Test-System Development Guide

A Comprehensive Handbook for Test Engineers

Agilent Technologies

Contents

Introduction 6

Section 1. Test System Design 6
Section 2. Networking Choices 6
Section 3. LXI: The Future of Test 6
Section 4. RF/Microwave Test Systems 6

Section 1. Test System Design 7

Overview 7

1. Introduction to Test-System Design 9

Introduction 9
Transforming test into a strategic advantage 9
Test-system considerations 10
Planning your test system 10
Control decisions 12
Planning for the future 16
Conclusion 16

2. Computer I/O Considerations 19

Introduction 19
Proprietary I/O versus industry-standard I/O 19
GPIB interfaces 20
USB interfaces 20
LAN interfaces 21
Which I/O interface should you use? 23
Conclusion 26

3. Understanding Drivers and Direct I/O 27

Introduction 27
History 27
Choosing and using instrument drivers 31
Conclusion 34

4. Choosing Your Test-System Software Architecture 35

Introduction 35
Gathering and documenting software requirements 36
Programming and controlling your instruments 38
Collecting and storing the test data 38
Designing the user interface 40
Choosing the development environment 42
Working with open standards 44
Developing a test sequence 46
Planning for software reuse 47
Conclusion 50

5. Choosing Your Test-System Hardware Architecture and Instrumentation 51

Introduction 51
System architecture 51
Choosing instruments for your test system 59
Example test system 61
Conclusion 64

6. Understanding the Effects of Racking and System Interconnections 65

Introduction 65
Choosing racks and accessories 65
Instrument layout 66
AC power distribution 73
Conclusion 74

7. Maximizing System Throughput and Optimizing System Deployment 75

Introduction 75
Upfront design decisions affect throughput 76
Fine-tuning your system for speed 84
Conclusion 86

8. Operational Maintenance 87

Introduction 87
Worldwide considerations 87
Calibration 89
Diagnostics and Repair 90
Cleaning 91
Upgrades and expansion 92
Conclusion 92

Section 2. Networking Choices 93

Overview **93**

9. Using LAN in Test Systems: The Basics 95

Introduction **95**
Coping with complexity **95**
Setting the standard **96**
Using LAN in test systems **98**
Conclusion **100**

10. Using LAN in Test Systems: Network Configuration and Basic Security 101

Introduction **101**
Creating a safe haven **101**
Understanding the pitfalls **101**
Designing the private, protected LAN **102**
Conclusion **105**

11. Using LAN in Test Systems: PC Configuration 107

Introduction **107**
Creating the right environment **107**
Exploring network settings in Windows XP and Vista **107**
Using multiple network connections **108**
Managing IP addresses **108**
Configuring LAN with Agilent IO Libraries Suite **109**
Conclusion **110**

12. Using USB in the Test and Measurement Environment 111

Introduction **111**
USB in the PC universe **111**
Agilent support for USB instrument connectivity **112**
Setting up USB instruments with the Agilent IO Libraries **113**
Conclusion **116**

13. Using SCPI and Direct I/O vs. Drivers 117

Introduction **117**
Deciding how to communicate **117**
Sketching the big picture **117**
Achieving communication **118**
Exploring the application alternatives **120**
Maximizing performance and flexibility **121**
Assessing I/O software alternatives **123**
Conclusion **124**

14. Using LAN in Test Systems: Applications 125

Introduction **125**
Scenario 1: Sharing instruments **125**
Scenario 2: Remote monitoring and data acquisition **127**
Scenario 3: Functional test systems **130**
Configuring a VPN **131**
Comparing network performance **133**
Conclusion **134**

15. Using LAN in Test Systems: Setting Up System I/O 135

Introduction **135**
Simplifying LAN-based instrument connections **135**
Assessing the Agilent IO Libraries Suite **135**
Connecting instruments to LAN **137**
Conclusion **140**

Section 3. LXI: The Future of Test 141

Overview **141**

16. Value, Performance and Flexibility: The Promise of LXI 143

Introduction **143**
Why test managers are asking for a new approach **143**
Addressing the challenges with LXI **144**
The advantages of LXI **146**
A closer look at LXI **150**
Exploring new possibilities with LXI **153**
Appendix 16A: Defining synthetic instruments **155**
Appendix 16B: Creating cost-effective measurement solutions with Agilent Open to test your way **156**

17. Transitioning from GPIB to LXI 157

Introduction **157**
Comparing system architectures **157**
Setting up an LXI system **159**
Simplifying software changes **160**
Conclusion **161**

18. Creating Hybrid Test Systems with PXI, VXI and LXI 163

Introduction **163**
Assessing modular systems **163**
Exploring LAN-based hybrid systems **165**
Going beyond hybrid to all-LXI **167**
Conclusion **168**

19. Assessing Synthetic Instruments 169

Introduction **169**
Reviewing the roots of SI **169**
Putting SIs in perspective **170**
Comparing present and future approaches **170**
Exploring the initial applications **173**
Utilizing current SI devices **174**
Conclusion **176**

Section 4. RF/Microwave Test Systems 177

Overview **177**

20. Optimizing the Elements of an RF/Microwave Test System 179

Introduction **179**
Letting the DUT define "future" **179**
Reviewing some essential considerations **180**
Translating requirements into optimized equipment choices **181**
Pulling it all together **184**

21. Six Hints for Enhancing Measurement Integrity in RF/Microwave Test Systems 185

Introduction **185**
Hint 1: Prioritize performance, speed and repeatability **185**
Hint 2: Review the nature and behavior of the DUT **187**
Hint 3: Understand, characterize and correct RF signal paths **188**
Hint 4: Be aware of everything connected to an instrument **189**
Hint 5: Examine the operational attributes of switches **191**
Hint 6: Accelerate measurement set up and execution **192**
Conclusion **192**

22. Calibrating Signal Paths in RF/Microwave Test Systems 193

Introduction **193**
Understanding vector and scalar calibration **193**
Performing vector calibration of network-analyzer paths **195**
Performing vector calibration of non-network-analyzer paths **196**
Performing scalar calibration of non-network-analyzer paths **197**
Conclusion **198**

Glossary of Test-System Development Terms 199

Introduction

The Agilent *Test-System Development Guide* is a comprehensive handbook for test engineers who need to maximize performance and flexibility while minimizing cost and complexity. Throughout, you'll find practical advice and real-world examples that illustrate the decisions involved in overall system architecture, networking solutions, and instrumentation hardware and software.

The Guide is divided into four sections, beginning with the basics of test system design, following by networking decisions, the new LXI instrumentation standard, and special considerations for RF/microwave tests:

Section 1. Test System Design

Starting with the fundamental philosophies of test system design, the eight chapters in this section cover I/O considerations, decisions regarding software and hardware architectures, racking and system interconnects, data throughout optimization, test planning, and various deployment issues.

Section 2. Networking Choices

These seven chapters explore the networking choices available for today's test systems. Local area networking (LAN) is covered in detail, including both network and PC configuration. The Universal Serial Bus (USB) is also covered as a networking option, as well as decisions regarding drivers and I/O software.

Section 3. LXI: The Future of Test

This section offers an in-depth analysis of LXI, a new measurement platform that combines the advantages of PC-based connectivity with the flexibility of card-based instrumentation—without the disadvantages of a conventional cardcage. LXI offers greater flexibility by incorporating a variety of current and future instrument form factors, lower costs and smaller footprint by eliminating the cardcage, and increased security through the use of a private LAN. This section explains why LXI can meet future test needs more effectively than current approaches and how to make the transition from GPIB-based systems.

Section 4. RF/Microwave Test Systems

RF/microwave test systems present a number of unique challenges, particularly in the face of increasingly complex devices and test requirements. This section offers advice on configuring test systems that balance the need for performance, speed, and repeatability.

Please visit **www.agilent.com/find/open** for the latest information on the products discussed in this handbook.

Section 1. Test System Design

Overview

The eight chapters in this section offer a comprehensive introduction to designing and deploying an automated test system:

1. **Introduction to Test System Design,** covers test-system philosophy and planning and discusses how test is used in three sectors: R&D, design validation and manufacturing.
2. **Computer I/O Considerations,** describes the advantages of using computer-industry standard I/O and explores the advantages and disadvantages of GPIB, USB and LAN interfaces for rack-and-stack test systems.
3. **Understanding Drivers and Direct I/O,** answers common questions about the use of drivers and direct I/O to send commands from a PC application to the test instrument.
4. **Choosing Your Test-System Software Architecture,** helps you choose the direction for your software based on the application you have in mind and the amount of experience you have. It explores the entire software development process, from gathering and documenting software requirements through design reuse considerations.
5. **Choosing Your Test-System Hardware Architecture and Instrumentation,** explores the hardware architecture decisions you must make before you begin building your system to ensure that it provides you with the performance and flexibility you need. It also discusses issues you should consider as you select instruments for your system.
6. **Understanding the Effects of Racking and System Interconnections,** discusses the important considerations for arranging your test equipment in a rack, including weight distribution, heat dissipation, instrument accessibility and ease of use. It also explores ways to minimize magnetic interference and conducted and radiated noise to maximize measurement accuracy.
7. **Maximizing System Throughput and Optimizing System Deployment,** discusses hardware and software design decisions that affect throughput, including instrument and switch selection, as well as test-plan optimization and I/O and data transfer issues. It also presents ways to optimize your system as you prepare to deploy it.
8. **perational Maintenance,** addresses key issues to consider once your system is operational, including worldwide deployment, calibration, diagnostics and repair, cleaning, upgrades and expansion.

1. Introduction to Test-System Design

Introduction

This chapter offers an overview of the process of designing test systems, beginning with a discussion of how carefully designed systems can transform test into a strategic competitive advantage. The chapter then walks you through the key factors to consider when designing a test system, choosing the level of automated control, and planning for future needs. It concludes with a comparative case study of testing power supplies using manual, semi-automated and automated control.

Transforming test into a strategic advantage

Functional test is fundamental to the electronics world. In the past, test has been treated as a necessary expense, but enlightened companies have realized that test can be a significant asset. A test system can be used for far more than simply verifying the limits of the device under test (DUT). Consider these possibilities:

- find the weaknesses of the device—before your customers do
- predict failures or out-of-spec trends in production
- search for the boundaries of the design—to stretch specifications or search for something you didn't know the product could do
- verify the long-term characteristics of the product
- optimize a production process
- test for environmental limits
- find the weaknesses in a competitor's product

Test can be used simply as a gating factor for "good" or "bad" devices, or it can be used to gain a competitive advantage. This chapter offers an overall view of how tests are made, techniques to optimize tests, and a number of methods you can use to your advantage. It covers the three primary sectors of the product life cycle that require test: R&D, design validation, and manufacturing. Other chapters cover such topics as hardware architecture, choosing instruments, software architecture, computer I/O and connectivity, assembling a test system, maximizing throughput, and optimizing deployment and maintenance.

A systematic test-system design process as outlined in this guide will assist you to quickly design a test system that produces reliable and repeatable results, meets your throughput requirements, and does so within your budget. For further information regarding test-system design, you can refer to the book from which much of the information in this chapter was derived: *Test-System Design, A Systematic Approach* by Tursky, Gordon, and Cowie (Prentice Hall, 2001).

The earlier a product weakness is discovered, the less expensive the consequences. That's one reason why the role of test changes with the stage of the product life cycle. When a product is first developed, the role of test is to verify that the design concept is viable. This calls for quick measurements, usually with hands-on use of discrete test instruments. Sometimes there is a need to load measurement data into an Excel spreadsheet for use in a lab report or for further analysis.

Excel is the most common software analysis tool for the R&D engineer. The connection is usually simple: a PC connected via GPIB or USB to an instrument or a small set of instruments. Simple software, such as Agilent IntuiLink, finishes the connection.

Once the design becomes more solid, there is a need to find its limits and weaknesses. That's where the design validation system comes in. To make the results more repeatable and less dependent upon operator expertise, the test system is automated using a PC and some sort of graphical software such as Agilent VEE or National Instruments LabVIEW.

Graphical software, often used for design validation testing, gives the engineer a more comprehensive set of tools for control and analysis, while at the same time creating a more repeatable measurement process that may include remote control of sources, measurements, and system switching. The same instruments used in the R&D bench system are often used in design validation. This gives continuity to the whole process, so that the initial R&D measurements can be compared to those made for design validation.

Textual software generally provides an effective programming environment for manufacturing test, as it enables the engineer to extract the highest throughput from the test system. In manufacturing, repeatability and reliability become paramount concerns. Again, if the same equipment can be used for all three test situations (R&D, design validation, and manufacturing), then the R&D engineer can more readily assist with any problems that may arise during manufacturing test.

The process of designing and integrating systems used for electronic test requires more than simply coding instrument commands to automate the measurements made on the R&D bench. The instruments are only one part of the complete test system; cables, software, test-plan documentation, and fixturing are equally important. The latter are especially prevalent in a manufacturing environment.

Test-system considerations

There are many factors to consider when developing a test system. The three main driving factors are test requirements, development time, and test cost. The factor that is most important will drive the other two. For example, if the test requirement is for a very accurate measurement, as in R&D or design validation, you must be willing to take a bit more time to achieve the required accuracy. On the other hand, the manufacturing manager would not be pleased if the test system were to perform more tests than required, or perform them at a higher-than needed level of accuracy, due to the obvious impacts on test-system cost and throughput.

Before the process to design a test system can begin, you must have a good understanding of the test application. This goes beyond simply understanding the device you are testing, as you must also be aware of other factors such as the skill level of the test system operator, the operating environment, and any standards requirements.

Planning your test system

Creating a comprehensive test plan allows you to take a big-picture view of the project and forces you to focus on meeting the objectives and requirements for the test system. The result is a considerable time saving in the development process.

Even in the R&D environment, there are times when it is useful to create a test plan, so that you can document and compare results after each design cycle. You must also consider the future for any test system you create today. It may be reasonable to create a dedicated and somewhat inflexible test system on some high-volume projects, but it is usually more appropriate to create a system that has the flexibility to adapt to future needs.

The test plan describes more than just the requirements of the DUT. It should also cover other areas of the test such as the level of experience required of the test system operator, calibration and maintenance requirements, physical limitations, and throughput requirements.

The first step in creating a test system is to seek out and compile all the information needed to create an overall test plan. Important information includes the following:

- functional and parametric tests to be performed
- DUT design validation criteria
- format and usage of test results, including sharing data throughout the enterprise
- number of tests
- DUT pin counts
- physical constraints such as size, environment, and available power
- heat buildup and power dissipation
- how the test system will be verified, maintained, and calibrated
- RF environment
- accuracy and resolution requirements
- throughput goals
- development time constraints
- software-development and runtime environment
- cost constraints
- continuity constraints with existing legacy systems

Among the decisions involved in determining the design of a test system, the most obvious is what it is you must test. This is usually defined in a test specification. The test specification should include a complete list of the product functions to be verified, operating parameters to meet, and any regulatory standards to adhere to.

Accuracy

System accuracy is a critical specification of any test system, and the overall test plan should include both the accuracy requirements of the test and the recommended margin. As a minimum, the test equipment should have twice the accuracy specified for the DUT. To maintain this margin requires that the operating temperature be maintained closely and that calibration cycles be followed faithfully.

Often, it is more cost effective to buy test equipment with a 10X accuracy margin so that calibration and maintenance requirements can be relaxed without affecting accuracy. In the "10X" case, you may even increase the product yield, since the product can come closer to its specification tolerance limits because you can count on the accuracy of the test system. Whatever the accuracy required, you must have confidence that you can rely on the results. Obviously, a calibration and maintenance plan is important for achieving the required test accuracy.

When determining instrument requirements, resolution must be specified as well as accuracy. Accuracy defines how close a measurement agrees with a standard value. Resolution indicates the smallest change that can be measured. There may be times when the absolute accuracy over an extended period is not as important as the resolution to measure small changes over the short term. Switching, fixturing, and cabling also add noise and crosstalk that can increase uncertainties.

Throughput

Throughput requirements will direct the necessary system capacity. Throughput is normally more important in the manufacturing environment than during design validation and rarely a concern in R&D. However, some complex designs require lengthy testing to be validated before going into production. A significant delay during R&D or design validation can cause a product launch to be delayed, and be costly in terms of missed market opportunity.

Downtime seriously degrades test-system throughput and can have a significant impact on product shipments. Predicting and preparing for wear-out mechanisms can reduce downtime. Further, using diagnostics or built-in test can help determine when the test system is about to fail. Such preventative maintenance procedures can result in big savings when they identify a test system failure before many DUTs are erroneously tested. In all cases, whether in R&D, design validation, or manufacturing, you should consider how you will handle downtime, either with spare test equipment or with a known path to repair or rental.

The overall test plan is a good place to describe what diagnostics the test system will require. It is easy to overlook test-system diagnostics as time consuming and costly to develop. Diagnostics are an important tool for maintaining throughput by reducing the downtime to repair failures. On most systems, a well-thought-out diagnostics approach will shorten test-system deployment time as well. Developing and following a calibration and maintenance plan in conjunction with the diagnostics is another way to prevent system failures that disrupt test-system throughput.

Results

Obviously, all tests must produce results. Sometimes this is merely a simple pass/fail indication, but often test results must be analyzed and archived. These requirements must also be defined in the overall test plan. If the test sequence is short, a few minutes or so, it is simpler to perform all data analysis after the test is over. However, if the test sequences are lengthy, some intermediate data analysis is recommended so that failing functions can be detected early enough to halt the test and avoid wasted time.

Hardware/software decisions

Once the requirements of the test system have been established in the test plan, then it is time to outline the design of the test system itself. The question is: What to consider first—software or hardware? In the past, the hardware provided the lead in test-system development. The test instruments that met the accuracy and throughput requirements were defined first, and then software was created to automate the test system.

But today, software can often be more expensive to develop than the cost of the hardware, so if test system cost is a driving factor, it is important to make sure that a new system can use as much existing software as possible.

The choice of programming languages may be based primarily on the experience of the programmer. Some find graphical languages such as Agilent VEE or LabVIEW easy to use. Others believe that textual languages such as C++, MATLAB or Visual Basic are easier to use, especially for complex test programs. If it is important to use existing textual test code, then a multi-language development environment like Microsoft® Visual Studio .NET is a definite advantage. For a thorough examination of test-system software options, see Chapter 4, *Choosing Your Test-System Software Architecture*.

In any case, it is critical to ensure that drivers exist for the selected equipment. If the required drivers and support are not available, the anticipated advantages provided by the selected language may not materialize. Driver issues are discussed in detail in "*Understanding Drivers and Direct I/O.*"

Control decisions

A major consideration for a test system is the level of automation to build into the system to control the test process. Manual control requires that a human operator make all of the test connections, set the instruments, and then record the data. Increasingly, even in simple R&D setups, most engineers prefer to use instruments under the control of a PC in order to have a record of the test.

Once the testing becomes more complex or repetitive, a fully automated test system is in order. A fully automated test system takes care of signal switching, measurement, recording, and even analysis of the results for pass/fail determination. Once the DUT is in the test fixture, the test system takes over and runs all of the tests. This is the ultimate in terms of test speed, reliability, and repeatability, but it is also the most expensive and time consuming to develop.

The type of control, either manual, semi-automated, or fully automated, should be determined early as it will influence which instruments you select. As shown in Table 1.1, many factors influence which control method is most suitable for your application.

Table 1.1. Comparison of test system control options

	Manual	Semi-automated	Automated
Instrument cost	Varies; can be higher than automated, since R&D typically needs more accuracy than production specs	Similar to manual	Depends on requirements; if space is paramount, cardcages can be used, but they are typically more expensive than standalone rack & stack instruments. Modular instruments may meet space needs with full compatibility to rack and stack instruments
Development cost	Very low; just hook up and go	Low or high depending upon how much is automated	High
Operator experience	Very high, often experienced engineers	High as the manual portions of the system may require an engineer	Low
Development time	Low	Low to high	High
Flexibility	High; changes can be made easily	Medium; some portions can easily be changed.	Low; changes require significant effort
Throughput	Low	Medium	High
Repeatability	Varies with expertise	Medium	High
System calibration	Rare; usually only each instrument is calibrated	Some system calibration may be possible	Full system calibration is possible
Self-check diagnostics	Individual instruments only, not system diagnostics	Individual instruments only, not system diagnostics	Common
Ease of instrument reuse	High	Medium	Low if card cage, medium if stand alone or modular instruments
Potential for human error	High	Medium	Low

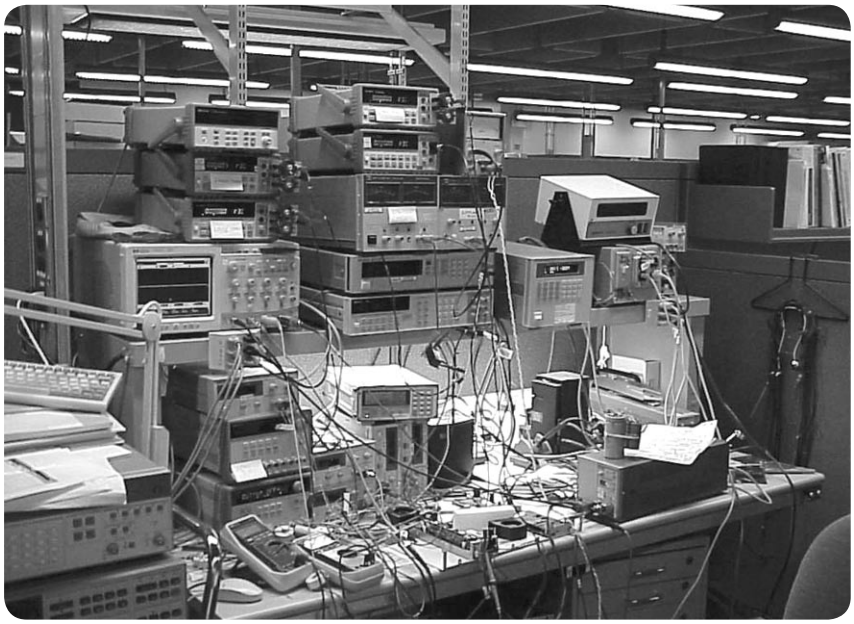
Manual control

A test system based on manual control depends entirely on the operator for all test functions (Figure 1.1). Connections between the DUT and instruments are made manually with test leads or cables. R&D engineers may follow procedures that are completely undocumented, but when using a manual control system for other test requirements, each instrument is normally manually operated by following a documented procedure. The results of each test are then manually recorded. This is a very flexible approach as it allows changes to the test system to be made very easily. On the other hand, it is a very slow method of testing and has significant problems with repeatability. For example, the engineer may make readings one time with the voltmeter at full scale, while the next reading might be at 1/10 of full scale, resulting in a slightly different answer.

