generally, are images. **Outputs** are attributes extracted from those images (edges, contours, identity of individual objects)
  - **Tasks:**
    - Segmentation (partitioning an image into regions or objects)
    - Description of those objects to reduce them to a form suitable for computer processing
    - Classifications (recognition) of objects

Digital Image Definition

- An image can be defined as a two-dimensional function f(x,y)
- x,y: Spatial coordinate
- F: the amplitude of any pair of coordinate x,y, which is called the intensity or gray level of the image at that point.
- X,y and f, are all finite and discrete quantities.

Fundamental Steps in Digital Image Processing:



Fundamental Steps in DIP: (Description)

Step 1: Image Acquisition

The image is captured by a sensor (eg. Camera), and digitized if the output of the camera or sensor is not already in digital form, using analogue-to-digital convertor

Fundamental Steps in DIP: (Description)

Step 2: Image Enhancement

The process of manipulating an image so that the result is more suitable than the original for specific applications.

The idea behind enhancement techniques is to bring out details that are hidden, or simple to highlight certain features of interest in an image.

Fundamental Steps in DIP: (Description)

Step 3: Image Restoration

- Improving the appearance of an image

- Tend to be mathematical or probabilistic models. Enhancement, on the other hand, is based on human subjective preferences regarding what constitutes a “good” enhancement result.

Fundamental Steps in DIP: (Description)

Step 4: Colour Image Processing

Use the colour of the image to extract features of interest in an image

Fundamental Steps in DIP: (Description)

Step 5: Wavelets

Are the foundation of representing images in various degrees of resolution. It is used for image data compression.

Fundamental Steps in DIP: (Description)

Step 6: Compression

Techniques for reducing the storage required to save an image or the bandwidth required to transmit it.

Fundamental Steps in DIP: (Description)

Step 7: Morphological Processing

Tools for extracting image components that are useful in the representation and description of shape.

In this step, there would be a transition from processes that output images, to processes that output image attributes.

Fundamental Steps in DIP: (Description)

Step 8: Image Segmentation

Segmentation procedures partition an image into its constituent parts or objects.

Important Tip: The more accurate the segmentation, the more likely recognition is to succeed.

Fundamental Steps in DIP: (Description)

Step 9: Representation and Description

- **Representation:** Make a decision whether the data should be represented as a boundary or as a complete region. It is almost always follows the output of a segmentation stage.
  - **Boundary Representation:** Focus on external shape characteristics, such as corners and inflections (انحناءات)
  - **Region Representation:** Focus on internal properties, such as texture or skeleton (هيكلية) shape

Fundamental Steps in DIP: (Description)

Step 9: Representation and Description

- Choosing a representation is only part of the solution for transforming raw data into a form suitable for subsequent computer processing (mainly recognition)

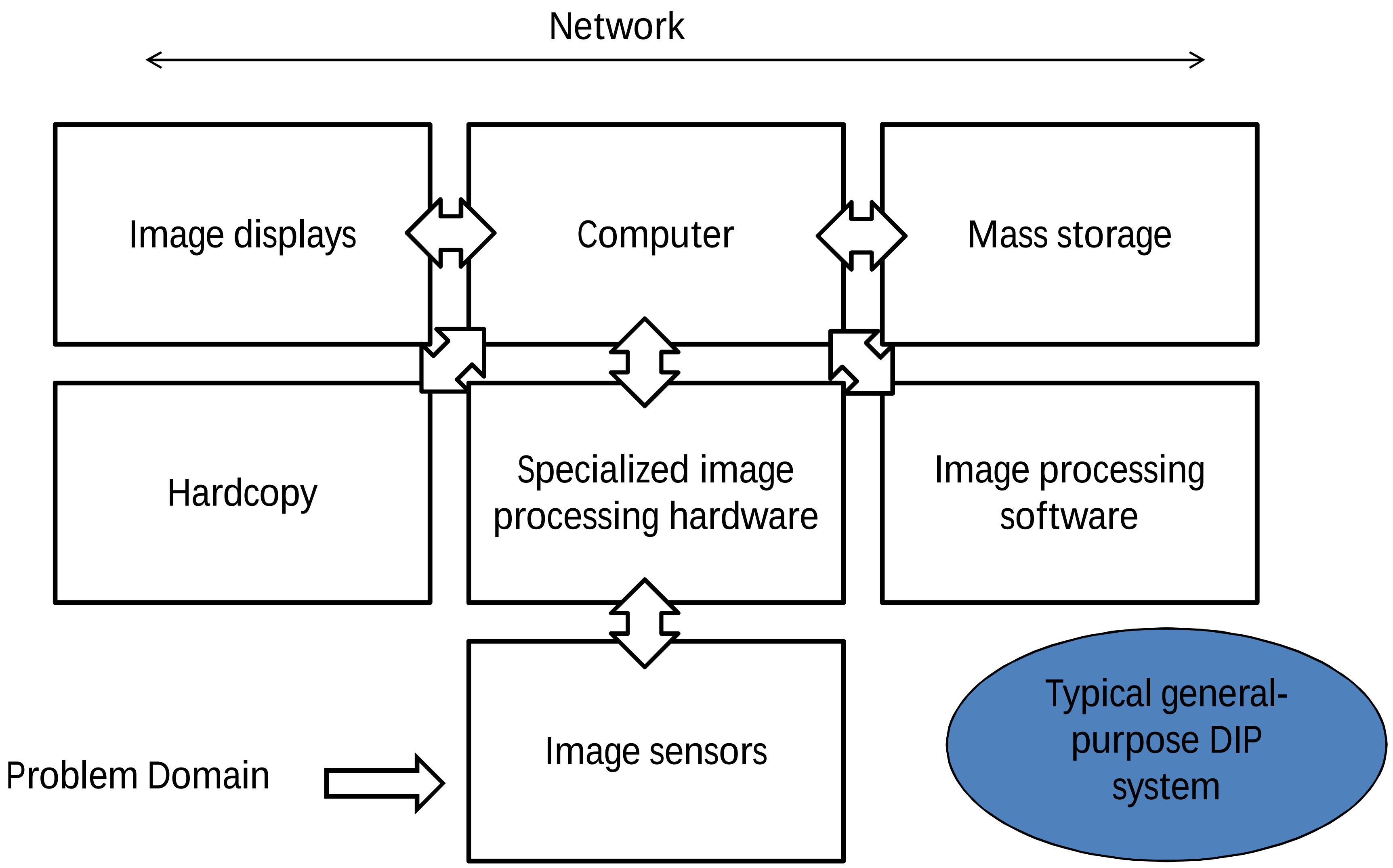
- **Description:** also called, ***feature selection***, deals with extracting attributes that result in some information of interest.

