

Software-Defined Test Fundamentals

*Understanding the Architecture of Modular,
High-Performance Test Systems*



Contents

Executive Summary	4
Architecture Layer No. 5: System Management/Test Executive	5
Architecture Layer No. 4: Application Development Software	6
Architecture Layer No. 3: Measurement and Control Services	7
Architecture Layer No. 2: Computing and Measurement Bus	8
Architecture Layer No. 1: Measurement and Device I/O	9
Chapter 1: Developing a Modular Test Software Framework	10
National Instruments Modular Test Software Framework	10
Chapter 2: Choosing the Right Software ADE for Your Automated Test System	13
Factors to Consider When Selecting an ADE	13
NI LabVIEW	15
NI LabWindows/CVI	18
Microsoft Visual Studio .NET (C++, Visual Basic .NET, C#, and ASP.NET)	20
Chapter 3: Choosing the Right Data Bus	22
Understanding Bus Performance	22
Instrument Control Bus Comparison (GPIB, USB, PCI, PCI Express, and Ethernet/LAN/LXI)	24
Conclusion: Instrument Bus Performance	26
Chapter 4: Modular Instruments Basics	28
Anatomy of a Modular Instrument	29
Effects of Front End	31
Bandwidth	31
Accuracy	31
Effects of ADC/DAC	33
Sampling Rate	33
Resolution	33
Instrument Types	36
Digital Multimeters (DMMs)	36
Arbitrary Waveform Generators (Arbs) and Digitizers	36
Dynamic Signal Analyzers (DSAs)	36
RF Analyzers and Generators	37
Conclusion	37
Chapter 5: PXI Modular Instrumentation Platform	38

Hardware Architecture	38
PXI Chassis	39
PXI Controllers	39
Software Architecture	43
PXI – Industry Standard for Modular Instrumentation	43
Why Customers Choose PXI	44
Extension of the PXI Platform: PXI Express	45
Chapter 6: Case Studies	46
U.S. Navy: Developing Digital Test Equipment for Navy Aircraft Communications Using NI LabVIEW and the PXI Platform	49
Sanmina-SCI Exceeds Throughput Goals with PXI Tester and Multithreaded Software	51

©2009 National Instruments. All rights reserved. CVI, FlexDMM, HS488, LabVIEW, Measurement Studio, MXI, National Instruments, NI, ni.com, and NI TestStand are trademarks of National Instruments. The mark LabWindows is used under a license from Microsoft Corporation. Windows is a registered trademark of Microsoft Corporation in the United States and other countries. Other product and company names listed are trademarks or trade names of their respective companies.

Executive Summary

When designing automated test systems and choosing a test system architecture, you need to take into account several design considerations and challenges including increased device complexity, shorter development cycles, decreased budgets, and test system longevity. For your test system, determine the factors that are most important and choose an architecture that best meets your needs.

A modular, software-defined test system architecture, like the one shown in Figure 1, is based on a layered approach with the following advantages:

- **Increased test system flexibility** deployable to a variety of applications, business segments, and product generations
- **Higher-performance architectures** that significantly increase test system throughput and deliver tight correlation and integration of instruments from multiple suppliers including precision DC, high-speed analog and digital, and RF signal generation and analysis
- **Lower test system investments** by reducing initial capital investment and maintenance cost while increasing equipment use across multiple test requirements
- **Increased test system longevity** based on widely adopted industry standards that implement technology upgrades to improve performance and meet future test requirements

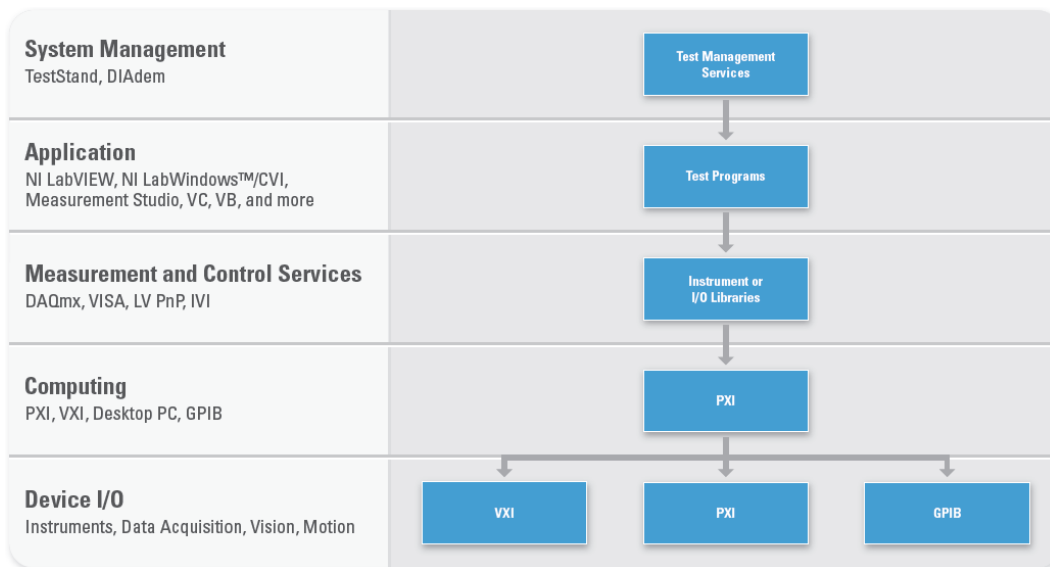


Figure 1. Five-Layer Architecture for Developing Test Systems

Architecture Layer No. 5: System Management/Test Executive

An automated test system requires the implementation of several tasks and measurement functions – some specific to the device under test (DUT) and others repeated for every device tested. To minimize maintenance costs and ensure test system longevity, it is important to implement a test strategy that separates the DUT-level tasks from the system-level tasks so you can quickly reuse, maintain, and change test programs (or modules) created throughout the development cycle to meet specific test requirements.

In any test system, operations are often *different* and *common* for each device tested such as system-level tasks.

Operations *Different* for Each Device

- Instrument configuration
- Measurements
- Data acquisition
- Results analysis
- Calibration
- Test modules

Operations *Common* for Each Device

- Operator interfaces
- User management
- DUT tracking
- Test flow control
- Results storage
- Test reports

Operations that are common for each device should be handled by the test executive. A test executive that handles the common operations can save time for your developers because they do not have to write the same code for multiple devices. They can spend their time writing the code to handle the operations that are different for each device. Using a test executive also ensures consistency among common operations and circumvents having multiple developers write the same type of code for several devices.

You can choose from several test executives. Some companies write their own test executive, and others opt to use commercially available software such as [NI TestStand](#). You should select the test executive that is best for your test system, whether you are creating a custom test executive or using one that is commercially available.

NI TestStand includes the Sequence Editor development environment for automated test system development, as shown in Figure 2.

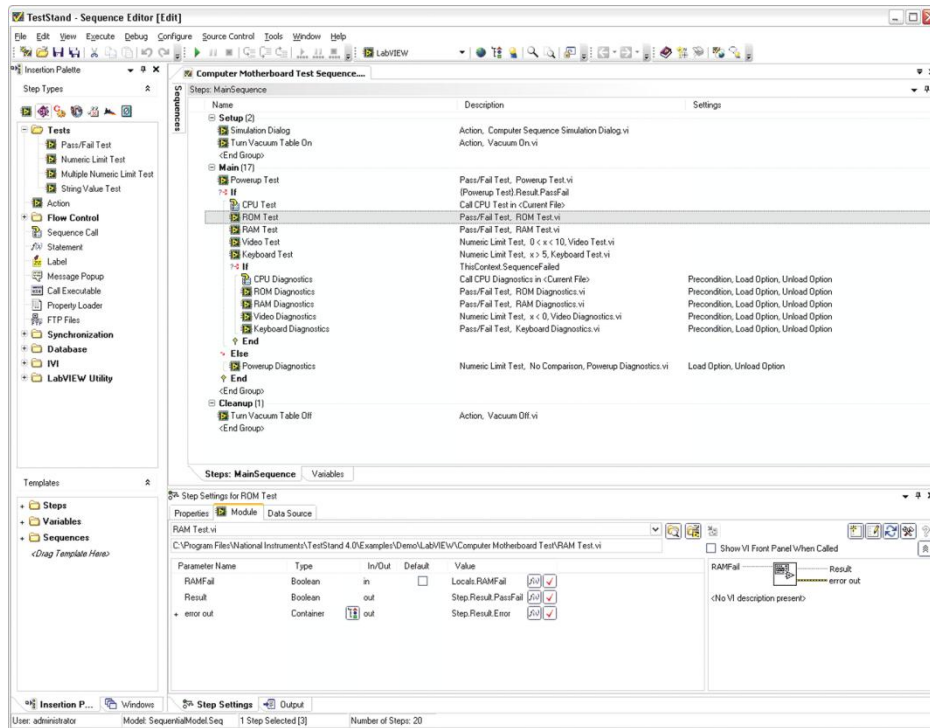


Figure 2. The NI TestStand Sequence Editor helps you shorten your automated test system development time.

With the NI TestStand Sequence Editor, you can create test sequences, which automate the execution of code modules written in any programming language. Each code module executes a test on the device under test and returns measurement information to NI TestStand. You can automatically log test result information in a report or database. In addition, systems written in NI TestStand can integrate with source code control, requirements management, and data management systems.

NI TestStand was designed to address four key areas: (1) simplify complex sequence development, (2) accelerate complex sequence development, (3) increase code and test system reusability and maintenance, and (4) improve test system execution performance. These focus areas have led to the adoption of NI TestStand in consumer electronics for validation and manufacturing test, military and aerospace applications, the medical industry, and IC characterization.

Architecture Layer No. 4: Application Development Software

The application development environment (ADE) plays a critical role in test system architectures. Using these tools, you can communicate with a variety of instruments, integrate measurements, display information, connect with other applications, and more. Ideally, the ADEs used to develop test and measurement applications provide ease of use, compiled performance, integration of a diverse set of I/O, and programming flexibility to meet the requirements for a range of applications.

You spend most of your development time working with an ADE, so it is critical to choose one that is easy to use, supports multiple platforms, and integrates easily with measurement and control services

such as drivers. Other features that you should consider when choosing an ADE for developing your test system are its presentation and reporting features, likelihood of product obsolescence, and the kind of training and support available worldwide.

Factors to consider when selecting an ADE include the following:

- Ease of use
- Measurement and analysis capabilities
- Integration with measurement and control drivers
- Training and support
- Multicore support
- Operating system independence
- Presentation and reporting features
- Protection against obsolescence
- Upgrades

Examples of ADEs include NI [LabVIEW](#), NI [LabWindows/CVI](#), and Microsoft Visual Studio.

Architecture Layer No. 3: Measurement and Control Services

Choosing test and measurement hardware with scalable software interfaces is another important layer in defining modular test architectures. Measurement and control services software provides modular software interfaces for configuring and programming your tests. Examples of these types of software interfaces include configuration managers such as NI Measurement & Automation Explorer (MAX), Virtual Instrument Software Architecture (VISA), Plug and Play instrument drivers, and Interchangeable Virtual Instrument (IVI) drivers. Other examples include instrument drivers provided by your instrument vendor such as NI-DAQmx.

Configuration Manager

A configuration manager, such as MAX, presents a unified system view of measurement hardware supported in the measurement and control services software. With MAX, you can define channel names to organize signals or specify scaling functions to convert digitized signals to measurement quantities. The key benefit of the configuration manager is the integration with the ADEs, which gives you the ability to easily integrate multiple measurements into a single application without tedious programming. With these configuration tools, you can avoid spending time configuring these measurement functions programmatically.

Instrument Connectivity

Integrating traditional instruments into the test software framework requires technologies such as Plug and Play instrument drivers and IVI to facilitate the communication with these instruments and their interchangeability. A Plug and Play instrument driver is a set of functions, or LabVIEW VIs, that control a programmable instrument. Instrument drivers help you get started using your instrument from your computer and save you development time and cost because you do not need to learn the programming protocol for each instrument. With open-source, well-documented instrument drivers, you can

customize your operation for better performance.

IVI implements a driver framework that facilitates instrument interchangeability by using a general API for each kind of instrument and separately implementing the driver to communicate with particular instruments. Separating the API from the particular driver implementation of each instrument gives you the ability to design a system using a particular IVI-compliant oscilloscope; after the system is deployed, you can change the brand and model of the instrument without having to rewrite the test application.

Programming Tools

Drivers can go beyond providing an easy-to-use API by adding tools to facilitate development and save you time. I/O assistants are interactive tools for rapidly creating a measurement or stimulus application. The DAQ Assistant, part of the NI-DAQmx driver, is an example of an I/O assistant. The DAQ Assistant presents a panel to the user for configuring common data acquisition parameters without programming. The combination of easy-to-use assistants and powerful programming environments is necessary to provide rapid development and the capabilities to meet a variety of application requirements.

Architecture Layer No. 2: Computing and Measurement Bus

At the center of every modern automated test system is a computer in the form of a desktop PC, server workstation, laptop, or embedded computer used with [PXI](#) and [VXI](#). An important aspect of the computing platform is the ability to connect (and communicate) with multiple instruments in a test system. Several instrumentation buses including GPIB, USB, LAN, PCI, and PCI Express are available for stand-alone and modular instruments. These buses have different strengths that make some more suitable for certain applications than others. For example, GPIB has the widest adoption for instrument control and availability of instrumentation; USB provides wide availability, easy connectivity, and high throughput; LAN is well-suited for distributed systems; and PCI Express delivers the highest performance.

The widespread use of the PC has generated the proliferation of high-performance internal buses including PCI and PCI Express, which offer the lowest latency and highest data throughput or bandwidth. The PCI bus provides up to 132 MB/s of bus bandwidth, and PCI Express, an evolution of PCI, can scale up to 4 GB/s to meet growing bandwidth needs and can provide complete software compatibility with PCI. Figure 3 illustrates the latency and bandwidth performance of the most popular instrument control buses. Oftentimes, you need a test system that incorporates multiple buses to maximize performance, longevity, and reusability.

