Industrial Network Basics: Simplifying I/O Terminology

There are many terms used for I/O technology in industrial automation: Remote I/O, Distributed I/O, Modular I/O, Expandable I/O, Block I/O, Conventional I/O and the list can go on.

Where is the I/O located?

Centralized I/O:

A majority of the I/O is located in the cabinet with the PLC and is situated on one continuous backplane.

Distributed I/O:

Small percentages of the I/O are located in many locations that are not the same as the controller. This data is usually collected over an industrial network. Distributed I/O solutions usually generate a total cost of ownership per point lower than other types.

Remote I/O:

This has evolved over time but in 2011, I believe this is the new definition. A big percentage of the I/O is in a single or few locations that are not the same as the location of the PLC. It is a hybrid of Distributed and Centralized I/O solutions. This data is usually collected over an extended backplane or an industrial network.

How is the I/O collected?

Conventional I/O:

Normally mounted inside a cabinet on DIN rail or rack; this utilizes a backplane providing communication and power supply to the I/O devices, which are unique to the version of the master device. This is usually found in a centralized or remote I/O configuration.

Expandable I/O:

Found both inside and outside the cabinet, expandable I/O solutions utilize the "slice" concept and use a backplane/sub-bus to communicate between I/O devices, which are unique to the version of the master device. These can implement large numbers of additional slices of different I/O types to be added including valve manifolds. It is usually used in a remote I/O configuration.

Block I/O:

A set number of I/O points typically 8 or 16 discrete points. Block I/O usually communicates over an industrial network and is rated for use on the machine, outside of a controls cabinet. It is almost always found in a distributed I/O configuration.

Modular I/O:

Normally found outside the cabinet fitted with an industrial network communications head with the ability to combine multiple types of I/O devices, including valve manifolds. Modular I/O is typically used in a distributed or remote I/O configuration.

What is distributed I/O and what are the benefits?

With ever larger systems, and increasingly complex processes, ensuring a centralised system for measuring and controlling all aspects of the plant is vital. Pooling data, which often originates from several different sites, into one single location is at best a challenge and at worst a logistical nightmare. Could distributed I/O be the answer? PIF investigates.

What is distributed I/O?

Distributed I/O systems include small field devices with a wide range of I/O options. These can include digital and analogue channels, temperature measurements, and counter inputs. These modular devices provide a flexibility distinctly lacking in traditional devices, like Programmable Logic Controllers (PLCs), that require a high density channel count and long cable runs.

What are the benefits of distributed I/O?

The beauty of the distributed approach is that it allows for short cable runs for signals (because measuring devices are closer to the sensors). Direct connection of sensors also eradicates signal conditioning, with the added benefit of standard signal inputs for specialised sensors.

In the Schneider (Modicon) world...

Remote I/O is racks and modules that are directly connected by multiconductor cables to the rack that contains the PLC processor. By using multiconductor (parallel) cables, the data throughput is very high and secure. Data, configuration and diagnostic information are "automatically" available. Data does not have to "packaged" into a serial protocol for sending to the I/O.

Distributed I/O is racks and modulues that are connected by some communciations protocol over a communications cable. Modbus, Modbus Plus, Modbus TCP/IP, Profibus, Interbus-S, etc. This type of I/O often does not provide as much information back to the processor because all info, including diagnostics, must be "packaged" into the comms and sent/received.

The main characteristics of a distributed I/O system are small field devices with a wide range of I/O options such as digital and analogue channels, temperature measurements and counter inputs. These modular devices give a flexibility that can not be achieved with traditional devices.

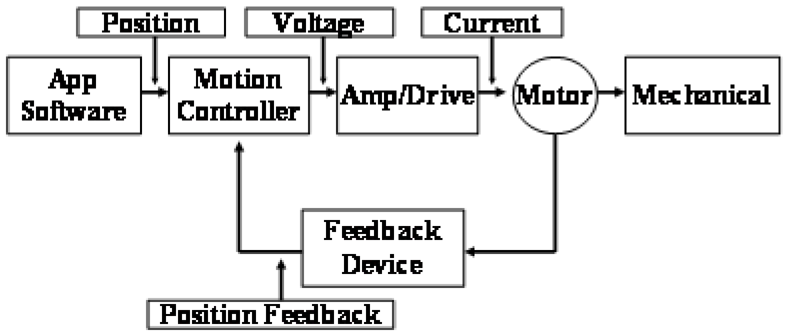
A significant benefit of this distributed approach includes short cable runs for signals since the measuring devices are located close to the sensors. Direct connection of sensors eliminates the need for signal conditioning while providing standard signal inputs for those unusual and specialised sensors.

Utilizing reliable and conventional communication buses such as Ethernet and RS485 with protocols such as Modbus/TCP, Modbus/RTU, Profibus, and even ASCII, modules can be integrated into existing networks and systems, giving a low cost, flexible upgrade path.

These advantages can be extended to the software, presenting data in a standard format for new and existing platforms such as OPC (Open Process Control), Modbus, SNMP (Simple Network Management Protocol), and the other protocols like email and SMS messages on the cellular network. With the use of appropriate gateway devices, modules can be connected to existing legacy systems such as a plant wide Distributed Control System (DCS) and simply reconfigured if the system is upgraded or replaced.

MOTION CONTROL:

COMPONENTS OF A MOTION CONTROL SYSTEM:



Application software: Used to command target positions and motion control profiles.

Motion controller: The motion controller acts as brain of the system by taking the desired target positions and motion profiles and creating the trajectories for the motors to follow, but outputting a ±10 V signal for servomotors, or a step and direction pulses for stepper motors.

Amplifier or drive: Amplifiers (also called drives) take the commands from the controller and generate the current required to drive or turn the motor.

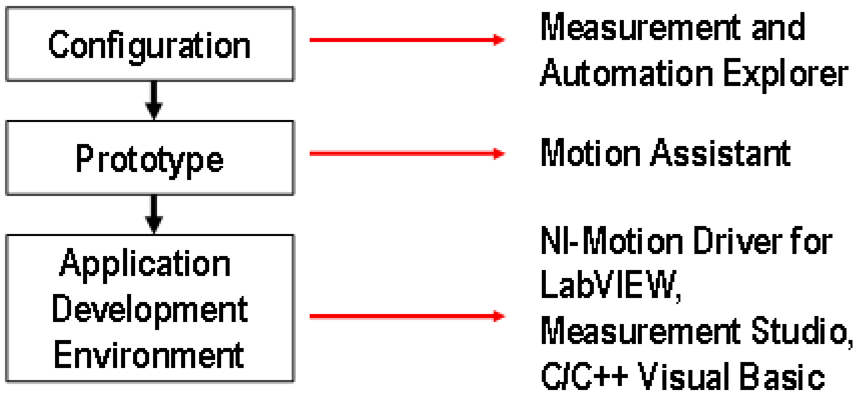
Motor: Motors turn electrical energy into mechanical energy and produce the torque required to move to the desired target position.

Mechanical elements: Motors are designed to provide torque to some mechanics. These include linear slides, robotic arms and special actuators.

Feedback device or position sensor: A position feedback device is not required for some motion control applications (such as controlling stepper motors), but is vital for servomotors. The feedback device, usually a quadrature encoder, senses the motor position and reports the result to the controller, thereby closing the loop to the motion controller.

SOFTWARE FOR CONFIGURATION, PROTOTYPING AND DEVELOPMENT:

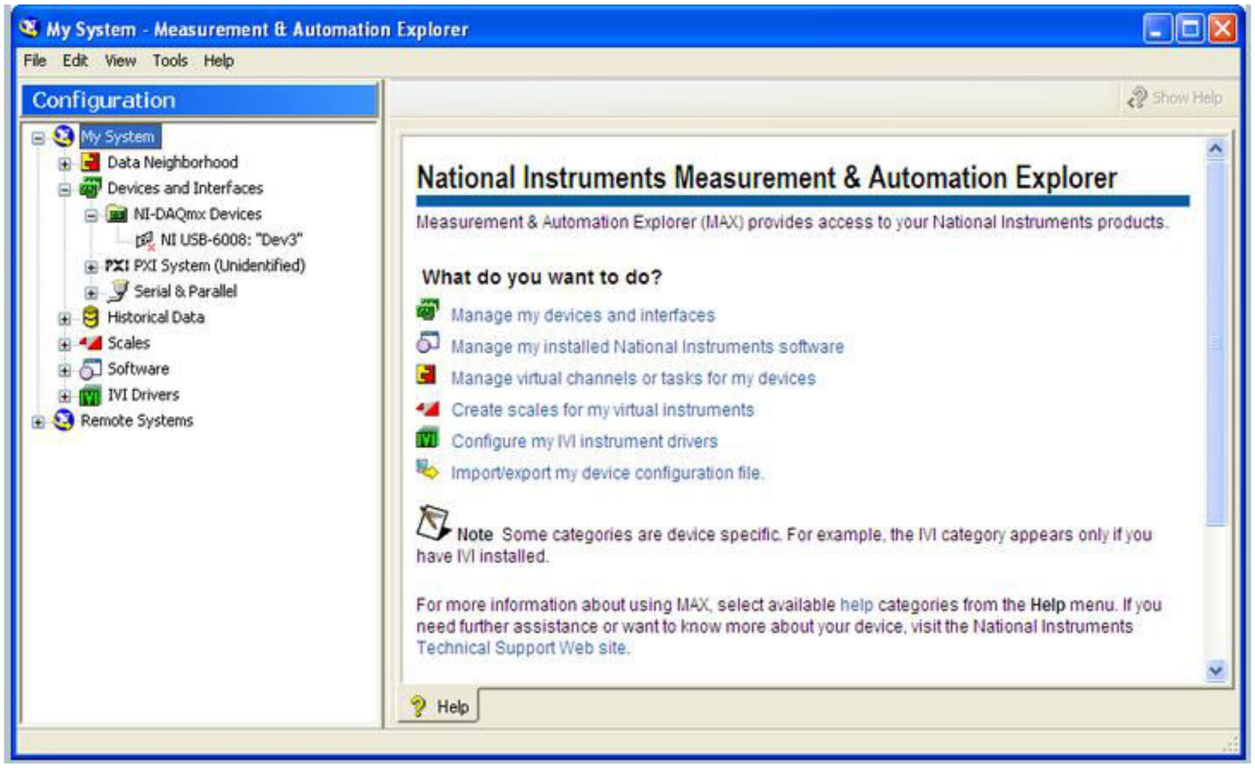
Application software is divided into three main categories—configuration, prototype and application development environment (ADE).