Fundamental Steps in DIP: (Description)

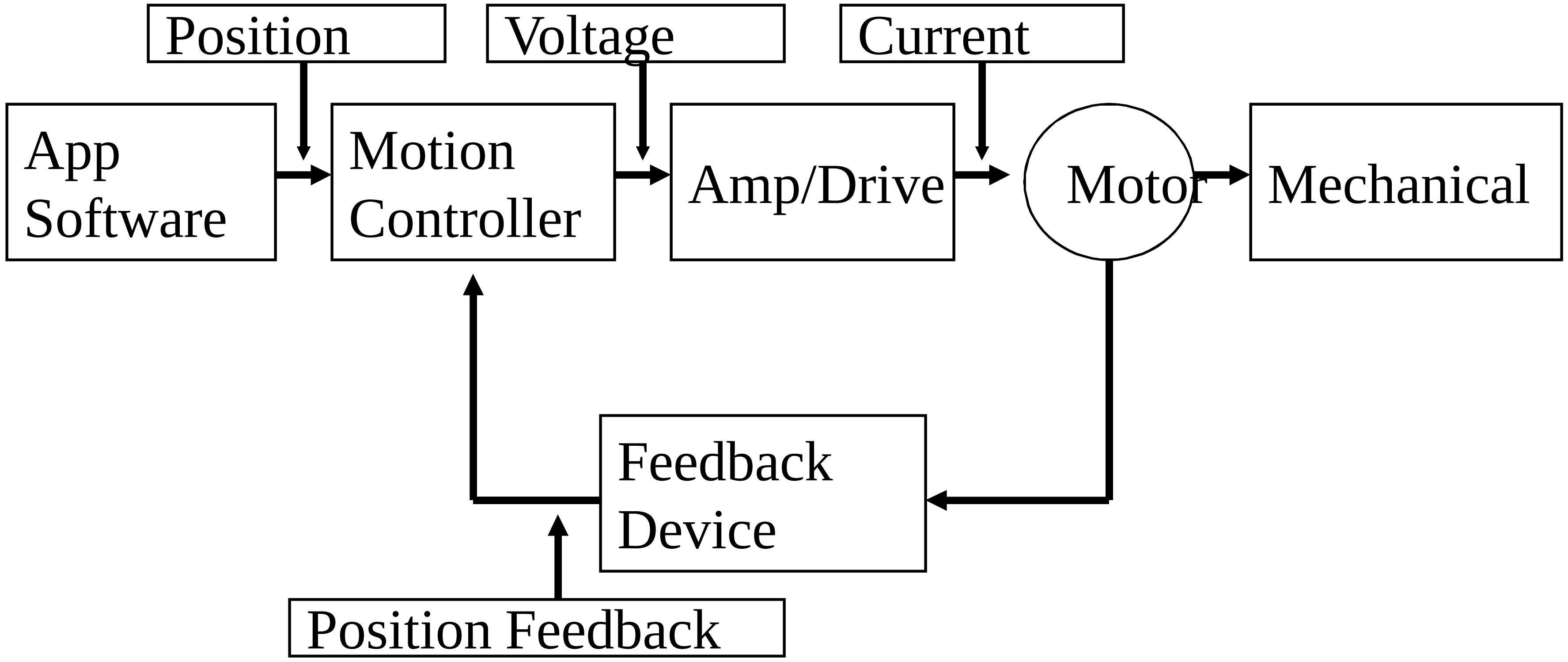
Step 10: Knowledge Base

Knowledge about a problem domain is coded into an image processing system in the form of a knowledge database.