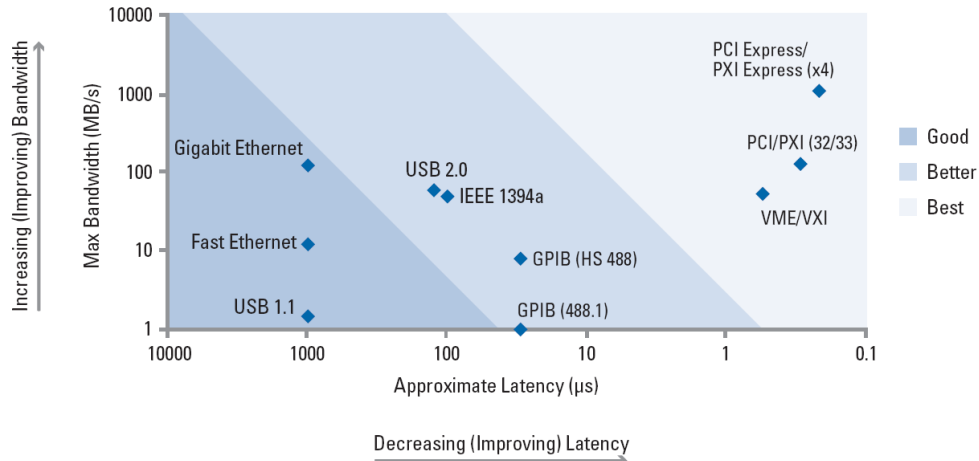


Figure 3. In an instrument control bus comparison, PCI and PCI Express provide the best bandwidth and latency or overall throughput performance.

Architecture Layer No. 1: Measurement and Device I/O

The final architecture layer, measurement and device I/O, includes the instruments and modules used in your test system. Depending on the computing and measurement bus you chose for layer No. 2, this layer can include VXI and/or PXI modules; GPIB, LAN, and/or USB benchtop instruments; and PCI or PCI Express instruments, as shown in Figure 4. Most automated test systems have several I/O devices connected by a mix of instrumentation buses. After defining the DUT or DUTs and the types of tests that you need to perform, you can choose the appropriate device I/O for your test system.

With a modular approach, you can define the test system measurement functionality and build systems that scale to meet future demands. Using a modular, software-defined approach, you can make custom measurements, perform measurements for emerging standards, or modify the system if requirements change to add instruments, channels, or new measurements. The combination of flexible, user-defined software and scalable hardware components is the core of modular instrumentation.



Figure 4. Example of a Hybrid Test System Based on PXI, LAN, and USB Instruments

Chapter 1: Developing a Modular Test Software Framework

Designing efficient test systems requires a modular software architecture and development tools optimized for test. To develop test systems faster and more cost-effectively, it is critical that you evaluate your test software architecture to maximize code reuse. Examining your test software architecture consists of evaluating the software development tools you are using and studying your approach to test code development. Understanding the importance of modular test software architectures and how to develop your tests as modules rather than building stand-alone applications significantly improves your test software reuse.

Incorporating modular test software architectures begins with choosing a software development environment that is designed for easily connecting to your instruments and quickly performing any type of measurement and analysis required for your tests. Such test software development tools include NI LabVIEW, LabWindows/CVI, and Measurement Studio for Visual Studio .NET. Using the proper test development environment, you can more easily share your test programs with others in your group and across test departments in your organization.

Test modules created by design engineers for prototype and validation tests are an integral asset to production test departments. Production test engineers can easily integrate tests developed during the prototype and validation phases into final automated test systems using industry-standard test management software such as NI TestStand.

NI TestStand provides built-in test management capabilities such as test module adapters for calling tests written in common test languages such as LabVIEW, LabWindows/CVI, C/C++, and Visual Studio .NET regardless of the function prototypes defined for each test. Maximizing code reuse between the product design and manufacturing teams reduces test development effort, so production schedules are met and demands for improved quality are upheld. The flexible NI TestStand module adapters ensure maximum code reuse across the product development cycle with minimal training and code.

National Instruments Modular Test Software Framework

The management level of the modular test software framework is responsible for directing the execution of the whole test system. The open software architecture of NI TestStand, a popular choice for test management software, greatly reduces the effort required to implement a scalable test software framework. It provides a completely modular and open architecture that you can use “as is” off-the-shelf or as individual components for designing your completely customized test systems based on NI TestStand. Figure 1 shows the NI TestStand architecture.

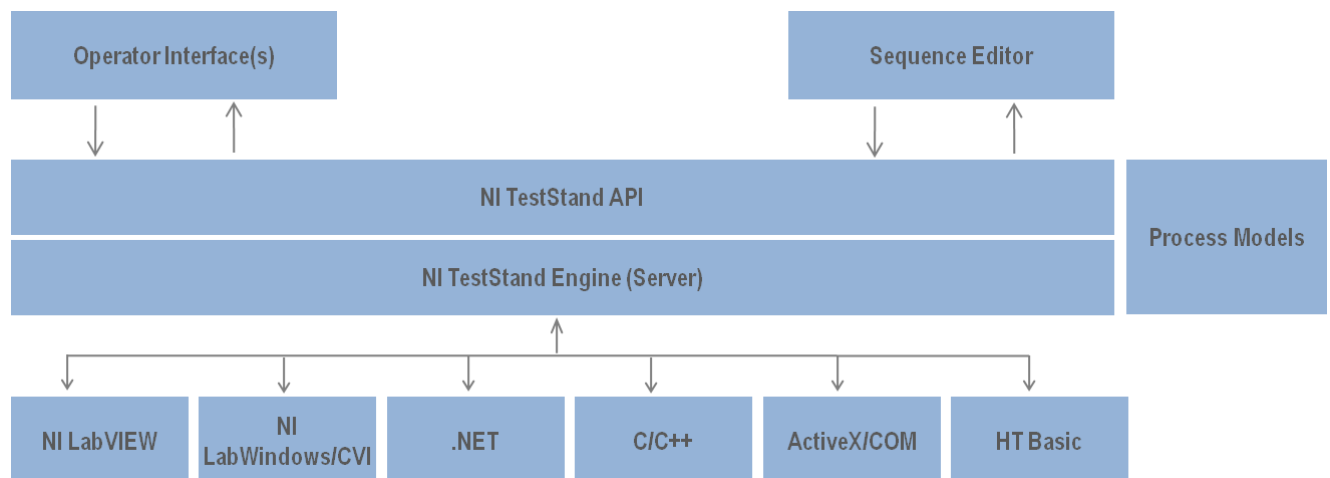


Figure 1. NI TestStand Test Management Software Architecture

The center of the NI TestStand architecture is the NI TestStand Engine, a powerful multithreaded test server with complete and extensively documented APIs. By communicating with the NI TestStand Engine, the module adapters provide an open language interface to automate tests written in any language. The process models offer superior modularity between the test code and the system-level functions that must be performed. The Sequence Editor provides an easy-to-use and powerful development environment for test sequences. Lastly, the operator interfaces are included in source code in multiple programming languages for rapid customization to match your exact needs.

NI TestStand Engine

All parts of the NI TestStand architecture are directed by the NI TestStand Engine, a set of libraries that export an ActiveX/COM API. With the API, you can perform any operation on the NI TestStand Engine programmatically by taking advantage of the more than 1,400 exported functions. The NI TestStand Engine implements multithreading so you can increase throughput by testing multiple units simultaneously. You also no longer have to include limit testing in your test code by implementing this functionality natively. By not including limit testing, test code becomes more flexible and reusable. Another feature of the NI TestStand Engine is that it implements flow control functionality much like any programming language. Finally, it increases the security of your test system by implementing multilevel user access and management.

Module Adapters

To call code written in different languages, the NI TestStand Engine uses the different module adapters available with NI TestStand. The module adapters provide an open language interface between the NI TestStand Engine and your test code written in LabVIEW, LabWindows/CVI, .NET, C/C++ DLLs, ActiveX/COM, and HT Basic. By calling code in different languages, you are able to take advantage of any legacy code as well as newer technologies. You can send and receive information from your code modules using an arbitrary number of parameters or by leveraging the NI TestStand API. Other module adapter features include stepping into code modules for debugging and leveraging code templates to improve programmer productivity.

Process Model

Testing a DUT requires more than just executing a set of tests. Usually, the test system must perform a series of operations such as identifying the DUT, logging results, and generating a test report. The set of such operations and their flow of execution are called a process model. Process models provide superior modularity between the test code and the system-level functions that must be performed by using these functions with multiple test sequences. NI TestStand is shipped with three process models that you can use as is or fully customize. The Sequential process model tests one unit at a time and the Batch and Parallel process models, featuring NI TestStand multithreading functionality, test more than one unit at the same time.

Sequence Editor

The Sequence Editor offers all the functionality and tools you need to develop the most sophisticated automated test systems. In the Sequence Editor, you can create, debug, and modify test sequence files. These files contain test steps that can include code modules developed in any test programming language. Furthermore, the Sequence Editor features a utility to build deployment packages to ease the distribution of test sequences and operator interfaces. The Sequence Editor also provides user management services, which prevents some users from accessing restricted functionality based on the privileges defined by your NI TestStand administrator. Figure 2 shows the Sequence Editor displaying a test sequence written in LabWindows/CVI.

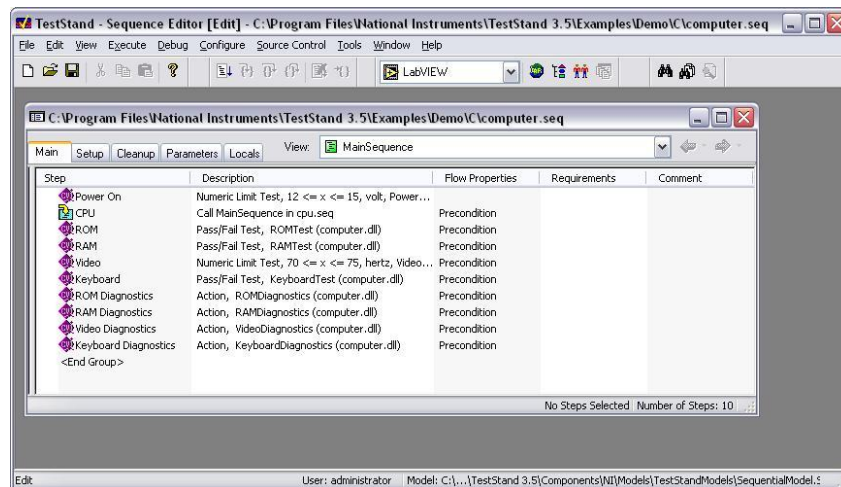


Figure 2. NI TestStand Sequence Editor

Operator Interfaces

Lastly, an operator interface is a customizable user interface for NI TestStand that you can use to execute and debug test sequence files created in the Sequence Editor. The operator interface is typically used on a manufacturing floor or if you need to deliver a custom look and feel to your test or validation system. NI TestStand operator interfaces use NI TestStand user interface controls, which facilitate development by fully implementing common features such as sequence file display and execution tracing. NI TestStand includes ready-to-run operator interfaces written in LabVIEW, LabWindows/CVI, C#, Visual Basic, and Visual Basic .NET.

Chapter 2: Choosing the Right Software ADE for Your Automated Test System

Application development environments (ADEs) play a critical, visible role in a test software framework. With these tools, the system developer designs and integrates the system that takes measurements, displays information to the end user, connects with other applications, and much more. Today, test engineers spend most of their development time working with an ADE. Hence, it is critical to select an ADE that not only is intuitive but can support multiple platforms and integrate easily with measurement and control services such as drivers. Other features you should consider when selecting an ADE for your test system development are its presentation and reporting features, how protected you are from the obsolescence of the product, and what kind of training and support is available worldwide. This paper discusses how three different ADEs – LabVIEW, LabWindows/CVI, and Visual Studio .NET – compare on these characteristics.

Factors to Consider When Selecting an ADE

Ease of Use for New Software Engineers

Because the ADE is where the heart of an automated system is developed, ease of use in these tools is critical to the productivity of a new software engineer. Ease of use goes beyond how quickly someone can get up and running. For example, developers should be able to easily integrate processing routines with multiple measurement devices, create sophisticated user interfaces, deploy and maintain an application, and modify the application as product designs evolve and system needs expand. Other ADE features needed include extensive documentation and example code.

Measurement and Analysis Capabilities

It is critical that the ADE used to develop a test system can seamlessly manage and process measurements. To do this effectively, the ADE should incorporate measurement data types directly in the environment so that the data is easy to use in additional processing routines. For maximum development productivity, the ADE should include comprehensive statistical and numerical analysis functions as well as high-performance signal processing and control algorithms common in measurement applications.

Multicore and Parallelism Support

Multicore technology has become a standard feature in automated test systems and a necessity for today's electronic devices that are processing unprecedented amounts of data. Multicore processors present new software challenges that must be overcome to fully take advantage of processing capabilities in a multithreaded application. An ADE must offer developers with programming techniques to create processes to execute in parallel.

Integration with Measurement and Control Drivers

Too often, test system developers assume a device driver alone is sufficient for effectively integrating their measurement devices. The driver is not enough; measurement and control drivers should be integrated as seamlessly as possible with the ADE. In the ideal case, the software that controls the measurement devices is transparent, appearing only as part of the ADE. This ideal implementation guarantees maximum development flexibility and a scalable architecture that organizations can deploy on all of the platforms the ADE targets.

Training and Support

The ease of use of an ADE can only go so far in making it simple for new users to learn the application. Hence, ADE vendors should provide manuals and online training for engineers to quickly learn how to use their products. Advanced users also might need classroom training to further their knowledge and learn more about system-level design concepts. This classroom training should give developers the opportunity to attain proof of their knowledge by going through a certification process. Another consideration to make when selecting an ADE is the type of vendor support you have access to when developing your application, such as phone and e-mail support. Furthermore, if you are going to standardize on an ADE worldwide, you want to consider whether your engineers around the world have access to support in their own languages.

Platform Independence

Test software applications today target several different architectures. It is important that you choose an ADE that is flexible enough to support all of these different architectures as seamlessly as possible. Different OSs such as Windows, Linux, and Macintosh can offer different benefits depending on the application. You should be able to port your code from one platform to the other. If the ADE does not support these multiple platforms, you must use different ADEs for different projects and spend crucial time porting your intellectual property from one platform to the other.


Presentation and Reporting Features

Test applications present many presentation and reporting challenges due to their emphasis on the graphical representation of data. The ADE should have multiple visual components for data visualization such as charts, graphs, knobs, and meters. Reporting should also be easy to facilitate the communication of the information acquired by the system. Some of the most popular reports, such as Microsoft Word and Excel, should be simple to generate. The communication of results should also be easy to implement by either publishing the application on the Web or logging information to a database.