Manual control is often the least expensive test-system control option to set up, since it may not include such items as a system switch, expensive software, or test fixtures. Also, the time and cost required to set up the test are very low. However, the instrument cost for manual control varies. Often, the R&D application calls for a more accurate measurement than the equivalent measurement needed in manufacturing and therefore requires rather expensive instruments.

The cost to conduct the test is usually very high. Manual control generally requires a skilled operator to follow the labor-intensive test procedures. System self-testing is almost impossible, and complex and frequent calibration is often required due to the high accuracies needed. Typically, only the individual instruments are calibrated and not the entire system. As a result, inexperienced engineers may believe that the overall system accuracy is better than it actually is.

Figure 1.1. A test system using manual control requires a skilled operator.



Repeatability is a concern with manual test systems. There are many opportunities for operator error to go unnoticed. These errors creep in when the operator is attaching cables, setting instruments, recording results, and even when transferring the results to other documents.

Even with these limitations, the manual approach can be useful. With due diligence while conducting the test and techniques such as using the same cables to increase repeatability, the manual approach can produce reasonably reliable results. Another advantage of manual control is the ease in which the test system can be reconfigured or the instruments used for other projects.

Additionally, a skilled engineer conducting the tests is constantly comparing the results against expectations, thereby providing a form of continuous verification of the test system. An incorrectly operating fully automated test system could continue to test for hours, days, or even weeks without detecting the problem, resulting in the shipment of incorrectly tested products.

Use manual control when

- a small number of devices are being tested
- cost of automation outweighs benefits
- speed of test is not critical
- test requirements may change regularly
- the delay to create an automated system is unacceptable
- skilled operators are available
- the instruments need to be easily disassembled for use elsewhere.

Semi-automated control

Semi-automated control is a common type of control approach used for test systems, and is useful in R&D, design validation, and manufacturing test (Figure 1.2). Test systems using this control approach have manual portions for flexibility where it is needed and automation where it makes sense. Those sections of the test system that are expected to change often or would be too expensive to automate can be manual. Those sections that will not change or would benefit from automatic data recording can be automated.

A semi-automated test system might require the operator to manually connect the DUT, provide instructions to the operator for the procedural steps, and automatically record the results. For example, a semi-automated system might have an oscilloscope and an RF source that are under computer control, with a power supply under manual control. The engineer would vary the voltage to the DUT via the power supply, run a set of tests at this voltage level, and then manually change the voltage and run another set of tests.

Figure 1.2. A test system using semi-automated control often uses a PC for the operator interface.

Semi-automated control is often much faster than manual control and produces a more reliable and repeatable result. This method of control can take advantage of simplified software development with Agilent's VEE or Visual Studio .NET for quickly creating the required automation.

The most common type of test equipment includes a fully functional front panel and a computer interface that allows both manual and automated use. This is a major benefit, even when automating, as you can always go back to a manual approach if you need to measure other parameters, troubleshoot the system, or conduct an experiment. These standalone instruments are beneficial when developing a fully automated test system for manufacturing as it is common to start with a semi-automated system and then increase the level of automation as experience and production volume increases.

Use semi-automated control when

- automation benefits will outweigh added costs
- test volume does not require full automation
- some flexibility in the test system is required
- reasonably repeatable results are required
- skilled operators are available or close by
- a move to full automation is anticipated but not yet required

Automated control

Fully automated test systems are the domain of complex design validation testing or the manufacturing test environment (Figure 1.3); they are rarely used in R&D. All of the instruments, signal switching, and connections to the DUT are controlled by computer. In some automated test systems, an operator may be required to manually install the DUT into a test fixture as a single action, but others have an automated handler to insert and remove the DUT from the test fixture.

Full automation is the most expensive control method in terms of software development time, but it also results in the highest throughput and most repeatable and reliable measurements by nearly removing the human-error factor from the test. The skill level required of the operator is usually much reduced.

Full system calibration and diagnostics are easier to implement in an automated system where software can reconfigure the test system to allow it to test and calibrate itself against an external traceable reference. Full system calibration can even calibrate the cables and connections instead of just the individual instruments.

Proper diagnostics designed into an automated test system can test most of the system. You can create a diagnostic device that plugs into the DUT fixture. This device will connect test stimulus signals to test measurement instruments. Diagnostic software you create will then configure the test system to verify operation through the same switches, cables, and connectors that are used for testing.

There must be compelling reasons to justify an automated test system. Not only is the initial development cost high, but any changes or upgrades to an automated system can be very expensive. The compelling reason for the expense is usually the high-volume requirements of manufacturing test, but there are times during R&D and design validation when the required accuracy is very high or the test is very complex, making it necessary to automate the test to remove potential human errors or speed up the test process.

Use fully automated control when

- high-volume manufacturing requires automation
- precision or repeatable tests are required to test the DUT
- reducing test time is critical
- test requirements are known and stable
- cost per test outweighs test-system development cost
- time is available for development
- skilled operators are not available
- accuracy or complexity requirements dictate automation

Figure 1.3. A fully automated test system requires minimal operator interaction.

Planning for the future

When making test-system design decisions, you should keep future needs in mind. Upgrades are a fact of life for a test system. They can be very expensive and time consuming but are often unavoidable. Naturally, any upgrades must justify the expense and effort required. Reasons for upgrades include

- accommodate changes in design of the DUT
- conduct additional tests
- obtain higher accuracy
- obtain higher throughput
- eliminate redundant tests
- rearrange the test sequence to detect failures earlier
- improve analysis
- automate more of the test
- decrease the skill level required to operate the test system
- replace obsolete equipment
- change reporting requirements
- upgrade the operating system
- conform to new standards
- add newly developed models
- repeatability is important

A few moments considering the future can have a significant impact on future options. For example, when selecting instruments for a manual system, there is usually very little added cost to select instruments that have computer interfaces. You may not need the interface today, but computer control is not possible without it (and could be costly, difficult, or even impossible to add at a later date).

Using open standards will increase the likelihood that test system components will be useable in the future. Proprietary interfaces have a habit of disappearing or not supplying the drivers you need for future software options. Using proprietary measurements made by specific equipment in a test system from manufacturers that do not supply future upgrade paths could make an entire test system obsolete if that exact instrument is no longer available.

Consider where the instrument architecture is in its lifecycle. For instance, is it a cardcage design based on a PC backplane that will soon be replaced? Are vendors designing new products to this architecture (or to its replacement)?

Following proper software design techniques resulting in well-written software that is easily understood, maintained, and modified is an obvious requirement for future upgrades. Good documentation is also critical to the future of a test system: Chances are you will not be the one that is tasked with future modifications.

Conclusion

Although test-system development is a complex task that can include many aspects of electronic and mechanical design, following a systematic approach and partnering with quality test equipment manufacturers will enable you to enhance your success while lowering the cost and time it takes to create the test system.

Case study: testing power supplies

This case study is an example of how a test system can evolve from R&D to design validation to manufacturing. Many of the same instruments are used in all three areas with the major difference being the type of control used. This is a common practice as the knowledge gained in each phase of product development is transferred to the next.

Manual control

When developing a product such as a power supply, the R&D engineer will create a test system as required to explore options and verify results. The test bench in Figure 1.1 is typical of such use. Many instruments are within reach and it is easy to rearrange them as needed. All of the connections to the DUT are made manually and each instrument is manually operated. This is an example of a test system with manual control.

The flexibility to quickly move from measurement to insight to next measurement, whatever that next measurement might be, is obvious. Standalone test instruments readily lend themselves to this usage model. The high level of skill required of the operator is also important. There is significant opportunity for error and confusion with a manually controlled system. R&D engineers are in their element at such a bench, but it falls short on reliability and repeatability when compared to other control methods.

The block diagram in Figure 1.4 shows the interconnection of the instruments for some of the tests used during the R&D phase of power supply development. Some of the standard tests measure output-voltage accuracy, output noise, load regulation, line regulation and output programming speed.

The test system diagrammed in Figure 1.4 is just one example of a manual setup for testing some aspects of the design. Other R&D engineers would have other manual setups on their benches to test for other parameters. In this case, the total R&D manual test system is actually distributed throughout the benches of the entire design team.

More-specialized tests will also be conducted at this stage. Loop gain (Bode plot) is used to evaluate the stability of the control loops used to regulate the output voltage and current of the power supply. Load transient response is measured by applying a load-current step change and monitoring the output voltage on the scope, also giving insight into the stability of the control loops. Voltage and current stress on the components are also measured so power can be calculated to ensure that no parts are over stressed. The temperature of individual components may also be measured.

As these measurements are made, the test system is rearranged, the cables are attached as required, the instruments are manually controlled, and the results are noted. Often, the exact configuration is not recorded, making an exact repeat of the measurement difficult. The cable connections are often made with probes and clip leads in a manner that is quick but not reliable. Even so, the advantages to a skilled operator far outweigh the problems associated with manually controlling a test bench.

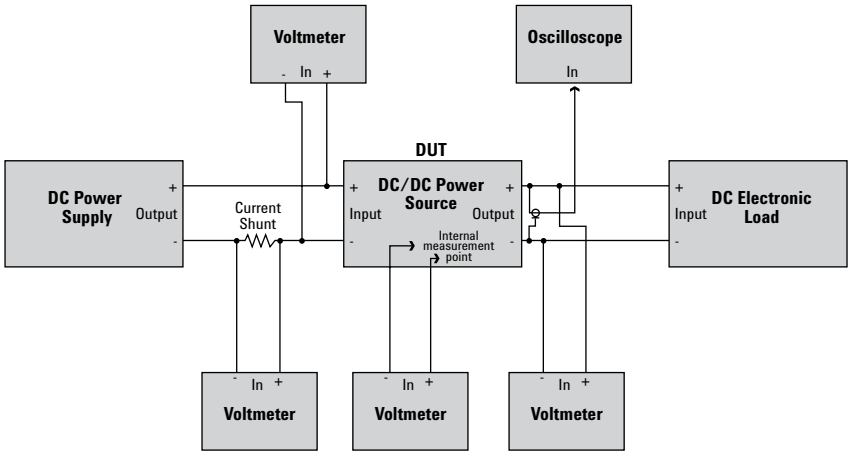
Semi-automated control

The design is "complete." Now it needs validation, so the test requirements are somewhat different. In this case, the same instruments are used, but a computer is added for semi-automated control. The block diagram of Figure 1.4 remains the same, but now a computer is connected to some of the instruments.

Many of the same measurements are made during design validation as were made during R&D. But now, more of them can be made to fully validate the design. For example, the output accuracy of the power supply under test can be checked at a variety of operating conditions. The input voltage, load current, and even the ambient temperature can be varied to ensure proper regulation of the output voltage and that the output noise is within requirements. The same tests can be conducted on multiple prototypes to ensure that the design is consistent across units. Further, these tests can be completed much faster and include automated data recording, enabling statistical analysis.

The repeatability and reliability of semi-automated control along with automated data gathering are a significant enhancement to manual control. By selecting instruments that include computer interfaces, automating portions of the test system is much easier. In many cases, the automation is merely a matter of having a computer perform the commands and read the results that were done by an operator.

Figure 1.4. Block diagram of a manually controlled test system used for R&D



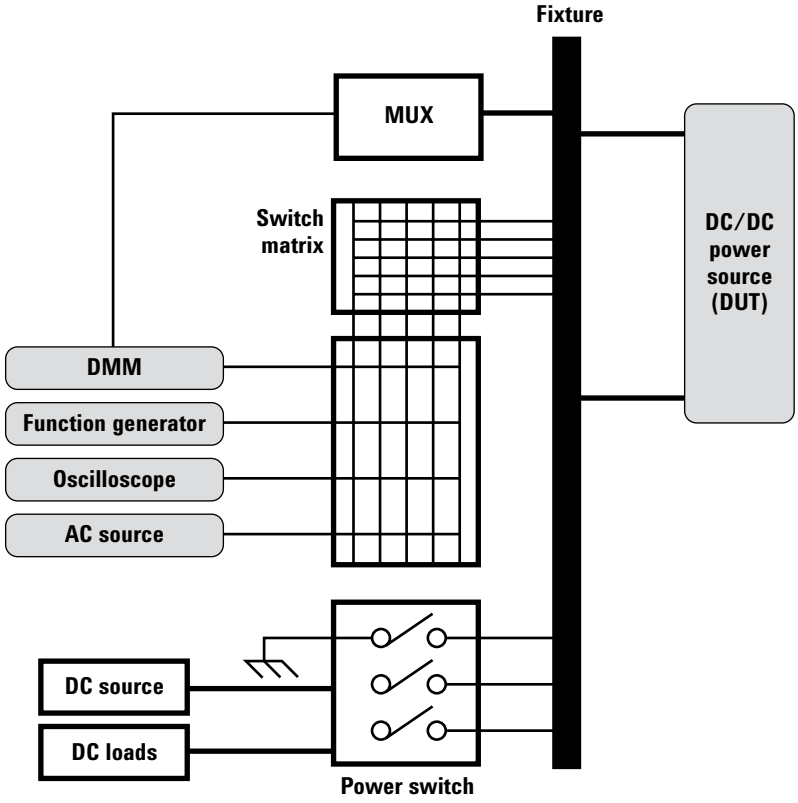
Automated control

The move to a fully automated test system may require additional instruments. The computer now controls all of the instruments as well as the reconfiguration of the interconnections for various tests. The digital multimeter, scope, and loads are still used, but now switches are employed to connect the DUT to the instruments. As the tests are performed, the computer uses the switches as required.

The block diagram in Figure 1.5 includes connections to the DUT and measurements that test the power supply in the manufacturing environment. The number of tests performed may approach those conducted during R&D and design validation but they are normally not as thorough. Manufacturing tests are often performed only at one operating point that is considered to be a worst-case condition. This maximizes the amount of information gained about the DUT in the minimum time.

The speed, repeatability, and reliability of the fully automated system can be significantly better than that of other test system control methods. Also, the skill level of the operator can be less. But the time and expense to create the system and make any changes usually makes automated test systems only feasible for manufacturing uses.

Figure 1.5. Block diagram of a fully automated test system.



2. Computer I/O Considerations

Introduction

Whether you plan to use your rack-and-stack test system for R&D, design validation or manufacturing, you are likely to program and control your system with a PC. For decades, the IEEE-488 bus, commonly known as the general-purpose instrumentation bus (GPIB), has been the standard interface for connecting test instruments to computers and for providing programmable instrument control. GPIB is still a common and useful technology, but now other I/O options are available. This chapter explores the various I/O options and helps you decide which interfaces make the most sense for your test system.

Proprietary I/O versus industry-standard I/O

Most of today's PCs offer built-in Ethernet-based local area network (LAN)and Universal Serial Bus (USB)connections. These industry-standard PC I/O technologies are much faster than previous PC I/O technologies such as RS-232, and therefore are much more suitable for automating and controlling test-and-measurement instruments. IEEE 1394, or FireWire interfaces, while not as ubiquitous as LAN and USB ports on today's computers, also are readily available.

Using these industry-standard interfaces for communicating with your test instruments can save you time and money and reduce headaches as you build your test system. Some benefits of using industry-standard I/O are immediate and obvious. For example, with USB, you don't have the additional expense of purchasing an I/O card, and you don't have to dismantle your PC to install the card. The LXI standard has been adopted by most instrumentation companies, facilitating the widespread use of LAN-based instruments.

There are other less obvious advantages to industry-standard I/O as well. Because the computer industry employs thousands of engineers who work on improving the throughput rate and data integrity of these interfaces, they are likely to continue to improve more rapidly than proprietary interfaces. Using industry-standard I/O also makes it easy to interchange instruments in your system with instruments from a variety of manufacturers.

Proprietary interface cards, such as MXI and MXI-Express from National Instruments are expensive, with typical price tags starting about US$1,000. You have to open up your PC housing to install them. And if you don't have an open expansion slot, you need to consider replacing your computer.

Because of the inherent advantages of industry-standard I/O and customer demand for it, instrument manufacturers are now providing LAN and USB interfaces to their test equipment. For example, the Agilent 33220A arbitrary waveform/function generator, introduced in early 2003, includes LAN, USB and GPIB interfaces. With the widespread adoption of LXI, most new instruments are likely to have a LAN interface.

If you want to use your existing GPIB instruments in a rack-and-stack test system, you don't necessarily need to use GPIB as your interface. Agilent also offers converters—USB/GPIB and LAN/GPIB—that allow you to use your GPIB-equipped test instruments with USB- or LAN-equipped PCs, eliminating the need to install a GPIB card in your PC. National Instruments also offers a FireWire/ GPIB converter. The next chapter looks at GPIB and the two main industry-standard interfaces, LAN and USB, and explores the applications where each is most appropriate. (FireWire interfaces are used primarily for VXI test systems. You will find more information about VXI in Chapter 5, Choosing your Test-System Architecture and Instrumentation.

GPIB interfaces

GPIB is the most common interface for programmable test-and-measurement equipment. It is still one of the best choices if you want to maximize throughput for a variety of block sizes. GPIB is a parallel bus that includes control lines, handshake lines, and 8 bi-directional data lines—specifically designed for instrument communications and control. GPIB supports up to 14 devices that can be connected to your PC. You can use either a star or a daisy-chain configuration for connecting multiple instruments (see Figure 2.1), but cable length is limited to 2 meters (times the number of devices) up to a maximum length of 20 meters.

You can achieve data transfer rates of more than 500 KB/s on a GPIB bus if you limit bus cable length to 1 meter (times the total number of devices), up to a maximum length of 15 meters. Longer cable lengths reduce the maximum data transfer rate to less than 500 KB/s.

When you use GPIB, configuring the instrument I/O bus is a relatively easy task. However, each instrument on the bus needs to have a unique address. This requirement means you may have to manually change an instrument's address when you configure your system.

GPIB has other drawbacks, too. GPIB cables and connectors are rather large, bulky, and relatively expensive. And because GPIB isn't a standard built-in PC interface, you have to open your PC housing and install an interface card in one of your PC's expansion slots.

To communicate with instruments over GPIB, you need to install an I/O software package. Plug and Play drivers, IVI-COM drivers, and VISA (Virtual Instrument Software Architecture) are examples. These packages support popular languages such as C and C++, Microsoft Visual Basic 6.0, Visual Basic .NET, MATLAB, Agilent VEE, LabVIEW, and others.

USB interfaces

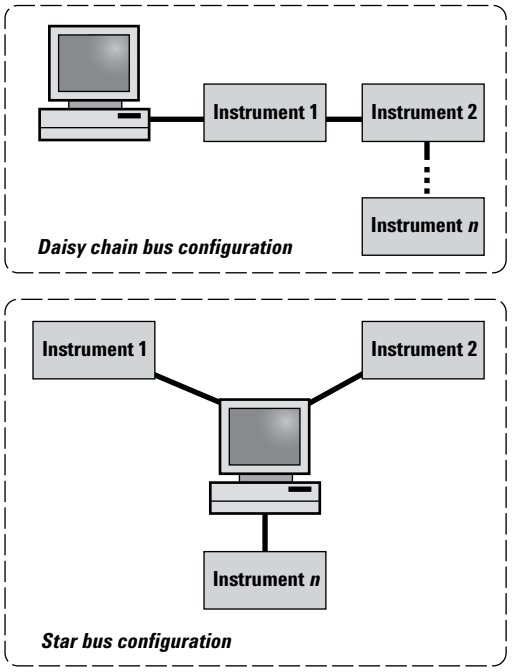
USB was originally intended as an alternative to the RS-232 serial interface and the Centronics parallel interface, an older standard I/O interface for connecting printers and certain other devices to computers. USB is suitable for a range of computer peripherals, from slow devices, such as mice and keyboards, to high-performance devices such as scanners, printers, and cameras. Now, USB is finding its way into test-and-measurement instrumentation, too.

USB is a serial interface bus that includes two power wires and a twisted pair to carry data. USB is capable of data transfer rates of about 12 Mb/s for v1.1, and up to 480 Mb/s for v2.0. In addition, v2.0 is fully backward-compatible with v1.1. The main difference is the data transfer rate.

USB is capable of supporting up to 127 devices on a given interface. If you use a GPIB-based system, you must ensure that instrument addresses are unique, but USB provides this function automatically. When USB devices are manufactured, they are given unique identifiers based on the manufacturer, the instrument serial number, and the product number. When the device is powered up and connected to a controller, the controller detects its presence automatically, and if the host-side software drivers are loaded, the instrument will be ready to communicate on the bus. USB devices also are "hot swappable," which means you don't have to shut down your PC to plug in or unplug an instrument.