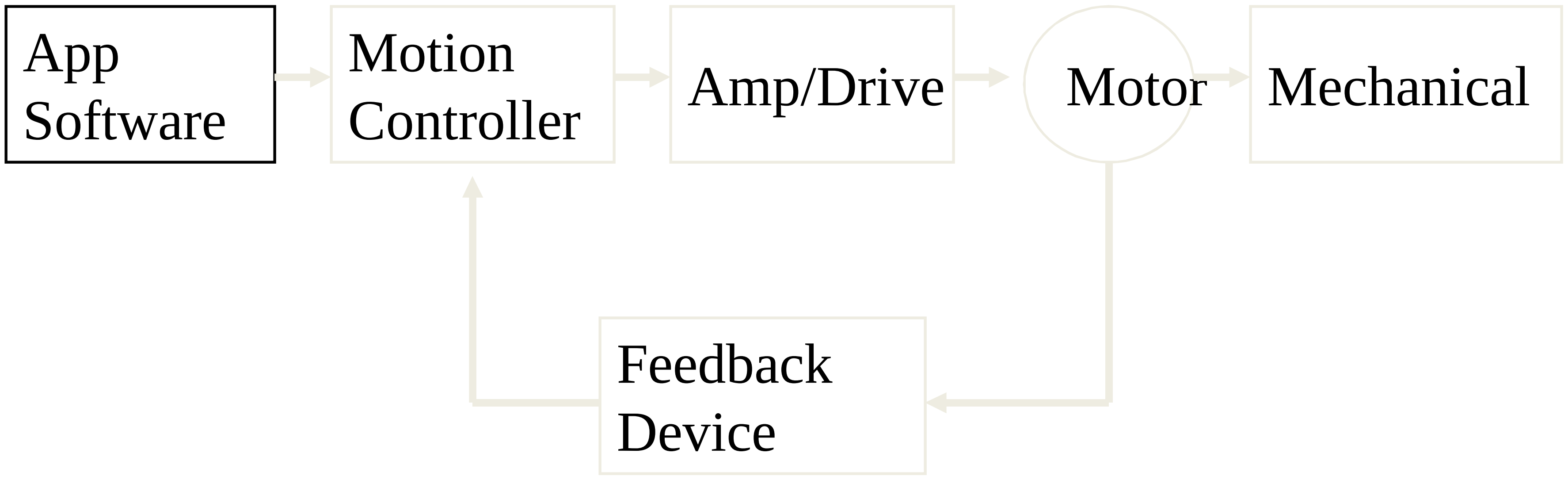
Components of an Image Processing System


Network
Image displays
Computer
Mass storage
Hardcopy
Specialized image processing hardware
Image processing software
Image sensors
Problem Domain
Typical general-purpose DIP system