Protection Against Obsolescence

Standardizing on an ADE for the development of your test system is a big commitment. It is important that your investment is protected from the obsolescence of the product. One of the characteristics you should consider is the product's track record of integrating with the latest software technologies and its ability to protect you against discontinuous shifts in test software development. Furthermore, the product should go through routine upgrades to add new functionality.

		NI LabVIEW	NI LabWindows/CVI	Microsoft Visual Studio
Factors	Ease of Use for New Engineers	●	◐	◐
	Measurement and Analysis Capabilities	●	●	○
	Multicore and Parallelism Support	●	◐	○
	Integration with Measurement and Control Drivers	●	●	○
	Training and Support	●	●	◐
	Platform Independence	●	◐	◐
	Presentation and Reporting Features	●	◐	○
	Protection Against Obsolescence	●	●	◐



○ Below ◐ Average ● Above

Table 1. Different ADEs feature different benefits and challenges during test system development.

NI LabVIEW

LabVIEW is a graphical development language that helps engineers and scientists create flexible and scalable test applications rapidly and at minimal cost. It uses a graphical development paradigm instead of relying on text-based programming. The LabVIEW graphical dataflow language and block diagram approach naturally represent the flow of your data and intuitively map user interface controls to your data, so you can easily view and modify your data or control inputs. Figure 1 depicts the block diagram of an application and its respective front panel written in LabVIEW.

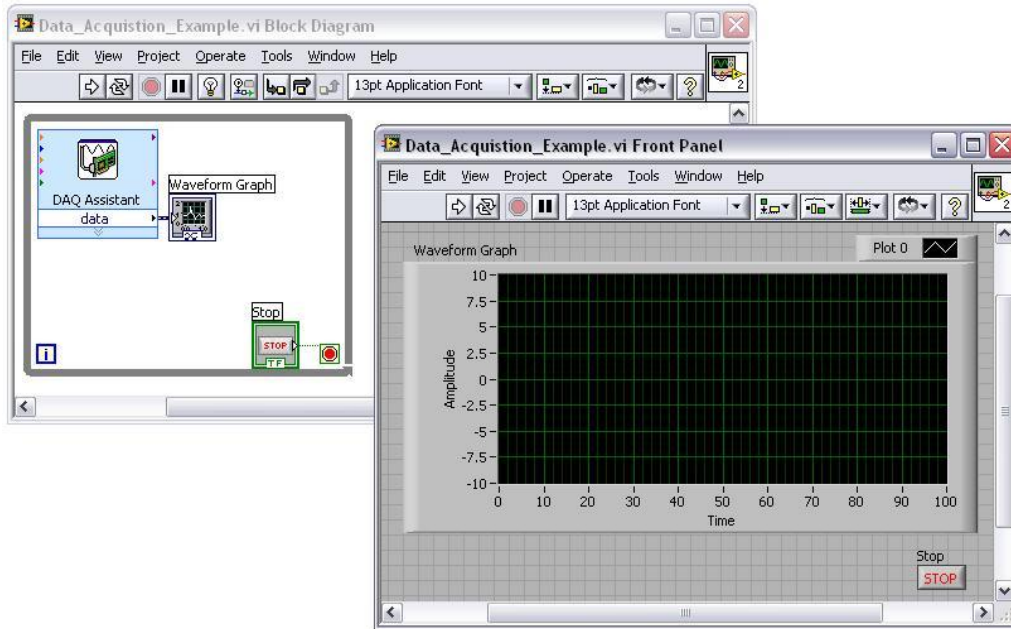


Figure 1. The LabVIEW ADE

LabVIEW also has features to help you reference the extensive documentation included with the product. Using the Context Help feature, you can leverage the graphical nature of LabVIEW to access a subVI's documentation by simply hovering over it. In addition, LabVIEW emphasizes the use of the hundreds of example programs available with the product and online as a means of demonstrating and teaching different features.

Despite the sophistication of the underlying algorithms, LabVIEW analysis tools are easy to use. More than 15 analysis Express VIs, such as the Spectral Measurements Express VI, reduce the complexity of implementing measurement analysis in your application through interactive configuration dialogs in which you can preview analysis results immediately. You can use these and other measurement analysis tools to input real-world, time-domain signals directly from data acquisition hardware and provide results ready for charting, graphing, or further processing. With these functions, you easily can determine signal characteristics such as DC/RMS levels, total harmonic distortion (THD/SINAD), impulse response, frequency response, and cross-power spectrum.

To benefit from the improved processing performance of multicore technology, however, engineers must be able to program their test code to target the different cores. LabVIEW is inherently parallel and can automatically generate programs optimized for multiple processing cores. In LabVIEW, two loops that do not share a data dependency automatically execute in separate threads, as shown in Figure 2.

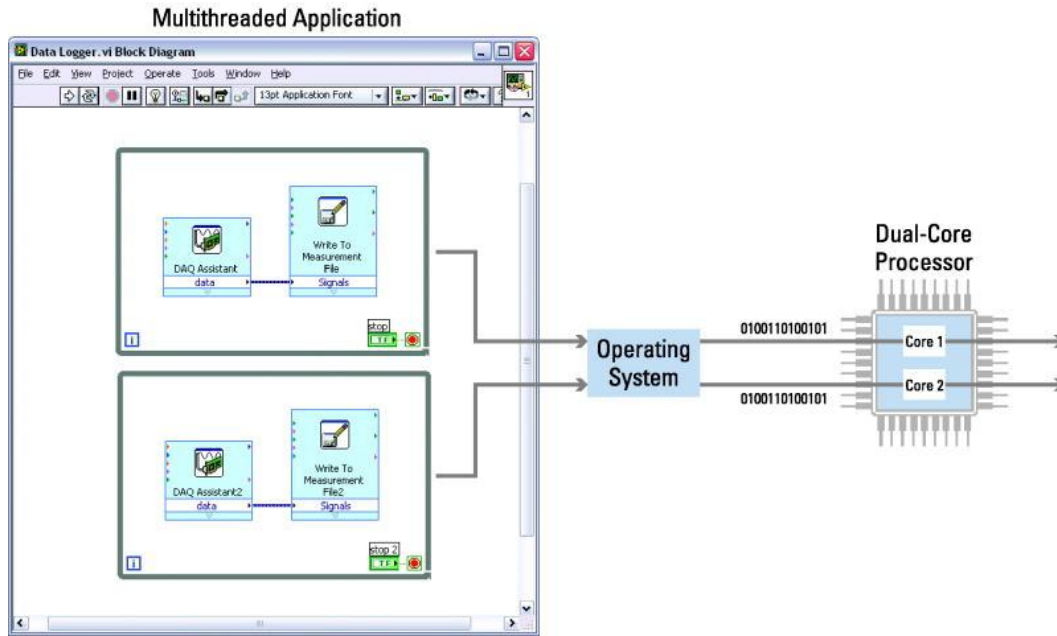


Figure 2. LabVIEW inherently handles the parallelization of code to target multiple cores.

Consider Table 2 as a representative example of the performance improvement achieved with a LabVIEW application on a dual-core processor versus a single-core processor, such as the 2.16 GHz Intel Core 2 Duo T7400 dual-core processor used in the new NI PXIe-8106 embedded controller. With LabVIEW, you can take advantage of existing code on multicore processors to enhance the performance of your test applications.

Benchmark	Single-Core 2 GHz	Dual-Core 2 GHz	Percentage Improvement
Program to find the prime numbers in the first 1,000,000 natural numbers	6.87s	3.59s	47.74%
Program to calculate 1,000 digits of Pi	3.96s	3.06s	22.73%

Table 2. By executing tasks in parallel, the same LabVIEW application, which finds all the prime numbers in the first 1,000,000 natural numbers, runs 47.74 percent faster on a dual-core processor.

LabVIEW offers tight integration with measurement and control drivers, which simplifies connecting to and communicating with thousands of instruments from hundreds of vendors. With this software, you can quickly acquire data from GPIB, serial, Ethernet, PXI, USB, and VXI instruments using instrument drivers, interactive assistants, and built-in instrument I/O libraries. Furthermore, LabVIEW includes easy-to-use libraries and interactive assistants to communicate with the National Instruments line of modular instruments and data acquisition products.

National Instruments offers LabVIEW training for any level of expertise. While basic courses target nonprogrammers and developers who want to learn the product, intermediate and advanced users also can find content that is useful for their level of expertise. On-site courses help organizations train a large number of developers quickly without having to leave the company. Online and self-paced courses target those developers who wish to increase their knowledge on their own time and at their own pace.

Although LabVIEW is usually seen as a Windows OS application, the product's original OS was the Macintosh. National Instruments ported LabVIEW to Windows as the OS's importance increased in the desktop PC industry. The NI commitment to ensure that LabVIEW supports new platforms continues today. LabVIEW continues to support both Windows and Macintosh but has also added support for Linux due to its increasing popularity among customers. Being able to run LabVIEW VIs on different OSs means that no matter which computing platform you work with, you can use your LabVIEW knowledge. LabVIEW can even run on other targets such as real-time systems and even field-programmable gate arrays (FPGAs) and digital signal processors (DSPs).

LabVIEW presentation and reporting features are a big part of why the ADE is so well-suited for test software development. LabVIEW contains multiple graphs, charts, meters, knobs, and switches in both 2D and 3D to help you represent measurement data graphically. The ADE also includes the LabVIEW Report Generation Toolkit, which you can use to create reports in Microsoft Word and Excel formats. If you need to communicate results by exporting the application through the Web, you can use LabVIEW remote panels to display the front panel over the Web on any browser. On the other hand, if you need to log the results of your measurements to a database, the LabVIEW Database Connectivity Toolkit offers a set of easy-to-use tools with which you can quickly connect to local and remote databases and perform many common database operations.

Finally, National Instruments has continually emphasized that it is committed to help its LabVIEW customers fight obsolescence. Even though a large amount of development effort has been focused on adding new features and integrating new technologies, running code from previous versions on newer versions has always been a priority. Running older code on newer versions of the product means that the valuable resources that were dedicated to creating previous applications are not wasted and can be leveraged in newer applications.

NI LabWindows/CVI

LabWindows/CVI is a proven test and measurement ANSI C development environment that greatly increases the productivity of engineers and scientists. Figure 3 displays the LabWindows/CVI development environment.

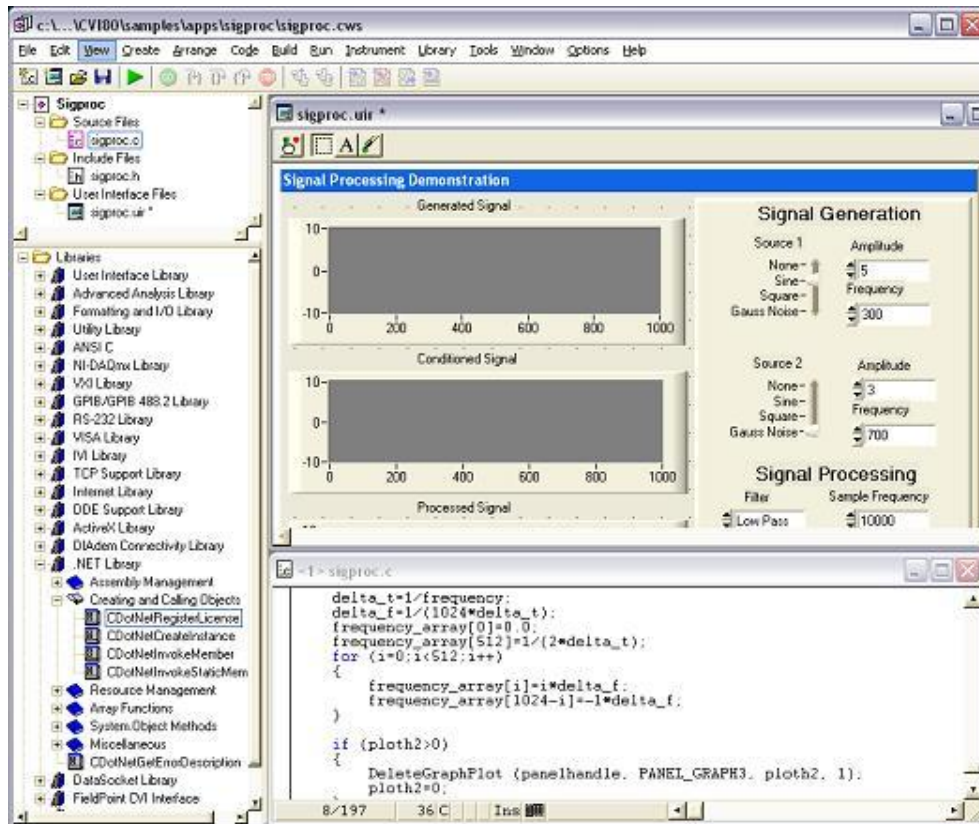


Figure 3. LabWindows/CVI features a complete workspace you can use to quickly develop, debug, and manage large applications.

Engineers and scientists use LabWindows/CVI to develop high-performance, stable applications in the manufacturing test, military and aerospace, telecommunications, design validation, and automotive industries. LabWindows/CVI streamlines development in these areas with hardware configuration assistants, comprehensive debugging tools, and interactive execution capabilities you can use to run functions at design time.

Toolkits such as the Advanced Analysis Library complement the analysis libraries included with LabWindows/CVI to help you analyze your measurement data. The LabWindows/CVI Advanced Analysis Library offers a comprehensive set of functions for analyzing your data. With these powerful analysis routines, you can easily convert raw data into useful information and build test applications. The Advanced Analysis Library includes functions for signal generation, 1D and 2D array manipulation, complex operations, signal processing, statistics, and curve fitting.

LabWindows/CVI is an industry leader in instrument control and connectivity because of the [NI Instrument Driver Network](#) of more than 8,000 instrument drivers from more than 200 vendors. You can use these drivers to easily program instrument control applications. With the Instrument I/O Assistant, you can generate code to communicate with devices such as serial, Ethernet, and GPIB instruments without using an instrument driver. The Instrument I/O Assistant offers a simple interface for quickly prototyping applications and autoparsing instrument data without any programming. You can easily

import the code generated into any application, which removes the tedium of writing instrument connectivity, basic communication, and string parsing code. In addition to the integrated NI-DAQmx Libraries, LabWindows/CVI provides the DAQ Assistant, an interactive interface to the data acquisition driver framework.