With USB, the computer schedules and initiates all transactions. If you are using a Windows NT® operating system, you will find that it does not support USB connections.

Figure 2.1. You can configure a GPIB bus in either a daisy-chain or star topology, or you can intermix these two configurations.



Configuring USB systems

USB cables and connectors are considerably smaller than their GPIB counterparts. However, device-interconnect configurations for USB are somewhat different from those usually seen in GPIB-based systems. Most USB instruments are equipped with a single USB connector, so you cannot daisy-chain multiple devices together. Instead, you need to use a hub to connect the devices to your computer, as shown in Figure 2.2. Not all test-and-measurement USB drivers are designed to work with hubs, so it is a good idea to check with the manufacturer.

Hubs provide expansion capability for USB, permitting multiple devices to be connected to a single USB port. These hubs are transparent to a controller, and you can cascade them up to five deep. Using hubs in your system offers several advantages. For example, many USB hubs include LED status lights that indicate which port is connected. Also, a hardware failure at the interface to one instrument, such as a shorted line, is unlikely to cause an entire bus to fail. This makes troubleshooting an I/O interface fault in a large system with many instruments a much easier task than having to disconnect each device in turn, as required in a GPIB-based system.

Making USB connections

Connecting USB instruments to a PC controller is also a simple task. USB is especially useful with laptops, since typically they do not have the PCI slots required to install GPIB PCI cards. Virtually every PC produced within the last few years has several USB ports already built in.

As with GPIB, communications with instruments via USB requires the installation of an I/O software package. Plug and Play drivers, IVI-COM drivers, VISA, and IntuiLink software—supporting C/C++, Visual Basic 6.0 and Visual Basic.NET—are available with USB support.

LAN interfaces

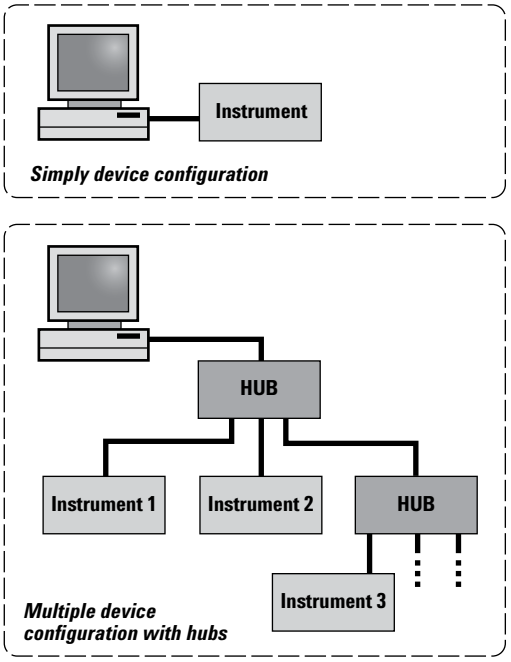
You also can connect your test-and-measurement instruments to a PC via a LAN interface. Ethernet LANs are almost universally available at industrial and commercial sites, and most PCs found in these facilities are already connected to a LAN. With the introduction of the LXI standard, Ethernet-based LAN interfaces for test equipment are becoming even more common than USB connections. Ethernet-based LANs commonly support data rates of 100 Mb/s to 1Gb/s.

USB and LAN interfaces share a number of features. They both operate in serial mode, and both use relatively small and inexpensive cables and connectors (especially when you compare the connector costs to those of GPIB).

You will want an Ethernet switch or router to interconnect multiple LAN instruments in a test system. Ethernet switches are readily available today—and are relatively inexpensive. Most provide network status, or activity indication with a series of LEDs.

Ethernet-based LAN devices typically need to be configured to operate properly on a network. However, instruments that support Dynamic Host Configuration Protocol (DHCP) provide the capability for test instruments to configure themselves automatically to operate on a network—if these services are available on the network. To simplify the configuration task, LXI instruments are required to support DHCP.

Figure 2.2. USB configurations with a single device and with multiple devices connected through a huband with multiple devices connected through a hub



Connection methods

You can connect LAN-enabled instruments several different ways. They may be connected directly to a site LAN (a workgroup LAN, intranet, or enterprise LAN), or they may be connected to a private LAN.

In private-LAN configurations, your PC and your test instruments are connected to each other via a LAN, but they are not connected to a site LAN. The simplest private-LAN configuration consists of a controller and only one instrument. See the first illustration in Figure 2.3. You also can connect multiple instruments in a private LAN, as shown in the second illustration in Figure 2.3.

If you plan to use your site LAN, rather than a private LAN, you need to be aware of two potential drawbacks:

1. Traffic on your site LAN can slow down your measurements.

2. If you are using a LAN interface for controlling your test system, it is possible that a faulty instrument could damage or disrupt the network, particularly when the instrument is turned on and tested for the first time. Controlling your test instruments via a private LAN is the safest approach, since it limits the range of potential disruption and access and maximizes performance.

For all setups, you can connect instruments to the LAN either with a conventional LAN cable or through a wireless adapter. Wireless routers and hubs also are available, as are wireless USB-to-LAN interfaces. See *Application Note 1909-3, Creating a Wireless LAN Connection to a Measurement System*.

Remote access

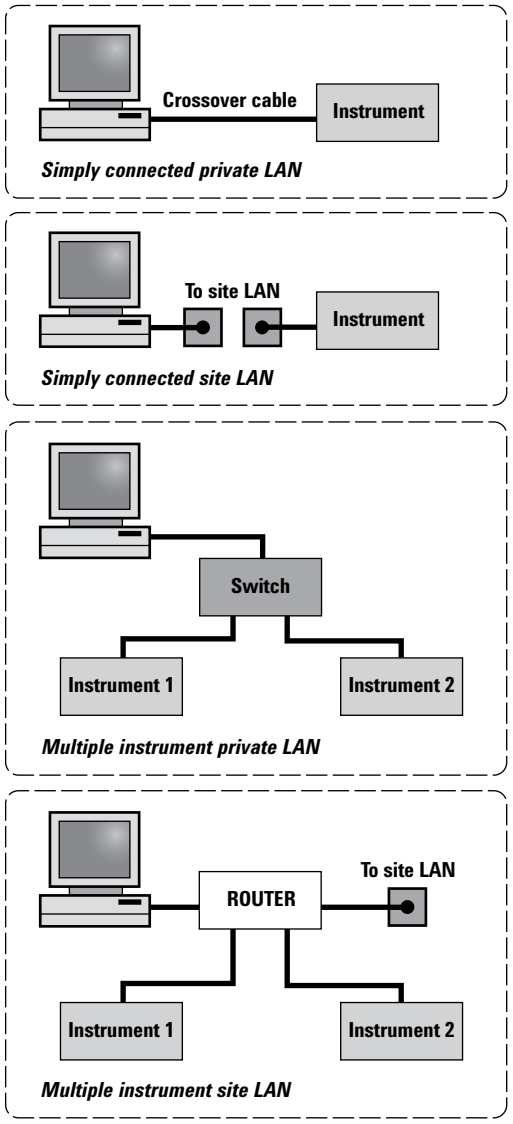
A site LAN has the potential for permitting any controller on the LAN to access instrumentation—either intentionally or unintentionally. If the site LAN can be accessed from physical locations outside of your facility, then others can access your instrumentation. This open access can be a valuable asset because it lets you remotely control instruments and systems almost as easily as if they were next door. You can use remote access capability to diagnose system and instrument faults from faraway locations. Multiple engineers can share the expensive test instruments and systems from remote locations.

However, this open access also can be a disadvantage. For example, if the site LAN is connected to the outside world to provide Internet access, you face a serious risk of exposure to undesired system accesses. Firewall software and/or using a router which requires specific device addressing rather than a switch or hub can provide protection.

If you want remote access to your test equipment, but security and controlled access are a system requirement, then you need a lockout feature. Some instruments, such as the 33220A function/arbitrary waveform generator, provide this feature via an Allow List. An Allow List is a list of remote LAN addresses that are permitted to communicate with the instrument. Any controller that attempts to access an instrument whose address is not on the Allow List is rejected. This feature provides a level of system security for those instances where your system is connected to a site LAN and is at risk for inadvertent access.

You can also use a virtual private network (VPN) for secure, remote access.

Figure 2.3. Single and multiple instrument configurations can be connected to private LANs and site LANs.



Instrument communication and operation over LAN

Instrument communication over an Ethernet-based LAN requires a software driver package if I/O is to be performed via Plug-and-Play, IVI-COM or VISA. It's also possible to use the TCP/IP's sockets or telnet to perform instrument I/O directly without a host-side driver. In fact, I/O operations using sockets provide the fastest data transfer rates, since the host-side driver is bypassed.

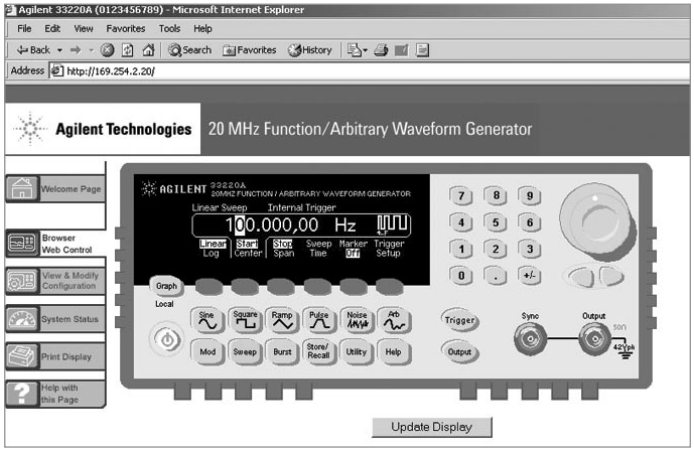
You can operate some LAN-enabled test instruments via a virtual front panel that appears on your PC screen. Typically, the display looks and acts like the actual instrument itself (see Figure 2.4), and you use your mouse to actuate buttons as if you were actually pushing front-panel buttons. The virtual instrument display mimics that of the actual instrument that may be thousands of miles away. Agilent LXI instruments allow both monitoring and control of instruments from your web browser.

Which I/O interface should you use?

To decide which I/O interface or interfaces you use in your test system, you will need to consider many factors. These include data transfer rates and block sizes, and costs for cables, routers, hubs, and PC I/O cards. Other factors include I/O driver availability, and programming requirements, as well as the need for possible remote system access.

Keep in mind that you do not have to choose a single I/O interface. Systems incorporating multiple interfaces are particularly useful if you have a mixture of older GPIB instruments and newer instruments with other interfaces built in. Today's advanced software tools that include VISA technology eliminate the need to talk to different kinds of I/O in different ways. A minor change to a single line of code is typically all that is required. However, do not mix interfaces on a single instrument—the input and output must be on a single interface—and make sure your software drivers know which instrument is using which interface.

Figure 2.4. Virtual front panel of the 33220A multifunction switch/measure unit



Gating factors on data rates

The data rates of a test system are determined by the slowest device/firmware/software in the system.

For example:

1. A high-speed instrument with integrated LAN controlled with an older computer will be limited by the computer processor speed and possibly memory depth.
2. A USB2 interconnect will operate at a USB1 rate if the instrument, USB hub and computer do not also support USB2.
3. An Instrument with a data transfer rate of 33K bytes/second will not transfer data any faster with USB, LAN or a computer that is able to transfer data at 1M bytes/second

To see an example system that incorporates multiple interfaces (RS-232, FireWire, USB, GPIB and LAN), see Chapter 5, Choosing Your Test-System Hardware Architecture and Instrumentation.

Real data rates

You will notice that individual I/O bus specifications for data transfer rates usually give only the theoretical maximum transfer rate. The actual data transfer rate that can be achieved for any given system depends on a number of factors. These factors include PC microprocessor speed, PC software and driver overhead, I/O card hardware, and instrument-specific hardware and firmware.

These variables make it difficult to predict the actual data transfer rate that might be expected for any given system configuration. Table 2.1 shows a relative comparison of data transfer rates for several data block sizes among GPIB, USB v1.1, USB v2.0, and LAN interfaces. These data were compiled using the Agilent Model 33220A function/arbitrary waveform generator and a Hewlett-Packard Kayak PC with an 800 MHz processor running on a Windows XP operating system.

For small data-block sizes of a few hundred bytes, there is no appreciable difference in bus speed, but the higher-performance buses (USB v2.0 and LAN) demonstrate a marked improvement in the time required to transfer large data blocks.

The differences in data transfer rates between small and large data blocks for any given interface are largely due to variations in the latency, or software overhead, required for each of the interfaces prior to the start of the actual data transfer.

If you're looking for high throughput in a test system, don't be swayed by the perception that high-speed interfaces will always get you there. In most test systems, the use model is one of "Close a channel; measure a point," then "Close another channel; measure another point." In this case, block transfer rate is meaningless.

The time to close the channel and make the measurement dominates the total time. GPIB's strong performance in this use model is one of the reasons it has lasted so long as an interface.

For a detailed look at data transfer rates of two different block sizes over the various interfaces, see *Application Note 1475-1, Modern Connectivity—Using USB and LAN Converters*. This application note compares the Agilent 82350B GPIB PC card, the 82357A USB/GPIB converter, and the E5810A LAN/GPIB gateway in terms of controller and operating system requirements, set-up steps, data transfer rates, allowable distances from instruments to the PC, etc. These details will help you choose the best interconnection method for your application. One of the benefits of having an instrument that supports multiple interfaces is the ability to easily compare the actual data transfer rate for each of the I/O interfaces in a given application. This permits you to select the interface that offers the optimum performance.

If the application program's I/O calls are written with a driver interface that provides a common set of programming commands independent of the interface, such as Agilent's VISACom, then it becomes a simple matter to direct the I/O calls to any of the three interfaces.

Table 2.1. Relative I/O times from a PC to an Agilent 33220A (data taken with a 1-meter cable on an HP Kayak XU800 with an 800 MHz processor running Windows XP)

Interface	Function change	Frequency change	4K arb	64K arb
LAN (socket)	100 ms	3 ms	8 ms	110 ms
USB 1.1	100 ms	4 ms	10 ms	185 ms
USB 2.0	99 ms	3 ms	8 ms	100 ms
GPIB	99 ms	2 ms	20 ms	340 ms

Comparing costs

Today, many companies are looking for ways to lower the cost of test. If this is true of your organization, implementation cost will be an important consideration in selecting an I/O interface for your test system.

New PCs typically have a LAN and several USB ports built in, but GPIB interfaces usually require a card that you must purchase separately. GPIB cards typically cost about US$500 and additional USB or LAN cards usually sell for US$10 to US$50.

Also, if you plan to use USB or LAN interfaces to connect multiple instruments in your system, you will need switches or hubs. These hubs can cost from US$25 to US$200 each, depending on features and the number of ports they support.

You also need to consider the cost of the cables for your test system. GPIB cables are relatively expensive, ranging in price from US$60 to US$150 each, depending on their length. USB cables, on the other hand, range from US$8 to US$30. LAN cables are usually the least expensive, typically costing less than US$10. Some can be found for as low as US$3.

You can make useful cost comparisons by assuming that all test instruments are able to support any of the three interfaces and computing the interface cost for your proposed test system. Today, few test instruments actually do support all three, since the industry is just beginning to provide instruments equipped with multiple computer-industry-standard interfaces. However, the I/O interface converters mentioned earlier permit GPIB-only instruments to be connected to USB- and LAN-based interfaces. For example, the Agilent 82357B USB/GPIB interface enables your PC to communicate with GPIB devices via the PC's USB port.

Similarly, the Agilent E5810 LAN/GPIB gateway provides a means to connect GPIB devices to a LAN (see Figure 2.5.) These converters can save you the cost of replacing your existing GPIB test instruments if you decide you want to use industry-standard I/O. However, these converters are appropriate only for applications where measurement speed is not critical, as they do slow the data transfer rate.

Let's look at an example of a test system designed to test the Agilent 33220A function/arbitrary waveform generator. The test system consists of a controller, a local printer, seven rack-and-stack instruments, a fully loaded 13-slot VXI mainframe, and support for testing three 33220A waveform generators.

As Table 2.2 shows, GPIB is the most expensive scheme to implement. Even with the added costs of USB and LAN hubs, their reduced cable costs and higher overall speed performance makes them more attractive alternatives.

From a systems perspective, hubs and switches also offer some I/O interface operational feedback that is lacking with GPIB systems. Also, the much smaller USB and LAN cables and connectors take up much less rack space, making system cabling easier. They also weigh less.

Figure 2.5. The Agilent E5810A LAN/GPIB Gateway and the 82357B USB/GPIB Converter.

Ease of implementation

USB is the simplest I/O to implement, and GPIB is also relatively straightforward, as long as you don't mind the hassle of opening your PC and installing an interface card. Since LAN has become common in home broadband applications, configuration is becoming much easier, but remains the most difficult of the three interfaces to implement. However, for many system developers, the advantages of LAN far outweigh the added development time required. Evaluate your own situation to decide if that is true for you.

Table 2.2. Typical costs for LAN, GPIB and USB interfaces

Interface	Single instrument		12-instrument system	
LAN	PCI card + cable	$30	PCI card + cables + 16-port switch	$300
USB	PCI card + cable	$60	PCI card + cables + 2 hubs	$225
GPIB	PCI card + cable	$600	PCI card + cables	$1600

Conclusion

With the new generation of test instruments offering a choice of interfaces, you need to decide which interface is best suited for your test system. Comparing costs, data transfer rates and ease of implementation will help you choose the interface most appropriate for your application (see Table 2.3). For R&D applications, where the number of instruments in a system is usually small and a quick and easy interface set-up is desired, USB is usually the best choice.

For design verification and manufacturing, USB and Ethernet-based LAN are good choices, although LAN is typically the better of the two alternatives for larger systems because of its data-throughput performance, cost, remote access, and ease of system assembly.

The added flexibility, remote system access and control, performance on a par with USB, captive cable connectors (which aren't found on USB), and the capability for wireless operation offered by the LAN approach can make LAN the most attractive choice for many systems applications.

Table 2.3. Advantages and disadvantages of GPIB, USB, and LAN interfaces

Interface	Advantages	Disadvantages
LAN	• Good data-throughput performance • Low cost • Remote access makes it easy to control system from remote location	• Requires LAN knowledge to set up
USB	• Quick, easy setup • Low cost • Good data-throughput performance	• Does not work with Windows NT
GPIB	• Ubiquitous interface on test instruments • Maximizes throughput for all block sizes	• PC expansion slot required • Must open PC housing to install card • Relatively expensive • Limited cable lengths permitted between computer and instruments

Get help configuring your I/O interfaces

Configuring an interface to connect your PC to an instrument or system can be a daunting task for someone who is not well versed in the intricacies of PCs, I/O technologies, and I/O inter-face configuration. In the past, this was especially so for LAN-based I/O that required a system to be connected to a site LAN. Fortunately, step-by-step guides such as Agilent's *USB/LAN/GPIB Interface Connectivity Guide* are now available to help you to configure your I/O interfaces. The *Connectivity Guide* describes in detail how to connect instruments to various interfaces, and how to configure your PC. It also includes programming examples and interface trouble-shooting tips. You can view the guide at **http://cp.literature.agilent.com/litweb/pdf/E2094-90009.pdf**

3. Understanding Drivers and Direct I/O

Introduction

This chapter answers common questions about the use of drivers and direct I/O to send commands from a PC application to the test instrument. It discusses how the driver came about, what the different software layers do in a system to help the instrument communicate to the PC, which drivers are compatible with various software languages and I/O software, and references for further study.

For the purposes of this discussion, a *driver* is a piece of software intended to simplify programming and accelerate test-system development by facilitating communication with an instrument. In contrast, *direct I/O* involves embedding specific instrument commands (typically called *SCPI commands*) in your test software and managing all of the input/output communication yourself.

Even if you have never programmed an instrument in a test system, you have used drivers on your PC. Digital cameras, printers and other peripherals all require a driver to talk to a PC. Moreover, if you've ever upgraded a PC, you may have found that the old printer driver no longer worked with the new operating system, and you need to go online to find a new one. Or you may have found that the printer didn't work exactly the same way it did under the old operating system. Similar issues exist with test and measurement equipment.

In a September 2001 survey, *Test & Measurement World* published a summary of engineers' worst headaches. Instrument drivers topped the list. Instrument manufacturers and various trade groups have been working on driver standards for some time, in an attempt to alleviate the frustrations of engineers who need to automate measurements and create test systems on a deadline. As a result of these efforts, we might expect finding and using appropriate drivers to be dramatically easier, but at the moment, complexities and incompatibilities are still troublesome.

This chapter answers common questions about the use of drivers and direct I/O to send commands from a PC application to the test instrument. It discusses how the driver came about, what the different software layers do in a system to help the instrument communicate to the PC, which drivers are compatible with various software languages and I/O software, and references for further study.

With new insight into these topics, you should be able to choose, install and use drivers more easily and reduce the amount of time you spend getting your instruments and computer applications to talk to each other.