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

Configuration:

National Instruments offers Measurement and Automation Explorer, an interactive tool for configuring not only motion control, but all other National Instruments hardware. For motion control, Measurement and Automation Explorer offers interactive testing and tuning panels that help you verify your system functionality before you program.

Measurement & Automation Explorer (MAX) provides access to your National Instruments CAN, DAQ, FieldPoint, GPIB, IMAQ, IVI, Modular Instruments, Motion, NI Switch Executive, VI Logger, VISA, and VXI devices.



National Instruments Measurement & Automation Explorer

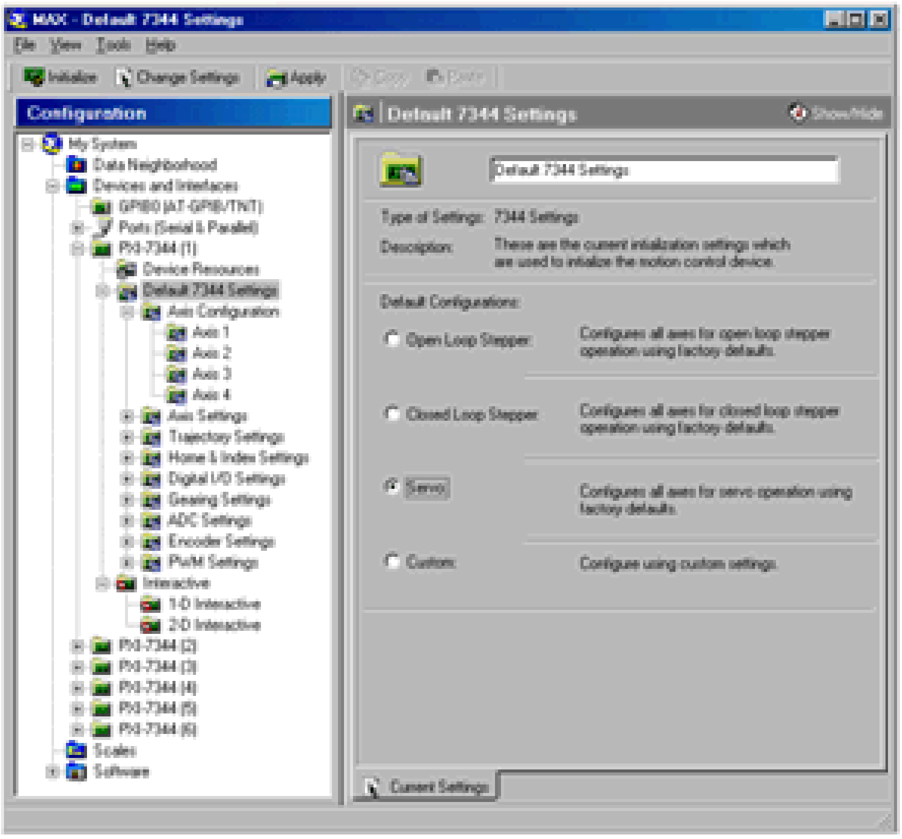
Measurement & Automation Explorer (MAX) provides access to your National Instruments products.

What do you want to do?

- Manage my devices and interfaces
- Manage my installed National Instruments software
- Manage virtual channels or tasks for my devices
- Create scales for my virtual instruments
- Configure my IVI instrument drivers
- Import/export my device configuration file.

Note Some categories are device specific. For example, the IVI category appears only if you have IVI installed.

For more information about using MAX, select available help categories from the **Help** menu. If you need further assistance or want to know more about your device, visit the National Instruments Technical Support Web site.



Default 7344 Settings

Default 7344 Settings

Type of Settings: 7344 Settings

Description: These are the current initialization settings which are used to initialize the motion control device.

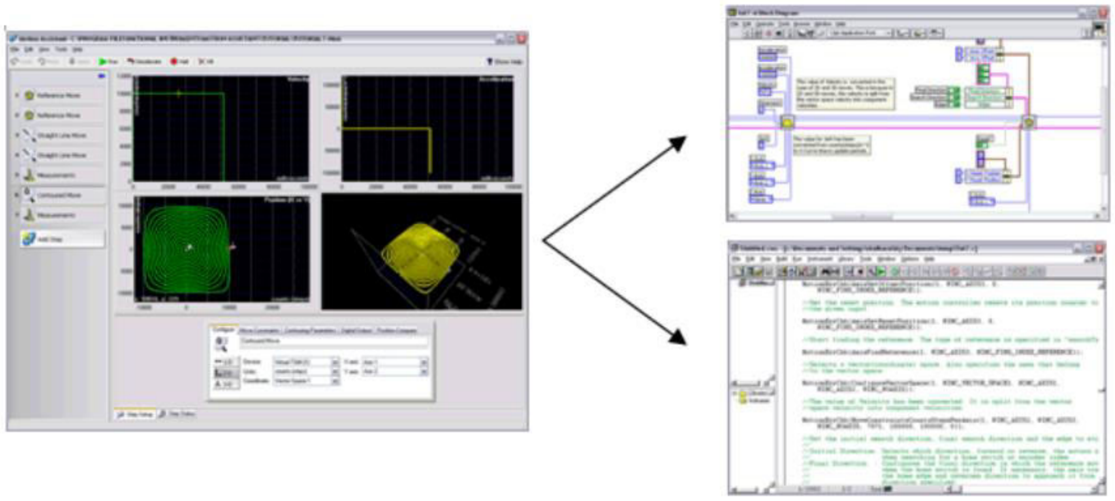
Default Configurations:

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.

Prototyping:

For prototyping, National Instruments offers a tool called NI Motion Assistant. NI Motion Assistant is an interactive tool with which you can configure moves using a point-and-click environment and generate LabVIEW code based on the moves you configure.

The key benefit of the NI Motion Assistant lies in the difference between configurable and programmable environments. With configurable environments, you can start your development without programming. You can think of the tasks in the NI Motion Assistant as prewritten blocks of code that you simply configure to meet your needs. Programmable environments, on the other hand, require you to use standard programming languages such as LabVIEW, C, or Visual Basic to accomplish your tasks. Unfortunately, many configurable environments may be limited in functionality or in the ability to integrate with other I/O outside motion. The NI Motion Assistant bridges the gap between programmable and configurable environments by offering all configurable system features as well as LabVIEW code generation.


The NI Motion Assistant helps you quickly prototype your application and then convert your project into LabVIEW VIs or C code for further development.

Development:

After the prototyping phase, the next step is to develop the final application code. For this, you use driver-level software in an ADE such as LabVIEW, C, or Visual Basic. For a National Instruments motion controller, you use NI-Motion driver software.

The NI-Motion driver software contains functions you can use to communicate with NI motion controllers in the Windows or LabVIEW Real-Time OS. NI-Motion also includes MAX to help you easily configure and tune your motion system.

For non-Windows systems, you can develop your own driver using the NI Motion Control Hardware DDK manual. It explains how to communicate on a low level with NI motion controllers. If you do not have the expertise or time to develop your own driver, National Instruments Alliance Partner Sensing Systems offers a Linux and VxWorks driver, and can create drivers for other OSs, such as Mac OS X or RTX.

Motion Controller:

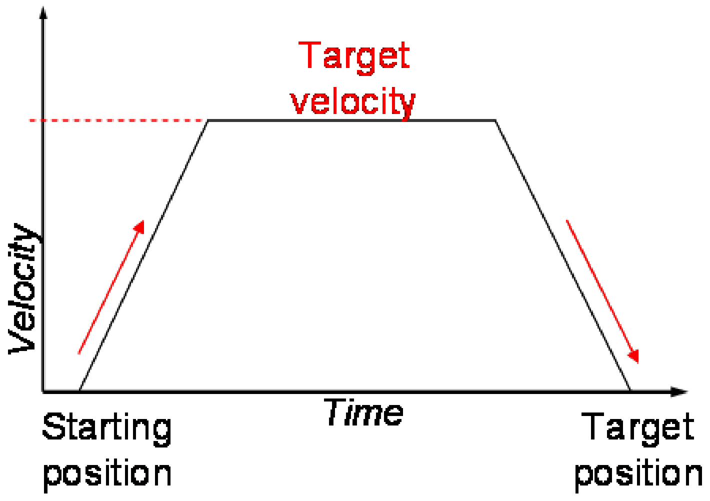
A motion controller acts as the brain of the motion control system and calculates each commanded move trajectory. Because this task is vital, it often takes place on a digital signal processor (DSP) on the board itself to prevent host-computer interference (you would not want your motion to stop because your antivirus software starts running). The motion controller uses the trajectories it calculates to determine the proper torque command to send to the motor amplifier and actually cause motion.

The motion controller must also close the PID control loop. Because this requires a high level of determinism and is vital to consistent operation, the control loop typically closes on the board itself. Along with closing the control loop, the motion controller manages supervisory control by monitoring the limits and emergency stops to ensure safe operation. Directing each of these operations to occur on the board or in a real-time system ensures the high reliability, determinism, stability, and safety necessary to create a working motion control system.