The training and support structure you can take advantage of for LabVIEW is also available for LabWindows/CVI. LabWindows/CVI has a variety of training courses that target different levels of expertise with the product. Organizations that need to train a large number of developers quickly without their having to travel can choose from on-site courses. Engineers who wish to increase their knowledge on their own time and at their own pace can select training in the form of online and self-paced courses. To complement the training opportunities for LabWindows/CVI, National Instruments applications engineers from local branches around the globe provide worldwide support.

By maintaining the backward compatibility of LabWindows/CVI, National Instruments helps to protect you from obsolescence. You not only can run C code developed many years ago or LabWindows/CVI code created in a previous version of the product but also run the applications faster with new optimizing compiler integration. The National Instruments commitment to ensure LabWindows/CVI offers backward compatibility is critical to industries that value longevity and continuity such as military and aerospace.

Microsoft Visual Studio .NET (C++, Visual Basic .NET, C#, and ASP.NET)

Visual Studio .NET offers a powerful ADE by supporting multiple programming languages such as C++, Visual Basic .NET, C#, and ASP.NET. With the option to select any of these programming languages, you can use the same tool and leverage the expertise of your developers even if their knowledge focuses on different programming languages. You can run applications developed in Visual Studio .NET on a PC as well as the Web by using the ASP.NET language.

Visual Studio .NET provides functionality to develop in different programming languages such as C++, Visual Basic .NET, and C#. By enabling these programming languages to compile to the Common Language Runtime, you can add libraries developed in different languages. On the other hand, the fact that the .NET platform is supported only by the Microsoft Windows OS means that you are limited in the number of OSs you can use to run your application. Furthermore, porting your application to another OS in the future might require rewriting the application in a different language.

By default, Visual Studio .NET does not include any functionality to integrate with measurement and control drivers or perform any analysis operations. Components such as those offered by NI Measurement Studio, as shown in Figure 4, can provide access to measurement and instrument drivers and analysis functionality. These components increase the ability of the ADE to integrate with instrument and measurement drivers by providing interactive assistants to generate code automatically. In contrast, there are certain features of the .NET framework that make it inherently difficult to communicate with some instruments. The .NET framework executes code in the Common Language Runtime, which prevents you from accessing the hardware. When you cannot access the hardware, it is

Chapter 3: Choosing the Right Data Bus

GPIB, USB, PCI/PCI Express, PXI/PXI Express, and Ethernet/LAN are some of the most popular communication buses available for automated test systems. The challenge for today’s test engineer is not to choose a single bus or platform on which to standardize every single application, but to choose a bus or platform appropriate for a specific application or even a specific part of an application. This chapter presents a head-to-head comparison of the most popular instrumentation buses to help you make informed decisions when choosing the bus and platform technologies that meet your application-specific needs. Specific bus technologies discussed below include GPIB, USB, PCI, PCI Express, and Ethernet/LAN/LXI.

Understanding Bus Performance

First, it is important to outline the relevant performance criteria for instrument control buses to set a baseline for evaluation and comparison.

Bandwidth

When considering the technical merits of alternative buses, bandwidth and latency are two of the most important bus characteristics. Bandwidth measures the rate at which data is sent across the bus, typically in MB/s (106 bytes per second). A bus with high bandwidth is able to transmit more data in a given period than a bus with low bandwidth. Most users recognize the importance of bandwidth because it affects whether their data can be sent across the bus to or from a shared host processor as fast as it is acquired or generated and how much onboard memory their instruments need. Bandwidth is important in applications such as complex waveform generation and acquisition as well as RF and communications applications. High-speed data transfer is particularly important for virtual and synthetic instrumentation architectures. The functionality and personality of a virtual or synthetic instrument is defined by software; in most cases, this means data must be moved to a host PC for processing and analysis. Figure 1 charts the bandwidth (and latency) of all the instrumentation buses examined in this paper.

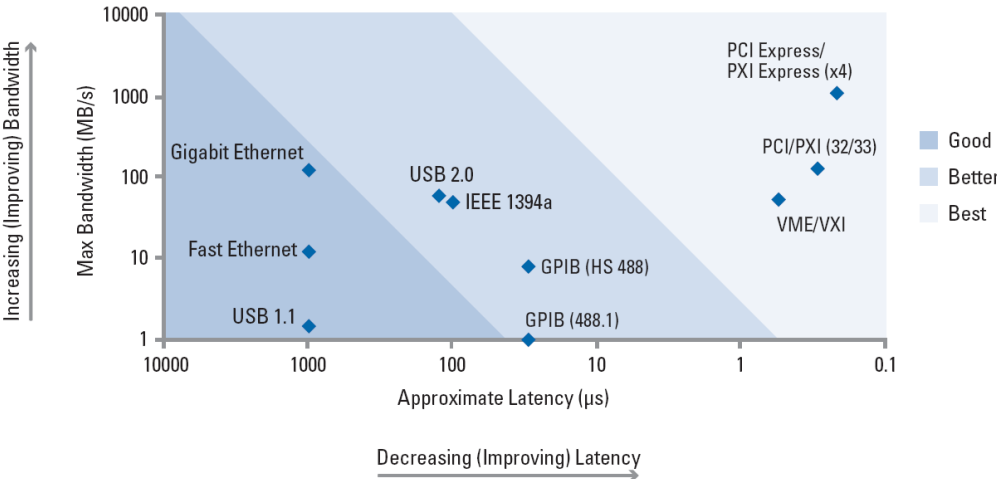


Figure 1. Bandwidth versus Latency for Instrumentation Buses

Latency

Latency measures the delay in data transmission across the bus. By analogy, if you compared an instrumentation bus to a highway, bandwidth corresponds to the number of lanes and the speed of travel, while latency corresponds to the delay introduced at the on and off ramps. A bus with low (meaning good) latency introduces less delay between the time data was transmitted on one end and processed on the other end. Latency, while less observable than bandwidth, has a direct impact on applications where a quick succession of short, choppy commands are sent across the bus, such as in handshaking between a digital multimeter (DMM) and switch, and in instrument configuration.

Message versus Register-Based Communication

Buses that use message-based communication are generally slower because this mode of communication adds overhead in the form of command interpretation and padding around the data. With register-based communication, data transfer occurs by directly writing and reading binary data to and from hardware registers on the device, resulting in a faster transfer. Register-based communication protocols are most common to internal PC buses, where interconnects are physically shorter and the highest throughput is required. Message-based communication protocols are useful for transmitting data over longer distances and where higher overhead costs are acceptable.

Long-Range Performance

For remote monitoring applications and for systems that involve measurement over a large geographical area, range becomes important. You can view performance in this category as a trade-off with latency because the error checking and message padding added to overcome the physical limitations of sending data over longer cables can add delays to sending and receiving the data.

Instrument Setup and Software Performance

Ease of use in terms of instrument setup and software performance is the most subjective criterion examined here. Nonetheless, it is important to discuss. Instrument setup describes the out-of-the-box experience and setup time. Software performance relates to how easily you can find interactive utilities or standard programming APIs such as VISA to communicate with and control the instrument.

Ruggedness of Connector

The physical connector for the bus affects whether it is suitable for industrial applications and whether additional effort is required to “ruggedize” the connection between the instrument and the system controller. Figure 2 presents photos of several instrumentation bus connectors.

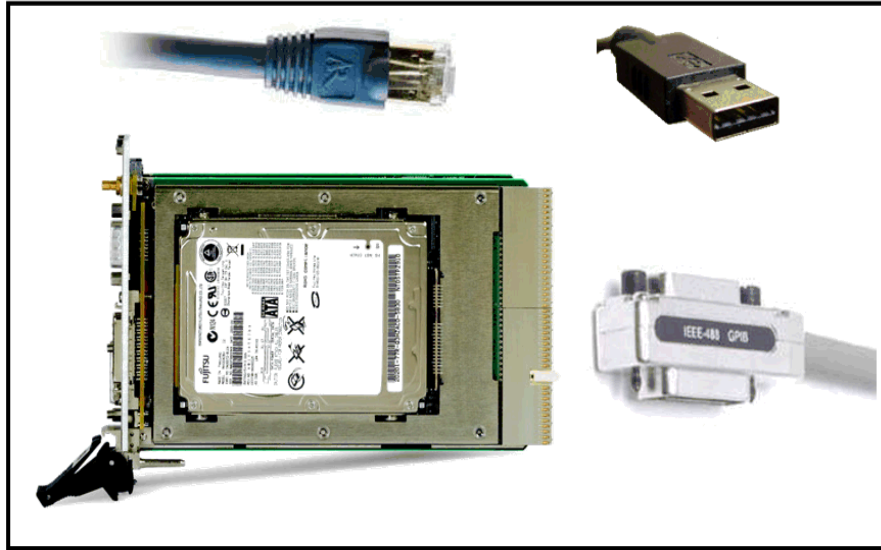


Figure 2. Connectors for Ethernet, USB, PXI, and GPIB (not to scale): The connector for PXI is an integrated part of the modular instrument on which it resides.

Instrument Control Bus Comparison (GPIB, USB, PCI, PCI Express, and Ethernet/LAN/LXI)

GPIB

First examine the IEEE 488 bus, familiarly known as GPIB (general-purpose interface bus). GPIB is a proven bus designed specifically for instrument control applications. GPIB has been a rugged, reliable communication bus for more than 30 years and is still the most popular choice for instrument control because of its low latency and acceptable bandwidth. It has the widest industry adoption, with a base of more than 10,000 instrument models featuring GPIB connectivity.

With a maximum bandwidth of about 1.8 MB/s, GPIB is best suited for communicating with and controlling stand-alone instruments. The more recent, high-speed revision, HS488, increased bandwidth up to 8 MB/s. Transfers are message-based, often in the form of ASCII characters. You can cable together multiple GPIB instruments to a total distance of 20 m, and bandwidth is shared among all instruments on the bus. Despite relatively lower bandwidth, GPIB latency is significantly lower (better) than that of USB and especially Ethernet. GPIB instruments do not autodetect or autoconfigure when connected to the system, though GPIB software is among the best available and the rugged cable and connector are suitable for the most demanding physical environments. GPIB is ideal for automating existing equipment or for systems requiring highly specialized instruments.

USB

USB (universal serial bus) has grown popular in recent years for connecting computer peripherals. That popularity has spilled over into test and measurement, with an increasing number of instrument vendors adding USB device controller capabilities to their instruments.

Hi-Speed USB has a maximum transfer rate of 60 MB/s, making it an attractive alternative for

instrument connectivity and control of stand-alone and some virtual instruments with data rates below 1 MS/s. Though most laptops, desktops, and servers may have several USB ports, those ports usually all connect to the same host controller, so the USB bandwidth is shared among all the ports. Latency for USB falls into the better category (between Ethernet at the slow end and PCI and PCI Express at the fast end), and cable length is limited to 5 m. USB devices benefit from autodetection, which means that unlike other technologies such as LAN or GPIB, USB devices are immediately recognized and configured by the PC when you connect them. USB connectors are the least rugged and least secure of the buses examined here. You may need external cable ties to keep them in place.

USB devices are well-suited for applications with portable measurements, laptop or desktop data logging, and in-vehicle data acquisition. The bus has become a popular communication choice for stand-alone instruments due to its ubiquity on PCs and especially due to its plug-and-play ease of use. The USB Test and Measurement Class (USBTMC) specification addresses the communication requirements of a broad range of test and measurement devices.

PCI

PCI and PCI Express feature the best bandwidth and latency specifications among all the instrumentation buses examined here. PCI bandwidth is 132 MB/s, with that bandwidth shared across all devices on the bus. PCI latency performance is outstanding; it is benchmarked at 700 ns, compared to 1 ms in Ethernet. PCI uses register-based communication, and, unlike the other buses mentioned here, it does not cable to external instruments. Instead, it is an internal PC bus used for PC plug-in cards and in modular instrumentation systems such as PXI, so distance measures do not directly apply. Nonetheless, you can “extend” the PCI bus up to 200 m by using NI fiber-optic MXI interfaces when connecting to a PXI system. Because the PCI connection is internal to the computer, it is probably fair to characterize the connector robustness as being constrained by the stability and ruggedness of the PC in which it resides. PXI modular instrumentation systems, which are built around PCI signaling, enhance this connectivity with a high-performance backplane connector and multiple screw terminals to keep connections in place. Once booted with PCI or PXI devices in place, Windows automatically detects and installs the drivers for modules.

An advantage that PCI and PCI Express share with Ethernet and USB is that they are universally available in PCs. PCI is one of the most widely adopted standards in the history of the PC industry. Today, every desktop PC has either PCI slots, PCI Express slots, or both. In general, PCI instruments can achieve lower costs because these instruments rely on the power source, processor, display, and memory of the PC that hosts them rather than incorporating that hardware in the instrument itself.

PCI Express

PCI Express is similar to PCI. It is the latest evolution of the PCI standard as Hi-Speed USB is to USB. Therefore, much of the above evaluation of PCI applies to PCI Express as well.

The main difference between PCI and PCI Express performance is that PCI Express is a higher-bandwidth bus that dedicates bandwidth to each device. Of all the buses covered in this tutorial, only PCI Express

offers dedicated bandwidth to each peripheral on the bus. GPIB, USB, and LAN divide bandwidth across the connected peripherals. Data is transmitted across point-to-point connections called lanes at 250 MB/s per direction for PCI Express Gen1. Each PCI Express link can be composed of multiple lanes, so the bandwidth of the PCI Express bus depends on how it is implemented in the slot and device. A x1 (by 1) Gen1 link provides 250 MB/s, a x4 Gen1 link provides 1 GB/s, and a x16 Gen1 link provides 4 GB/s dedicated bandwidth. It is important to note that PCI Express achieves software backward compatibility, meaning that if you are moving to the PCI Express standard, you can preserve your software investments in PCI. Also extensible by external cabling, PCI Express continues to evolve. In 2007, the PCI-SIG (PCI Special Interest Group) introduced PCI Express Gen2, which doubles the bandwidth of PCI Express Gen1.