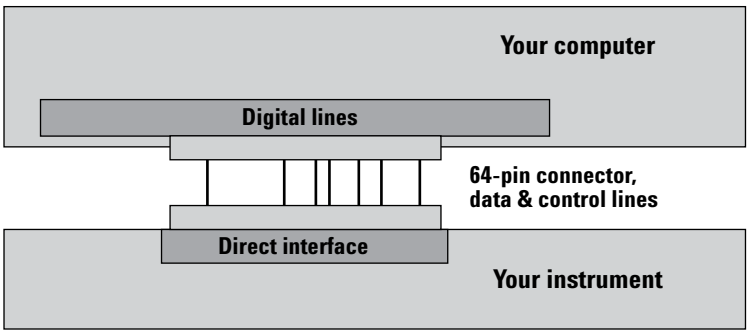
History

The history of automated measurements dates back to at least 1970, when instruments began to be connected via imaginative schemes to devices resembling computers. One popular I/O format involved connecting a large cable to the instrument (Figure 3.1). Each line on the cable represented a function or range, and the line was simply grounded at the proper time. The device, say a voltmeter, would return a value using binary coded decimal (BCD) 1-2-4-8 format, or a quainter 1-2-2-4 format. Needless to say, the programming syntax of instruments at this time was anything but standardized. However, since everything was hardwired, the process was straightforward and immediate.

GPIB

In 1971, development began on a standard hardware interface. The idea was to be able to trigger multiple instruments at once and still allow both slow and fast instruments to "talk" on the same bus without

Figure 3.1. Early instrument control utilized hard-wired commands.



conflict. The first products to use this bus were released in 1972. The same year this new bus was dubbed Hewlett-Packard Interface Bus (HP-IB). In 1975, IEEE adopted it as a standard with little modification, and IEEE-488 was born. A variant of the original interface is now popularly known as General Purpose Interface Bus (GPIB).

With GPIB and a desktop computer (actually at the time it was called a "desktop calculator"), the need arose for a common communication language. Limited processing power in the 'calculators' demanded a simple syntax, so ASCII commands were chosen. A DMM might be sent what was affectionately termed "R2D2 code." Here's an example:

```
"F1R2T1"
```

The command means "Go to the dc volts Function, the 1 volt Range and Trigger a reading." Different manufacturers had unique ways to interpret the command strings, based on their instruments' capabilities. If you had to replace a product with one from another manufacturer, or even a new-generation product from the same manufacturer, it could mean completely rewriting the entire program. Later versions of IEEE 488 elevated the standard from being a hardware-only standard to one that also specified protocol.

SCPI

In 1989, seeing a need for more clarity and interchangeability that was available with simple ASCII, Hewlett-Packard introduced a programming language known as Test & Measurement Systems Language (TMSL). Within less than a year, nine T&M manufacturers had met to generate a universal approach to instrument control, using TMSL as the basis. The outcome was Standard Commands for Programmable Instruments (SCPI) (Figure 3.2).

Today, SCPI is still the most-used form of instrument control. In SCPI, the instrument programming syntax became much more robust and predictable. SCPI defined a strict hierarchy, and every command was associated with a concomitant response. These were defined for source, sense and switch devices. Here's an example of SCPI code:

```
CONF:VOLT:DC 0.3,0.003
```

This command tells the instrument to configure itself to get ready to read a 0.3 volt dc signal with 3-millivolt resolution. It should be obvious from this statement that SCPI commands require some intelligence on the other end of the wire, as not every voltmeter has a 0.3 V range. The commands need to be parsed by the voltmeter and this parsing adds a small layer of delay time to the system.

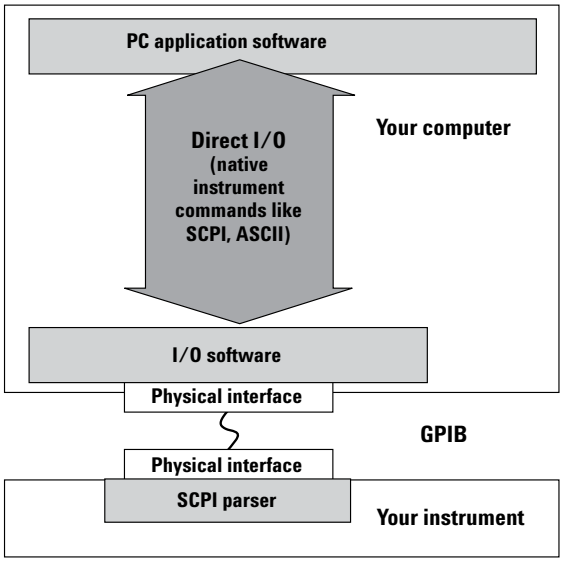
One advantage of SCPI is that the list of commands typically covers 100 percent of the instrument's programmable functions, no matter how arcane. For a friendly tutorial on SCPI, go to **ftp://ftp.agilent.com/pub/mpusup/pc/iop/hpibtut/ib5_scp.html.**

The I/O software: SICL and VISA

Instrument commands aren't the whole story. It takes more "layers" of software to communicate with a computer. Before you send the instrument a command, you need to define the I/O path, route the information through the proper I/O card, find out where the instrument is on the bus and speak to the instrument in the syntax of the I/O you're using. Assuming the GPIB I/O card in the computer is at address 7 and the DMM is at address 22 on the bus, the simple BASIC command might be:

```
ASSIGN @Dvm to 722 !
```

Figure 3.2. Compared to "R2D2" code, SCPI commands standardize programming and make life easier for the programmer. SCPI commands can access virtually any programming function in the instrument, but the parser does add small delays to the process.



This tells the computer where to send the command.

```
OUTPUT @Dvm;
"TRIG:SOURCE:INT" !
```

This sets the trigger source to internal.

The above will work with a GPIB interface, but if you try the same thing using RS-232, the syntax is very different. Switching between GPIB and RS-232 would require rewriting some code.

SICL

Standard Instrument Control Library (SICL) I/O software was subsequently developed to address the challenges of updating or reusing code. SICL was developed by HP to make software as I/O-independent as possible. It adds a layer on top of the instrument code; this layer checks to see what I/O is used and alters the syntax accordingly (Figure 3.3). The code looks the same, regardless of I/O type. All you have to do is use one line of code to declare the I/O type at the beginning of the program.

SICL is not the only I/O software available today. AGILENT VISA, NI-VISA and NI-488 and VISA-COM (from Agilent) perform similar functions. That's a dizzying array of choices, so for now let's concentrate on VISA. While SICL software was created to communicate with Agilent interfaces only, VISA was created to work industry-wide and is now the preferred programming interface.

VISA

In the late 1980's, there was a move to build standardized card cage instruments. This movement led to a software and hardware standard known as VME Extensions for Instrumentation (VXI). Based on the VME standard, VXI made special modifications for software, shielding, triggering, power supplies and analog performance. VXI was adopted by hundreds of instrument manufacturers who produced a wide variety of plug-in cards. VXI's interchangeability at the card level brought about the need for common I/O software, similar to HP's SICL, but implemented as an industry-wide standard. Largely derived from the SICL library, the VISA syntax was born.

Virtual Instrument Software Architecture (VISA) was created by the VXI*plug&play* Foundation to standardize I/O software across physical interfaces and between various vendors (Figure 3.4). In most cases, test systems are not solely VXI, but rather hybrids of VXI and "rack and stack" architectures, so it was not enough to create I/O software exclusively for VXI. For that reason, the VXI*plug&play* specifications were extended to include traditional standalone instruments as well as both types[1] of VXI instruments.

1 VXI has two types of instruments, distinguished mostly by their local intelligence. "Message-based" cards" can react to a high-level message, and usually have on-card parsing. "Register-based" cards are just what the name implies: cards that have directly programmable registers. Message-based cards can do more, but are inherently slower, since they must interpret complex commands.

Figure 3.3. SICL I/O software reduces a test engineer's programming burden by making it easier to change I/O types (USB, LAN, GPIB, USB, VXI, RS-232, etc) without recoding the program. SICL adds a software layer, which has a small effect on system speed.

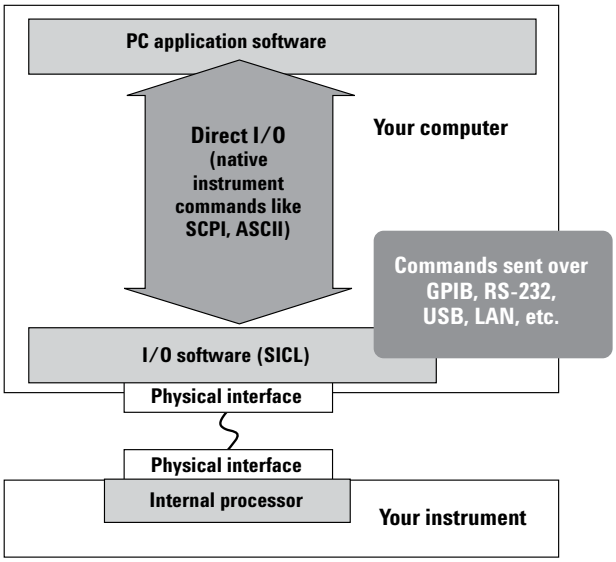
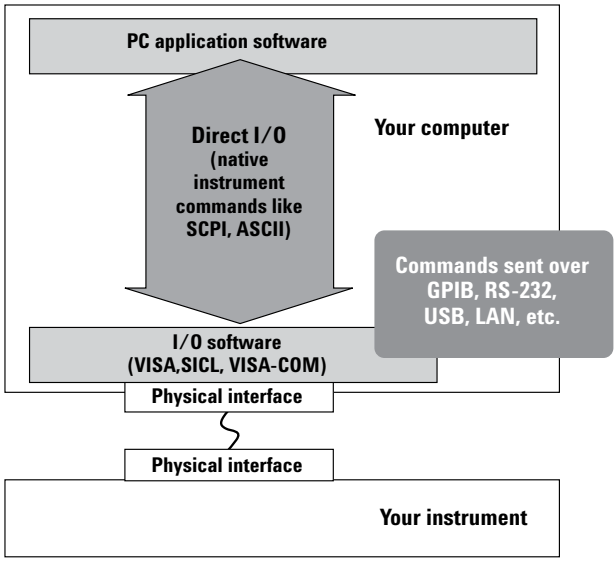


Figure 3.4. VISA is the most popular form of I/O software. Drawing heavily on the work done for SICL, VISA was created to serve multiple T&M suppliers and be a universal standard. VISA-COM is a new variant of VISA.



Today's two main suppliers of VISA are Agilent Technologies and National Instruments. (In 1999, the engineers from HP Test & Measurement who were involved in instrumentation were split from HP in the new venture now known as Agilent Technologies.)

VISA I/O software uses common terminology and syntax to connect to and control instruments. A VISA library supports complete control of instrument across the physical interfaces GPIB, RS-232, USB, LAN and VXI.

The VISA library provides the capability of SICL, in a way that conforms to industry standards. A program written to work with Agilent's VISA library will work with implementations of VISA from other vendors. For those accustomed to using SICL, Agilent's implementation of VISA is provided along with its SICL libraries. (Since the introduction of VISA, programming based on the SICL library has gradually been phased out in favor of the industry-standard VISA library.)

To program a new test system, the test engineer installs the appropriate I/O library along with the application programming language. VISA was originally developed to be used with C and C++, but can also be called from any language that can call arbitrary Windows dynamic-link libraries (DLLs), including Microsoft® Visual Basic. Agilent provides header files to facilitate the use of VISA in Visual Basic.NET and C#. These can be downloaded from **http://www.agilent.com/find/iolib**.

PC industry adds language independence

As I/O development was proceeding in the T&M industry, the PC industry was making big strides in I/O-independence and language-independence. In 1994, Microsoft stated: "The Component Object Model (COM) is a software architecture that allows components made by different software vendors to be combined into a variety of applications. COM defines a standard for component interoperability, is not dependent on any particular programming language, is available on multiple platforms, and is extensible."[2]

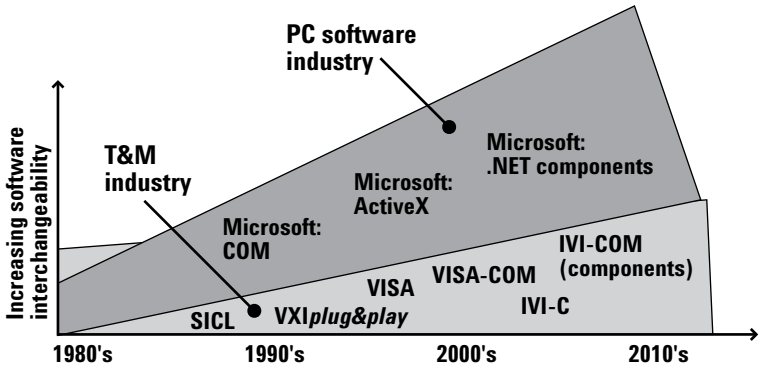
In February, 2001, Microsoft introduced .NET, its 3rd generation of component technology. .NET has been applied to Microsoft's integrated development environment, Visual Studio®.NET, as well as MS Office, other applications, operating systems and web services.

The benefits of these PC software technologies are compelling, but should the test and measurement industry embrace PC operating systems?

Detractors point out the frequent operating system upgrades in the PC industry relative to T&M languages. However, as Figure 3.5 indicates, COM—which is integral to .NET components—has been around longer than most T&M standards. It seems only logical to take advantage of the investments Microsoft has made to create this paradigm shift. With 3,000 engineers working for three years on the first version of .NET, Microsoft's investment is twenty times that of the leading T&M language. Similar correlations apply to software. Visual Basic has over 6,000,000 users, and C/Visual C++ has 1,000,000 users worldwide. This will result in an unprecedented body of software the average engineer will be able to leverage.

2 Dr. Dobb's Journal, Microsoft Corp. December, 1994.

Figure 3.5. PC Software Overtakes T&M Software in interchangeability. The millions of people using Visual Studio software will afford the engineer an unprecedented pool of available intellectual property.



VISA-COM

To incorporate this programming language independence, Agilent initiated a VISA-COM standard as a companion to the VISA specification. VISA-COM software makes VISA services available in a language-independent COM component architecture. As a result, you are free to pick from popular I/O configurations, but now you also have the freedom to choose from a list of software languages, including C++, C# and VB.NET.

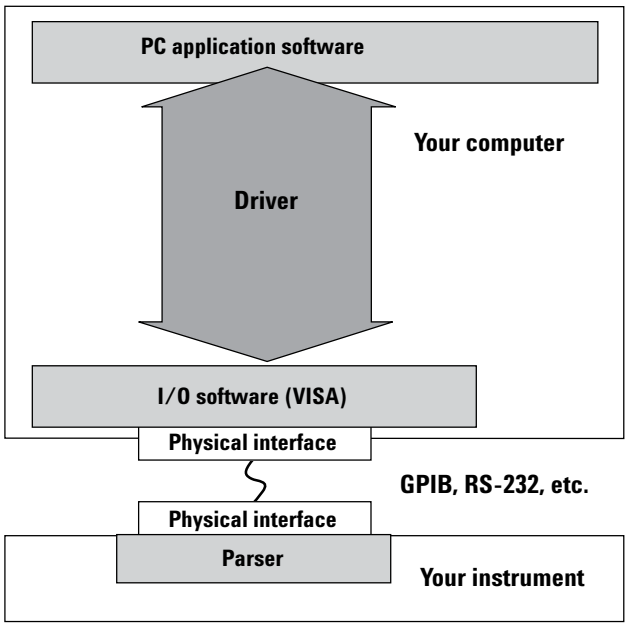
When using Agilent VISA-COM, you also need to install Agilent VISA. Agilent IO libraries are shipped along with Agilent instruments, software and I/O products.

Choosing and using instrument drivers

By managing both the overall communication between the PC and the instrument as well as all the details of command syntax and instrument functionality, drivers are clearly essential considerations in test-system development (Figure 3.6). Without drivers, you're forced to either memorize or look up the direct I/O SCPI commands related to the particular instrument being programmed. If you intend to code in a proprietary language, then you need to know how those commands fit. For simple applications, this approach works well, but as application complexity increases, using direct I/O can become difficult and time consuming. Programming a direct communication path usually requires you to know a specialized computer programming language and its programming environment and to be familiar with proper command sequences and interrelationships between commands. You also need to know how to load and configure various I/O libraries and parse instrument responses that may be in the form of binary data or screen graphics. Whether you have these competencies or not, when today's product design cycles are measured in months rather than years, it doesn't make sense to spend several of those months coding a new test system, unless very high volume production is the goal.

However, even will all these potential disadvantages, there are times when using direct I/O can be a better choice than using a driver; see "When should I use a driver?" on page 33.

Figure 3.6. The driver is, among other things, a programming aid that works between the PC application and the I/O software. It can save enormous amounts of development time and prevent coding mistakes.



Drivers come in many forms and offer various levels of functionality. A driver can be as simple as a list that pops up when you hit the next "dot" in Visual Basic, or it could be as elaborate as a "panel driver" that displays a virtual front panel on the screen of your computer to help you set up the instrument (Figures 3.7 and 3.8).

Driver coverage

A simple DMM may have only 25 commands, whereas a more complex instrument may have hundreds. You can imagine how expensive it is to write an intelligent driver that anticipates all the possible permutations of instrument setup, triggering, sourcing and measurement. And that's why you'll seldom see a driver that covers every command in the instrument.

Instrument manufacturers take their best guess at the commands you are likely to use and craft the driver accordingly. A typical IVI driver (see "IVI drivers") covers about 40-60 percent of the instrument's command list. This may sound like a small number, but consider this: Agilent surveyed customers who used the 3852A Data Acquisition/Switch Unit, a complex instrument with over 300 distinct commands. By poring over customers' code, we found they rarely used more than 5 percent of the available commands. This is an extreme case, but it tells you that 40-60 percent coverage is a good start. And even if a driver doesn't incorporate a particular command, there are ways to send commands directly yourself (Figure 3.9).

Figure 3.7. Agilent's T&M Programmers Toolkit using a VXIplug&play WIN32 power supply driver in VB.NET.

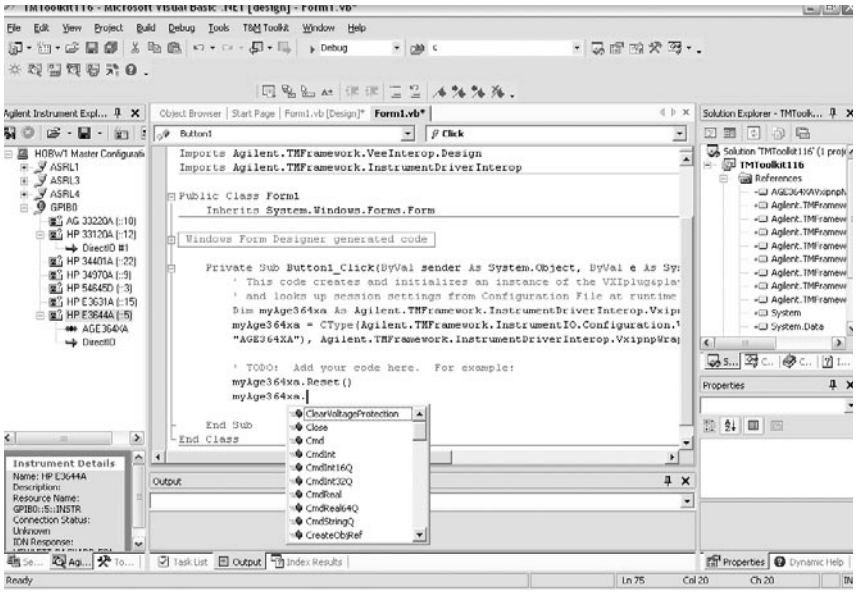


Figure 3.8. A tiny but interesting program, written in VEE. With its intuitive interface, VEE is the fastest T&M graphical language to learn. Fill in the boxes, and the VEE panel driver generates code for you. See **http://www.agilent.com/find/vee.**

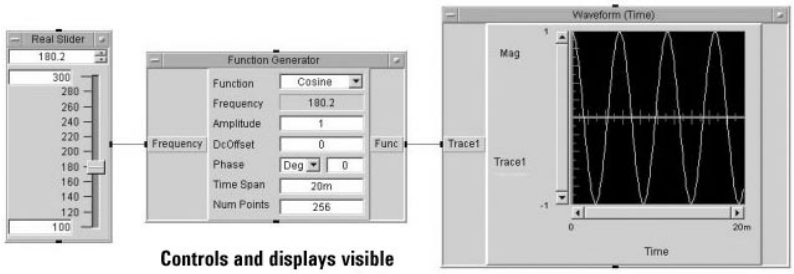
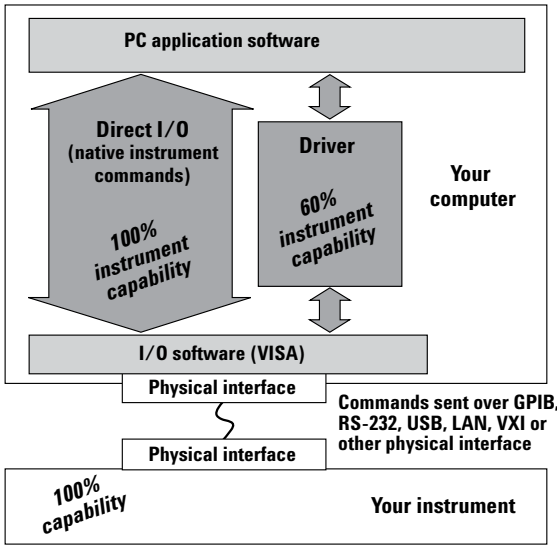


Figure 3.9. If you are using a driver and need to access instrument functions the driver doesn't have, you can send direct SCPI or ASCII commands, or go through the driver with pass-through commands to control the instrument directly. This gives you the convenience of drivers, with the 100 percent coverage of direct I/O. To avoid command conflicts, this technique requires in-depth knowledge on the part of the programmer.