Calculating the Trajectory

The motion trajectory describes the motion controller board control or command signal output to the driver/amplifier, resulting in a motor/motion action that follows the profile. The typical motion controller calculates the motion profile trajectory segments based on the parameter values you program. The motion controller uses the desired target position, maximum target velocity, and acceleration values you give it to determine how much time it spends in the three primary move segments (which include acceleration, constant velocity, and deceleration).

For the acceleration segment of a typical trapezoidal profile, motion begins from a stopped position or previous move and follows a prescribed acceleration ramp until the speed reaches the target velocity for the move.



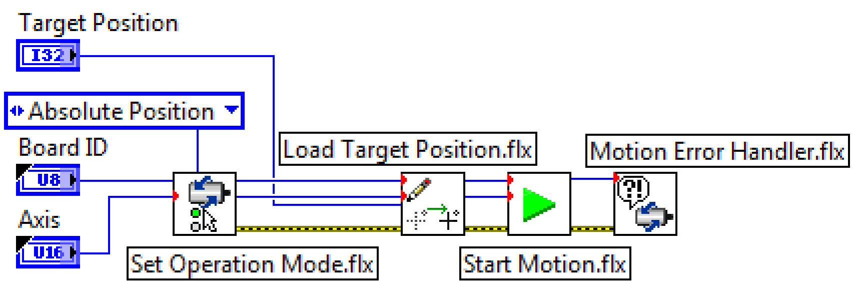
Selecting the Right Motion Controller:

NI offers three main families of DSP-based motion controllers, including the low-cost NI 733x series, the mid-range NI 734x series, and the high-performance NI 735x series. The NI 733x low-cost controllers offer four-axis stepper motor control and most of the basic functions you need for a wide variety of applications, including single and multiaxis point-to-point motion. The NI 734x series is the mid-range series that offers up to four axes of both stepper and servo control, as well as some higher-performance features such as contouring and electronic gearing. The NI 735x series is the most advanced series that offers up to eight axes of stepper and servo control, extra I/O, and many powerful features including sinusoidal commutation for brushless motors and 4 MHz periodic breakpoints (or position triggers) for high-speed integration.

Move Types:

Single-Axis, Point-to-Point Motion

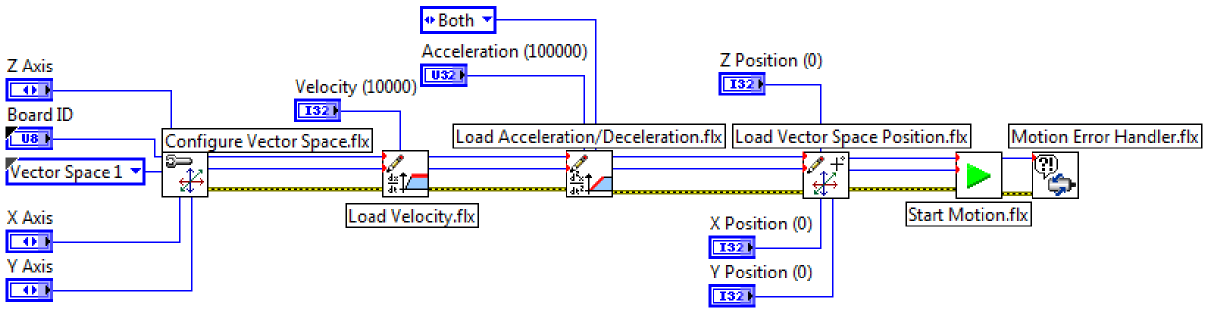
One of the most commonly used profiles is the simple, single-axis, point-to-point move, which requires the position to which the axis needs to move. Often it also requires the velocity and acceleration (usually supplied by a default setting) at which you want the motion to move. Figure 6 shows how to move a single axis in LabVIEW using the default velocity and acceleration.



Single-Axis, Point-to-Point Motion in LabVIEW

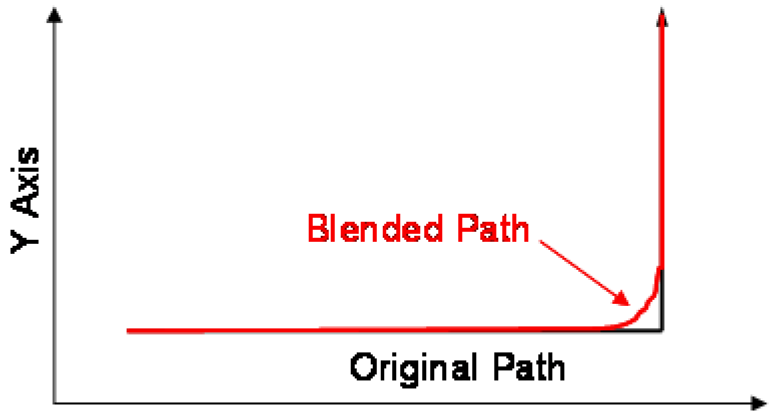
Coordinated Multiaxis Motion:

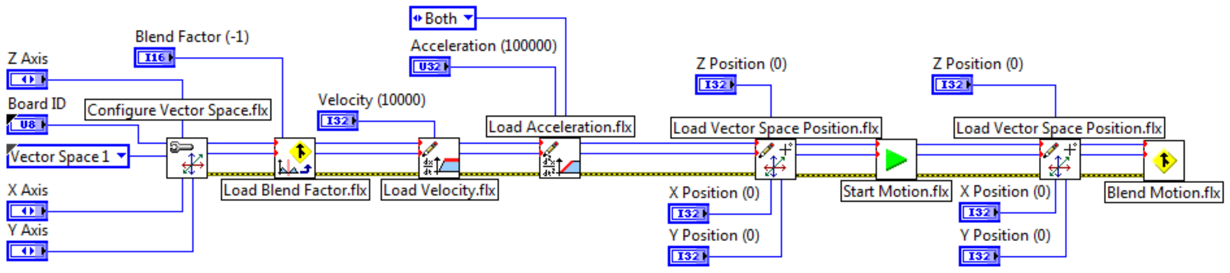
Another type of motion is coordinated multiaxis motion, or vector motion. This move is often point-to-point motion but in 2D or 3D space. Vector moves require the final positions on the X, Y, and/or Z axes. Your motion controller also requires some type of vector velocity and acceleration. This motion profile is commonly found in XY-type applications such as scanning or automated microscopy. Figure 7 shows how to accomplish a three-axis move using LabVIEW. For more information on coordinated motion, view the examples in the LabVIEW Multiaxis.llb library in NI-Motion driver software.



Blended Motion

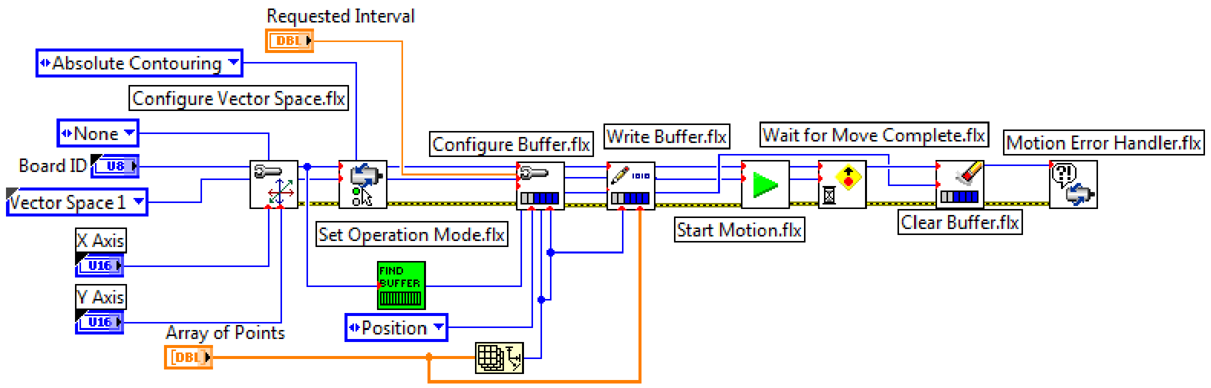
Blended motion involves two moves fused together by a blend that causes the moves to act as one. Blended moves require two moves and a blend factor that specifies the blend size. Blending is useful for applications requiring continuous motion between two different moves. However, in blended motion, your system does not pass through all of the points in your original trajectory. If the specific position along the path is important to you, consider a contouring motion.





Contoured Motion:

With contouring, you can supply a position buffer and create a smooth path or spline through them. Contouring holds an advantage over blending in that it guarantees that the system passes through each position.



Electronic Gearing

With electronic gearing, you can simulate the motion that would occur between two mating gears without using real gears. You use electronic gearing by supplying a gear ratio between a slave axis and a master axis, encoder, or ADC channel.

Motor Amplifiers and Drives:

The motor amplifier or drive is the part of the system that takes commands from the motion controller in the form of analog voltage signals with low current and converts them into signals with high current to drive the motor. Motor drives come in many different varieties and are matched to the specific type of motor they drive. For example, a stepper motor drive connects to stepper motors and not servo motors. Along with matching the motor technology, the drive must provide the correct peak current, continuous current, and voltage to drive the motor. If your drive supplies too much current, you risk damaging your motor. If your drive supplies too little current, your motor does not reach full torque capacity. If your voltage is too low, your motor cannot run at its full speed.

Motors and Mechanical Elements:

Motor selection and mechanical design is a critical part of designing your motion control system. Many motor companies offer assistance in choosing the right motor, but it is helpful to know some basics about motors before you start looking. Table 1 describes different motor technologies.