High-speed, internal PC buses were designed for rapid communication. Consequently PCI and PCI Express are ideal bus choices for high-performance, data-intensive systems where large bandwidth is required, and for integrating and synchronizing several types of instruments.

Ethernet/LAN/LXI

Ethernet has long been an instrument control option. It is a mature bus technology that has been widely used in many application areas outside of test and measurement. 100BASE-T Ethernet has a theoretical maximum bandwidth of 125 MB/s. Gigabit Ethernet, or 1000BASE-T, increases the maximum bandwidth to 125 MB/s. In all cases, Ethernet bandwidth is shared across the network. At 125 MB/s, Gigabit Ethernet is theoretically faster than Hi-Speed USB, but this performance quickly declines when multiple instruments and other devices are sharing network bandwidth. Communication along the bus is message-based, with communication packets adding significant overhead to data transmission. For this reason, Ethernet has the worst latency of the bus technologies featured in this tutorial.

Nonetheless, Ethernet remains a powerful option for creating a network of distributed systems. It can operate at distances of up to 85 to 100 m without repeaters and, with repeaters, it has no distance limits. No other bus has this range of separation from the controlling PC or platform. As with GPIB, autoconfiguration is not available on Ethernet/LAN. You must manually assign an IP address and subnet configuration to your instrument. Like USB and PCI, Ethernet/LAN connections are ubiquitous in modern PCs. This makes Ethernet ideal for distributed systems and remote monitoring. It is often used with other bus and platform technologies to connect measurement system nodes. These local nodes may themselves be composed of measurement systems relying on GPIB, USB, and PCI. Physical Ethernet connections are more robust than USB connections but less so than GPIB or PXI.

LXI (LAN eXtensions for Instrumentation) is an emerging LAN-based standard. The LXI standard defines a specification for stand-alone instruments with Ethernet connectivity that adds triggering and synchronization features.

Conclusion: Instrument Bus Performance

Despite the conceptual convenience of designating a single bus or communication standard as the “ultimate” or “ideal” technology, history shows that several standards are likely to continue to coexist

because each bus technology has unique strengths and weaknesses. A compilation of the performance criteria from the previous section, Table 1 shows that no single bus is superior across all measures of performance.

	Bandwidth (MB/s)	Latency (μs)	Range (m) (without extenders)	Setup and Installation	Connector Ruggedness
GPIB	1.8 (488.1) 8 (HS488)	30	20	Good	Best
USB	60 (Hi-Speed)	1000 (USB) 125 (Hi-Speed)	5	Best	Good
PCI	132	0.7	Internal PC Bus	Better	Better Best (for PXI)
PCI Express	250 (x1) 4000 (x16)	0.7 (x1) 0.7 (x4)	Internal PC Bus	Better	Better Best (for PXI)

Table 1. Bus Performance Comparison

You can exploit the strengths of several buses and platforms by creating hybrid systems. Hybrid test and measurement systems combine components from modular instrumentation platforms such as PXI and VXI and stand-alone instruments that connect across GPIB, USB, and Ethernet/LAN. One key to creating and maintaining a hybrid system is implementing a system architecture that transparently recognizes multiple bus technologies and takes advantage of an open, multivendor computing platform, such as PXI, to achieve I/O connectivity.

The other key to successfully developing a hybrid system is ensuring that the software you choose at the driver, application, and test system management levels is modular. Though some vendors may offer vertical software solutions for specific instruments, the most useful system architecture breaks up the software functions into interchangeable modular layers so that your system is tied neither to a particular piece of hardware nor to a particular vendor. This layered approach provides the best code reuse, modularity, and longevity. For example, VISA (Virtual Instrument Software Architecture) is a vendor-neutral software standard for configuring, programming, and troubleshooting instrumentation systems comprising GPIB, VXI, serial (RS232/485), Ethernet, USB, and/or IEEE 1394 interfaces. It is a useful tool because the API for programming VISA functions is similar for a variety of communication interfaces.

With hybrid systems, you can combine the strengths of many types of instruments, including legacy equipment and specialized devices. Despite the appeal of finding a one-size-fits-all solution for instrumentation, reality requires that you fit the instruments and associated bus technologies to your specific application needs.

Chapter 4: Modular Instruments Basics

Modular instrument is the term given to the modular hardware that fits into the virtual instrumentation architecture, as illustrated by Figure 1.

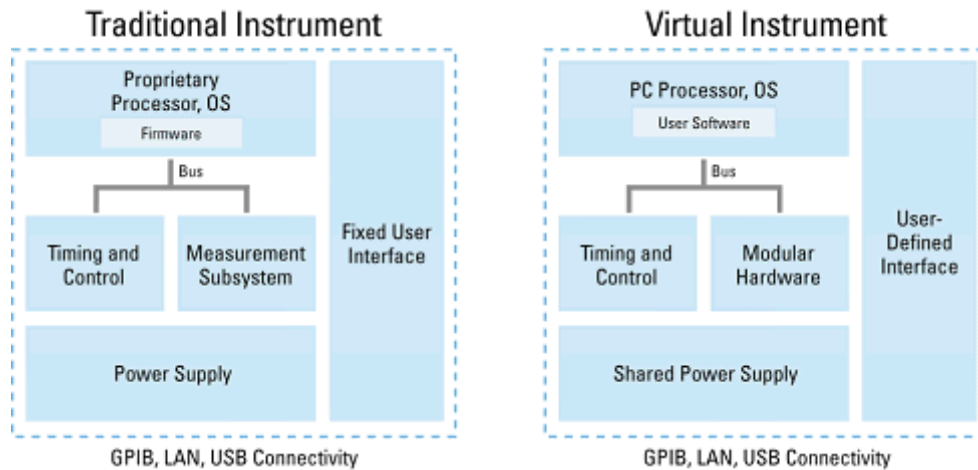


Figure 1. Comparing Traditional and Virtual Instrumentation Architectures.

To understand how modular instruments work, you need to know the similarities and differences between the two approaches shown in Figure 1.

As you can see, both approaches have measurement hardware, a chassis, a power supply, a bus, a processor, an OS, and a user interface. Because the approaches use the same basic components, the most obvious difference from a purely hardware standpoint is how the components are packaged. A traditional, or stand-alone, instrument puts all of the components in the same box for every discrete instrument. By contrast, in a well-designed modular instrumentation system, many of the components – such as the bus, power supply, OS, and user interface – are shared across instrument modules instead of duplicating these components for every instrument function. These instrument modules can also include different types of hardware, such as oscilloscopes, function generators, digital, and RF.

While you can design modular instruments for a variety of platforms, this tutorial examines various specifications with respect to modular instruments designed for the PCI eXtensions for Instrumentation (PXI) platform – a rugged platform for test, measurement, and control supported by more than 70 member companies.

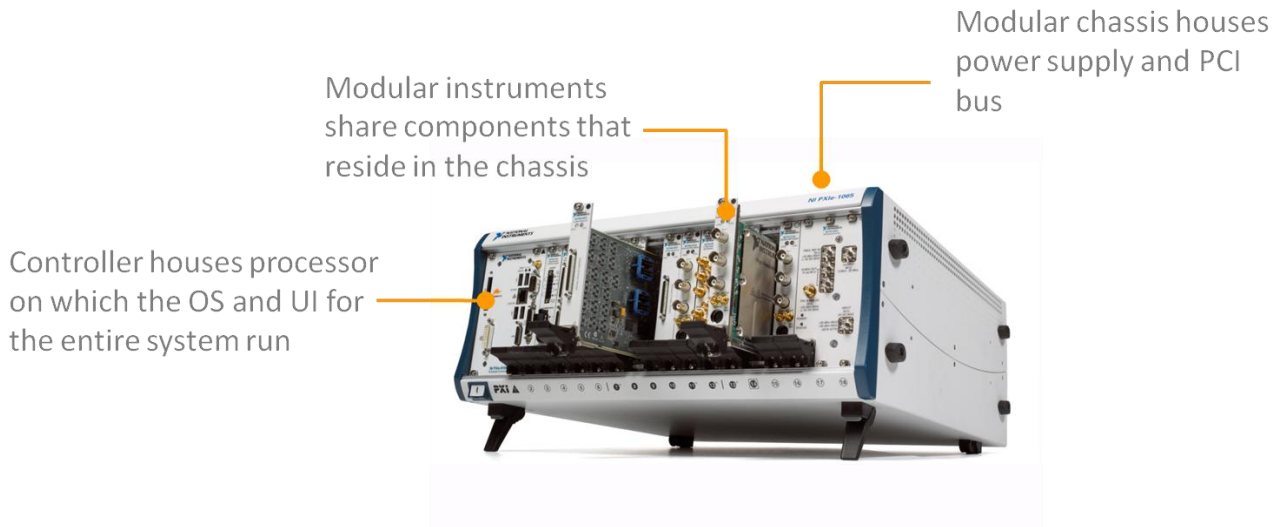


Figure 2. The PXI platform supports modular instrumentation.

Anatomy of a Modular Instrument

To choose the best possible set of instruments for your application, it is important for you to understand the various components that make up a modular instrument and the impact that these components have on fundamental instrument specifications such as bandwidth, resolution, accuracy, and sampling rate.

This tutorial describes the anatomy of a modular instrument with inputs and provides insight into how the components of such an instrument can impact various specifications.

Modular instruments typically consist of four main components – analog inputs, analog front end, analog-to-digital converter (or digital-to-analog converter in the case of instruments that have outputs), and chassis connection.

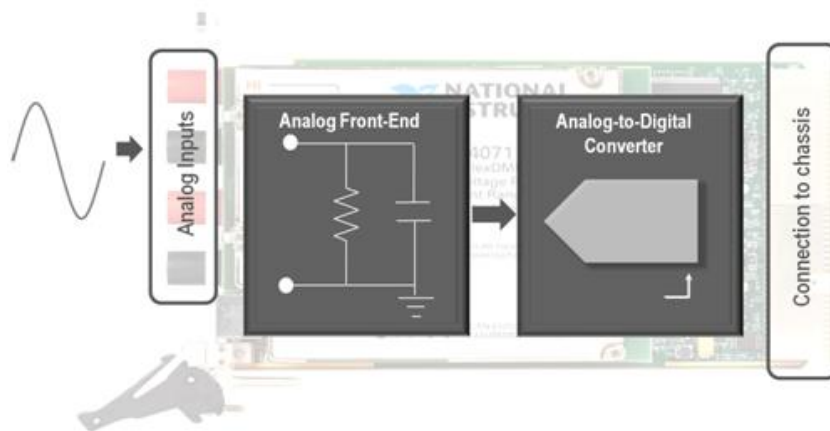


Figure 3. Anatomy of a Typical PXI Modular Instrument.

Analog Inputs: All modular instruments have input connectors. This is where the analog signal enters the device. The type of connector, however, varies from instrument to instrument. DMMs for example use banana connectors because they provide a solid connection with the input signal and, thereby, reduce noise and ensure accurate DC measurements. On the other hand, many digitizers use BNC connectors due to their higher bandwidth.



Figure 4. DMM connectors are optimized for accuracy. Digitizer connectors are optimized for bandwidth.

Connection to chassis: Modular instruments made for the PXI platform also have a PXI or PXI Express connector at the back. This component connects the instrument to the chassis backplane and enables synchronization between multiple instruments in a chassis using the trigger lines on the backplane.

Analog front end: All instruments also feature an analog front end. This is essentially the analog circuitry that is required to condition the input signal. The analog front end is typically optimized either for bandwidth or accuracy. The front end of a DMM, for example, has an oven-stabilized onboard reference that is used to calibrate the DMM before every measurement to ensure maximum accuracy. The front end is also optimized to reduce noise.

On the other hand, the analog front end of higher-speed instruments such as digitizers and arbitrary waveform generators is designed using components that are impedance-matched to reduce attenuation at higher bandwidth values.

Typically, there is a trade-off between designing an instrument's front end for accuracy and bandwidth. Thus, instruments with front ends that have high bandwidth typically have low accuracy. The same is true for the opposite case.

Analog-to-digital converter (ADC): After the input signal passes through the front-end circuitry, it is converted into digital values that can be read by the computer using an analog-to-digital converter (ADC). The instrument's ADC determines how fast it can sample an input signal. It can also help determine the instrument's resolution.

Note: In the case of instruments that output a signal, the ADC is replaced with a digital-to-analog converter (DAC).

Effects of Front End

The previous section of this paper discussed how the analog front end of an instrument can impact its bandwidth and accuracy. This section defines the terms bandwidth and accuracy and explains how these specifications can impact your signal measurements.

Bandwidth

The bandwidth of an instrument is more clearly defined as the frequency at which a sinusoidal signal is attenuated by the analog front end to 70 percent of its original amplitude. This is also commonly known as the 3 dB point. Bandwidth of an instrument depends on the bandwidth of its front end, which takes the form of an RC circuit and acts as a lowpass filter.

For example, Figure 5 shows a 100 MHz, 1 V_{pk-pk} sine wave passing through the front end of an instrument with a bandwidth of 100 MHz. The signal that emerges from the front end is a 0.7 V_{pk-pk} sine wave, which is essentially the original signal attenuated by 30 percent.

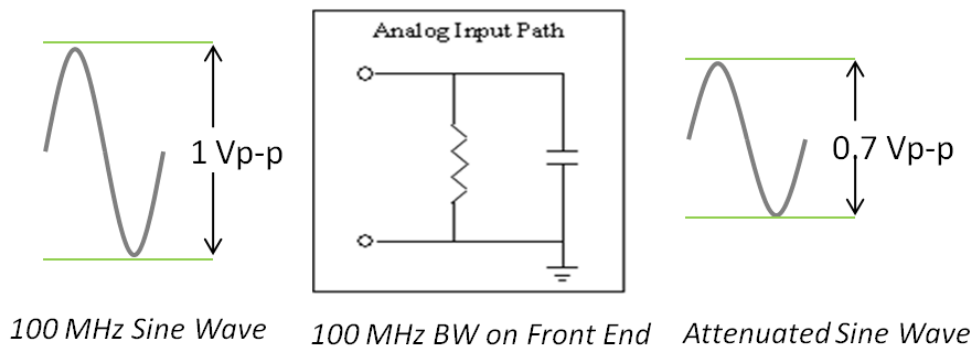


Figure 5. 100 MHz Sine Wave Passing through an Instrument with a Bandwidth of 100 MHz.