Driver evolution

There are three basic generations of drivers: proprietary T&M drivers, traditional T&M drivers and component PC drivers (Figure 3.10). These represent the past, present, and future of driver technology. In the past, instrument drivers were custom-designed to function with a vendor's own application development environment (ADE). A considerable body of legacy application programs uses these proprietary drivers, but for new development, engineers today have better choices.

When you need to accelerate test system design and deployment, Agilent recommends the new IVI-COM driver and the VXI*plug&play* WIN32 driver for instrument control. IVI-COM is the only component PC driver built on the PC standard COM architecture; the IVI-COM standard is being led by Agilent and other instrument companies. A component driver built on COM works in all popular PC languages and most T&M languages, uses the most popular types of I/O can be used in the latest .NET technologies and is backward-compatible.

When should I use a driver?

Use an instrument driver if

- A driver is available that works with your development environment and I/O software, and supports the majority of instrument features you want to use.
- You want easy access to commonly used instrument functions because the instrument commands are typically organized in a hierarchical structure.
- You want to simplify the process of developing and maintaining your code over time, because there is a single point of interface to update or change.
- Software interchangeability is important to you.
- You need to simplify maintaining the system when instruments need to be exchanged.

Use direct I/O if

- You have instrument programming experience or access to programming experts.
- You need to use instrument features not supported by the available drivers (the other 40–80 percent of the instrument capability).
- You need the absolute maximum in system throughput speed.
- You need to control the exact configuration of the instruments in your system.
- You have a large volume of legacy SCPI-based code.

Figure 3.10. The three generations of drivers represent varying degrees of language independence. IVI-COM is the newest and the one supporting the widest variety of software environments.

Instrument driver families					
Component PC (based on PC standards)	**Traditional T&M** (based on T&M standards)			**Proprietary T&M** (specific to one language)	
IVI-COM	**IVI-C (via NI)**	**WIN VXI*plug&play***	**LabWindows/ CVI Plug&Play**	**VEE Panel Drivers**	**LabVIEW Plug&Play (VXI*plug&play* GWN)**

IVI drivers

In 1998, test and measurement companies formed the Interchangeable Virtual Instrument (IVI) Foundation[3] to address the high cost of developing and maintaining test system software and the need to evolve technology more rapidly through the use of better drivers. The foundation comprises end-user test engineers, equipment manufacturers and system integrators with many years of experience building test systems.

IVI classes

The goal of hardware interchangeability led IVI to the concept of instrument classes. The idea is simple: If you use a spectrum analyzer, it certainly would save time if you could program every instrument in the spectrum analyzer class the same way, no matter who built it. Both the specification and any specific driver that implements it are called an IVI Class Driver (IVI-C Class or IVI-COM Class).

As of this writing, the IVI Foundation has defined the following instrument classes: DC Power Supply, Digital Multimeter (DMM), Function Generator/Arbitrary Waveform Generator, Oscilloscope, Power Meter, RF Signal Generator, Spectrum Analyzer and Switch. Others are under development.

This work makes it much simpler for the engineer to program instruments from separate suppliers whenever those instruments conform to a particular class.

3 For additional information, you can visit the IVI Foundation website at: **www.ivifoundation.org**.

Finding drivers and technical advice

Instrument vendors typically provide drivers on a CD with new products and offer their most up-to-date instrument drivers on their Web pages. For downloads or more information on Agilent drivers, I/O software, connectivity and application software, join us at the Agilent Developer Network: **www.agilent.com/find/adn**. Note that we do not post drivers written by other parties. Because you are at the mercy of whoever created the driver, it is a good idea to use a driver supplied by the same vendor who made the equipment.

Third-party software and systems integration companies that support the test-and-measurement industry can provide driver development tools and services. Two such companies are Pacific Mindworks (**www.pacificmindworks.com**) and Vektrex (**www.vektrex.com**).

For advice on mixing I/O hardware and I/O software from different suppliers, see **ftp://ftp.agilent.com/pub/mpusup/pc/binfiles/iop/m0101/readme/trouble/niinfo.htm.**

Conclusion

If the project you are pursuing is not complex, there are often situations where you don't even know you are using a driver. Indeed, the ultimate goal of T&M companies is to keep this process entirely transparent. In the meantime, if you do get embroiled with issues of driver selection, note there can be tradeoffs between speed of development and speed of execution. The industry is working through these issues by instituting faster I/O and software aids, such as tools to keep track of instrument states. The whole idea is to give you both fast programming and fast throughput.

If you choose to use a driver, computer industry-standard IVI COM drivers and a Visual Studio .NET-compliant development program such as the Agilent T&M Programmers Toolkit give you significant leverage. The T&M applications you develop will show significant hardware and software interchangeability, while being easily maintainable and extensible. Plus, the intellectual property you create during the development process will be widely transferable to other projects.

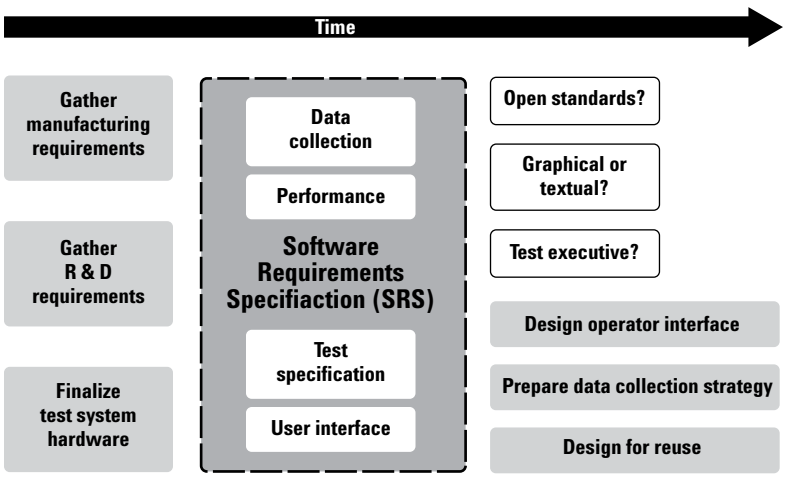
4. Choosing Your Test-System Software Architecture

Introduction

This chapter will help you understand the tools required to design, develop and deploy the software component of your test system (see Figure 4.1). The information presented here will help you choose the direction for your software based on the application you have in mind and the amount of experience you have. We will explore the entire software development process, from gathering and documenting software requirements through design reuse considerations.

- **Gathering and documenting software requirements.** Before gathering and documenting your software requirements, finalize your test system hardware design. Once finalized, start working with your R&D and manufacturing teams to collect the information you need to create software requirements specifications (SRS).
- **Programming and controlling your instruments.** The control of instruments is rapidly evolving from proprietary test and measurement standards to open, computer-based industry standards. This trend affects the hardware that connects the PC to the instrument as well as the software and drivers that control the instrument.
- **Collecting and storing test data.** Data collection is the science of obtaining, moving and formatting data. The integrity of your test system depends on obtaining the right data at the right time.
- **Designing the user interface.** One of the most important (and easily overlooked) aspects of test systems is the graphical user interface (GUI). This is what the test engineers, operators and technicians see when they interact with your software.
- **Choosing the development environment.** The software environment and tools you choose will have a significant impact on the overall cost of your test system. When choosing your software environment, consider more than just the purchase price of the software. Also, consider how easy it is to learn and use the software, how hard it is to connect to other languages, devices or enterprise applications, as well as support and maintenance costs. Over the life of a test system, software support and maintenance costs alone can exceed hardware costs.
- **Working with open standards.** Today, the industry trend is to move away from closed, proprietary development environments. More and more people are embracing open, industry-standard development environments as their platform of choice for test-system development projects. Making the right choice now will give you the flexibility and capabilities you need in the future.
- **Developing a test sequence.** Test executives are applications designed to run a series of tests quickly and consistently in a predefined order. Of the 93 percent of test-system developers who use test equipment, approximately 37 percent use a commercial test executive for test sequencing, while the remaining 56 percent use a "homegrown" test executive.
- **Planning for software reuse.** Designing for code reuse means you and your co-workers won't have to re-create your software components every time you start a new project. Instead, you can build up a company knowledge base of best ideas, best practices, and software components. This knowledge base will bring uniformity and consistency to your company's product testing functions.

Figure 4.1. Test-system software development process overview



This chapter will provide you with a solid overview of the test system software architecture as outlined above. For more in-depth information, refer to the sources listed throughout this document. Now, let's get started with the first phase of choosing your test-system software architecture—gathering and documenting your software requirements.

Gathering and documenting software requirements

The Software Requirements Specifications (SRS)[1] is a prioritized list of required test-system software capabilities and information on the software's external interfaces, performance requirements, system attributes and design constraints. Typically, some requirements "musts" are essential and others "wants" can be traded for time (e.g., to meet a project deadline).

The IEEE identifies the following areas you should address in your SRS:[2]

- **Functionality.** What is the software supposed to do?
- **External interfaces.** How does the software interact with people, the system's hardware, other hardware and other software?
- **Performance.** What is the speed, availability, response time and recovery time of various software functions?
- **Attributes.** What are the portability, correctness, maintainability and security considerations?
- **Design constraints.** What industry standards need to be followed? Does a specific language need to be used? What about internal policies for database integrity, resource limits and operating environments?

Ideally, the SRS will describe WHAT you need the software to do, not HOW the software will do it. In other words, you can look at the software as a "black box" that controls a set of external resources such as instruments, a computer monitor and other components (see Figure 4.2).

The SRS will include implementation details only if those requirements are imposed externally. For example, your company may require that a portion of the system be implemented in a specific programming language.

A good SRS answers the following questions:

1. What measurements and tests are required to exercise the device under test (DUT)?
2. How will the measurements and tests be performed given the available instruments and devices?
3. What types of data need to be collected?
4. Where will the data be stored?
5. What are the external constraints (e.g., performance and time specifications)?
6. How will the operators, test engineers and technicians interact with the software?

Within the product development lifecycle, the R&D department should provide a formal list of testing requirements to the test-development department. The System Requirements Specifications, also referred to as a Project Requirements Specification, refers to the system as a whole and therefore is different from the Software Requirements Specifications. Furthermore, the manufacturing department will have its own requirements, such as safety standards. It is the combination of R&D and manufacturing specifications that determine the hardware requirements of a test system and provide the basis for the Software Requirements Specifications.

It's important to note that trying to build or design software while the test system hardware is still in a state of flux typically results in additional software re-work and re-design. This is one of the challenges you will face in the real world of test-system development!

1 May be referred to as an ERS or simply as "the requirements."

2 For more information, refer to the IEEE Standard 830-1998 *"Recommended Practice for Software Requirements Specifications"* and the IEEE Standard 1233-1998 *"Guide for Developing of System Requirements Specifications"* located on the IEEE web site (**http://standards.ieee.org**).

Figure 4.2. Scope of the SRS

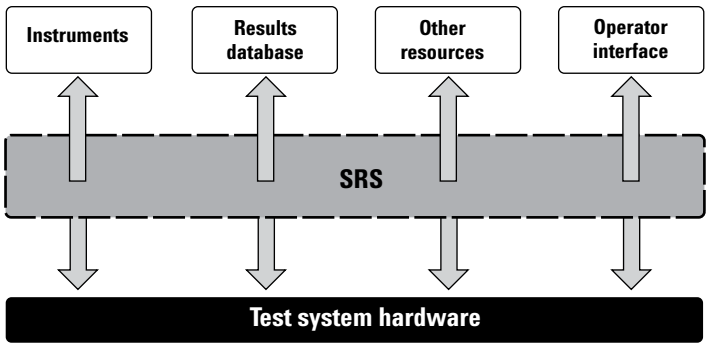


Figure 4.3 provides an SRS template and a requirements example. As shown in the template, SRS is more than requirements. Document within the SRS what the software is meant to do and provide definitions for the terms you are using. Document the external constraints imposed upon you and the external resources you have available. Describe your users in detail and the modes of operation for each user class. Finally, include appendices and an index. Once you've completed these tasks, you're ready to describe the specific requirements. The requirements example (user interface of a test sequencer) is a snippet from a larger set of requirements divided by function. The words "MUST" and "HIGH WANT" are a way of ranking the relative importance of the requirements. You can break up requirements into more manageable hierarchies based on function, program mode, or some other classification system that will make the requirements section easier to navigate.

Figure4.3. SRS template and requirements

Example SRS template

Table of contents
1 Introduction
- 1.1 Purpose
- 1.2 Scope
- 1.3 Definitions, acronyms and abbreviations
- 1.4 References
- 1.5 Overview

2 Overall description
- 2.1 Product perspective
- 2.2 Product functions
- 2.3 User characteristics
- 2.4 Constraints
- 2.5 Assumptions and dependencies

3 Specific requirements
Appendices
Index

The IEEE says that requirements must be correct, unambiguous, complete, consistent, ranked for importance, verifiable, modifiable and traceable. You can see that the above format meets a number of those goals, but some additional practices are necessary to meet them all. If you refer to requirements in more than one place, you will need to cross-reference them using a unique number (3.4.3, for example) so that if a requirement changes, you will know where to fix it elsewhere in the document.

Each written requirement needs to be verifiable and unambiguous to ensure the test program behaves as expected. As you write the SRS, refer to the System Requirements Specifications whenever possible. This is called backward-traceability, helping to explain why certain requirements are included and not just an arbitrary restriction.

The SRS must describe what testing resources (instruments) are required (e.g., the type of voltmeter, switches, computer monitor, etc.) and whether any factory resources are needed (e.g., a results database). In addition, you need to define within the SRS the data collection method, user interface requirements, performance constraints and, most importantly, the specific DUT test requirements. For example, if you need to perform a specific resistance measurement and you know you have an Agilent 34401A multimeter, the SRS would specify a single-sample 4-wire measurement including a description of the proper switching path, thus ensuring access to the pins on the DUT.

In order to accurately describe the test-system software user interface requirements, you should develop specific use cases for the different users of the test system (e.g., operators, test engineers, managers, etc.). Use cases are scenarios describing the users' interactions with the software. Taking the time to develop well-written requirements specifications up front will save you time later in the development process. The SRS process forces you to think about the scope of your project and helps to identify poorly understood areas of your software. This means you will spend less time re-writing and re-testing software due to confusion over what was truly required in the first place. A well-written SRS will help ensure that the project portion you want to contract out or redistribute will not require re-work on your part.

Example requirements

3.4 User interface functionality:
3.4.1 (MUST) The UI allows the user to create, modify, run and debug sequences.
3.4.2 (MUST) The UI allows users to view and export, load and store sequence run result data.
3.4.3 (MUST) The UI represents sequences in a hierarchical manner, which may be expanded or collapsed to view or hide internal details of the sequence.
3.4.4 (HIGH WANT) The UI can represent shared (used several places) sequences separate from the main sequence hierarchy.
3.4.5 (HIGH WANT) The UI will use graphical icons to denote variations in state of sequence items.

Programming and controlling your instruments

When designing your test-system architecture, you need to think about how your PC will communicate with different instruments. The two most important factors are deciding how to physically connect the PC to other instruments and deciding what software will you use to control and communicate with other instruments. Refer to Chapter 2 for advice on choosing an I/O option and to Chapter 3 for advice on choosing and using drivers and other instrument communication software.

Collecting and storing the test data

Data collection is the science of identifying, collecting, formatting and distributing important information about the behavior of your test system and the devices it tests (see Figure 4.4). Quality data collection and analysis is the foundation for controlling your manufacturing and test processes–the ultimate goal of a manufacturing test engineer. Quality data also can be used to support many functions throughout your organization and support products throughout their development lifecycle.

Communicating results of a test sequence is one use of test data. Test data also may be used to ensure regulatory standards are met, document performance standards, or provide traceability for the DUT. Given these applications and others, you may want to collect more data than your R&D or manufacturing colleagues request.

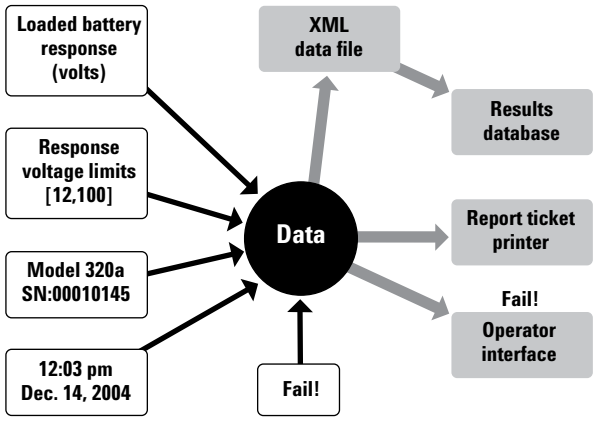
In addition to external data requirements, recorded data can be used to debug a test sequence in ways debugging runs cannot. Debugging means slowing down and subtly changing the behavior of your test sequence. This means a defect you see in a normal run may not show up in a debugging run (and vice-versa). One way to reduce the burden of diagnosing test software, and its associated DUT, is to always collect the data you need to debug a problem. You will need to balance the benefits of collecting extra data with the costs in performance and time for your test software.

Just as important as the standard types of data (e.g., test limits, measured values and pass-fail judgments) are the contextual data. Contextual data are used to communicate everything relevant to the DUT's testing environment. This includes the test-system configuration, software version, driver versions and other factors.

The more variables you record, the more correlation points you and your colleagues can analyze during debug. For example, in one particular manufacturing test situation, a DUT would fail in the afternoon. The test engineer was able to correlate the time of day to the time of the failure and use that information to look more closely at a photoelectric component of the DUT. It turned out that sunlight would strike that component directly at certain times of the day, causing the component to charge a capacitor and cause the test to fail. A DUT may fail due to the temperature variations or relative humidity. Capturing contextual information and measurement conditions can save days of effort.

You want to ensure the writing or formatting of your data does not affect the behavior of your test system. Today's PCs use a variety of caching techniques that can dramatically affect how long it takes for a given file or network I/O command. If the time it takes to cache your data varies between each test run, you will get inconsistent test results. For that reason, it's a good idea to keep your data in RAM until the end of your DUT testing and then do your formatting and data transmission.

Figure 4.4. Overview of the data collection process



Data is useless unless it can be understood. Good data is

- **Identifiable.** Information identifies the circumstances surrounding the data and the condition in which it was collected.
- **Searchable.** The data posses regular structure or fields that are uniquely identifiable, making it easy for a script or software tool to identify and compare across multiple records or datasets.
- **Transformable.** Raw data must be interpreted and displayed (insight is the goal). This means that software algorithms can perform operations on some or all of the fields of your data and create a new data format or data visualization based on your original data.
- **Permanent.** Data must remain available and comprehensible. Relational databases tend to be the best choice for long-term storage of data as these databases are highly searchable. If your company does not already have a database for manufacturing information, you may want to consider a database solution. This decision depends on your company's data storage policies, practices and budget.[3]

Table 4.1 lists some common data file formats and relevant characteristics.

Binary formats have the fundamental issue of not being self-describing. In addition, you need to acquire a separate software application to interpret the data. Depending on the software application you use for interpreting the data, you also may be limited in the number of transformation functions.

Text files are hard to search and transform, and are not very identifiable. Since plain text files do not have regular fields, a text search for the number 12, for example, could return the hour twelve, the limit value 12, or the DMM measurement 12.

3 Tufte, Edward R. *"The Visual Display of Quantitative Information."* Graphics Press, 2001.

Table 4.1. File data format comparisons

	Binary	Unformatted text	Comma-separated variables (.csv)	XML (Extensible Markup Language)
Identifiable	Only with special tools	Only for small data sets	Needs good column format design	No major issues
Searchable	Only with special tools	Difficult and error-prone	No major issues	Excellent, but requires XML expertise
Transformable	Only with special tools	Difficult and error-prone	No major issues	Excellent, but requires XML expertise
Permanent	Only with special tools	Only for small data sets	No major issues	No major issues
Example: spreadsheet analysis	Only with special tools	Not importable	Supported by Excel, others	Excel 2003 format available

Comma-separated value (dot-csv) text formats are a good choice since they are easy to import into Microsoft Excel. With Microsoft Excel, it's easy to make a table of results with the rows containing the results and each column containing a unique description. Another advantage is most data analysis software can easily read this format. The downside of this format is that it cannot store hierarchical data or easily parse data sets. You must decide up front as to the number and types of columns, with each column containing one unique data field.

XML[4] is self-describing, very transformable, and has excellent search characteristics. There is an XML language called Extensible Stylesheet Transforms (XSLT) that can apply arbitrary algorithms to convert your XML data into new XML formats, HTML, or simple text formats.[5] A number of data analysis programs, including Microsoft Excel 2003, can import XML data.[6] If you fail to output your data in the right XML format for an analysis tool, you can write a relatively small XSLT that will convert all your XML data into the desired format. XSLT also provides a powerful search feature, making it much easier to identify data values or data structures.

4 Extensible Markup Language: **http://w3.org/xml**.

5 Holzner, Steve. *"Inside XML."* New Riders, 2000.

6 XML in Microsoft Office: http://www.microsoft.com/ presspass/press/2002/Oct02/10-25XMLArchitectMA.asp.

The manufacturing test industry has already begun adopting XML. Some test executive applications support XML data logging. There is a standard called IPC 2547[7] that defines an XML format for communication of manufacturing test data.

Figure 4.5 is an example of a standard test run in XML format. You will still want to know the test sequence ID, the variant of the test, if the test limits are modifiable on the "PowerTest" and the hardware configuration of the test system.