	Pros	**Cons**	**Applications**
Stepper Motors	Inexpensive, can be run open loop, good low-end torque, clean rooms	Noisy and resonant, poor high-speed torque, not for hot environments, not for variable loads	Positioning, micro movement
Brushed DC Servo Motors	Inexpensive, moderate speed, good high-end torque, simple drives	Maintenance required, no clean rooms, brush sparking causes EMI and danger in explosive environments	Velocity control, high-speed position control
Brushless Servo Motors	Maintenance-free, long lifetime, no sparking, high speeds, clean rooms, quiet, run cool	Expensive and complicated drives	Robotics, pick-and-place, high-torque applications

After determining which technology you want to use, you need to determine the torque and inertia at the motor shaft.

Feedback Devices and Motion I/O:

Feedback Devices

Feedback devices help the motion controller know the motor location. The most common position feedback device is the quadrature encoder, which gives positions relative to the starting point. Most motion controllers are designed to work with these types of encoders. Other feedback devices include potentiometers that give analog position feedback, tachometers that provide velocity feedback, absolute encoders for absolute position measurements, and resolvers that also give absolute position measurements. When using National Instruments motion controllers, you can use quadrature encoders and potentiometers.

Motion I/O

Other I/O that is important in motion control includes limit switches, home switches, position triggers, and position capture inputs. Limit switches provide information about the end of travel to help you avoid damaging your system. When a motion system hits a limit switch, it typically stops moving. Home switches, on the other hand, indicate the system home position to help you define a reference point. This is important for applications such as pick-and-place

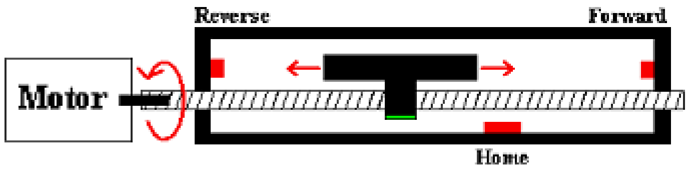


Image Acquisition And Processing:

Vision Basics contains the information about digital images. It contains information about the properties of digital images, image types, file formats, the internal representation of images in IMAQ Vision, image borders and image masks. The next important aspect is the display. It contains information about image display, palettes, regions of interest and nondestructive overlays. The last topic is about system setup and calibration. It describes how to set up an imaging system and calibrate the imaging setup so that you can convert pixel coordinates to real-world coordinates.

Digital Images:

Digital Images contain information about the properties of digital images, image types, file formats, the internal representation of images in IMAQ Vision, image borders and image masks. An image is a 2D array of values representing light intensity.

Image types in the IMAQ Vision libraries can manipulate three types of images: grayscale, color and complex images. A grayscale image is composed of a single plane of pixels. Each pixel is encoded using one of the following single numbers:
- An 8-bit unsigned integer representing grayscale values between 0 and 255.
- A 16-bit signed integer representing grayscale values between –32,768 and +32,767.
- A single-precision floating point number, encoded using four bytes, representing grayscale values ranging from –∞ to ∞.

A color image is encoded in memory as either a red, green and blue (RGB) image or a hue, saturation, and luminance (HSL) image. Color image pixels are a composite of four values. RGB images store color information using 8 bits each for the red, green and blue planes. HSL images store color information using 8 bits each for hue, saturation and luminance.

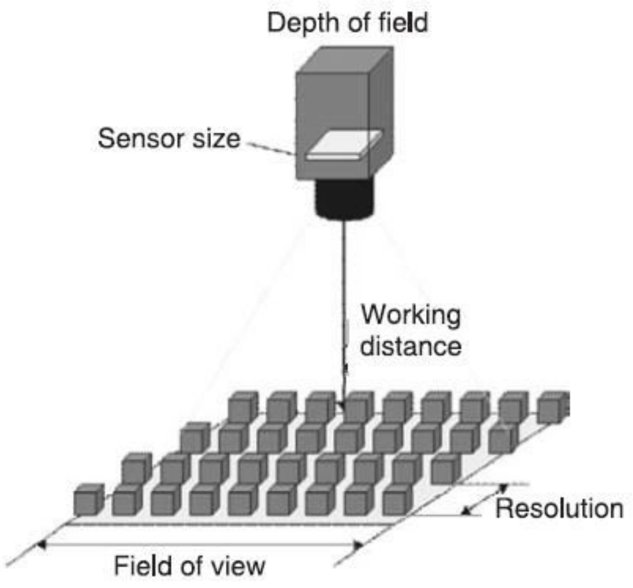
An image file is composed of a header followed by pixel values. Depending on the file format, the header contains image information about the horizontal and vertical resolution, pixel definition, and the original palette. Image files may also store information about calibration, pattern matching templates, and overlays. The following are common image file formats:
Bitmap (BMP)
● Tagged image file format (TIFF)
● Portable network graphics (PNG)—Offers the capability of storing image information about spatial calibration, pattern matching templates and overlays.
● Joint Photographic Experts Group format (JPEG)
● National Instruments internal image file format (AIPD)—Used for saving floating-point, complex and HSL images.

Many image processing functions process a pixel by using the values of its neighbors. A neighbor is a pixel whose value affects the value of a nearby pixel when an image is processed.

Display:
Displaying images is an important component of a vision application because it gives you the ability to visualize your data. Image processing and image visualization are distinct and separate elements. Image processing refers to the creation, acquisition and analysis of images. Image visualization refers to how image data is presented and how you can interact with the visualized images. A typical imaging application uses many images in memory that the application never displays.

At the time a grayscale image is displayed on the screen, IMAQ Vision converts the value of each pixel of the image into red, green and blue intensities for the corresponding pixel displayed on the screen. This process uses a color table, called a palette, which associates a color to each possible grayscale value of an image. IMAQ Vision provides the capability to customize the palette used to display an 8-bit grayscale image. A palette is a pre-defined or user-defined array of RGB values.

System Setup and Calibration:
Five factors comprise a imaging system: field of view, working distance, resolution, depth of field, and sensor size.



Fundamental parameters of an imaging system.

Resolution—The smallest feature size on your object that the imaging system can distinguish.
Pixel resolution—The minimum number of pixels needed to represent the object under inspection.
Field of view—The area of the object under inspection that the camera can acquire.
Working distance—The distance from the front of the camera lens to the object under inspection.
Sensor size—The size of a sensor's active area, typically defined by the sensor's horizontal dimension.
Depth of field—The maximum object depth that remains in focus.

IMAQ Vision has two algorithms for calibration: perspective and nonlinear. Perspective calibration corrects for perspective errors, and nonlinear calibration corrects for perspective errors and nonlinear lens distortion. Learning for perspective is faster than learning for nonlinear distortion. The perspective algorithm computes one pixel to real-world mapping for the entire image.

IMAGE PROCESSING AND ANALYSIS:

Image processing and analysis contains image analysis with information about histograms, line profiles, and intensity measurements. Image processing provides information about lookup tables, kernels, spatial filtering and grayscale morphology. Operators contain information about arithmetic and logic operators that mask, combine and compare images. Frequency domain analysis contains information about frequency domain analysis, the fast Fourier transform, and analyzing and processing images in the frequency domain.

Image Analysis:
Image analysis combines techniques that compute statistics and measurements based on the graylevel intensities of the image pixels. A histogram counts and graphs the total number of pixels at each grayscale level. From the graph, you can tell whether the image contains distinct regions of a certain gray-level value. A histogram provides a general description of the appearance of an image and helps identify various components such as the background, objects and noise. The histogram is a fundamental image analysis tool that describes the distribution of the pixel intensities in an image. Use the histogram to determine if the overall intensity in the image is high enough for your inspection task. You can use the histogram to determine whether an image contains distinct regions of certain grayscale values. You also can use a histogram to adjust the image acquisition conditions.

In saturation too little light in the imaging environment leads to underexposure of the imaging sensor, while too much light causes overexposure, or saturation, of the imaging sensor. Images acquired under underexposed or saturated conditions will not contain all the information that you want to inspect from the scene being observed. It is important to detect these imaging conditions and correct for them during setup of your imaging system. You can detect whether a sensor is underexposed or saturated by looking at the histogram. An underexposed image contains a large number of pixels with low gray-level values. This appears as a peak at the lower end of the histogram. An overexposed or saturated image contains a large number of pixels with very high gray-level values.

For example, a bright object with uniform intensity appears in the plot as a plateau. The higher is the contrast between an object and its surrounding background, the steeper is the slopes of the plateau. Noisy pixels, on the other hand, produce a series of narrow peaks. Intensity measurements measure the grayscale image statistics in an image or regions in an image.

Image Processing:
Image processing contains information about lookup tables, convolution kernels, spatial filters and grayscale morphology. The lookup table (LUT) transformations are basic image-processing functions that highlight details in areas containing significant information at the expense of other areas. These functions include histogram equalization, gamma corrections logarithmic corrections, and exponential corrections. Use LUT transformations to improve the contrast and brightness of an image by modifying the dynamic intensity of regions with poor contrast. A LUT transformation converts input gray-level values from the source image into other gray-level values in the transformed image.

Seven predefined LUTs are available in IMAQ Vision: Linear, Logarithmic, Power 1/Y, Square Root, Exponential, Power Y and Square. A convolution kernel defines a 2D filter that you can apply to a grayscale image. A convolution kernel is a 2D structure whose coefficients define the characteristics of the convolution filter that it represents. In a typical filtering operation, the coefficients of the convolution kernel determine the filtered value of each pixel in the image.

Grayscale morphology is morphological transformations extract and alter the structure of particles in an image. They fall into two categories:
● Binary morphology functions which apply to binary images
● Grayscale morphology functions which apply to gray-level images

Operators:
Arithmetic and logic operators can mask, combine and compare images. Operators perform basic arithmetic and logical operations on images. Use operators to add, subtract, multiply and divide an image with other images or constants.
We can perform logical operations, such as AND/NAND, OR/NOR, and XOR/XNOR, and make pixel comparisons between an image and other images or a constant. Common applications of these operators include time-delayed comparisons, identification of the union or intersection between images, correction of image backgrounds to eliminate light drifts, and comparisons between several images and a model.