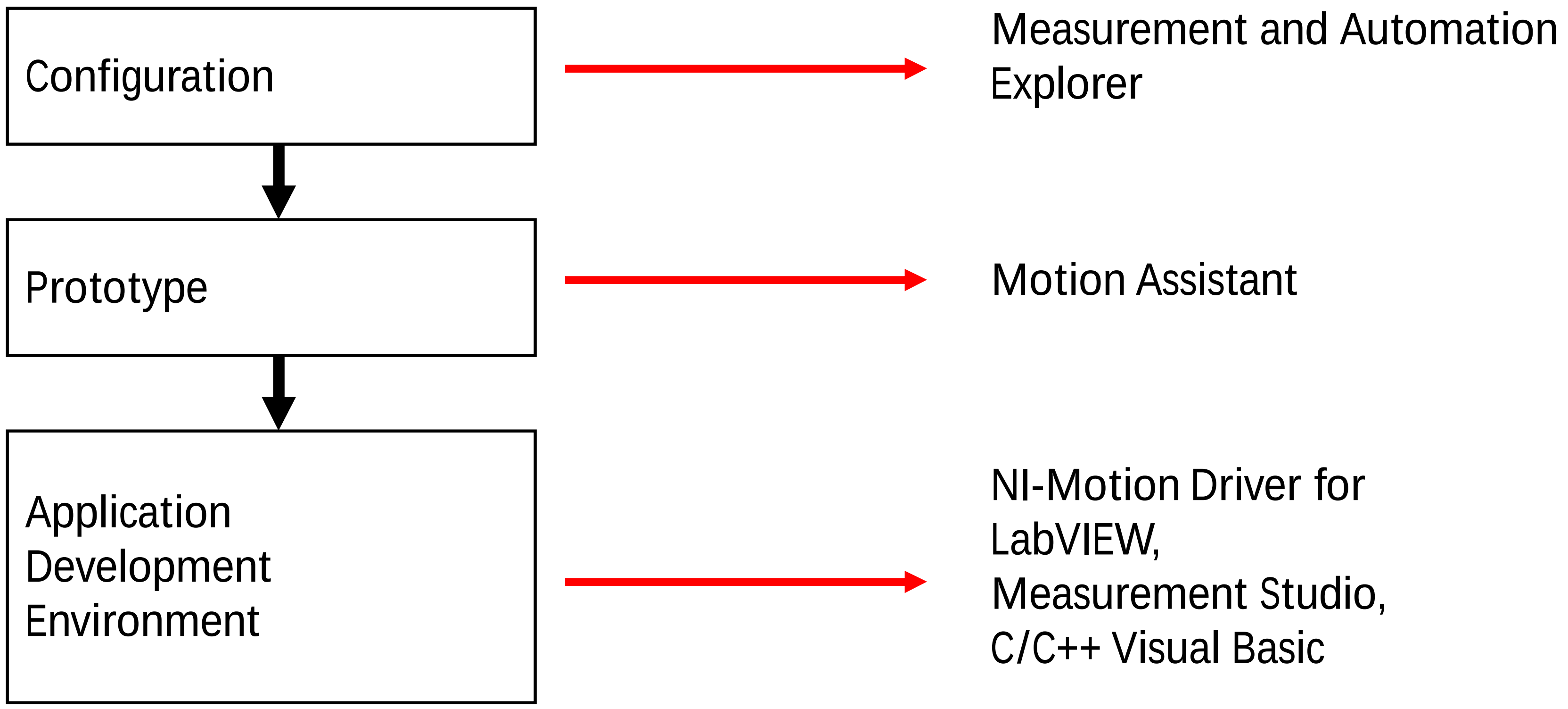
Elements of a Motion System



Application Software

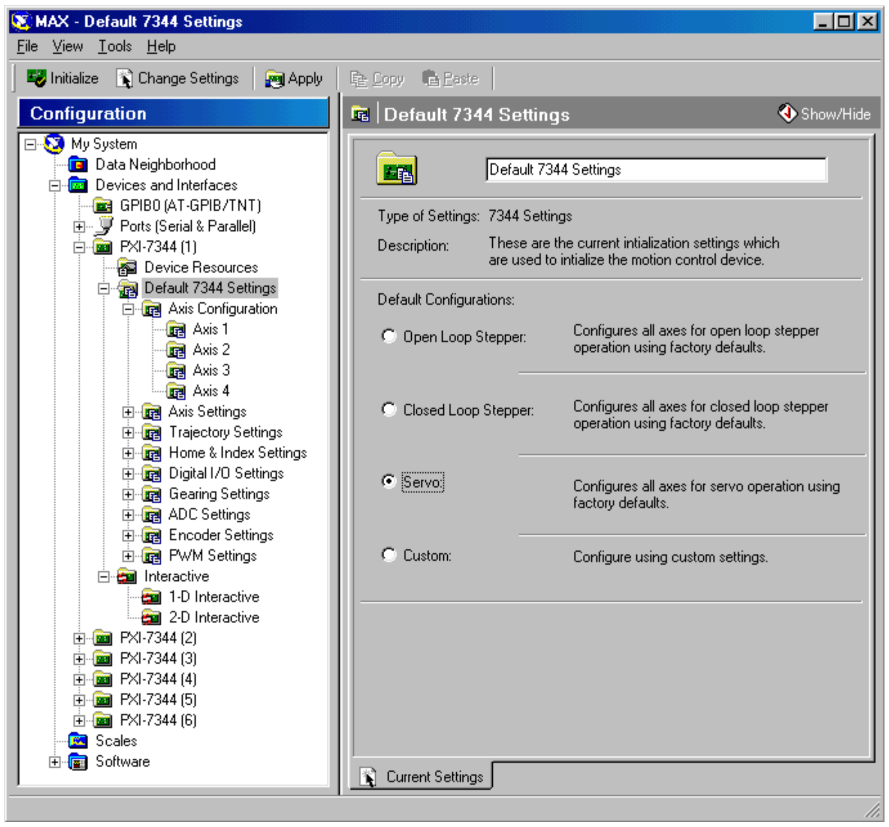


Software Overview


Configuration
Measurement and Automation Explorer
Prototype
Motion Assistant
Application Development Environment
NI-Motion Driver for LabVIEW, Measurement Studio, C/C++ Visual Basic

Configuring a Motion Control System

- Measurement & Automation Explorer
  - Use to configure all National Instruments devices
  - Tune servo motors
  - Test configuration
  - Configure signal routing
  - Test NI vision and data acquisition devices



Default 7344 Settings

Type of Settings: 7344 Settings

Description: These are the current intialization settings which are used to intialize the motion control device.

Default Configurations:

Open Loop Stepper:	Configures all axes for open loop stepper operation using factory defaults.
Closed Loop Stepper:	Configures all axes for closed loop stepper operation using factory defaults.
[✓ Servo:]	Configures all axes for servo operation using factory defaults.
Custom:	Configure using custom settings.