If this were a .csv file, we would have to create a field for every record to answer those questions. Using XML, we can insert a record type called <TestSequence ID="32"> and fully describe the current test sequence in that record. We can then add an XML attribute called "IDREF" to refer to that test sequence record in our <TestRun> records.

In summary, the data format you choose will have a large impact on its value over time. You need to consider how easy or difficult it will be for someone else to read and interpret the data once you are no longer involved in the project.

7 IPC 2547: http://webstds.ipc.org/2547/2547.htm

Designing the user interface

When a user displays generated by a test system should vary according to the class of user, such as operator, test engineer, technician, or service and calibration engineer. A well-written SRS defines the commands and/or menu selections available to each user class. You will want to provide each user class with only the capabilities and information those people need to do their jobs. The more choices you provide, the greater the possibility for confusion and mistakes.

To ensure security, you can create a unique login for each of the users. Each user login should be linked to the appropriate class.

You can verify that your GUI meets the users' needs with a methodology called "User-Centered Design," or UCD, which consists of prototyping and storyboarding.[8,9] In general, a test system's GUI should be able to

1. Customize its behavior based on the user class.
2. Provide or allow input of detailed information about the DUT.
3. Provide information about the state of the system.
4. Provide operations for controlling the system's state and potentially its configuration.
5. Display the DUT testing results.

For an operator, the interface you design should always show the state of the test system (e.g., running a test, paused or stopped). For example, you could use a large color-coded graphic on the PC monitor in conjunction with lights mounted on the test system. The operator also will need a way to control the state of the test system as well as a way to input DUT information (unless this is done automatically via a bar code scanner).

As a general rule, test program should have the following features:

1. Commands for starting and stopping the test sequence.
2. Commands for sending test results to various kinds of printers (defect report ticket, etc.).

8 Vredenburg, Karel, et al, *"User-Centered Design, an Integrated Approach."* Prentice Hall PTR, 2002.

9 Norman, Donald A., *"The Design of Everyday Things."* Basic Books, 2002.

Figure 4.5. XML report file

```
<?xml version="1.0" ?>
- <TestReport xmlns="urn:Agilent/EPSG/Casper/Production">
  - <Sequence name="FastTestA.tsq">
      <DUT serialnumber="0000100245" model="101" />
      <TestEnv operator="Joe1" host="fasttest3" date="1/22/04" time="01:24:31 pm GMT" />
      <Result success="0" message="PowerTest: Voltage outside Expected Range" elapsedSeconds="109" />
      <Test name="PowerTest" success="0" />
    </Sequence>
  - <Sequence name="FastTestA.tsq">
      <DUT serialnumber="0000100246" model="101" />
      <TestEnv operator="Joe1" host="fasttest3" date="1/22/04" time="01:29:03 pm GMT" />
      <Result success="1" elapsedSeconds="124" />
      <Test name="PowerTest" success="1" />
    </Sequence>
  </TestReport>
```

3. Control of the behavior of the test sequence (i.e., picking a DUT variant from a drop-down list).

4. A way to display a more detailed description of test results. The quality of a test results message can help in providing a quick diagnosis of a user error or a recurring hardware problem and may ultimately eliminate the need for a test engineer to visit the factory floor.

The user interface shown in Figure 4.6 was designed for an operator in a low-to-mid-mix/high-volume test application. The operator starts by logging into the test system, selecting the name and version of the test plan and entering the DUT information. The test status portion of the display is a little less prominent and visible than recommended for a manufacturing test environment, which may necessitate the addition of test status lights to the test system.

The system message field displays the test result information as well as instructs the user on what to do next. To help the test engineer during the debugging process, the system message field also can display error messages.

The user interface shown in Figure 4.7 was designed for a high-mix, very low volume testing situation (e.g., cell phone base stations). It also can be used for test sequence development or debugging. The class of user for this interface is highly skilled and possesses detailed knowledge of the purpose and function of the available tests, the DUT, and the test system configuration. An unskilled test operator would not be able to use this interface effectively.

The two GUIs were created with the same test software, though they vary considerably in complexity. The operator GUI in Figure 4.6 hides unnecessary choices and information critical to the software developer.

Figure 4.6. Low-mix, high-volume user interface

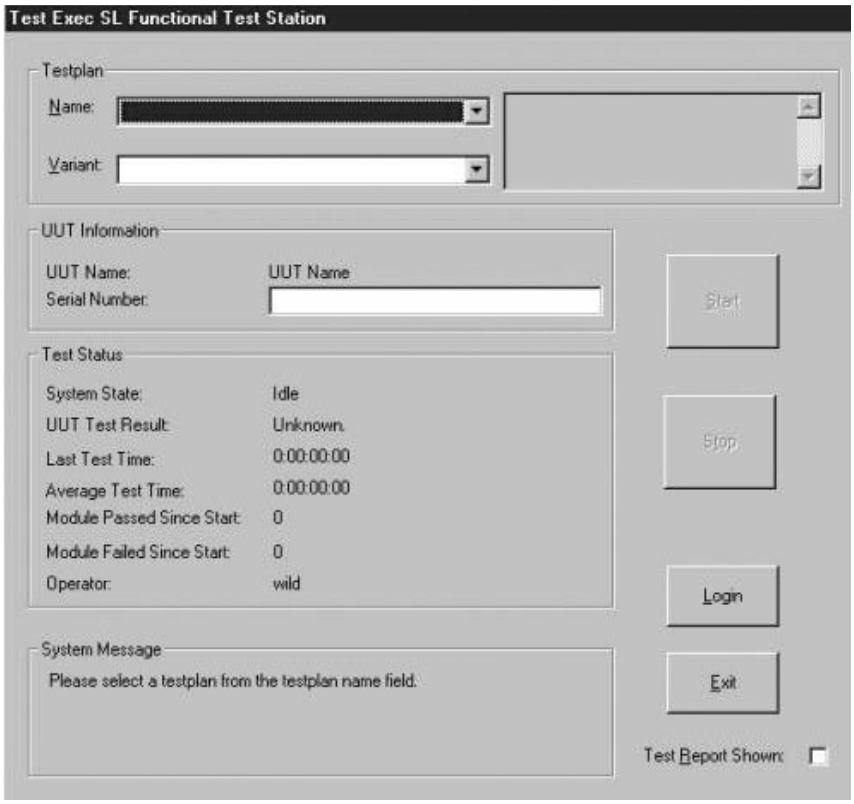
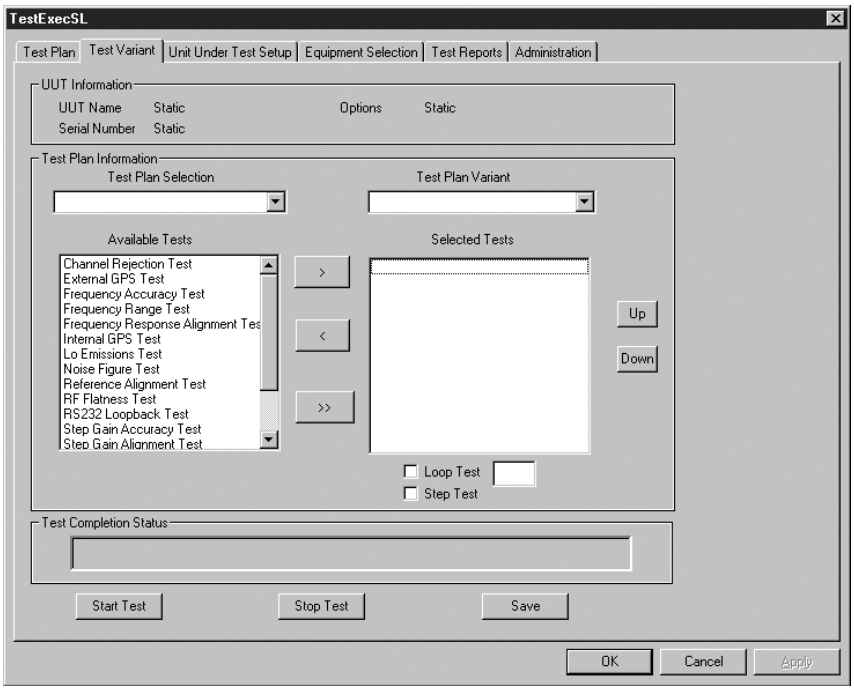


Figure 4.7. Software developer's interface



Choosing the development environment

The next step in choosing your test system software architecture is to select a software development environment. The software environment and tools you choose will have a significant impact on the overall cost of your test system. When choosing your software environment, consider more than just the purchase price of the software. You need to consider how easy it is to learn and use the software, how hard it is to connect to other languages, devices or enterprise applications, as well as support and maintenance costs. Over the life of a test system, just software support and maintenance costs can exceed hardware costs.

You have a number of options when it comes to software development environments, from writing everything yourself in a language such as C, C++, C#, VB, VB .NET, Agilent VEE, MATLAB or LabVIEW, to using an off-the-shelf test executive with pre-written third party tests. The software environment you choose needs to accomplish two goals: 1) meeting your time-to-first test requirements and 2) meeting your test-throughput requirements. How fast can you get your test system up and running, and how can you get the greatest throughput?

Software development environments can be grouped into two categories: graphical or textual. Graphical environments, such as Agilent's VEE Pro 7.5 (see Figure 4.8) or LabVIEW, are considered easy for engineers to learn and use, largely because of engineers comfort with the schematic environment. In addition, it is easier to modify small to medium size graphical programs versus textual programming languages. Historically, textual programming languages ran faster in the manufacturing environment and yielded higher throughput. Today, there is less difference between the runtime speeds of a graphical environment and a textual environment..

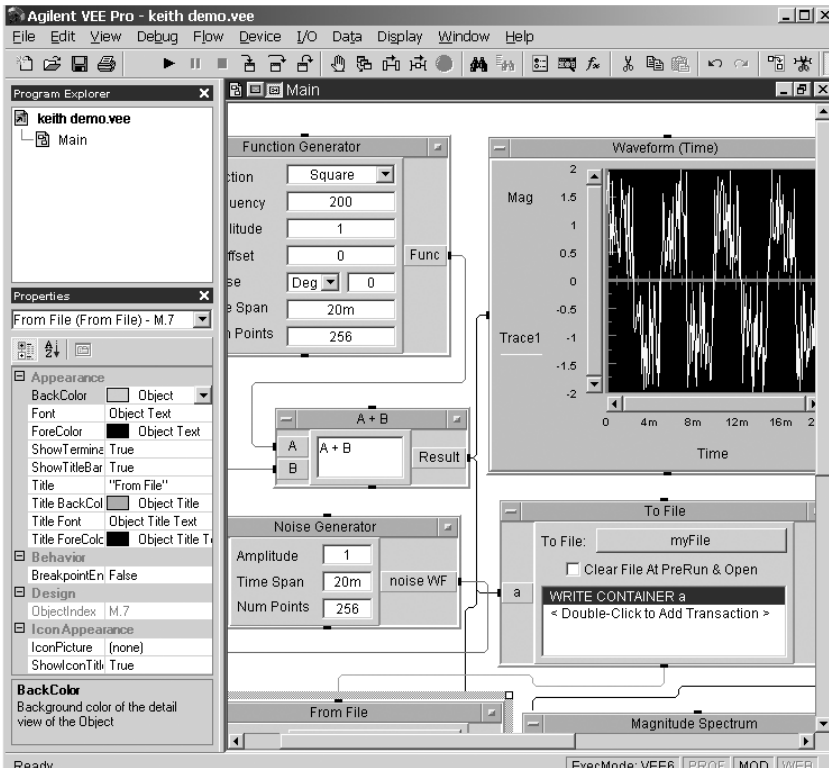
Even though graphical environments are easier to use than textual environments, textual environments are used more commonly in manufacturing test systems. Only about 22 percent of the half million-plus users who write code for test and measurement equipment use a graphical programming language.

Graphical or textual programming?

Before you can decide on which development environment is best for your application, it's important to understand the use model of each in greater detail.

Graphical programming is accomplished by manipulating images, called icons or objects, and the lines that connect these images. The images represent pre-made commands while the lines represent the program flow, control points, and /or how data are generated and consumed. The icons and interconnecting lines are contained within the integrated development environment (e.g., the software program).

Figure 4.8. Agilent VEE Pro graphical programming environment



Many graphical programming environments provide the ability to create compiled or packaged programs that do not need the programming environment to run. There are several graphical programming environments targeted at test and measurement engineers. These programs tend to have extensive I/O and instrument drivers, and T&M-specific math and graphing operations.

Some of the advantages of graphical programming languages over textual languages are as follows:

1. **No complex syntax.** The program instructions, typically presented as a group of icons connected by lines, are more immediately understandable.
2. **Easier visualization of the paths of execution and interaction.** Multiple concurrent activities rely on what is called a data-flow model, where a command needs to have all its data available before it will execute. This is easier than using multithreaded programming techniques in textual programming languages such as C++ or Java.
3. **Real-life metaphors.** The icons representing the commands can use metaphors (images) that represent real-world equivalents of the actions carried out by the icon. Most test engineers find graphical programming to be more intuitive and user-friendly than textual programming.
4. **Rapid prototyping.** With the intuitive nature of a graphical programming language, it can be easier to quickly build a prototype of your system. The prototyping capability is less useful when dealing with a large test system, but prototyping can aid development of systems of any size.[10]
5. **Ability to share and learn existing programs easily.** Using real-life metaphors as visual cues can make it easier to share and learn existing programs and increase productivity.[11]

10 Rahman, Jamal and Lothar, Wenzel, *"The Applicability of Visual Programming to Large Real-World Applications,"* 1995, http://www.computer.org/conferences/vl95/html-papers/wenzel/paper.html.

11 Blackwell, Alan F. and Green, T.R.G., "*Does Metaphor Increase Visual Language Usability?*," IEEE Symposium on Visual Languages VL'99, 1999, pp. 246-253.

Textual programming languages use special words and syntax to represent the program's operations and flow. Most, but not all textual programming languages are based on open standards. This means you will have a choice of vendors when it comes to your programming environment and software tools. Textual programming languages have a much larger set of third-party drivers, tools, and add-ins because they are based on open standards and are more widely used than graphical languages. This benefits the test engineer.

Some of the advantages of textual programming languages over graphical languages are as follows:

1. **Ability to handle large programs.** Textual programming languages are better suited for creating larger, more comprehensive programs.

Agilent VEE Pro and T&M Programmers Toolkit

Agilent VEE Pro

- Description: Easy to use, powerful graphical instrument programming environment
- Applications: Data acquisition, design, low volume manufacturing test
- Purpose: Graphical program creation to acquire and analyze instrument data
- Features: Easy test-system control, sequencing, support of Microsoft .NET framework, MATLAB® analysis and visualization, full support of ActiveX

Agilent T&M Programmers Toolkit

- Description: Test code development (in VB .NET, C++ or C#) integrated into Visual Studio .NET
- Applications: Design characterization, design validation, manufacturing
- Purpose: Writing complex programs with a variety of drivers in a PC standard environment
- Features: Instrument I/O and communication, test code debug, data collection, display and analysis, support for IVI-C, IVI-COM, VXI*plug&play* drivers

2. **Simpler navigation of large programs.** For larger programs, textual programming languages are easier to navigate and comprehend. A person can observe only about 50 graphical objects at a time before the information becomes too complex or too small to see.[12] If a user is forced to move around in a program to see all its objects, he or she can lose track of the control and data lines and find it difficult to understand the overall flow of the program. With that said, you can improve the understandability of large graphical programs by breaking up the program's large operations into smaller suboperations. This is called functional decomposition and is achieved by putting a series of commands into a "black box". You then send commands to the functional block and receive its output as appropriate. Make sure the graphical program you use supports this functional decomposition[13] if you plan on working with larger programs in a graphical environment.

3. **Higher system throughput.** The faster runtime speeds of a textual programming language can improve overall system throughput. However, be aware that the time spent during instrument operations will often have a greater impact on throughput than the choice of programming environment. For example, time lost through inefficient signal switching between the test system and the DUT can far outweigh any time savings earned through choice of programming language.

4. **Greater choice of development environments.** For example, there are few graphical programming languages that have development environments provided by multiple vendors. This means that today's graphical languages are less likely to have the advantages created by competition between vendors.

Graphical programming tends to be easier to learn and comprehend while textual programming is more pervasive and open. Table 4.2 summarizes the differences between the two programming environments.

Working with open standards

In addition to choosing between graphical and textual programming, you need to consider whether the environment you choose will be based on industry standards or propriety, vendor specific technology. C++, Visual Basic, and C# are all examples of industry standard programming environments. Agilent VEE Pro and NI LabVIEW are examples of proprietary development environments although Agilent VEE Pro 7.0 does allow for access into industry standard technologies such as .NET.

Choosing between proprietary and open standards

Several factors to consider when deciding between an industry standard and a proprietary development environment are 1) cost, 2) industry support, 3) upgradeability, and 4) extensibility.

Development environments for open standard programming languages have a greater feature set and are less expensive than their proprietary counterparts. Simply stated, an open standard environment tends to create greater competition, which in turn tends to drive down prices and create innovation.

Open-standard languages generate a lot of interest from both software tool vendors and open-source developers. Both of these groups spend considerable time understanding the needs of the test-system programmer and, as a result, develop both free and for-pay tools and applications to meet those needs. A good example is the tremendous number of C and C++ libraries available on the market, both from vendors and from end-users. These libraries save development time and money given that it is faster and less expensive for a developer to buy the domain-specific software (such as mathematical analysis libraries) than create it from scratch.

Table 4.2. Graphical versus textual programming

	Graphical	Textual
Free and open	Few open standards, less extensible	Dominated by open standards, very extensible
Rapid prototyping	Excellent T&M prototyping features	Some code wizards, (T&M Programmers Toolkit, for example) but slower
T&M support	Several graphical environments targeted at T&M, many drivers	Several T&M-specific 3rd-party tools available, many drivers
3rd-party tools	Hundreds	Tens of thousands
Learnable and shareable	Easy to pick up and use programs	Only small or very-well-designed programs are easy to share

12 Begel, Andrew, *"LogoBlocks: A Graphical Programming Language for Interacting with the World,"* 1996, http://www.cs.berkeley.edu/~abegel/mit/begel-aup.pdf.

13 Glinert, E. P., *"Visual Programming Environments,"* IEEE Computer Society Press, 1990.

Open standard environments also have a time-to-market advantage, as most proprietary environments cannot quickly take advantage of emerging technologies. Emerging programming technologies are developed with the most common open standard programming languages in mind. It takes longer for a vendor to release a new version of proprietary software that takes advantage of new technology.

The .NET framework

The .NET framework is an open, multi-platform, multi-vendor set of software technologies for programming computers. The C# language has been submitted to a standards body as an open language. The underlying .NET "Common Language Infrastructure" technology, also an open standard, is available in multiple operating systems, including Microsoft's Windows and Linux.

The .NET technology has excellent support and applicability to both web development and PC software development environments. The .NET technology has many of the advantages of Java language without many of Java's drawbacks. For example, the .NET technology eliminates programmer memory leaks, makes software deployment easier, and provides a rich Application Programming Interface (API) for system and GUI development. The .NET technology is fully compiled via a Just-In-Time (JIT) compiler. The JIT compiler takes the operating system (OS) and platform independent code and creates optimized, machine-level code for the target platform.

While there is some additional overhead required to load the .NET framework runtime, programs written with .NET are comparable, or run faster, than their C/C++ counterparts.[14] The reason programs can run faster in the .NET environment is due to the inefficiencies inherent in the linker operation of older languages.[15]

A survey of programmers and a number of case studies have shown significant improvements in productivity via the .NET environment over the programmers' old environment.[16]

The .NET Framework (the collection of API services and helper code used by the .NET languages) is not the same thing as Visual Studio .NET. Visual Studio .NET is Microsoft's development programming environment with support for the .NET technologies. As shown in Figure 4.9, there are multiple .NET development environments and programming languages available from a number of different vendors and supported on multiple platforms.

The best-known .NET languages are C# and Visual Basic (VB) .NET. C# is a lot like Java in structure and features, but its syntax is meant to be an evolution of C++. A C++ programmer familiar with object orientation and exception handling could easily move to the C# programming environment.

VB .NET is an upgrade to Visual Basic 6. Engineers with existing VB 6 applications must use an upgrade wizard to migrate to VB .NET. Once the upgrade process is complete, access to .NET applications and the additional power and flexibility provided by .NET can be achieved.

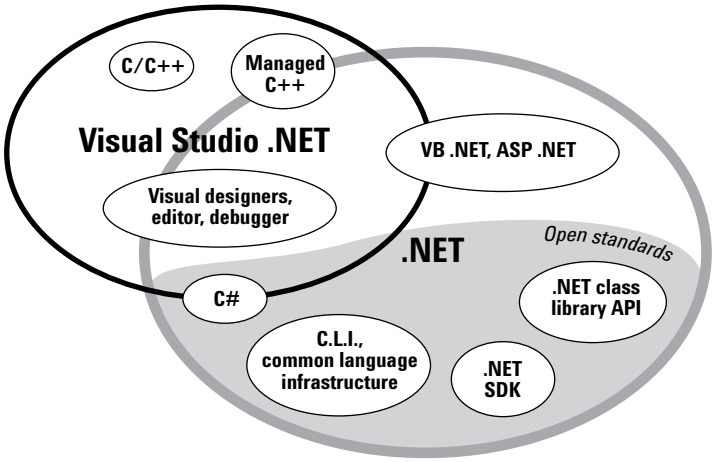
Microsoft's C++ language also has been enhanced to include a new version called Managed C++. Managed C++ makes it easier to execute calls within the .NET software. Microsoft provides the original unmanaged C++ in Visual Studio .NET as well.