Frequency Domain Analysis:
Information about converting images into the frequency domain using the fast Fourier transform, and information about analyzing and processing images in the frequency domain are important. Frequency filters alter pixel values with respect to the periodicity and spatial distribution of the variations in light intensity in the image. Unlike spatial filters, frequency filters do not apply directly to a spatial image, but to its frequency representation. The frequency representation of an image is obtained through the fast Fourier transform (FFT) function which reveals information about the periodicity and dispersion of the patterns found in the source image.

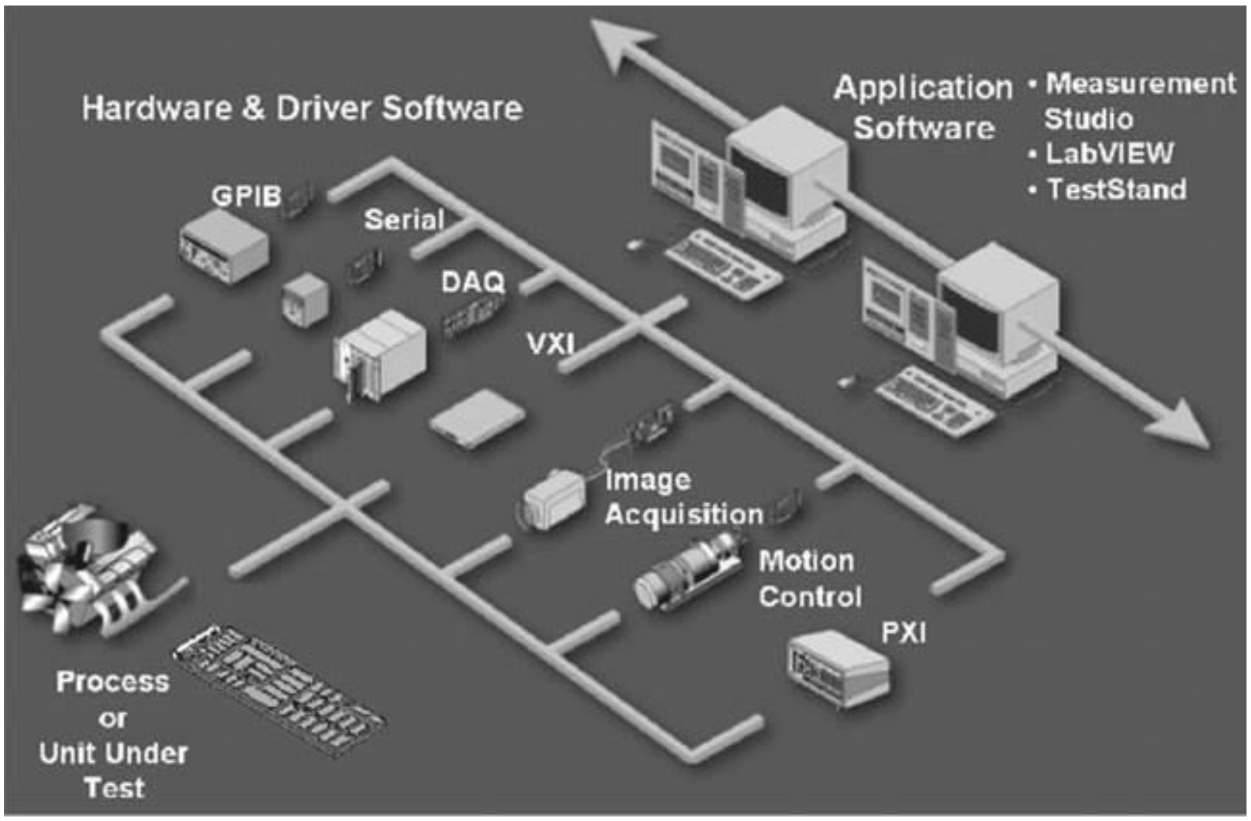
Frequency processing is another technique for extracting information from an image. Instead of using the location and direction of light-intensity variations, you can use frequency processing to manipulate the frequency of the occurrence of these variations in the spatial domain. This new component is called the spatial frequency which is the frequency with which the light intensity in an image varies as a function of spatial coordinates.

Use a low pass frequency filter to attenuate or remove, or truncate, high frequencies present in the image. This filter suppresses information related to rapid variations of light intensities in the spatial image. An inverse FFT, used after a lowpass frequency filter, produces an image in which noise, details, texture, and sharp edges are smoothed.

A high pass frequency filter attenuates or removes, or truncates, low frequencies present in the complex image. This filter suppresses information related to slow variations of light intensities in the spatial image.

A mask frequency filter removes frequencies contained in a mask specified by the user. Using a mask to alter the Fourier transform of an image offers more possibilities than applying a lowpass or highpass filter. The image mask is composed by the user and can describe very specific frequencies and directions in the image.

INSTRUMENT CONTROL:
National Instruments software, including LabVIEW and Measurement Studio, delivers PCbased data analysis, connectivity, and presentation power to new levels in measurement and automation applications. National Instruments hardware and software connect the computer to your application. By providing an extensive hardware selection, including data acquisition and signal conditioning devices, instrument control interfaces (such as GPIB, Serial, VXI and PXI), image acquisition, motion control and industrial communications interfaces, National Instruments offers the widest range of solutions for practically any measurement.



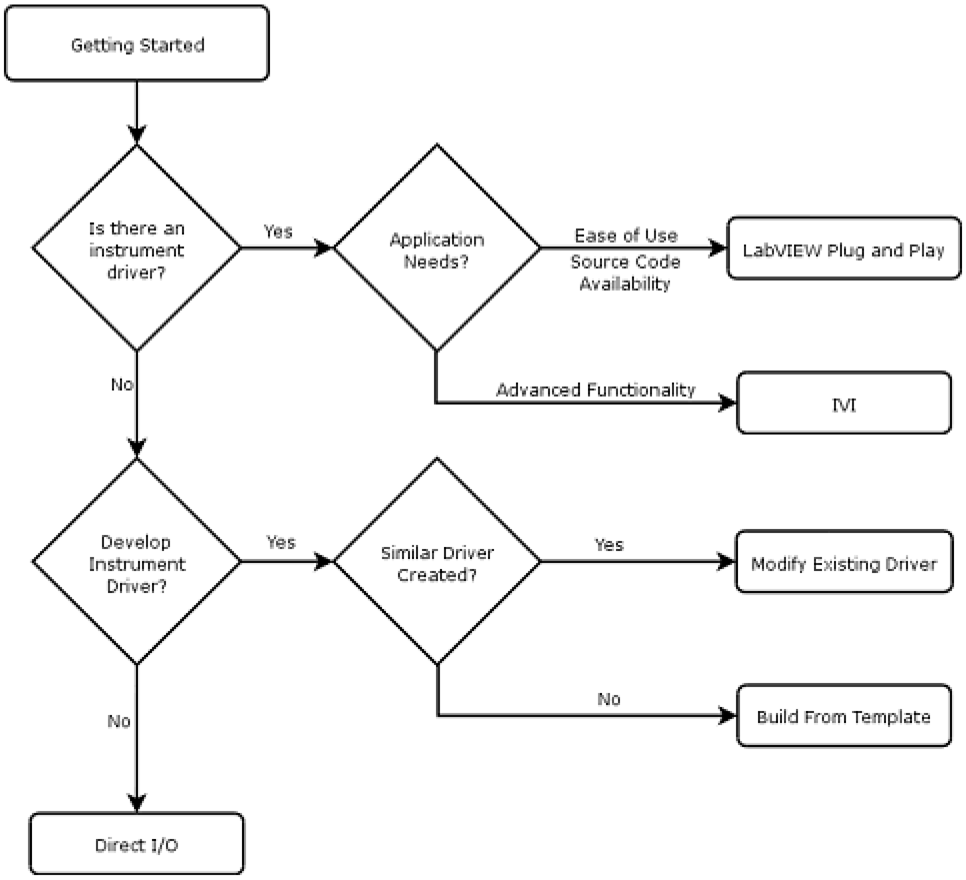
Computer-based measurement and automation

The most common categories of instrument interfaces are GPIB, serial, modular instruments and PXI modular instruments. Additional types of instruments include image acquisition, motion control, USB, Ethernet, parallel port, NI-CAN and other devices.

consider the following issues with PC control of instrumentation:
● Type of connector (pinouts) on the instrument
● Cables needed—null-modem, number of pins, male/female
● Electrical properties involved—signal levels, grounding, cable length restrictions
● Communication protocols used—ASCII commands, binary commands, data format
● Software drivers available

Instrument Control in LabVIEW:

With a wide array of instrument connectivity interfaces, it is important to choose the proper one for your application. The following flowchart will guide you through the choices you need to make when choosing the appropriate interface.



Flowchart on Choosing the Correct Interface

Instrument Drivers

In LabVIEW, an instrument driver is a set of VIs that communicates with an instrument. Each VI corresponds to a programmatic operation, such as configuring, reading from, writing to, and triggering an instrument. LabVIEW instrument drivers simplify instrument control and reduce test program development time by eliminating the need for you to learn the complex, low-level programming commands for each instrument. For more detailed information on how to use an instrument driver in LabVIEW.