NI-Motion Driver Software

- For Windows and LabVIEW Real-Time Systems
  - VIs and functions for LabVIEW, Visual Basic, and C
  - Measurement and Automation Explorer
    - Configuration of motion and other components
    - Motor Tuning
- For Non-Windows Systems
  - Motion Control Hardware DDK
  - Linux and VxWorks drivers from Sensing Systems (www.sensingsystems.com)

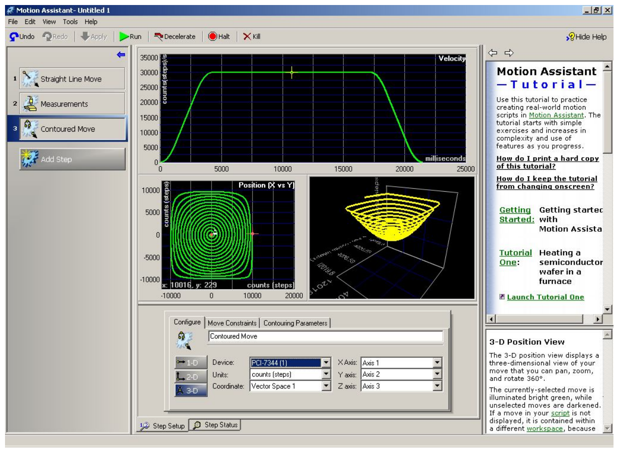
NI Motion Assistant Prototyping Software

Configurable

- Easy to use
- Interactive
- No programming required

Programmable

- Flexible
- Easier integration with other components



Motion Assistant
—Tutorial—

Use this tutorial to practice creating real-world motion scripts in Motion Assistant. The tutorial starts with simple exercises and increases in complexity and use of features as you progress.

How do I print a hard copy of this tutorial?

How do I keep the tutorial from changing onscreen?

Getting Started: Getting started with Motion Assista

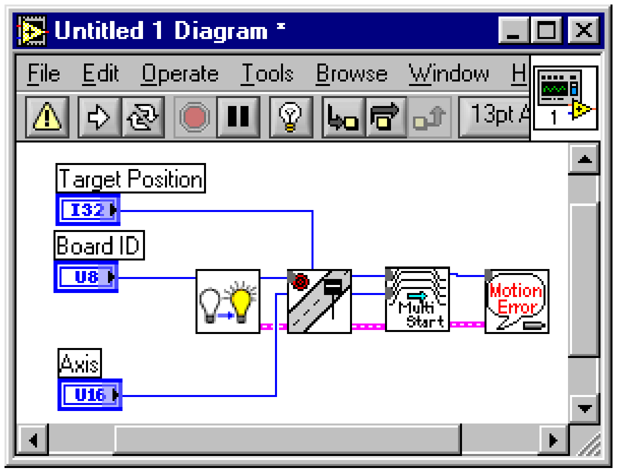
Tutorial One: Heating a semiconductor wafer in a furnace

Launch Tutorial One

3-D Position View

The 3-D position view displays a three-dimensional view of your move that you can pan, zoom, and rotate 360°.

The currently-selected move is illuminated bright green, while unselected moves are darkened. If a move in your script is not displayed, it is contained within a different workspace, because



Purpose of the Motion Controller

- Calculate the trajectories for each commanded move on the board to prevent host computer interference
- Provide the torque commands to the motor drive or amplifier
- Monitor limits and emergency stops for supervisory control
- Close the PID loop

Motor Drive Requirements

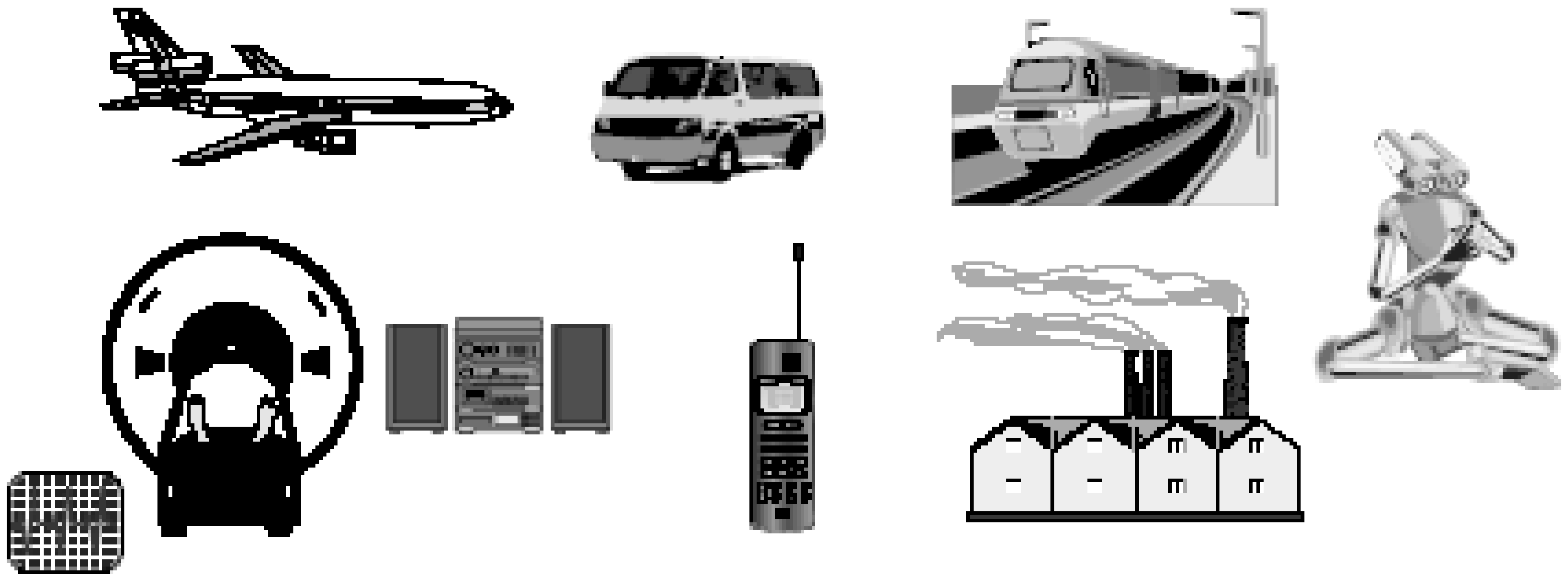
- The motor drive must match the motor technology
  - For example, 2-phase stepper motors require 2 phase stepper motor drives, etc…
- The motor drive must provide adequate:
  - Peak current
  - Continuous current
  - Voltage