Accuracy

There is always some uncertainty in any measurement made by an instrument. Accuracy is a measure of this uncertainty. To better understand accuracy, take a look at the example of the Omega chronometer watch series. These watches feature a mechanical movement with an average daily variation range between -4 and +6 seconds per day or 69.4 parts per million (ppm). This is an accuracy of 99.99 percent, the highest accuracy attainable by a mechanical movement.

Just like watches, instruments have a certain degree of accuracy. Some of the most accurate instruments available on the market are DMMs. The front end of some DMMs can offer less than 50 ppm of accuracy. This means some DMMs can offer an uncertainty of less than 50 μ V on a 1 V input signal. Accuracy is usually represented as an offset on your measurement.

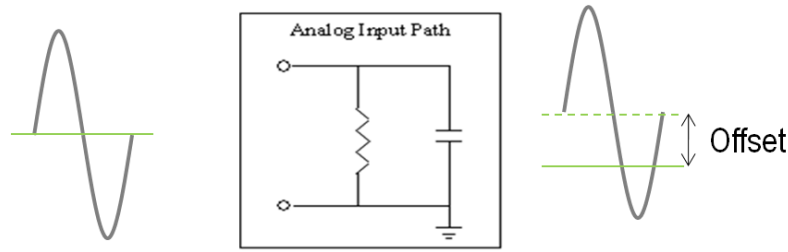


Figure 6. Accuracy is represented as an offset on your measurement.

Different instruments offer different accuracy for different measurement ranges and conditions. It is therefore very important to carefully evaluate the specifications listed in the datasheet to determine whether a particular product suits your measurement needs.

For example, assume that you are trying to measure a 7 V signal with the NI PXI-4070 6½-digit DMM 90 days after it has been calibrated and at an ambient temperature that is within $\pm 5^\circ$ of the temperature at which the DMM was calibrated. Based on this information, you can calculate the accuracy of the DMM using Table 1, which is included in the specifications sheet of the product.

DC Voltage \pm (ppm* of reading + ppm of range)

Range	Resolution	Input Resistance	24 Hr [†] $T_{cal} \pm 1^\circ\text{C}$	90 Day [‡] $T_{cal} \pm 5^\circ\text{C}$	2 Year [‡] $T_{cal} \pm 5^\circ\text{C}$	Tempco ^{°C} (0 °C to 55 °C)	
						Without Self-Cal	With Self-Cal
100 mV**	100 nV	>10 G Ω , 10 M Ω	10 + 10	30 + 20	40 + 20	4 + 5	0.3 + 0.3
1 V	1 μV	>10 G Ω , 10 M Ω	6 + 2	20 + 6	25 + 6	2 + 1	0.3 + 0.3
10 V	10 μV	>10 G Ω , 10 M Ω	4 + 2	20 + 6	25 + 6	1 + 1	0.3 + 0.3
100 V	100 μV	10 M Ω	6 + 2	30 + 6	35 + 6	4 + 1	0.3 + 0.3
300 V	1 mV	10 M Ω	6 + 6	30 + 20	35 + 20	4 + 3	0.3 + 0.3

* 1 ppm (part per million) = 0.0001%.
[†] Relative to external calibration source.
[‡] Using internal self-calibration; specifications valid over the entire operating temperature range.
 ** With offset nulling and 100 ms aperture.
 T_{cal} = temperature at which last self-calibration or external calibration was performed.
 Tempco = temperature coefficient.

Table 1. DC Voltage Accuracy of the PXI-4070 6½-Digit DMM

From the table, you can determine that the accuracy specifications for these conditions is $\pm(20$ ppm of reading + 6 ppm of range). Thus:

$$\text{Accuracy} = \pm(\text{ppm of reading} + \text{ppm of range})$$

$$\text{Accuracy} = \pm(20 \text{ ppm of } 7 \text{ V} + 6 \text{ ppm of } 10 \text{ V})$$

$$\text{Accuracy} = \pm \left(7 V \times \frac{20}{1000000} + 10 V \times \frac{6}{1000000} \right) = 200 \mu V$$

Therefore, the reading should be within 200 μV of the actual input voltage.

Effects of ADC/DAC

This section examines in greater detail the impact that sampling rate and resolution – specifications affected by the ADC of an instrument – can have on your measurements.

Sampling Rate

Sampling rate is the maximum speed at which the ADC of an instrument can convert the input signal into digital values that represent the voltage level. One consideration to make when evaluating the sampling rate of an instrument is the Nyquist Theorem, which states that a signal must be sampled at a rate greater than twice the highest frequency component of interest in the signal to capture the highest frequency component of interest. Even at a rate of 2X, the signal appears significantly distorted, as shown Figure 6, where a 100 MHz sine wave is being acquired by an ADC that can sample at only 200 MS/s or 200 MHz.

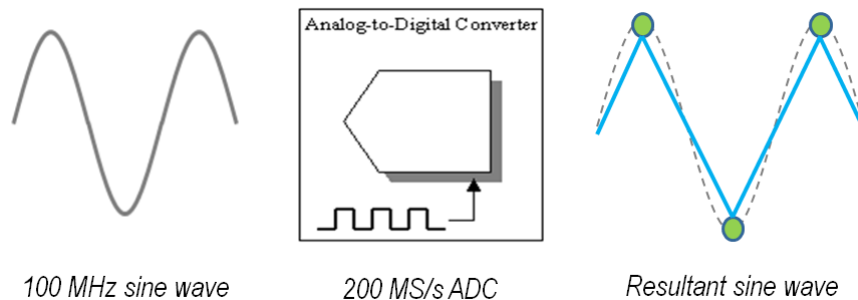


Figure 6. The 100 MHz sine wave sampled by 200 MS/s appears distorted.

To obtain an accurate representation of the signal, the instrument must have a sample rate at least 5X the speed of the signal.

Resolution

Resolution is another instrument specification that is directly impacted by the ADC. It is defined as the smallest amount of input signal *change* that an instrument or sensor can detect reliably. Typically, in analog instruments such as digitizers, arbitrary waveform generators (arbs), and dynamic signal analyzers (DSAs), resolution is represented in bits. But in the case of precision DC instruments, such as DMMs, it is represented in digits.

Bits

Expressed in bits, the resolution of analog instruments (digitizers, arbs, DSAs, and so on), is a characteristic that is directly tied to the ADC (or DAC in the case of arbs) used in the instrument. An 18-bit digitizer, for example, uses an 18-bit ADC. In analog instruments, you can calculate the smallest

possible change in the input signal that can be detected, also known as the least-significant bit (LSB), using the following formula:

$$\text{Least Significant Bit (LSB)} = \frac{\text{Input Range}}{2^{\text{number of bits on ADC/DAC}}}$$

For example, Figure 7 shows a 10 V_{pk-pk} sine wave passing through a three-bit ADC. This ADC can use a total of 2³ different discrete values to represent any voltage value that it converts. Thus, for an input signal with a range of 10 V, the smallest possible voltage detectable by the ADC or LSB can be calculated as 10/2³ or 1.25 V.

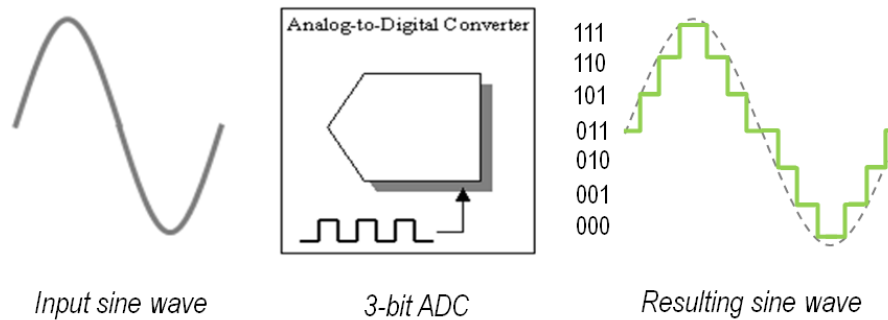


Figure 7. Sine Wave Passing through a Three-Bit ADC.

If the same signal were to pass through an ADC with a resolution of eight bits, then the smallest possible voltage that is detectable by the ADC is 10/2⁸ = 39 mV, and for a 24-bit ADC the value would be 10/2²⁴ = 596 nV.

Digits

The resolution of precision DC instruments such as DMMs is represented in digits. The number of digits is used to specify the number of meaningful counts, or unique digitized values, the DMM is capable of producing. For a traditional benchtop DMM, the number of digits indicates how many decimal places the DMM can display on its digital readout. The number of digits is often specified as the number of full digits, digits that can take any value from 0 to 9, and a single overrange digit, referred to as the ½ digit. The overrange digit is limited to only specific values, generally 0 or 1. For example, a 6½-digit DMM has seven decimal places on its readout. The most significant digit on the display can take on a value of 0 or 1; the rest can range from 0 to 9.

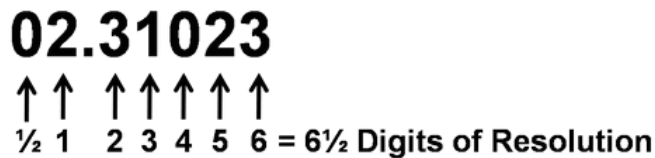


Figure 8. Example Reading Obtained from a 6½-Digit DMM.

There is a loose relationship between digits and bits. For example, for a noise-free DMM that uses a 12-bit ADC to digitize signals, you can calculate the digits of resolution using the following formula:

$$\text{Digits of Resolution} = \log_{10}(\text{Number of LSB}) = \log_{10}(2^{12}) = 3.61 \text{ digits}$$

In reality, however, a noise-free DMM does not exist. Noise may reduce the number of LSBs, therefore reducing the number of digits. As a result, you need to account for the noise level when calculating the effective number of digits (ENOD) for a DMM. You can do this using the following formula:

$$\text{ENOD} = \log_{10}\left(\frac{\text{Total Span}}{\sqrt{12} \times \text{RMS Noise}}\right)$$

For example, if a DMM set on the ± 10 V range (20 V total span) shows readings with an rms noise level of 70 μ V, you can calculate its effective absolute units of resolution and the ENOD using the following formula:

$$\text{ENOD} = \log_{10}\left(\frac{20 \text{ V}}{\sqrt{12} \times 70 \mu\text{V}}\right) = 4.92 \text{ digits}$$

Thus, this DMM can be called a five-digit DMM. You can calculate the minimum number of counts (equivalent to LSBs) required for building a five-digit DMM using the following formula:

$$\text{Count} = \frac{\text{Total Span}}{\sqrt{12} \times \text{RMS Noise}} = \frac{20 \text{ V}}{\sqrt{12} \times 70 \mu\text{V}} = 82478$$

The minimum number of bits needed is, therefore, 17 ($2^{16} = 65,536$, $2^{17} = 131,072$).

Table 2 relates bits, counts, and ENOD to conventional digits of resolution for DMMs. As evidenced by the table, bits, counts, and ENOD are deterministically related. A direct mathematical relationship between ENOD and digits does not exist because digits are used only as an approximation.

Digits	3 ½	4	4 ½	5	5 ½	6	6 ½	7
ENOD	3.01	3.61	4.21	4.81	5.42	6.02	6.62	7.22
Counts	1,024	4,096	16,384	65,536	262,144	1,048,576	4,194,304	16,777,216
Bits	10	12	14	16	18	20	22	24

Table 2. Relating Bits, Counts, and ENOD to Conventional Digits of Resolution.

Instrument Types

This section of the tutorial describes the five main instrument types – DMMs, digitizers, arbitrary waveform generators, RF analyzers and generators, and dynamic signal acquisition boards – and explains for which specifications each type is optimized.

Digital Multimeters (DMMs)

A DMM is similar to a digitizer or data acquisition hardware with some important differences. Unlike the front end of digitizers, the front end of DMMs is designed to lower noise and increase accuracy.

Additionally, DMMs provide accuracy through averaging. This often slows down their speed. Figure 1 illustrates basic DMM functionality. Like a digitizer, a DMM has a fast sample clock that sets the rate at which samples are acquired. However, unlike a digitizer, a DMM does not return every analog-to-digital conversion. Rather, multiple samples are buffered and then averaged to return a single measurement.

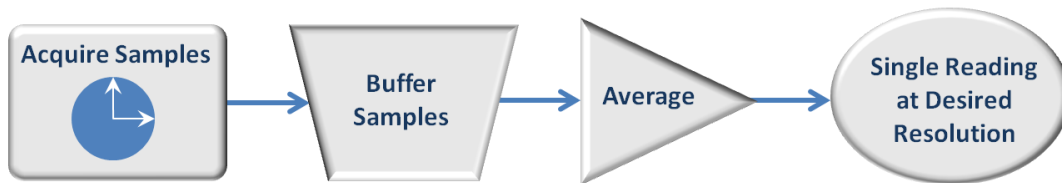


Figure 9. DMMs average various samples to obtain a single reading.

The number of samples collected for averaging is determined by the aperture time. By increasing the aperture time of a DMM, you can effectively increase the number of samples being averaged per measurement and thus reduce noise and increase resolution.

Through averaging, DMMs offer higher resolution. However, averaging also reduces the speed of DMMs. Thus DMMs are designed for high-precision but not high-speed applications.

Arbitrary Waveform Generators (Arbs) and Digitizers

Arbs and digitizers are different in that arbs are used for signal generation while digitizers are used for measurements. However, because both instruments are designed to work with high-speed signals, they share many of the same design concerns. For starters, when designing the front end for an arb or digitizer, engineers pay special attention to the characteristic impedance of the components used. By matching impedances of all components used in the signal path, engineers are able to design high-bandwidth front ends.

Additionally, both arbs and digitizers use ADCs/DACs that are high-speed rather than high-resolution. Typically, these devices have sampling rates in the GS/s range generally have resolution of between eight and 12 bits. Thus arbs and digitizers are optimized for high-speed applications but not high-precision applications.

Dynamic Signal Analyzers (DSAs)

DSA devices are specifically designed for sound and vibration applications. These devices use high-resolution rather than high-speed ADCs. Even so, they are different from DMMs. While DMMs are

specifically optimized for measuring DC voltages, currents, and resistances, DSA devices are optimized for measuring AC signals.