Instrument I/O Assistant:

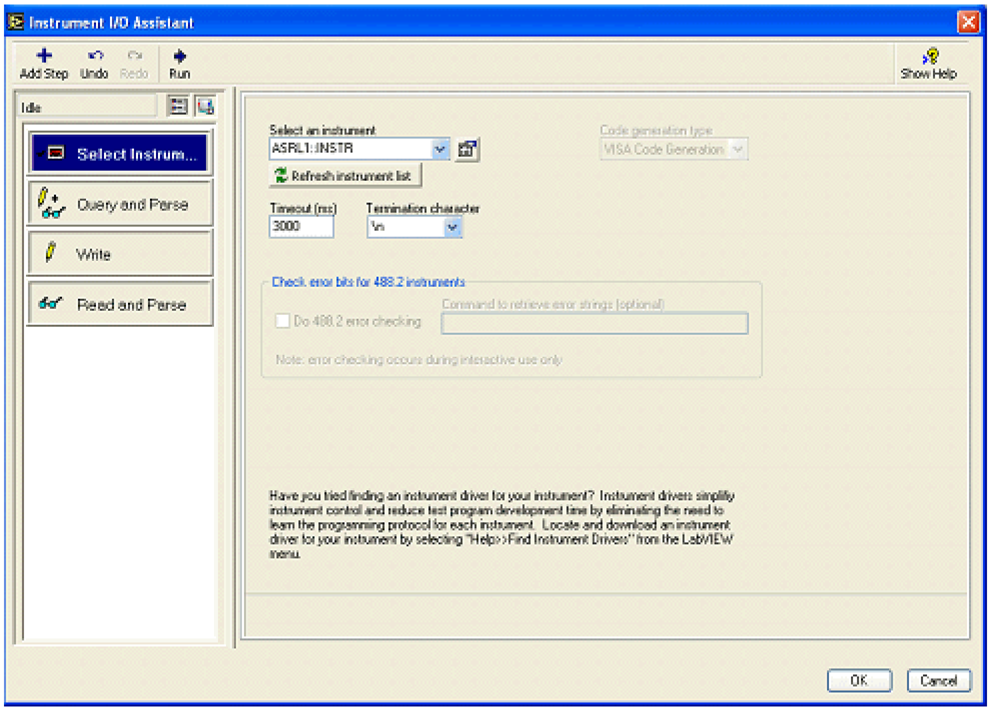
The Instrument I/O Express VI, found on the Functions>>Instrument I/O palette, launches the Instrument I/O Assistant, which you can use to communicate with message-based instruments and graphically parse the response. For example, you can communicate with an instrument that uses a serial, Ethernet, or GPIB interface.

Using the Instrument I/O Assistant

The Instrument I/O Assistant organizes instrument communication into ordered steps. To use the Instrument I/O Assistant, you place steps into a sequence. As you add steps to the sequence, they appear in the step sequence window. Use the view associated with a step to configure instrument I/O. Four steps are available in the Instrument I/O Assistant.

-**Select Instrument**—Use this step to select the instrument you want to communicate with and to configure basic instrument properties. This step appears in the step sequence window when you launch the Instrument I/O Assistant and must always be the first step in any Instrument I/O Assistant sequence.

-**Query and Parse**—Use this step to send a command to the instrument, read a response from the instrument, and parse the returned data.
-**Write**—Use this step to send a command to the instrument.
-**Read and Parse**—Use this step to read a response from the instrument and parse the returned data.



Select an instrument: ASRL1::INSTR
Code generation type: VISA Code Generation
Timeout (ms): 3000
Termination character: \n
Do 488.2 error checking
Command to retrieve error strings (optional)
Note: error checking occurs during interactive use only

Have you tried finding an instrument driver for your instrument? Instrument drivers simplify instrument control and reduce test program development time by eliminating the need to learn the programming protocol for each instrument. Locate and download an instrument driver for your instrument by selecting "Help>>Find Instrument Drivers" from the LabVIEW menu.

Instrument I/O Assistant

VISA API:
VISA is a standard I/O API for instrumentation programming. VISA can control GPIB, serial, USB, Ethernet, PXI, or VXI instruments, making the appropriate driver calls depending on the type of instrument you use so you do not have to learn

instrument-specific communication protocol. Before you begin using VISA, make sure you choose the appropriate method of instrument control.
Creating a Typical VISA Application

Use the I/O controls on the Controls>>I/O and Controls>>Classic>>Classic I/O palettes to specify the instrument or device resource you want to communicate with. Use the VIs and functions on the Functions>>Instrument I/O>>VISA palette to build VIs that control instruments.

Creating an Instrument Driver:
Use the Instrument Driver Project Wizard to create a new instrument driver project. Select Tools»Instrumentation»Create Instrument Driver Project to launch the Instrument Driver Project Wizard. After creating the new instrument driver, follow the Instrument Driver Modification Instructions in LabVIEW Help to complete the driver.

Use the Instrument Driver VI Wizard to create an instrument driver VI and insert the VI into the instrument driver project library. Right-click the project library file in the Project Explorer window and select New»Instrument Driver VI from the shortcut menu to launch the Instrument Driver VI Wizard.

CONTROL DESIGN AND SIMULATION TOOLS:
Control Design Toolkit: It provides a library of VIs and LabVIEW MathScript functions that you use to design, analyze and deploy a controller for a linear time-invariant dynamic system model. This toolkit includes frequency response analysis tools such as Bode, Nyquist and Nichols plots; time response analysis tools such as step and impulse response analysis; classical design tools such as Root Locus; and state-feedback design tools such as Linear Quadratic Regulators and pole placement. In addition, the Control Design Toolkit supports PID design, lead-lag compensators, predictive and continuous observers, and recursive Kalman filters for stochastic system models that incorporate measurement and process noise. You can also use the Control Design Toolkit and the LabVIEW Real-Time Module to deploy a discrete controller to a real-time target.

PID Control Toolkit: It offers PID and fuzzy logic control functions that you can combine with the math and logic functions already in LabVIEW to graphically develop control algorithms and programs for automated control.

Simulation Module: It provides VIs, functions and other tools that you use to construct and simulate all or part of a dynamic system model. This module supports both nonlinear and linear dynamic system models and includes tools for trimming and linearizing nonlinear models. The LabVIEW Simulation Module includes functions for describing continuous and discrete transfer function, zero-pole-gain and state-space models, as well as nonlinear phenomena such as friction, deadband

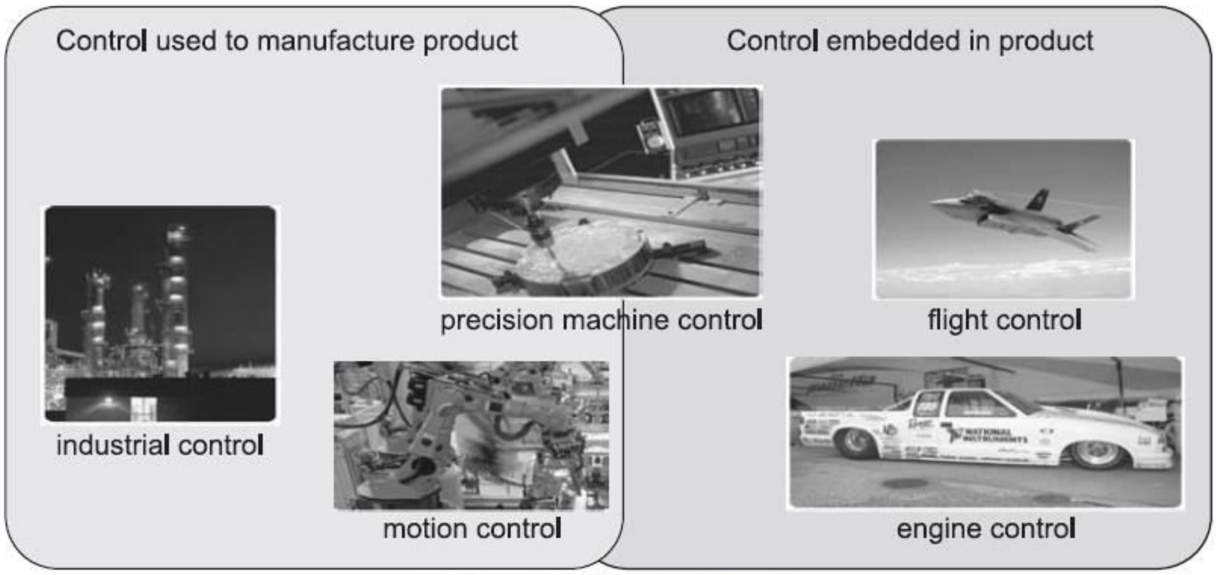
and backlash. You can interact with a model by using any of the VIs and functions included with LabVIEW itself. You also can use the Simulation Module and the LabVIEW Real-Time Module to deploy a continuous or discrete model to a real-time target.

Statechart Module: It assists in large-scale application development by providing a framework in which you can build, debug and deploy statecharts in LabVIEW. With the LabVIEW Statechart Module, you can create a statechart that reflects a complex decisionmaking algorithm. Then, you can generate the block diagram code necessary to call the statechart from a VI. The Statechart Module supports hierarchy, concurrency and an event-based paradigm. If you install the appropriate LabVIEW module, you can execute statecharts on supported real-time targets and National Instruments FPGA devices.

System Identification Toolkit: It combines data acquisition tools with system identification algorithms for accurate plant modeling. Use the LabVIEW System Identification Toolkit with National Instruments hardware, such as NI DAQ devices, to stimulate and acquire data from a plant and then identify a dynamic system model. This toolkit provides VIs that support parametric, nonparametric, partially-known, and recursive model estimation methods; AR, ARX, ARMAX, output-error, Box-Jenkins, transfer function, zero-pole-gain, and state-space model forms; and Bode, Nyquist, and pole-zero analysis.

CONTROL DESIGN TOOLKIT:
LabVIEW has a PID control toolkit that can be used to solve these applications. However, while PID control is sometimes sufficient for a given control application, there are still several cases where this doesn't make sense. In some cases, a more complex higher-order controller is required or the controller must compensate for non-linear behavior in the plant system. Sometimes the PID control algorithm is incapable of achieving the required performance. Finally, because the PID control algorithm is a single-input single-output controller, it cannot be used to handle complex multi-input multi-output systems. For instance, to balance and control the position of a helicopter, you need to read in several inputs and control several outputs.