NI Motor Drive Products

- MID series Stepper and Servo drives
  - MID-7604/2 four and two axis 1.4 Amp/phase stepper drive
  - MID-7654/2 four and two axis 5 Amp continuous servo drive
- Universal Motion Interface (UMI) for easy connectivity to 3rd party drives
- See Drive Advisor at ni.com/motion/advisors for info on third party drives

What is an Embedded System?

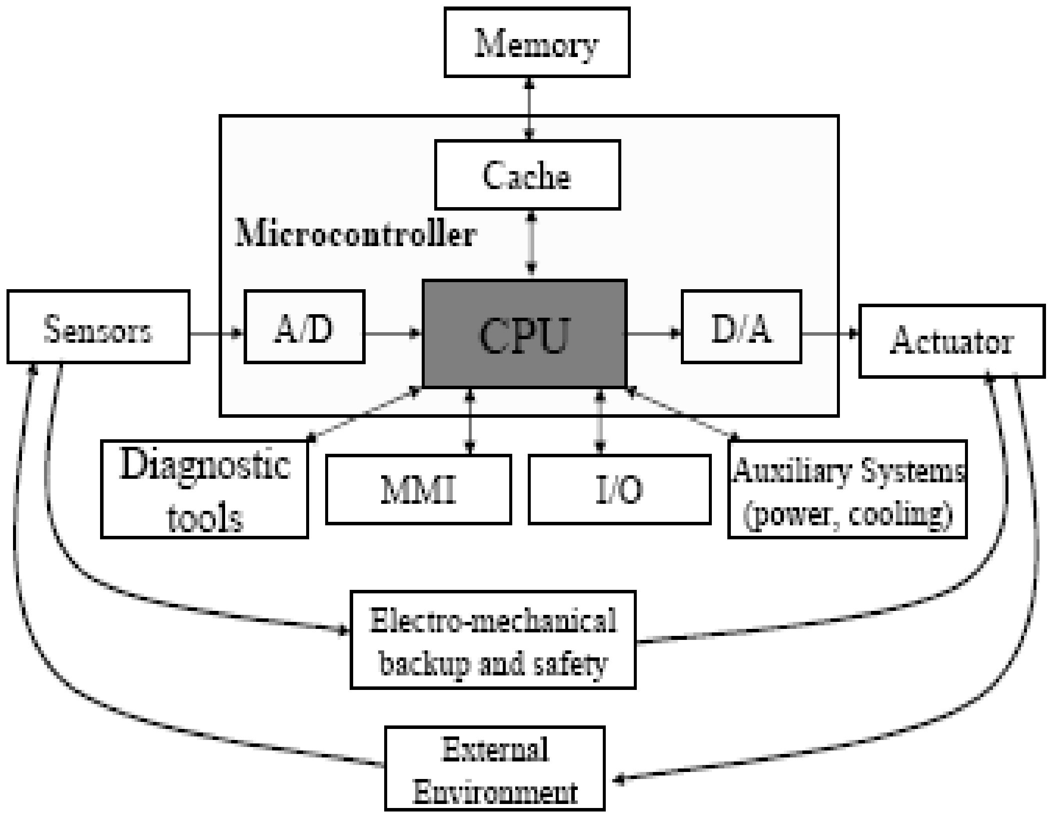
- Computer purchased as part of some *other* piece of equipment
  - Typically dedicated software (may be user customizable)
  - Often replaces previously electromechanical components



An Embedded Control System Designer's View

- Measured by:

  Cost, Time to market, Cost, Functionality, Cost & Cost.