14 Wilson, Matthew, *"Does C# Measure Up?"* Windows Developer, Volume 2, Issue 13, Fall 2003, http://www.wd-mag.com/wdn/webextra/2003/0313

15 Johnson, Mark S. and Miller Terrence C., *"Effectiveness of a machine-level, global optimizer,"* 1986, http://portal.acm.org/citation.cfm?id=13321&dl=ACM&coll=portal

16 http://www.microsoft.com/net/casestudies

Figure 4.9. Programming languages within the .NET framework



One significant advantage of .NET over older programming technologies is its extensibility. Microsoft engineered .NET so that it avoids a lot of the DLL installation frustrations Windows programmers experienced in the past. There are already a large number of third-party tools for .NET. Many of these third-party controls (i.e., advanced graphing visual controls) are useful to test-system programmers. Additionally, several test and measurement vendors, including Agilent Technologies, National Instruments, and Measurement Computing, have released .NET-compatible tools. For a complete list of released .NET-compatible tools, refer to Microsoft's .NET partner web site at www.vsippartners.com.

Agilent Technologies' first add-in for Visual Studio .NET is called the Test and Measurement Programmers Toolkit (see the sidebar on page 43). The T&M Programmers Toolkit provides I/O tools, graphing and mathematical libraries, T&M specific help and example generators, and .NET wrappers for instrument drivers and other software. The T&M Programmers Toolkit is fully integrated into the Visual Studio environment. For more information on Agilent's solutions, go to **http://www.agilent.com/find/toolkit** or **http://www.agilent.com/find/iolib.** To download .NET-related I/O source files, which also work with the Agilent I/O Libraries, go to the Agilent Developer Network (ADN) web site at **http://www.agilent.com/find/adn.**

Developing a test sequence

In a survey of more than 2,500 test and measurement equipment users, 93 percent of the respondents said they use multiple test instruments and /or are connecting their test instruments to a PC. Of that, 37 percent said they use a commercial test executive for test sequencing. The remaining 56 percent of these respondents use internal or "home grown" software for test sequencing.

A test executive is a software application designed to run a series of tests quickly and consistently in a predefined sequence. If any of the tests within the test sequence fail, then the DUT fails. Over the years, test executives have improved considerably both in terms of flexibility and capabilities. First-generation test executives were language-specific and not powerful enough for a mission critical manufacturing environment. Second-generation test executives, such as Agilent's TxSL and NI's TestStand are more powerful but more expensive. They also lack the flexibility required for a low-volume, high-mix manufacturing environment.

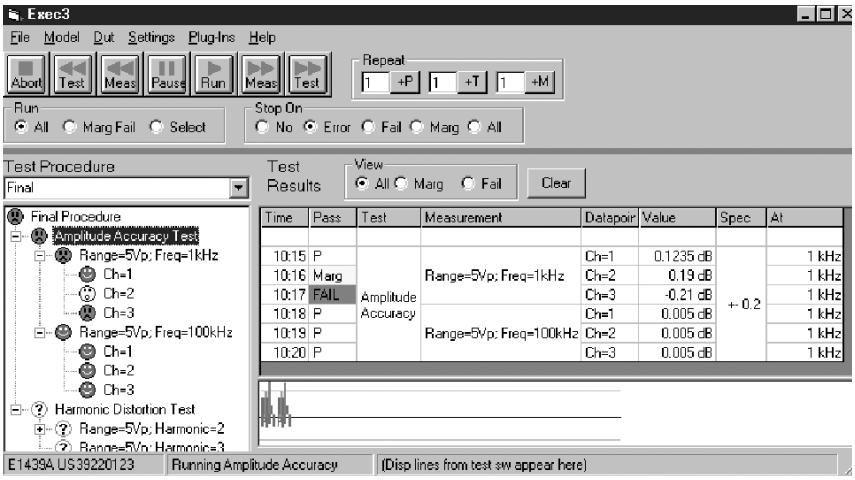
Each of the tests within the test sequence is a separate module. Commercial test executives come with a standard set of test modules and allow the user to create additional test modules from scratch (as well as customize existing test modules). Test executives control the data to and from the test module and, after collecting and analyzing all of the data, determine if the DUT passed or failed.

One reason for using a test executive is it provides a structured framework for manufacturing test systems. Test executives work best in medium- to low-mix, and medium- to high-volume manufacturing test environments.

Test executives are written so that sequence design, individual test design, and test limits and configuration management are treated as separate tasks. Keeping the three tasks separate results in greater flexibility, higher quality, and an increased opportunity for code reuse. It is the test executive that provides the infrastructure and helper services required to connect each of the separate tasks into a complete program.

One of the most important features of a test executive is its test sequencer. As shown on the left side of Figure 4.10, the test sequencer is a sequence of tests that can be manipulated in design mode. Various test executives provide different levels of flexibility in this sequence, such as "test looping."

Figure 4.10. The test executive test sequencer



At a minimum, test executives should perform the following tasks.

1. Capture the results (and any extra data) using their own data collection model.
2. Keep track of the test limits and test setup data, passing the setups to the tests at execution time.
3. Provide limit checkers.
4. Provide run-time analysis of the test results (pass or fail reporting).

Additionally, test executives may include a software repository for maintaining the test modules (and for encouraging the reuse of tests). With a software repository, the test engineer can look for a specific test by doing a search within the test module repository. If all the engineers in a company settle on one test executive, it then becomes possible to share test modules between different product and manufacturing groups.

Test executives may use a switching model that makes it possible to map the physical layout of the test system's control and data lines (and any switch boxes) to the DUT and instrument's I/O pins. This allows the test engineer to think in terms of logical connections between instruments and the DUT, rather than worry about how the system is wired.

Finally, some test executives include tools for building the operator interface. While this feature tends to be less flexible than using one of the development environments discussed earlier, it does provide a fast and simple alternative.

Planning for software reuse

Aside from the use of standard libraries and operating system API's, most software reuse tends to be opportunistic. A typical reuse scenario is when a programmer encounters a problem and remembers a similar problem handled by a co-worker. The programmer searches through the old source code of previous programs to find the desired code. If the code is found, the programmer decides how and if the software can be adapted to the current test situation. After modifications are made, the software must then to be re-verified. Most of the time, adapting software in these situations is faster than creating software from scratch.

The scenario above could have been improved with a systematic software reuse approach. The advantages of a systematic approach is in the reduced time it takes to search, find, verify, and adapt test code for new test situations. A systematic reuse approach requires following specific coding and architectural styles, as well as adherence to standardized company policies and practices.

Discussing all of the considerations for implementing a complete companywide systematic reuse program is outside the scope of this chapter, but there are decisions you can make to help achieve a more systematic approach for yourself, your team, and even your company. Reuse considerations should begin after you've gathered system requirements and before you begin the software development.

Professional test executive or custom software?

How do you decide if you should create your own test executive or buy an off-the- shelf version? Here are a few factors you will need to consider.

1. The first thing to look at is whether you need a test executive at all. If you don't have a relatively fixed sequence of tests, test executives are probably not right for you.
2. If your company has an internal test executive, or more likely, several internal test executives, you'll need to investigate their quality, features, availability of support, and the collection of tests or other auxiliary software available for them.
3. If you find a reasonable choice, it doesn't hurt to look at the cost of porting existing code over to use a professional test executive.
4. You may decide to use a professional test executive because of its support, quality or features.
5. A professional test executive most likely will have better outsourcing characteristics. Third-party software contractors and consultants may already have experience with such a test executive, and third-party libraries may be available.
6. A professional test executive should include a complete set of documentation.

If you choose to go with a professional test executive, make sure it's from a company that provides high-quality service and support.

The design reuse process

The first step in the design reuse process is to complete a domain analysis. This is accomplished by 1) systematically analyzing the functions and parts of your software domain, and 2) using this information to develop a software architecture with well-defined component types and algorithms.

Next, you will want to look for natural boundaries in your software. One software design practice of finding and documenting the natural boundaries is known as Design Patterns.[17] To find the natural boundaries, look to those areas where one type of activity or data set links with another type of activity or data set. These areas can then be grouped into separate modules and documented accordingly. Once documented, the same type modules can then be swapped for one another.

Once you have identified, collected and documented your modules, components and /or individual parts, you will need to thoroughly test them before they go into the repository (or are passed on to your co-workers). This will save you and your co-workers from problems later in the process.

Finally, reusable components are reusable only if your co-workers know they exist. You need a repository (such as a relational database) for your modules where anyone in your team, division, or company (if appropriate) can browse and search for them based on what the components are and what they do.

A design reuse example

A good model for design reuse of individual test modules is the test executive—here's why.

1. Some test executives break test software up into swappable tests, sequencers, limit checkers, test sequence and test limit data.[18]

2. Test executives rely on the concept of modules. For example, you can have a module that provides the ability to perform a single pass or fail judgment, including the sequencer data type, the sequence execution operation, and the test types.

3. Test executives allow reuse of tests in different test sequences with no change to the test code. The sequencer provides the necessary data to the tests to customize their operation for the current test sequence.

4. Test executives keep the tests in separate modules or files from the test sequencer or test executive application. This allows you to easily swap tests in and out without recompilation.

5. Some test executives allow you to write your own custom limits checkers or sequencers.

All of these modules are able to interoperate because test executives use well-defined application programming interfaces (APIs) for each module. The modules are placed on natural boundaries between different types of data and functions within the test executives.

You can achieve similar reuse success in your own code with good architecture influenced by the natural boundaries of your software's functions, types and data. To accomplish this, put information that changes frequently, such as the limits for a test, into a Data File. Put less flexible elements, such as a test class, into Types or "classes." Functions, or "procedures," should be reserved for the least flexible elements.

Design reuse and .NET

While the definitions of the boundaries of your software domain are not specifically influenced by the programming language or software environment, some environments are better than others in helping to keep your software modular and swappable.

.NET provides software tools that make it easier to develop a formal software reuse program within your department or company. Since .NET is object-oriented, it's good at representing boundaries between different types of objects, such as tests or sequencers. Nonobject-based languages, such as C, require you to keep track of which functions apply to which objects, without much context-sensitive help or compile-time error checking.

.NET also includes improved versioning and deployment features. In addition, .NET has the ability to tell Windows that you will only accept a certain version of an external library. This eliminates one of the common frustrations with earlier versions of Windows where you rely on an external library (DLL), but then the DLL changes and your software no longer functions correctly.

17 Shalloway, Alan and Trott, James R., *"Design Patterns Explained: A New Perspective on Object-Oriented Design,"* Addison-Wesley Pub Co, 2001.

18 This is a good example of a design pattern specific to the test and measurement domain.

Design reuse benefits

In summary, the reasons for implementing a design reuse program include improved software quality, increased software development efficiency, and better use of expert knowledge.

Design reuse improves quality in a couple of different ways. First, software errors are reduced as a result of the extra architectural analysis, improved system design, and flexibility and transparency. With good reuse policies implemented throughout the organization, you have access to thoroughly tested and verified components, reducing the opportunities for creating new defects.

Design reuse increases software development efficiency by reducing duplication of effort. Components need to be designed, implemented and tested only once. Good reuse practices make it easier to reuse an existing component as opposed to re-writing or even re-creating a new component.

Design reuse takes advantage of an organization's expert knowledge. For example, most software developers spend time specializing on a particular set of skills and will write components based on those skills. With time, the set of available components for reuse becomes the set of the best knowledge of your organization. The company's expert skills and deep knowledge will be evident in a rich set of reusable software components.

These benefits are not theoretical. The Software Engineering Laboratory at the National Aeronautics and Space Administration's (NASA) Goddard Flight Center achieved significant benefits by implementing software reuse in the development of software products in its Flight Dynamics Division. According to the software engineering lab, NASA realized a 35 percent reduction in the effort needed to deliver a line of code, a 53 percent increase in daily productivity, and an 87 percent increase in code quality.[19]

19 Proceedings of the Sixteenth Annual NASA/Goddard Software Engineering Workshop: Experiments in Software Engineering Technology, Software Engineering Laboratory, December 1991.

Design reuse summary

Systematic design reuse across your company requires that your management value the extra efforts required by designing for reuse. Failure to invest and do the job right the first time will lead to frustration and wasted time down the road. One or more repositories of software components must be made available to all the engineers who will need them. You also need to be aware of any copyright or patent limitations of the code you plan to reuse. For example, if your software is written under contract with another company, they may have exclusive rights to that code.[20]

20 Defter, Frank W, et al, *"Software Reuse: Major Issues Need to Be Resolved Before Benefits Can Be Achieved,"* United States General Accounting Office, 1993, http://www.defenselink.mil/nii/bpr/bprcd/vol2/272c.pdf.

Conclusion

Before you begin writing code for your test system, you need to make a number of important decisions about the system's software architecture. You will want to start by creating a detailed software requirements specification that defines what you want the system to do and how it should operate. The SRS should include an outline of how you will gather, store, analyze and present your data as well as how end users will interact with your system.

Another important decision you need to make upfront is which programming environment and language you will use for writing your code. Using a standards-based environment such as Visual Studio .NET maximizes your flexibility and helps you prolong the useful life of your software. By combining Microsoft's Visual Studio .NET with Agilent's T&M Programmers Toolkit, you can wrap objects written in a variety of languages such as Agilent VEE Pro. This allows you to pull them forward into your new programming environment, making the most of your legacy code investment.

Whether you choose a graphical or textual environment will depend on the size and complexity of your system, your skill set, your company standards, and the size of your programming team. The decision usually comes down to which environment—graphical or textual— will make you more productive. Textual environments are almost always the best choice for creating code for large, high-throughput manufacturing test systems because they offer the most power and flexibility, and they allow faster throughput.

Finally, you need to decide whether to use an off-the-shelf test executive or write your own test routines. Test executives can speed up your test system development and lower your costs but will require an up-front training investment. If you are only performing a few tests, you may want to consider writing your own code.

5. Choosing Your Test-System Hardware Architecture and Instrumentation

Introduction

This chapter explores the hardware architecture decisions and design choices you must make before you begin building your system to ensure that it provides you with the performance and flexibility you need. It also discusses issues you should consider as you select instruments for your system.

A test system is essentially a group of subsystems that work together to test a particular device or range of devices. You need to make important decisions about each of the subsystems before you begin ordering test instruments or building your system. The way these subsystems communicate and interrelate has a huge effect on the cost, performance, maintainability and usability of your system. The time you spend upfront defining the architecture of your system is likely to save you time later that you might spend debugging software and tracing down the cause of faulty measurements. Ultimately, careful planning will help you ensure accurate testing of your DUT.

When you design a test system, you need to consider many of the same issues that architects consider when they design buildings: esthetics, safety, heat, size, cost, future expansion, optimal location of parts, and so on. Once you have decided how to approach these high-level issues, your test requirements will guide you in designing a system for the range of devices you expect to test.

This chapter explores the system architecture decisions and design choices you must make to ensure your test system provides you with the performance and flexibility you need. It also discusses issues you should consider as you select instruments for your system.

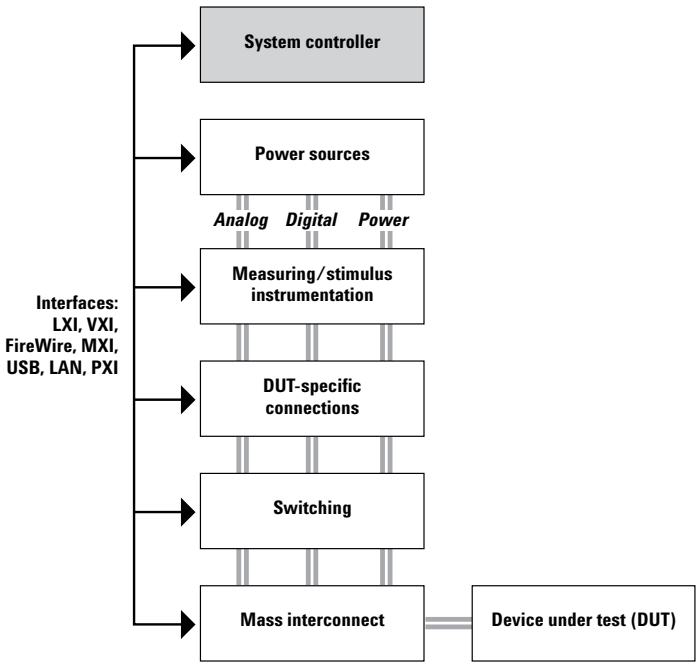
System architecture

The architecture you choose for your test system will depend on whether you plan to use it for R&D, design validation, or manufacturing test. In R&D, for example, you are probably performing parametric tests that will not be repeated on hundreds of devices under test (DUTs). In design validation, you need to be able to adapt to pinouts that are changing often, but the speed of each individual test is not particularly critical. In high-volume manufacturing, you've got hundreds to thousands of DUTs to test, and you want to test them as fast and as inexpensively as you can. The architecture of your test system will be different in each of these situations. In an R&D environment, you might not use all of the subsystems listed below, but for design validation, production validation or manufacturing test, typically you will need to make decisions about six major subsystems (see Figure 5.1):

- Instrumentation (measuring and stimulus instruments)
- Computing (computer, software and I/O)
- Switching (relays that interconnect system instrumentation and loads to the device under test, or DUT)
- Mass interconnects (DUT-to-system wiring interface)
- Power sources (power to the DUT)
- DUT-specific connections (loads, serial interfaces, etc.)

The test engineer's challenge is to choose these subsystems carefully and put them together efficiently. Let's look at each of the subsystems individually.

Figure 5.1. A generic test-system architecture



Instrumentation type: rack-and-stack or cardcage?

There are two major types of instruments for test systems, rack-and-stack and cardcage. Rack-and-stack instruments are standalone test instruments that can be used independently. For test systems, they are frequently stacked in a rack (hence the name) to save floor space, and typically, engineers use external PCs to control them. Newer LXI instruments often come in both traditional rack and stack formats as well as in smaller modular formats.

Cardcage instruments

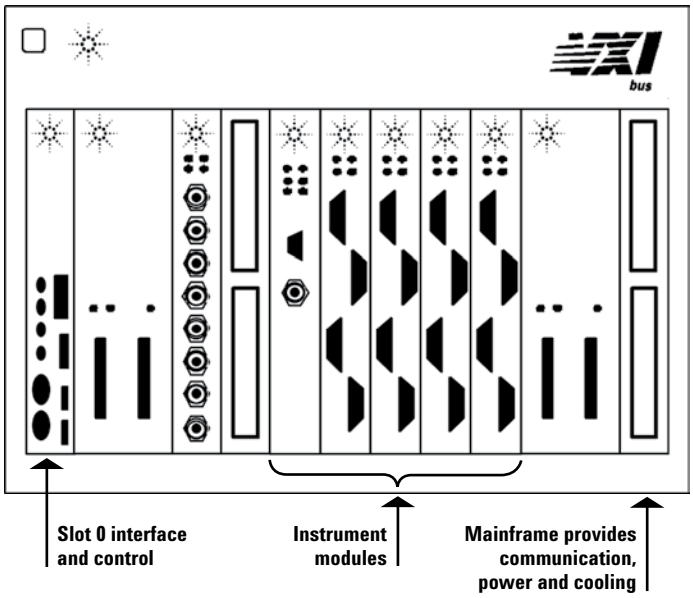
Cardcage instruments, as their name implies, are modular test instruments on plug-in cards. You insert the cards in a cardcage, or mainframe, and control them either with an embedded controller (a plug-in card that is a PC) or an external PC. Card-cage systems are often mixed with rack-and-stack instruments to provide all the functions needed in a test system.

VXI is a standard, open architecture for cardcage systems that allows instruments from different manufacturers to operate in the same mainframe (see Figure 5.2). The VXIbus (VMEbus eXtensions for Instrumentation) was developed by a consortium of test-and-measurement companies to meet the needs of the modular instrument market. The VXI standard was patterned after the VMEbus standard, but it was defined specifically as a new platform because VME did not meet the needs of the instrument community, particularly with respect to noise rejection and triggering. VXI instruments typically offer more performance and speed than other instrument types.

Another cardcage architecture is called PXI (PCIbus eXtensions for Instrumentation). While PXI cards are very small, they typically lack the accuracy and performance of VXI or rack-and-stack instruments. If you are considering using a PXI system, be sure to investigate whether you will need to purchase additional signal-conditioning equipment. Also, PXI is based on a PC backplane with no electromagnetic interference (EMI) or cooling specs, and therefore it is not as well suited to be a quiet measurement environment. Note that PXI is also transitioning to PXI Express, so be sure to look at your needs in the future to determine if you should purchase a hybrid PXI/PXI Express cardcage. See the sidebar on page 54 to compare attributes of PXI, VXI and rack-and-stack systems.

Another cardcage architecture is compact PCI (CPCI). CPCI technology is the basis for PXI, although PXI adds triggering options not available in PCI. CPCI and PXI cards can be interchanged to some extent. CPCI cards tend to be used in industrial PCs, because they are rack mountable and more rugged than other card types.

Figure 5.2. VXI mainframe with modular test instruments on plug-in cards



Racked instruments

Racked instruments can take up more space than cardcage instruments, but typically they are less expensive because they are produced in higher volumes. It is easy to find high-quality, high-reliability standalone instruments that are suitable for use in systems. Lately, test-equipment manufacturers have been putting more thought into how their standalone test instruments work in a system environment, making rack-and-stack architecture easier to implement. Agilent, for example, offers "system-ready" test instruments that incorporate standard protocols and optimized features like shielding, filtering, high-speed I/O and on-board intelligence and memory. Also with the large percentage of hybrid (cardcage plus rack-and-stack systems) and the introduction of LXI modular rack and stack instruments, there are more choices to optimize space vs. usability.