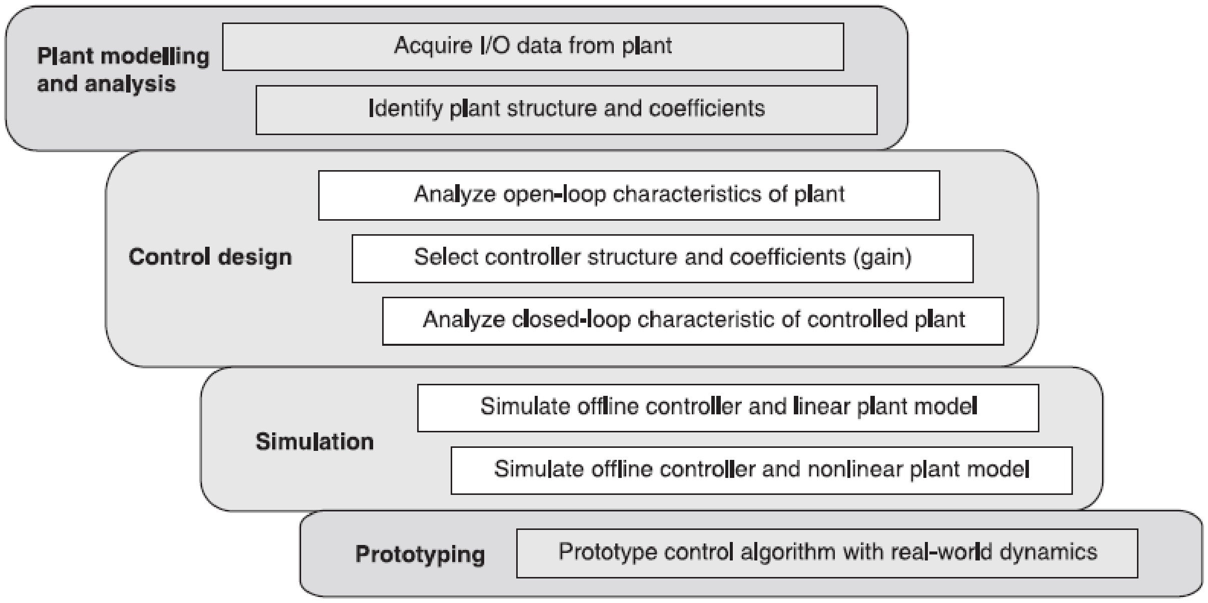
The control system used for processing may consist of a Programmable Logic Controller (PLC) executing a PID algorithm, or a Distribute Control System (DCS) for a larger process control. In this case, the control system is used to manufacture a product. A control system can also be part of an end-product being manufactured. This has been seen primarily in the automotive and aerospace industries with electronic control units and flight control systems. However, control systems are now finding their way into other end products such as precision motor controllers for computer hard drives and white goods like washing machines.


Control used to manufacture product
Control embedded in product
precision machine control
flight control
industrial control
motion control
engine control

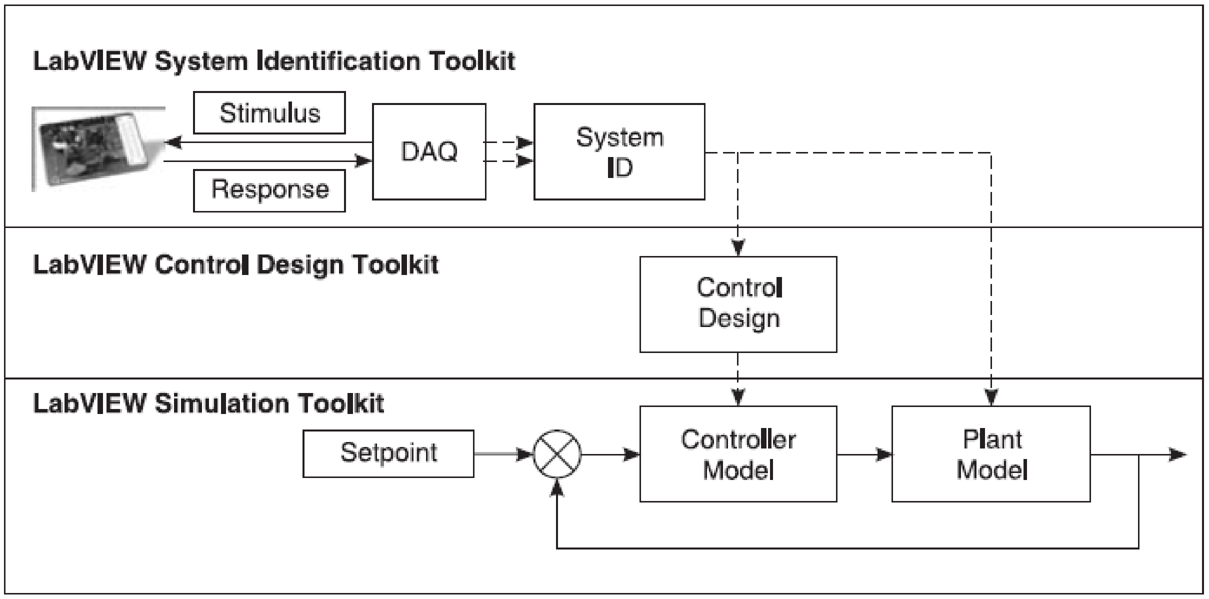
Applications of control systems.

Using the LabVIEW System Identification Toolkit, a design engineer will begin the control design process by creating a model of a plant. This begins with acquiring stimulus response data from the plant using standard data acquisition hardware and then selecting an plant structure that fits this data. From there, the engineer will find the right coefficients for the plant model that accurately represent the plant.

Next, the engineer will use the LabVIEW Control Design Toolkit to develop a controller for the plant. First, the engineer will analyze the open loop characteristics of the plant and then select a basic control structure to control this plant. From there, he/she will determine the right coefficients for this controller. Finally, the engineer will connect the controller with the plant and analyze the close-loop characteristics of the controlled plant.


Plant modelling and analysis
Acquire I/O data from plant
Identify plant structure and coefficients
Control design
Analyze open-loop characteristics of plant
Select controller structure and coefficients (gain)
Analyze closed-loop characteristic of controlled plant
Simulation
Simulate offline controller and linear plant model
Simulate offline controller and nonlinear plant model
Prototyping
Prototype control algorithm with real-world dynamics

Model-based control design process.


LabVIEW System Identification Toolkit
Stimulus
Response
DAQ
System ID
LabVIEW Control Design Toolkit
Control Design
LabVIEW Simulation Toolkit
Setpoint
Controller Model
Plant Model

LabVIEW control design tools.

SIMULATION MODULE:

Simulation is a process that involves using software to recreate and analyze the behavior of dynamic systems. You use the simulation process to lower product development costs by accelerating product development. You also use the simulation process to provide insight into the behavior of dynamic systems you cannot replicate conveniently in the laboratory. For example, simulating a jet engine saves time, labor, and money compared to building, testing, and rebuilding an actual jet engine.

OLE for Process Automation (OPC) OPC is the interoperability standard for the secure and reliable exchange of data in the industrial automation space and in other industries. It is platform independent and ensures the seamless flow of information among devices from multiple vendors. The OPC Foundation is responsible for the development and maintenance of this standard.

The OPC standard is a series of specifications developed by industry vendors, endusers and software developers. These specifications define the interface between Clients and Servers, as well as Servers and Servers, including access to real-time data, monitoring of alarms and events, access to historical data and other applications.

When the standard was first released in 1996, its purpose was to abstract PLC specific protocols (such as Modbus, Profibus, etc.) into a standardized interface allowing HMI/SCADA systems to interface with a "middle-man" who would convert generic-OPC read/write requests into device-specific requests and vice-versa. As a result, an entire cottage industry of products emerged allowing end-users to implement systems using best-of-breed products all seamlessly interacting via OPC.

Initially, the OPC standard was restricted to the Windows operating system. As such, the acronym OPC was borne from OLE (object linking and embedding) for Process Control. These specifications, which are now known as OPC Classic, have enjoyed widespread adoption across multiple industries, including manufacturing, building automation, oil and gas, renewable energy and utilities, among others.

With the introduction of service-oriented architectures in manufacturing systems came new challenges in security and data modeling. The OPC Foundation developed the OPC UA specifications to address these needs and at the same time provided a feature-rich technology open-platform architecture that was future-proof, scalable and extensible.
Today the acronym OPC stands for Open Platform Communications.

OLE for Process Control (OPC): Object Linking and Embedding (OLE) for process control (OPC) is a set of standards developed by a joint collaboration of leading automation industry suppliers. OPC's primary mission is to define a uniform interface for use with any organization or custom software package.