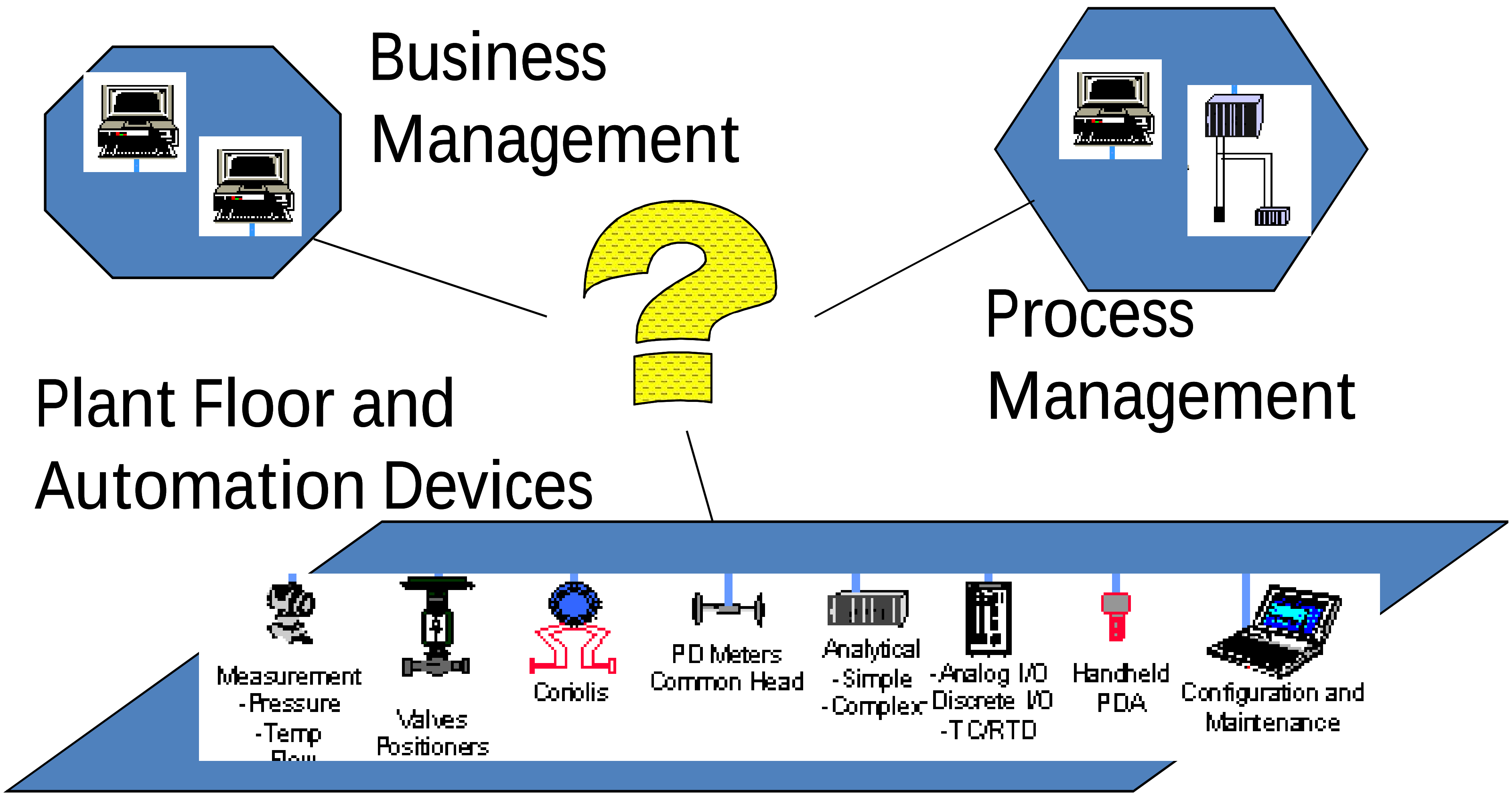
What is OPC

- A “real world” application of object technology used to devise a standard communication system to enable industrial plant floor devices and business software applications to communicate via a standard common protocol.

Communication

- Communication and integration of plant information systems:
  - Many different plant floor devices and process control software applications from many different vendors
  - Business Information and Management software