RF Analyzers and Generators

RF instruments are different from all other instruments. They consist of an ADC and an upconverter or a downconverter, and they have components that are impedance-matched to ensure minimum losses and attenuation at high frequencies. RF instruments have not been addressed in this tutorial due to their complexity. To learn more about RF instruments and how they work, visit ni.com/rf.

Conclusion

Bandwidth, accuracy, resolution, and sampling rate are often the first factors you must consider when selecting modular instruments because instruments that are optimized for accuracy and resolution typically have lower sampling rates and bandwidths. Figure 10, which uses two axes – resolution and sampling rate – to illustrate how various NI instrument types stack up against each other, is a good place to start when short-listing instruments for your application.

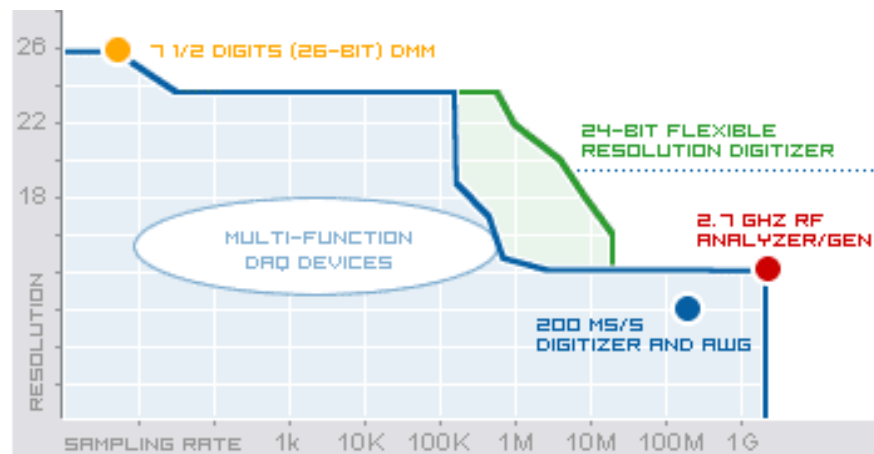


Figure 10. Resolution versus Sampling Rate Chart for NI Modular Instruments.

However, it is important to note that choosing instruments is often far more complex than simply evaluating the four specifications outlined in this document. You must also consider several other instrument specifications, such as noise, power, isolation, and dynamic range. Additionally, various instruments offer specific features that can prove to be extremely valuable for specific applications. The NI PXI-4071 7½-digit DMM, for example, offers a feature known as offset compensation that eliminates parasitic voltage and resistance values from a circuit to obtain highly accurate low-level resistance measurements. Other instruments such as the NI PXI-5154 feature equivalent time sampling, which increases the apparent sampling rate of the device by creating a picture of a waveform using a series of samples taken from repetitive waveforms.

Though this tutorial can provide a starting point, choosing the right modular instrument for your application involves thoroughly evaluating all the specifications and features of various products.

Chapter 5: PXI Modular Instrumentation Platform

PXI (PCI eXtensions for Instrumentation) is a rugged PC-based platform for measurement and automation systems. PXI combines PCI electrical-bus features with the rugged, modular, Eurocard packaging of CompactPCI, and then adds specialized synchronization buses and key software features. PXI is both a high-performance and low-cost deployment platform for measurement and automation systems. These systems serve applications such as manufacturing test, military and aerospace, machine monitoring, automotive, and industrial test.

Developed in 1997 and launched in 1998, PXI was introduced as an open industry standard to meet the increasing demands of complex instrumentation systems. Today, PXI is governed by the PXI Systems Alliance (PXISA), a group of more than 60 companies chartered to promote the PXI standard, ensure interoperability, and maintain the PXI specification. For more information on the PXISA, including the PXI specification, refer to the PXISA Web site at pxisa.org.

Hardware Architecture



Figure 1. A PXI system features three basic components: chassis, system controller, and peripheral modules.

PXI Chassis

[PXI chassis](#) provide the rugged and modular packaging for systems. Chassis available in both 3U and 6U form factors, generally ranging in size from four to 18 slots, are available with special features such as DC power supplies and integrated signal conditioning. The chassis contains the high-performance PXI backplane, which includes the PCI bus and timing and triggering buses. Using these timing and triggering buses, you can develop systems for applications requiring precise synchronization.

PXI Controllers

As defined by the PXI Hardware Specification, all PXI chassis contain a system controller slot located in the leftmost slot of the chassis (slot 1). Controller options include remote controllers from a desktop, workstation, server, or laptop computer as well as high-performance embedded controllers with either a Microsoft OS (Windows XP/2000) or a real-time OS (LabVIEW Real-Time).

PXI Embedded Controllers

Embedded controllers eliminate the need for an external PC, therefore providing a complete system contained in the PXI chassis. PXI embedded controllers are typically built using standard PC components in a small, PXI package. For example, the NI PXI-8110 controller has the Core 2 Quad Q9100 2.26 GHz processor, with up to 2 GB of DDR2 RAM, a hard drive, and standard PC peripherals such as ExpressCard, USB 2.0, Ethernet, serial, parallel, and GPIB ports. There are two types of PXI embedded controllers:

- PXI embedded controllers with Windows
- PXI embedded real-time controllers

PXI Embedded Controllers with Windows: [PXI embedded controllers with Windows](#) come with standard PC features such as integrated CPU, hard-drive, RAM, Ethernet, video, keyboard/mouse, serial, USB, and other peripherals, as well as Microsoft Windows and all device drivers already installed. Because the controllers have Microsoft Windows, the user experience is no different than a PC or laptop computer. It has application software similar to that available on your PC or laptop computer such as Microsoft Office Word, Excel, and PowerPoint.

PXI Embedded Real-Time Controllers: [PXI embedded real-time controllers](#) also come with standard PC features along with a real-time OS such as LabVIEW Real-Time or VxWorks to deliver real-time, deterministic, and reliable I/O for measurement, automation, and control. Because RT Series PXI controllers are configured and programmed over the Ethernet, you can distribute a real-time application across the network and remotely monitor it. These controllers are designed for applications requiring deterministic and reliable performance and can be run under headless operation (in other words, no keyboard, mouse, or monitor).



Figure 2. NI PXI-8110 Core 2 Quad Q9100 2.26 GHz Processor Embedded Controller: Notice the familiar PC peripherals such as keyboard/mouse and monitor connections as well as the hard drive, USB 2.0, Ethernet, serial, ExpressCard, and other standard PC peripherals.

Embedded controllers are ideal for portable systems and contained “single box” applications where the chassis is moved from one location to the next. For more information, refer to [PXI controllers](#).

PXI Remote Controllers

There are two types of PXI remote controllers:

- Laptop control of PXI
- PC control of PXI

Laptop Control of PXI: With ExpressCard MXI (Measurement eXtensions for Instrumentation) and PCMCIA CardBus interface kits, you can control PXI systems directly from laptop computers. During boot-up, the laptop computer recognizes all peripheral modules in the PXI system as PCI boards. Using ExpressCard MXI, you can control your PXI system with a sustained throughput of up to 214 MB/s.



Figure 3. Laptop Control of PXI – ExpressCard MXI Interface Kit.



Figure 4. Laptop Control of PXI – PCMCIA CardBus Interface Kit.

You now have the advantage of mobile PXI systems for applications such as field tests; in-vehicle data logging; noise, vibration, and harshness testing; and nondestructive testing with laptop control of PXI. You can purchase any [ExpressCard MXI compatible laptop](#) or [PCMCIA CardBus compatible laptop](#) to remotely control your PXI system.

PC Control of PXI: With MXI-Express and MXI-4 interface kits, you can control PXI systems directly from your desktop, workstation, or server computers. During boot-up, the computer recognizes all peripheral modules in the PXI system as PCI boards.

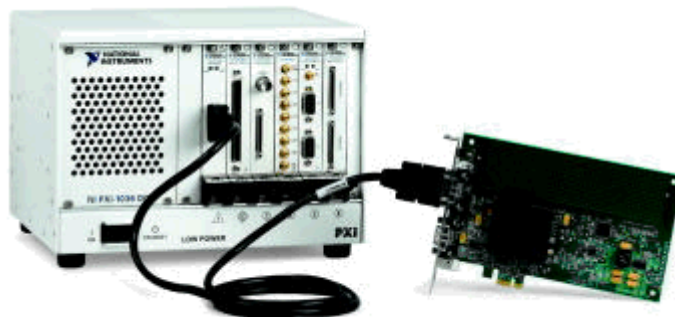


Figure 5. Remote control with two-port MXI-Express provides simultaneous control of two PXI chassis with combined throughput of 160 MB/s.

Using MXI-Express, you can control your PXI system with a sustained throughput of up to 832 MB/s. With the MXI-Express two-port interface kit, you can control two PXI systems simultaneously from a single PC.



Figure 6. Remote control with MXI-4 provides PC control of PXI as well as multichassis PXI systems.

With PXI remote controllers, you can maximize processor performance with minimized cost by using a desktop computer or laptop to remotely control a PXI system. Because all remote control products are software-transparent, no additional programming is required.

PXI Peripheral Modules

National Instruments offers more than 300 different PXI modules. Because PXI is an open industry standard, nearly 1,500 products are available from more than 70 different instrument vendors.

- | | |
|---------------------------------|---------------------------|
| Analog input and output | Boundary scan |
| Bus interface and communication | Carrier products |
| Digital input and output | Digital signal processing |
| Functional test and diagnostics | Image acquisition |
| Prototyping boards | Instruments |
| Motion control | Power supplies |
| Receiver interconnect devices | Switching |
| Timing input and output | RF and communications |

Because PXI is directly compatible with CompactPCI, you can use any 3U CompactPCI module in a PXI system. A categorized list of modules offered by National Instruments and its PXI product partners is available at ni.com/pxi.

PXI also preserves investments in stand-alone instruments or VXI systems by providing standard hardware and software for communication to these systems. For example, interfacing a PXI system to GPIB-based instrumentation is no different with a PXI-GPIB module than it is with a PCI-GPIB module. The software is identical. Additionally, a number of methods are available to build hybrid systems interfacing PXI, USB, LAN/LXI, VXI, and stand-alone instruments.

Software Architecture

Because PXI hardware is based on standard PC technologies, such as the PCI bus, and standard CPUs and peripherals, the standard Windows software architecture is familiar to users as well. Development and operation of Windows-based PXI systems is no different from that of a standard Windows-based PC. Additionally, because the PXI backplane uses the industry-standard PCI/PCI Express bus, writing software to communicate with PXI devices is, in most cases, identical to that of PCI devices. For example, software to communicate to an NI PXI-6251 multifunction data acquisition module is identical to that of an NI PCI-6251 board in a PC. Therefore, existing application software, example code, and programming techniques do not have to be rewritten when moving software between PC-based and PXI-based systems.



Figure 7. Two Different Packages – One Software Standard: In software, communication with a PXI module (bottom) is identical to that with a PCI board (top).

As an alternative to Windows-based systems, you can use a real-time software architecture for time-critical applications requiring deterministic loop rates and headless operation (no keyboard, mouse, or monitor). Additional information on using LabVIEW Real-Time with PXI systems is available at ni.com/realtime.

PXI – Industry Standard for Modular Instrumentation

PXI, based on PCI and next-generation PCI Express, is the fastest-growing test and measurement standard since GPIB. PXI best meets modular instrumentation demands now and in the future, with more than 60 vendors in the PXI Systems Alliance and more than 1,500 products available today. Primarily, all instruments in a PXI system share the same power supply, chassis, and controller. Alternative approaches duplicate the power supply, chassis, and/or controller for every instrument, adding cost and size and decreasing reliability. With PXI, the controller can be a high-performance slot 0

embedded controller, desktop PC, laptop, or server-class machine. When you require faster processing, you can easily upgrade the controller of a PXI system. To reuse existing equipment, you can use PXI to control USB, GPIB, LAN/LXI, serial, and VXI instruments.

Modular instruments require a high-bandwidth, low-latency bus to connect instrument modules to the shared processor for performing user-defined measurements. PXI meets these needs with bandwidth up to 2 GB/s for each slot as specified by the PCI Express Gen1 specification. Future generations of PCI Express will increase this further. Actual per-slot bandwidth will vary by manufacturer depending on whether a manufacturer uses x4 or x8 PCI Express interfaces and whether these interfaces are Gen 1, 2, or 3.

Take a modular RF acquisition system, for example. PXI can stream two channels of 100 MS/s, 16-bit IF data directly to a processor for computation. Neither LAN nor USB can meet these requirements, so these instruments always include an embedded, vendor-defined processor. Hence high-bandwidth standards such as PXI provide a truly software-defined approach required for modular instrumentation.

Why Customers Choose PXI

Higher Throughput

Every application is unique and has very specific needs. However, bandwidth and latency are two important attributes of a platform for many applications. Latency tends to dominate single-point operations, such as digital multimeter/switch scanning, and bandwidth tends to dominate data streaming applications, such as waveform stimulus/response. PXI provides high speed for a wide range of applications with both high bandwidth and low latency via the PCI/PCI Express bus.

Timing and Synchronization

Many measurement and automation applications require advanced timing and synchronization capabilities that you cannot implement directly across PC standard I/O buses such as PCI/PCI Express, Ethernet/LAN, USB, and so on. PXI offers advanced timing and synchronization features to meet your application needs:

- 100 MHz differential system reference clock
- 10 MHz reference clock signal
- Differential star trigger
- Star trigger bus with matched-length trigger traces to minimize intermodule delay and skew
- Trigger bus to send and receive high-speed timing and triggering signals
- Differential signals for multichassis synchronization
- Support for industry standards including GPS, 1588, and IRIG-B

System Reliability

The PXI specification defines requirements that make PXI systems well-suited for harsh environments. PXI features the high-performance IEC (International Electrotechnical Commission) connectors and rugged Eurocard packaging system used by CompactPCI. The PXI specification also defines specific cooling and environmental requirements to ensure operation in industrial environments. Modularity

makes it easy to configure, reconfigure, and repair your PXI systems, resulting in very low mean time to repair (MTTR). Because PXI is modular, you can update individual modules and components without replacing the entire system.

Lower System Costs

Because PXI is a PC-based platform, it delivers high-precision instrumentation, synchronization, and timing features at an affordable price. The low cost of PC components is only the beginning of the savings you gain from using PXI. With PXI, you use the same OS and application software like Microsoft Excel and Microsoft Word in your office and on the production floor. The familiarity of the software eliminates training costs and the need to retrain personnel every time you implement a new system. Because the foundation of PXI is PC technology, you benefit from low component costs, familiar software, and system reuse.