Or

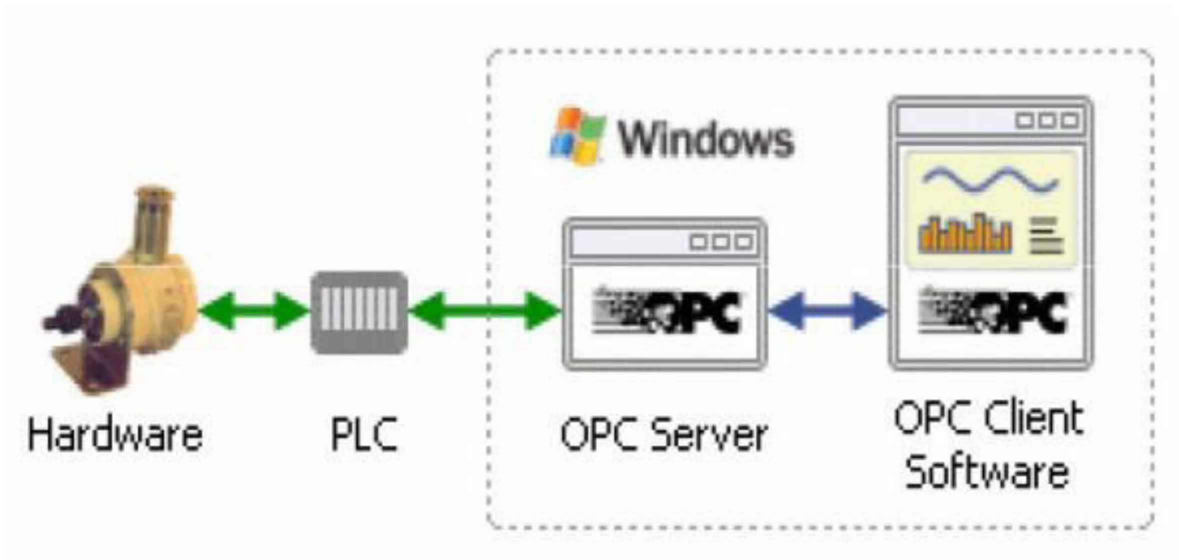
OPC (OLE for Process Control) is a communication standard based on OLE / COM technology. This is a special means of data exchange between MS-Windows applications. The OPC Foundation, which was created on the initiative of the main operators of the automation industry, publishes and maintains specifications that more specifically satisfy the industrial requirements of this sector.

OPC means “Open process control” or “Open platform communications” It is a standard that defines data communication between devices from different manufacturers .It Requires an OPC server to communicate with OPC clients .OPC allows “plug and play”, provides benefits as it reduces installation time and the opportunity to choose products from different manufacturers.

Different standards: “Real-time” data (OPC DA), Historical data (OPC HDA), Alarm & Event data (OPC AE), etc.

OPC is a standard interface to communicate between numerous data sources, including devices on a factory floor, laboratory equipment, test system fixtures, and databases. To alleviate duplication efforts in developing device-specific protocols, eliminate inconsistencies between devices, provide support for hardware feature changes, and avoid access conflicts in industrial control systems, the OPC Foundation defined a set of standard interfaces that allow any client to access any OPC-compatible device. Most suppliers of industrial data acquisition and control devices, such as Programmable Logic Controllers (PLCs) and Programmable Automation Controllers (PACs), are designed to work with the OPC Foundation standard.

Why to use OPC?



The purpose of the standard is to provide a common means for data access applications from any source, from field devices or from other files or applications. The great motivation to create the OPC ™ OLE for the Process Control standard is the need to establish a standard mechanism for communication between different data sources, either from field devices or even from other data files.

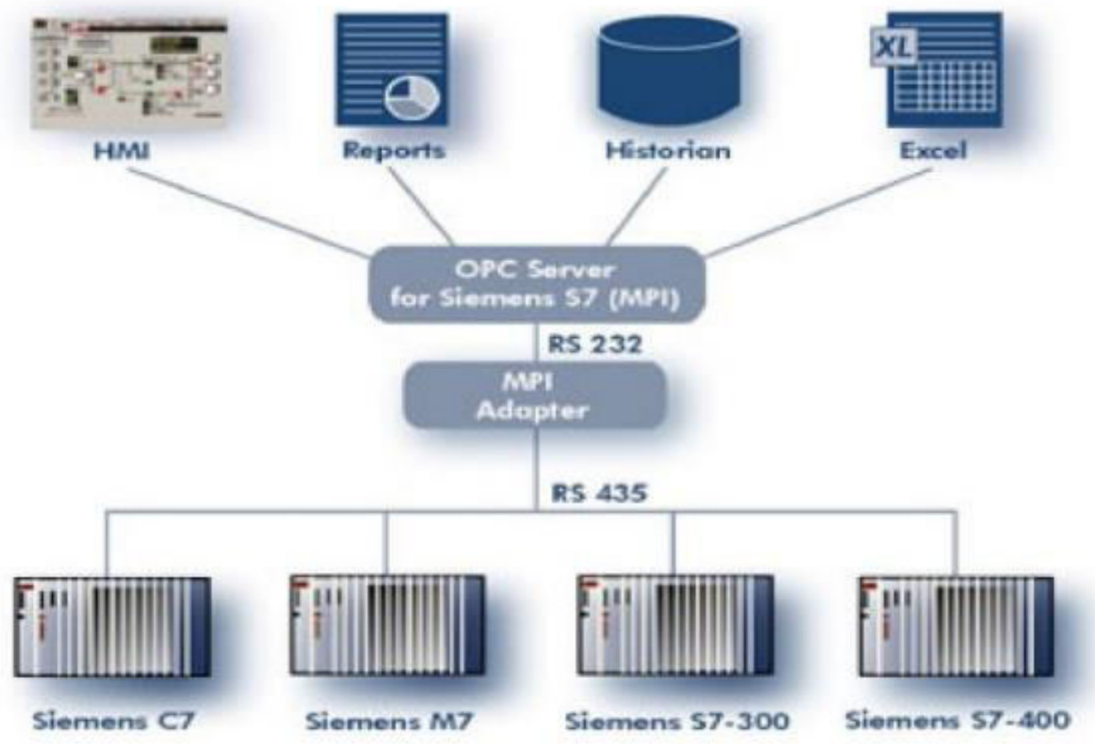
Information architecture in an Industrial Manufacturing Process involves three levels:

Field Management: with the advent of intelligent field equipment, a wide variety of data equipment, such as configuration and control data, may be users or even other applications.

Process Management: the use of Scada and SDCD control systems allows the decentralized control of industrial processes. The data provided can be considered together to allow effective and integrated of the entire Industrial Process.

Business/ Information Management: is the integration of factory floor information and individual management data from each controlled process with the company's corporate data, administrative and financial aspects. The data and information can be used by client applications to optimize the management and integration of the entire process of manufacturing.

Architecture of OPC applications:
OPC ™ - OLE for Process Control, implements two large modules: OPC Server and OPC Client. While the OPC Server specifies standard interfaces for direct access to equipment or applications, the OPC Client specifies the default interface for applications to access the collected data.



The data access server has three divisions:
Server – Contains all of the group objects
Group – Maintains information about itself and contains and organizes the OPC items

Item – Contains a unique identifier held within the group. The identifier acts as a reference for the individual data source, as well as value, quality, and timestamp information. The value is the data from the source. The quality status gives information about the device. The timestamp is the time that the data was retrieved.

There are three types of OPC data servers:

Data Access Server: direct access to equipment data;

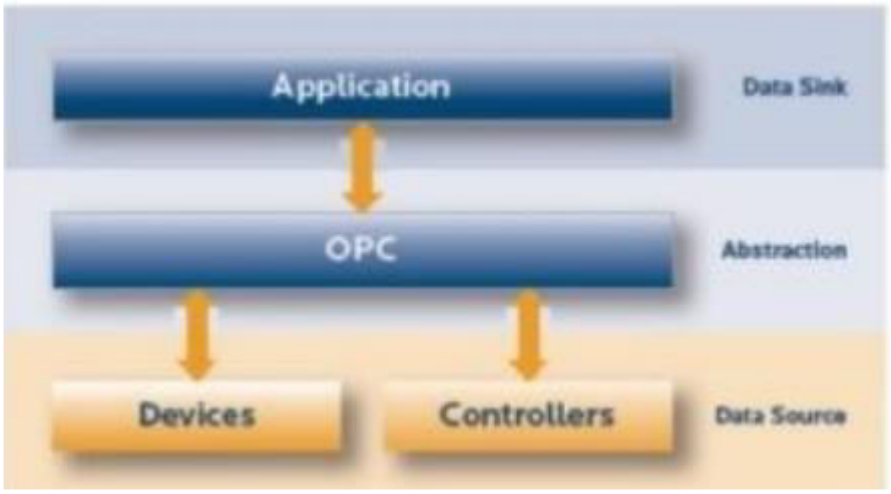
Alarm & Event Server: notification mechanism for OPC clients for specific events and alarm conditions;

Historical Data Server: reading, processing and editing historical data. OPC applications are written in several languages, such as Visual Basic, Delphi, Power Builder, etc. The OPC servers are written in C or C ++, taking advantage of the encapsulation characteristics of these languages, providing "objects" that can be accessed through any application

All specifications have been made to facilitate the development of OPC servers and can also be written in another language.

Access to OPC servers is done through the OLE / COM ™ and OLE / DCOM ™ components provided by the Microsoft ™ Windows ™ operating system.

How OPC Communication works? OPC can be represented as an "abstraction" layer that sits between the Data Source and the Data Sink, allowing them to exchange data without knowing anything about each other.



The OPC "device abstraction" is realized by using two, specialized OPC components called an OPC Client and OPC Server. Each of which is described in a following section. What's important to note is that just because the Data Source and Data Sink can communicate with each other via OPC does not mean their respective native protocols are no longer necessary or have been replaced by OPC. Instead, these native protocols and/or interfaces are still present, but only communicate with one of the two OPC components. In turn, the OPC components exchange information amongst each other and the loop is closed. Data can travel from the Application to the Device without having one talk directly to the other.

(OLE) - Object Linking Embedding: An object is an information unit that can be created and manipulated by users. It has intrinsic behaviour specified by its type, with its own commands and facilities. In other words, objects are data/software modules that can be included in software packages. They can be LINKED or EMBEDDED. Linked Objects: Stored separately and can be “partitioned” by multiple applications; Embedded Objects: are stored together with the applications. In this case, they are for the exclusive use of the same ones.

Benefits of using OPC:

- Simple to develop;
- Flexibility to “accommodate” the features of multiple manufacturers;
- High level of functionality;
- Enables efficient operation.
- Hardware manufacturers must only develop a set of components to access their equipment;
- Software developers do not have to rewrite drivers because of changes in equipment;
- Users have more options to develop international and integrated systems;