Information Islands



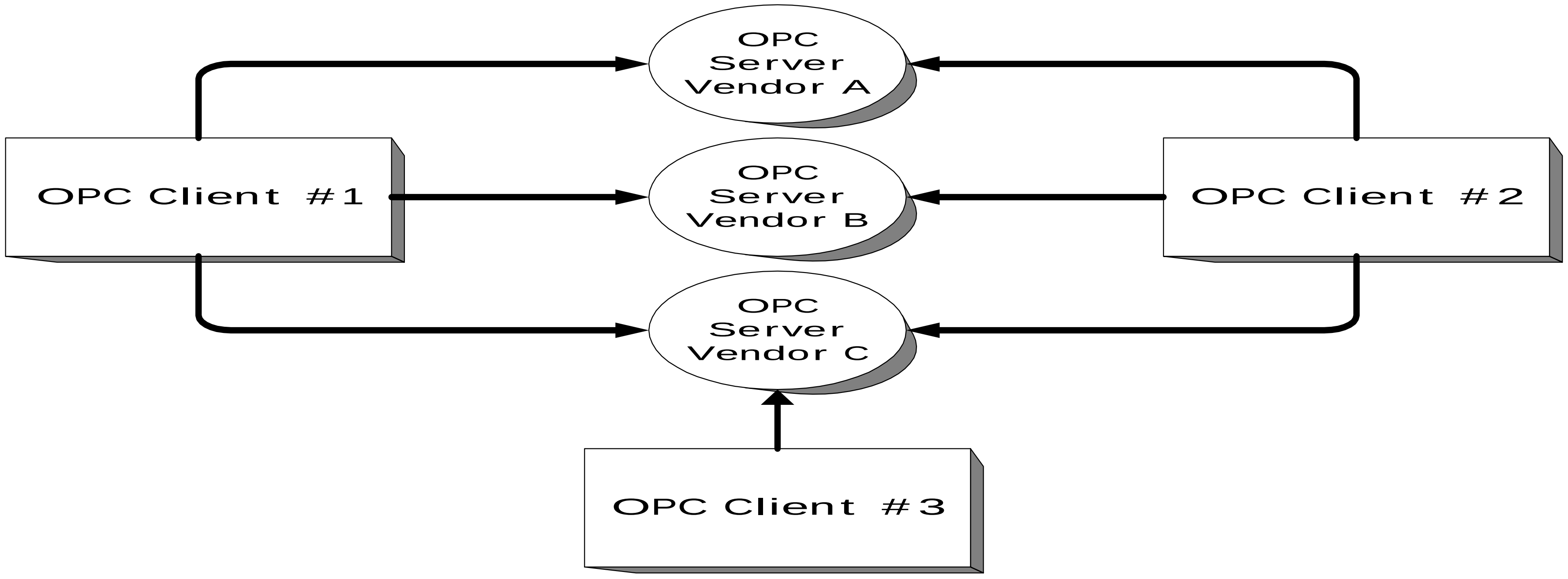
Business Management
Process Management
Plant Floor and Automation Devices
Measurement -Pressure -Temp
Valves Positioners
Coriolis
PD Meters Common Head
Analytical -Simple -Complex
-Analog I/O Discrete I/O -TC/RTD
Handheld PDA
Configuration and Maintenance

A Better Way: OPC

- The OPC Standard now places the burden on hardware vendors to develop a single OPC “driver” (server) that is responsible for data collection and distribution for their device(s).
- Provide data to clients in a standard manor.

Client / Server Relationship

- Software developers can now write clients that can communicate with hardware OPC servers using a common, efficient protocol



Technology (TLA Guide)

- COM - Component Object Model
  - Provides objects as reusable, binary components
  - objects “expose” a set of interfaces that client applications can use to access the object’s services.
  - Implementation is encapsulated behind interfaces (allows object to change without requiring client recompilation)

Technology (continued)

- DCOM - Distributed Component Object Model
  - remote objects appear to be local
- OLE - Object Linking and Embedding
  - based on COM (e.g. an OLE object implements certain COM interfaces)
  - provides integration among applications

Technology (continued)

- OLE Automation
  - Allows components to easily be used by high level custom programs (e.g. written in VB or VBA)
  - Set of special COM interfaces
    - automation or ActiveX objects implement these “automation” interfaces
  - Automation Controllers
    - Clients that can integrate those objects

OPC and COM

- OPC Specifications contain defined COM interfaces
- Server and Client Interfaces
  - Data Access
  - Historical Data
  - Alarms and Events
  - etc.
- Implementation and development

HMI /SCADA (Supervisory Control and Data Acquisition) System

What is SCADA?

- Supervisory Control and Data Acquisition
- Supervisory
  - Operator/s, engineer/s, supervisor/s, etc
- Control
  - Monitoring
  - Limited
  - Telemetry
  - Remote/Local
- Data acquisition
  - Access and acquire information or data from the equipment
  - Sends it to different sites through telemetry
  - Analog / Digital

Elements of SCADA

Elements of a SCADA system

- Sensors and actuators
- RTUs/PLCs
- Communication
- MTU
  - Front End Processor
  - SCADA server
  - Historical/Redundant/Safety Server
  - HMI computer
  - HMI software

SCADA server

SCADA Server

- It can be a Web server
- Data logging
- Analyzing data
- Serve the clients through a firewall
- Clients connected in the corporation or connected outside through internet
- Real-time decision maker
- Asks RTU for information

HMI Computer

Human Machine Interface Computer

- ➢ Access on the SCADA Server
- ➢ Control the system
- ➢ Operator Interface
- ➢ Software
  - ▪ User friendly
  - ▪ Programmable (C, C++)

DCS

DCS – Distributed Control System

- ➢ Process oriented – tendency to do something
- ➢ Not event oriented – does not depend on circumstances
- ➢ Local control over the devices
- ➢ Subordinate to SCADA