Extension of the PXI Platform: PXI Express

PXI Express technology is the latest addition to the PXI platform. The PXI Express specification integrates PCI Express signaling into the PXI standard, which increases backplane bandwidth from 132 MB/s to 6 GB/s, a 45 times improvement. It also enhances PXI timing and synchronization features by incorporating a 100 MHz differential reference clock and differential triggers.

The PXI Express specification adds these features to PXI while maintaining backward compatibility:

- **Software:** PCI Express uses the same OS and driver model as PCI, resulting in complete software compatibility between PCI-based systems (such as PXI) and PCI Express-based systems (such as PXI Express). This software compatibility is ensured by the PCI Special Interest Group (PCI-SIG), a group composed of member companies, such as Intel, who are committed to the development and enhancement of the PCI and PCI Express standards.
- **Hardware:** PXI Express chassis provide hybrid peripheral slots that accept both PXI Express peripheral modules and hybrid-slot-compatible PXI peripheral modules. These peripheral slots deliver signaling for both PCI and PCI Express.

You can use code you have written for previous PXI systems with PXI Express systems because PXI Express maintains complete software compatibility with PXI. Software compatibility includes OSs such as Windows XP and Linux, application software such as Microsoft Office, and user code such as LabVIEW VIs and C++ projects. For more information, refer to [PXI Express](#).

Chapter 6: Case Studies

Microsoft Uses NI LabVIEW and PXI Modular Instruments to Develop Production Test System for Xbox 360 Controllers

Author(s): D.J. Mathias, Microsoft

Product Used: LabVIEW, Modular Instruments, Oscilloscopes/Digitizers, PXI/CompactPCI

The Challenge: Developing a comprehensive, low-cost production test system for the Microsoft Xbox 360 wired and wireless controllers.

The Solution: Using a flexible, automated test system based on Microsoft Windows XP, Microsoft SQL Server, National Instruments LabVIEW, and NI PXI modular instruments to test the functional performance of the Xbox 360 controller, both wired and wireless versions.

Designing Powerful Controllers for a New Generation of Gaming

In 2001, Microsoft deployed a PXI-based end-of-line functional test system for the original Xbox controller using NI LabVIEW and PXI modular instruments.

The system tested device communication and monitored data packets at the bit level to verify that all controller-functional messages were within specification. The system also monitored signals at the chip level to analyze the electrical signals for parameters such as rise/fall times, minimum/maximum voltage levels, and current draw.

In May 2005, Microsoft announced its latest innovation for digital entertainment and gaming, the Xbox 360, along with a new line of Xbox 360 wired and wireless controllers. The Xbox 360 wired controllers use a versatile, low-cost USB interface to communicate to the main game console. With the USB interface, the system easily accepts additional peripherals such as dance pads and steering wheels. The Xbox 360 controller-functional test system needed to perform similar tests to those of the original Xbox controller test system, but it required higher-performance signal capture to qualify the signal integrity of the new controller and ensure a high-quality user experience. With the latest NI modular instruments, including the NI PXI-5124 12-bit, 200 MS/s digitizer, we met the increased functional test requirements for the Xbox 360 controller. Using the LabVIEW graphical development environment, we created more than 100 tests, implemented Ethernet communication, and incorporated a data storage interface to our Microsoft SQL Server database.



Microsoft uses PXI and LabVIEW to ensure a quality gaming experience with the Xbox 360.

PXI Modular Instruments for Design Validation and Production Test

Using PXI instrumentation and LabVIEW, we built the test system in our Xbox 360 controller design validation lab and recently deployed it to our production line. During the validation and production cycle, the following NI PXI-based modular instruments provided us with a broad range of measurement functionality:

- PXI-5124 high-resolution digitizer for USB communication interface analysis
- PXI-4472 dynamic signal acquisition module for vibration feedback motor analysis
- PXI data acquisition modules for general-purpose analog I/O measurements
- PXI-6509 digital I/O module for general-purpose I/O control

We rapidly adapted the test system capabilities to meet our requirements for both the validation lab and production test by taking advantage of the broad range of PXI functionality, PXI modularity, and the PXI software-centric measurement approach.

The PXI-5124 high-resolution digitizer is a key component in the Xbox 360 controller end-of-line functional test system. The 200 MS/s real-time sampling rate and 12 bits of resolution on the PXI-5124 digitizer helped us verify the signal integrity of the USB communication between the controller and the Xbox 360 console with confidence. The high-resolution input and high-speed sampling rate are important features that make the digitizer a low-cost, quality solution – and a better option compared with higher-cost and lower-resolution oscilloscopes – to capture, monitor, and analyze the Xbox 360 controller USB signals, audio signals, and serial data signaling.

NI LabVIEW Interfacing with Microsoft SQL Server, TCP/IP, and ActiveX Controls

Functional test is a key component to any production line. The challenge in developing a production line functional tester is to package as many parallel test scenarios as possible within the given production cycle time. With the new functional test system for the Xbox 360 controller, we implemented a test strategy that resulted in a 100 percent increase in our test throughput per test station.

We used LabVIEW to run multiple tests in parallel to maximize test coverage during the given production cycle time, and we used the LabVIEW Database Connectivity Toolkit to connect to our Microsoft SQL Server database to store every DUT parameter. As each Xbox 360 controller rolls off the production line, each completed test sends more than 110 data parameters to the dedicated Microsoft SQL Server for post-test analysis to implement future production line and device enhancements. Using the integrated TCP/IP and support for embedded ActiveX controls in LabVIEW, we communicated to the USB and wireless controllers through our custom interfaces. Overall, LabVIEW helped us develop an optimized end-of-line production test system for the Xbox 360 controller with data storage to our Microsoft SQL Server, communication through TCP/IP, and programmatic interaction with ActiveX controls.

Microsoft Sees Results Using NI LabVIEW and PXI Modular Instruments

At Microsoft Corporation, we developed a versatile validation and end-of-line production test system for the Xbox and Xbox 360 controllers using Microsoft Windows XP, LabVIEW, and PXI. With the PXI-based system, we can achieve reliable production line testing and store all parameters to our Microsoft SQL Server. Using the high-resolution input and high sampling rate of the PXI-5124 digitizer, we acquire our test signals with 12 bits of resolution at data rates up to 200 MS/s, which provides a low-cost automated test system. Finally, using the power of the PC, we continue to easily upgrade and maintain our system today and for future development.

For more information, contact:

D.J. Mathias
Microsoft
One Microsoft Way
Redmond, WA 98052
Tel: 1-800-MICROSOFT

U.S. Navy: Developing Digital Test Equipment for Navy Aircraft Communications Using NI LabVIEW and the PXI Platform

Author(s): Lawrence M. David Jr. - ALE System Integration, Terry Stratoudakis, P.E. - ALE System Integration

Product Used: LabVIEW, PXI-8196 RT, PXI-4060, PXI-6542, High-Speed Digital I/O, PXI-8196, Digital Waveform Editor, Digital Multimeters, NI-HSDIO

The Challenge: Developing a small, versatile test system that mimics the onboard communications of a military aircraft and analyzes the communications for accuracy and completeness.

The Solution: Using the NI LabVIEW graphical programming environment, NI Digital Waveform Editor software, and PXI hardware to design and develop a flexible and comprehensive test system.

Using the NI LabVIEW graphical programming environment, NI Digital Waveform Editor software, and PXI hardware to design and develop a flexible and comprehensive test system.

“The analysis capabilities of LabVIEW were instrumental in filtering and cross-referencing results from multiple categories of tests to pinpoint circuitry defects.”

When a local high-tech electronics firm was awarded the contract to supply a communications interface hub for a Navy surveillance aircraft, it was also tasked with designing digital test equipment (DTE) to verify the initial functionality of the interface and provide ongoing verification for 20 years of field maintenance. Using a PXI platform from National Instruments, ALE System Integration provided expertise in hardware integration and software development for the system.

System Requirements

The interface unit (the end product being tested) was designed to coordinate all the aircraft’s digital and analog signal routing, including the internal intercom, external radio, radar, and all digital instrumentation. The interface needs to correctly process all appropriate signals while ignoring corrupted signals and noise. To verify this functionality, the DTE needed to inject all valid and invalid signals and interpret the interface’s responses.



Using the National Instruments PXI platform, ALE System Integration developed a system to test communications equipment onboard naval aircrafts.

In addition to verifying the aircraft's functionality, the DTE needed the capability to perform a self-test operation and the flexibility to perform a one-time design verification including all flight-worthiness tests; additional electromagnetic interference tests; and a series of physical tests such as the highly accelerated life test (HALT), explosive atmosphere, salt atmosphere, and thermal cycling. The DTE also had to be able to inject and interpret the wide range of signals onboard the aircraft including analog audio, serial (9600 baud), high-speed digital (5 MHz), and MIL-STD 1553, a military standard serial data bus that features a dual redundant balanced line physical layer, time division multiplexing, and a half-duplex command/response protocol.

Designing the DTE with PXI Hardware from National Instruments

The DTE was implemented using an NI PXI-8196 embedded controller containing the following modules: NI PXI-6513, PXI-6542, PXI-2569, PXI-6511, and PXI-4060. The PXI-6542 module was used at a clock speed of 20 MHz allocating four bits per tick of the 5 MHz DUT clock, thus improving test accuracy. The NI-HSDIO software greatly reduced development time. The system also included a Condor QPC-1553 from GE Fanuc Embedded Systems that included LabVIEW drivers to further simplify software development.

We also used two programmable power supplies: a three-phase AC supply (GPIB) at 400 Hz to mimic the aircraft's power and a DC supply (USB) to mimic internal supply circuitry. Both programmable power supplies were interfaced to the system via the USB and GPIB ports of the PXI-8196 embedded controller.

Analyzing Digital Test Data with LabVIEW

With a graphical user interface (GUI) developed using LabVIEW software, the technician experiences improved flexibility in configuring test runs of any of the 10 categories of tests in addition to a DTE self-test. The analysis capabilities of LabVIEW were instrumental in filtering and cross-referencing results from multiple categories of tests to pinpoint circuitry defects. One constraint of this project was the simultaneous development of the test system and the interface unit. However, two major advantages of using LabVIEW to develop this system were the prototyping and debugging capabilities including custom probes and highlighted execution. The code developed for the test system was reused to conduct the ongoing tests for the system.

We were able to develop the test system with ongoing changes to the specification and give invaluable support to our client in their own development process. The key to this success was the NI Digital Waveform Editor software. With this tool, our client created digital waveform files using the editor, which we then fed into the test system. The resulting digital data file was easy for our client to review. Overall, this was a great experience for our client. Using the LabVIEW and PXI platform, we delivered software faster than our client expected, and we were able to integrate seamlessly with their team.

For more information, contact:

Terry Stratoudakis, P.E.
ALE System Integration
United States
terry@aleconsultants.com

Sanmina-SCI Exceeds Throughput Goals with PXI Tester and Multithreaded Software

Author(s): Mike Oehrlein, Sanmina-SCI Corporation

Product: Digital Multimeters, LabWindows/CVI, Modular Instruments, NI TestStand, PXI/CompactPCI

The Challenge: Developing a compact and high-speed functional test station that performs parallel calibration on eight medical devices.

The Solution: Using National Instruments LabWindows/CVI, NI TestStand, and high-performance PXI modular instruments to develop an automated and modular production tester with database and statistical process control capability for medical device calibration.

Test System Requirements

Sanmina-SCI, one of the world's leading contract manufacturers, recently developed a test application for a medical device that measures blood glucose levels by measuring the current and impedance characteristics produced from an electrochemical reaction. We needed to build a functional test and calibration system that complied with FDA regulations, calibrated the DC amplifier circuits of the measurement engine, and verified the operation of other critical support circuits contained within the device under test. Additionally, in real time, the system had to switch a variety of input loads into the measurement engine to emulate the complex task of simulating blood response. Lastly, the system had to reliably and repeatedly process up to 83,000 devices per week while maintaining a cycle time of effectively 30 seconds per device.

Because of these requirements, we chose a flexible software framework that facilitated multithreaded parallel testing; efficient communication with commercial databases; and a custom software interface with strict user management for administrators, supervisors, engineers, and manufacturing operators.

Compact, High-Speed Test Solution

The Sanmina-SCI design team knew from previous experience that a design solution based on traditional instrumentation alone could not meet test-throughput needs. For this application, the design team needed to build a hybrid test system based on PXI and GPIB modular instruments as well as a custom FPGA-controlled interface to the device under test that could test eight devices in parallel. The completed test system included eight NI PXI-4070 6½-digit FlexDMM devices, eight NI PXI-6533 high-speed digital I/O modules, an NI PXI-6508 general-purpose digital I/O module, a GPIB-based power supply, and an LCR meter. The FlexDMM delivered the throughput and accuracy the design team needed to acquire all of the voltage and current measurements on the device under test. The design



High-Speed PXI-Based Test System for Calibrating Medical Devices

team used the PXI digital devices to communicate with the DUT and write the calibration values to the onboard EEPROM.

We chose NI TestStand for test management software because it provided test sequencing of LabWindows/CVI and C# .NET modules, native parallel testing support, and a comprehensive user management framework. Its flexible and open source code operator interface also gave our design team full control over the look and feel of the operator interface. Rather than worry about thread management and synchronization, we could use NI TestStand to focus on the details of the test management process. We used LabWindows/CVI for test module development because of the built-in test and measurement functions and user interface controls, in addition to the SQL database connectivity capabilities. With LabWindows/CVI, we could create highly efficient ANSI C source code that integrated with modular instruments and GPIB. By using commercially available software such as LabWindows/CVI and NI TestStand, we kept software development time and costs to a minimum.

Exceeded Yield Expectations

The PXI-based tester provided a compact calibration system that met our high-speed throughput requirements while exceeding the initial system yield requirements by achieving yields greater than 95 percent in production. The customer was satisfied with the completed test system and has plans to expand the system by building additional stations to satisfy product demand and adapting fixtures and software for the next-generation devices.

For more information, contact:

Mike Oehrlein
Sanmina-SCI Corporation
13000 S. Memorial Pkwy
PO Box 1000
Huntsville, AL 35807
Tel: (256) 882-4800, ext. 8587
www.sanmina-sci.com