There are many benefits of using system-ready rack-and-stack instruments in your test system. For example, they can reduce your system development time because troubleshooting a system is easier when you use instruments that are capable of standalone operation. You can use an instrument in standalone mode to run preliminary checks to ensure you are getting good test results before you have the entire system set up. You cannot do the same with cardcage instruments, so it is more difficult to differentiate between hardware and software problems.

In some organizations, using a standard set of racked instruments throughout the product lifecycle can lower the barriers to effective communication and cooperation among organizations with different responsibilities. For example, R&D engineers may use benchtop instruments as they develop and fine-tune product designs. When they turn to design validation testing—or in the case of larger organizations, when they turn their pre-production prototypes over to the design validation department— it is helpful to use the same instruments, even though the tests are more likely to be automated or semi-automated at the design validation stage. If it is the same engineer doing the validation testing, he or she is already familiar with instrument operation and already trusts the test results the instrument generates. If R&D and design validation are handled by different engineers or different organizations, using the same test instruments can facilitate effective communication and shared problem solving. You get the same benefits if you use the same test system architecture when the product moves to manufacturing.

Making a choice

The decision you make about which instrument architecture to use will be influenced by several factors. If you are building a system from scratch, you will want to look at overall system performance and cost. However, if you already have a collection of either rack-and-stack or cardcage instruments, reusing them and adding to your collection may be more cost effective than starting over. Also important is whether you have access to rack-and-stack or cardcage systems-building expertise. If all the expertise in your company is with cardcage architecture, it may not make sense to switch to rack-and-stack, even if the equipment cost is less. If you decide to stay with an existing cardcage setup for your system, you may want to consider migrating to a hybrid system, adding rack-and-stack instruments to gain the capabilities or performance you need. You will need to evaluate the specific circumstances to make the best decision.

Another factor to consider is the cost of maintaining your system. Look into typical repair costs and the cost of keeping spare parts and extra instruments/cards on hand.

"Choosing instruments for your system" on page 59 offers more detailed information about choosing the right instruments for your system.

Comparison of instrumentation types

	Rack and stack	VXI	CPCI	PXI	See notes:
Standalone use	Yes	No	No	No	1
Accuracy	****	***	**	**	2
Price	$$	$$$$	$$$	$$$	3
Burst speed	** to ****	****	****	****	4
Single-point measurement speed	**	***	**	**	5
GUI response time	****	**	**	**	6
Footprint	**	**	****	****	7
Ease of use and integration	****	*	*	*	8
Shielding	****	***	*	*	9

1. Standalone use

With an internal PC, a cardcage can operate standalone, but you need a monitor if you require an operator GUI. Cost of an embedded PC is several times that of a standard PC. In any case, card cages generally require some form of computer communication in order to be useful, while rack-and-stack instruments can be used to check out the system without a computer present.

2. Accuracy

Cardcages have power supplies that must be shared among several subsystems. Rack-and-stack instruments are optimized to one use, so they are designed to have the right power supply for the job at hand, and analog circuitry that is not subject to cage-imposed restrictions. Rack-and-stack instruments are designed to minimize magnetic interference so they are less likely to induce currents that would disrupt sensitive instruments. As a result, rack-and-stack systems typically outperform cardcage systems in terms of accuracy, crosstalk, noise, and other factors.

3. Price

Cost of a bench-top system is usually lower when instruments are not rack-mounted. When instruments are rack-mounted, system cost depends on the configuration of the rack.

4. Burst speed

Burst speed is the speed at which the instrument can move a large amount of data from a single channel across some bus or I/O port to the computer. Burst communication is used in data acquisition more than it is used in functional test. Cardcages typically shine in this arena, although recent improvements in I/O speed, such as the adoption of fast LAN, have blurred the distinction between backplane and external I/O.

5. Single-point measurements

Single-point measurement speed is the time it takes to make a single measurement, switch channels and then make another measurement. This is the predominant mode used in functional test. You'll find more information about test-execution speeds in the "Measurement speed" section on page 60.

6. GUI response time

When a cardcage communicates to the PC, the PC must often do double duty as it processes the data and also updates the GUI. In some rack-and-stack instruments, these operations happen in parallel, giving the operator more real-time update capability. This is especially true with an oscilloscope, where lack of immediate feedback can be annoying.

7. Footprint

PXI and CPCI systems have the smallest footprints. However, many instrument functions are not fully realizable in PXI, so engineers typically adopt a hybrid approach of rack-and-stack plus PXI instruments. Once you have a rack for part of your system, you use the same amount of floor space as you would for a full rack-and-stack system, so you lose the space-saving advantage offered by the small form factor of the PXI cards

8. Ease of use and integration

If a racked system has been designed to accommodate a reasonable amount of expansion space (a good idea to plan for unforeseen future needs), adding instruments to a rack is not a lot more complicated than adding an instrument to a cardcage. A more important consideration is the ease of adding additional cables to an existing architecture. For example, whether you use a cardcage or several racked instruments, their inputs and outputs are usually connected into a switching system or a mass interconnect. If the system has been designed to handle such new instruments, integration will only take a few minutes. If the system has to be redesigned to handle the new instrument, it can take days.

9. Shielding

Dedicated rack-and-stack instruments are typically well shielded. Since they are designed for a specific purpose, they are frequently more noise-free than their card-cage counterparts. VXI has specific shielding specifications, and these are lacking in PXI and CPCI. While it is possible to shield PXI, the implementation is left up to the vendor, so placing a new vendor's product in a slot may result in unwanted interference with nearby instruments.

The computing subsystem

Before you consider the questions surrounding the computing subsystem, you need to decide whether you will control your system manually, semiautomatically or with a fully automated control system. These issues are addressed in Chapter 1, *Introduction to Test-System Design.* The information in this computing subsystem section is for test engineers who have decided to use either automated or semi-automated control.

For systems that use rack-and-stack test instruments, you will most likely use an external or racked PC that is cabled to the instrumentation. For test systems that use card-based instruments, you need to decide whether to use an embedded PC (one that fits inside an instrumentation cardcage) or an external PC. At first glance, the embedded PC may seem like a good choice. It fits inside an existing cage, so it uses rack space efficiently, and it is directly connected to the backplane, so data transfer speeds are excellent. Unfortunately, embedded PCs cost a lot more than external ones, and typically they do not have room to hold many modern peripherals. The technology used in embedded PCs tends to lag the technology of the general computer industry, so embedded PCs often are at least a generation behind in processor type and speed.

If you use an external PC, you will get more computing power for your money. In addition, most external PCs come with industry-standard interfaces like LAN, USB and FireWire built-in. If you use a PC with these interfaces, you can lower the cost of your test system by using test instruments that support these interfaces, or shorten setup time by using USB/GPIB or LAN/GPIB converters. This topic is covered in detail in Chapter 2, *Computer I/O Considerations.*

In manufacturing environments, cost is typically a critical concern, especially when you are implementing hundreds of identical test systems. The lower initial cost of external PCs typically makes them a better choice for manufacturing test systems, and the fact that they are typically less expensive to service than embedded controllers adds to their appeal. 1U or 2U rack-mountable PC controllers are now available that can be a good trade of size and cost.

Another major computing consideration is the choice of software and application-development and runtime environments. Computing subsystem decisions related to software are covered in Chapter 4, *Choosing the Test-System Software Architecture.*

Switching

Switches, or relays that interconnect system instrumentation and loads to your DUT, are an integral part of most automated test systems. Choosing the proper switch type and topology will impact the cost, speed, longevity, safety and overall functionality of your test system. For a thorough examination of switching in test systems, see *Application Note 1441-1, Test System Signal Switching.*

The types of relays you choose for your low-frequency switching subsystem are important, as they affect the type of circuits and systems you can test. Reed relays and FETs are the best choice for high-speed systems, and of the two, reeds have higher voltage and current ratings. Reed relays are excellent choices to connect measurement instruments and low-current stimulus to the DUT. They are very fast (typically about 0.5 to 1.0 ms), although they can have a higher thermal offset voltage than armature relays. Use armature relays (which typically switch in 10-20 ms) for higher-current loads. When you use armature relays, group your tests so the relays stay connected to perform as many readings as possible at one time. Because armature relays are relatively slow, you will want to avoid connecting and disconnecting them multiple times.

Switching topologies can be divided into three categories based on their complexity: simple relay configurations, multiplexers and matrices. The best one to use depends on the number of instruments and test points, whether connections must be simultaneous or not, required test speed, cost considerations and other factors.

A matrix arrangement of reed relays provides an excellent way to allow any instrument to be connected to any pin on your DUT, and it permits easy expansion as you add new instruments to your system or more pins appear on your DUT. Matrices use more relays than multiplexers, so they tend to cost more. If you don't need to connect multiple instruments to any pin, a multiplexer is a suitable solution. If you have a 1 x 20 multiplexer for example, you can connect a test instrument to 20 pins, but you can't hook anything else to those 20 pins. With those same 20 relays in a matrix, you can connect four instruments to five pins in any combination.

In manufacturing test and design validation systems you often need banks of general-purpose relays of varying current capability. You can use such relays to connect DUT inputs to ground or to a supply, or through resistors to simulate dirty switches. You also can use them to provide ways to disconnect output loads in order to allow parametric tests on output transistors, as shown in Figure 5.3.

You also need to think about where to place and how to arrange your switches. While relay cards can be placed in a cardcage that is intended for high-performance instruments, it is a waste of valuable real estate. The high-speed backplane in a modular cage is more suited to the control of high-speed instruments, not simple relays. If you place relays in a separate box that is tuned for that purpose, it will be easier to expand the high-performance instrumentation while allowing room separately for denser relay cards, more relay cards or a bigger switchbox. It also makes a clearer delineation between the instrumentation and the switching subsystems, which makes it easier to keep your system organized.

Placing the DUT interface panel (mass interconnect or feedthrough panels) in front of a switching subsystem that has the plug-in cards facing the interface panel accomplishes two goals: 1) It minimizes rack space, because the switchbox and mass interconnect are in the same plane, and 2) it reduces wire length from the switching to the DUT. If the box you choose has cards in the rear, reverse-mount the switchbox using the rails on the rear of the rack, as shown in Figure 5.4. There are two negatives to this approach: the front panel of the switching instrument is not accessible from the front of the system, and it can be harder to reach the plug-in cards for service. However, once a system is operational, it is seldom necessary to operate a switchbox from its front panel, and cards can be accessed by pulling the instrument out the back or by removing the side panel of the system.

Figure 5.3. Switched loads allow parametric measurements

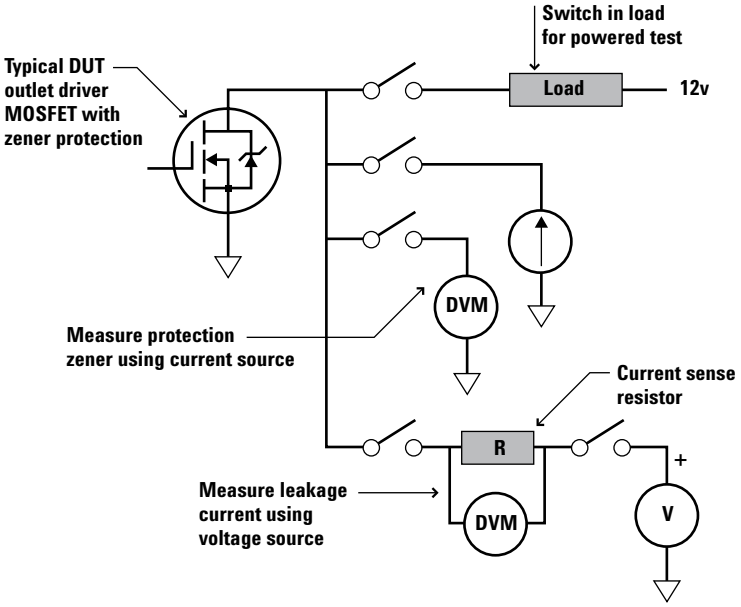


Figure 5.4. Rear-mounting the switching subsystem reduces rack space and minimizes cable lengths

Tips for successful switching

1. Place system switching in a box dedicated for that use, such as the Agilent 34980A switch/measure unit or the 34970A data acquisition/switch unit. Placing all system switching in one place minimizes cost and helps to keep your system organized. Allow enough room to expand the switchbox to a larger size or to provide room for another one as your needs grow.

2. Inside the switchbox, create an instrumentation matrix. For example, create a 16 x N switch matrix, connecting instruments to the 16 "rows", and your DUT to the "N" (column) side, allowing one matrix column per DUT pin. By making N an expandable number, in increments of, say, 16 or 32, you can handle modules that are close to your immediate needs with a way to easily expand to higher-pin-count modules in the future. When you need new instruments, simply connect them to a new set of rows. No additional wiring is needed. Since most instrumentation is low current and must be scanned across multiple points quickly, choose fast reed relays or FET switches for this architecture.

3. Also inside the switchbox, allocate a set of general-purpose relays for power supply and load connections. These relays are generally too big to allow economical creation of a high-current matrix that could programmatically assign any DUT pin to any load. Therefore, bring such relay connections out directly to an interface panel where they can be connected to the appropriate pins. When you are designing the switching for your test system, you may want to build in some safety features. Particularly if you are working with high voltages or high currents, you might want to include a switch to disconnect all signals, to minimize the chance for potentially serious accidents.

Mass interconnects

A mass interconnect panel is a DUT-to-system wiring interface that allows you to use fixtures instead of wiring each connection separately. When you are designing a functional test system for a design lab, it is tempting to leave out a mass interconnect, since the product design changes so much and the extra time to rewire a fixture is not productive. It also is not as likely that you will make identical measurements on large numbers of devices. Simple clip leads may suffice, especially for small DUTs. Interface panels are relatively expensive—using one can easily double the cost of a system— but there are a couple good reasons for adding one to your design-validation, production-verification or manufacturing test system:

- A mass interconnect provides a physical location for mounting interface components such as terminal blocks, fuses, custom electronics/interfaces/conditioning, etc., between the system and the DUT. You can mount these components either to the interface frame or to a shelf attached to the frame.
- Device measurements are less likely to change due to random movements of wires.
- Using terminal blocks on the interface makes it easy to make wiring changes as the DUT changes, allows easy connection of multiple resources to common points, and provides easy test connections for debugging the system.

For design validation, production validation and manufacturing test, mass interconnects are typically well worth the investment. They provide a fast and robust means of changing connections to different DUTs using the same system.

You can obtain more information about mass interconnects from the three major manufacturers: Virginia Panel, MAC Panel and Everett Charles Technologies/TTI Testron.

Power sources

DUT power is an integral component of a test system, whether it is a simple bias supply or an advanced system power source. Depending on your application, your DUTs can require anything from a few milliwatts to many kilowatts. There are many power supplies available for providing power to a DUT. Choosing the right one is more complicated than simply picking the right voltage and current level.

Testing your DUT will be a lot less frustrating if you choose a reliable system power source that provides a stable voltage source to power the DUT and built-in measurement capability to verify DUT performance under various operating conditions.

When you select your DUT power source, consider the following:

- Number of outputs needed
- Settling time
- Output noise
- Fast transient response
- Fast programming, especially down-programming response
- Remote sensing—compensate for voltage drop in wiring

- Built-in, accurate, voltage and DC current measurement or waveform digitization
- Small size—it's possible to get linear performance (low noise) out of a switched power supply to free up rack space
- Triggering options
- Programmable output impedance
- Multiple outputs and sequencing of outputs
- Over-voltage protection
- Over-current protection
- Lead lengths
- Safety due to exposed voltages

Your choice of supply can dramatically impact system throughput, since waiting for power supplies to settle can be one of the most time-consuming elements in a typical test plan.

DUT-specific connections

Many DUTs require components to be connected to their outputs in order to adequately stress the unit (Figure 5.5). These can take the form of resistive or reactive output loads such as resistors, light bulbs or motors, or complicated, simulated loads such as the dynamically varying current in a camera battery. In most cases, it is wise to provide a place to put such loads in a system, such as a slide-out tray on which small, discrete loads can be mounted. Some DC-programmable loads (the size and shape of a power supply) can be rack mounted. Such loads are often connected to the DUT through relays to allow the DUT to be completely disconnected from all test system resources. If you decide to use relays, locate the loads close to the switching subsystem to minimize cable lengths.

Other architectural considerations

In addition to the foregoing decisions, make sure your planning also takes into account AC power distribution, cooling, ergonomics, safety, and future expansion.

AC power diistribution

If you are designing a system that you expect to replicate and ship to areas of the world that have different power requirements, you will probably want to include a power distribution unit in your system to make it easier to convert to the appropriate scheme. Power distribution units give you a way to route power, detect power line problems, and filter the input, and they provide the potential for adding uninterruptible power supplies and an emergency off (EMO) switch input.

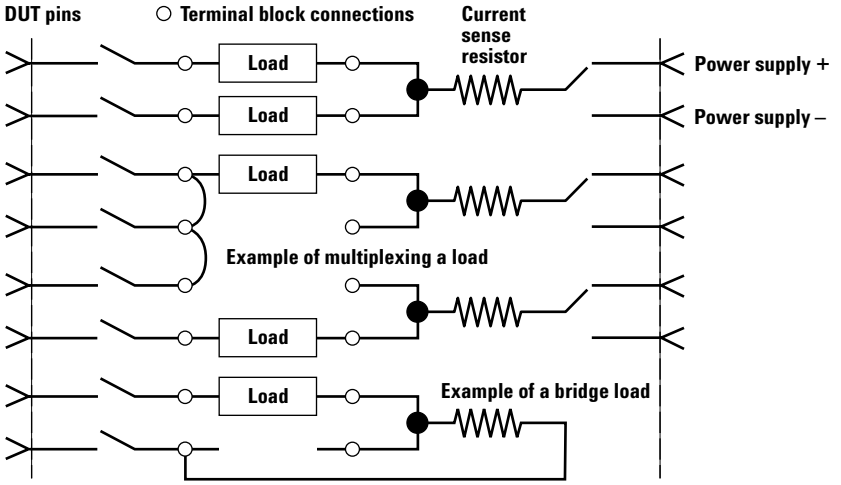
Cooling

If you do not pay attention to cooling, temperatures in a rack can easily exceed environmental conditions specified for your test instruments. When this happens, your instruments can fail prematurely and your measurement results can be jeopardized. Temperature gradients are also something to consider. If one end of the rack is ten degrees hotter than the other end, even if the overall temperature is within instrument specifications, the resulting gradient can cause some unwanted thermocouple effects or slow drift errors.

You can use extractor fans to draw air through your system to remove heat. If you cannot create enough airflow to remove the heat with a fan, you may need to consider air conditioning your rack. There are standard NEMA enclosures that can be used for this purpose.

If you are using rack-and-stack test instruments, it is important to think through how you place the instruments in the rack. Test instruments typically pull air in on one side or through the bottom and exhaust hot air out the other side or the top. Be careful not to position an instrument's air intake adjacent to another instrument's exhaust vent. You will find more information about racking test instruments in Chapter 6, Understanding the Effects of Racking and System Interconnections

Figure 5.5. Simplified diagram showing ways you can connect loads in various configurations. A "bridge load" connects a load between two pins on the DUT, rather than between an output and ground or an output and power.



Ergonomics

As you make decisions about your system architecture, keep in mind the operator's comfort and convenience. Provide adequate work space at the correct height, depending on whether the operator will be sitting or standing. Put displays at a comfortable height and if appropriate, provide the ability to tilt the display to reduce glare and eyestrain. Make sure illumination is adequate for the tasks that need to be performed. Provide for left-handed and right-handed operators by allowing a mouse to be placed on either side of the keyboard.

Safety

If you are working with high voltages, consider using interlocks to prevent accidents. Take precautions to deal with static electricity. For moving parts that could cause bodily harm, consider using deadman switches (two switches, both of which must be engaged for the equipment to run) and EMO switches (a single switch to turn off the entire system in an emergency). Position heavy equipment low in the rack and watch how you distribute weight in the rack to prevent it from tipping over. Also consider how weight distribution would change if you were to remove an instrument for maintenance.

Future expansion

To maximize the re-usability of a functional test system, you need to design it in such a way that in the future it will be able to accommodate more instruments, more switches and bigger DUTs that require more power, without a complete re-design. To maximize your long-term flexibility, use open standards whenever possible. Make sure to allow 20 percent to 30 percent extra room in a cardcage, or 20 percent extra room in your rack to accommodate instrument additions. See Chapter 1, Introduction to Test-System Design, for more ideas about planning for future expansion.

Choosing instruments for your test system

The measurement and stimulus instruments you choose for your system—whether they are rack-and-stack instruments or instruments on a card—will be driven largely by the functional and parametric tests you need to perform, and whether you are using manual, semi-automated or fully automated control for your test system.

Identify your needs

In all cases, it is wise to start by making a thorough list of the inputs and outputs of each of the devices you plan to test and the parameters you will measure. Note the accuracy and resolution you need for each measurement as well. Once the list is complete, check to make sure it does not contain redundant or unnecessary tests. Then identify possible test instruments for the required measurements and look for opportunities to use the same piece of test equipment for multiple measurements.

The types of instruments you need will vary depending on your application. However, there are several universal questions that you must answer in order to select measurement and stimulus instrumentation properly:

1. AC stimulus. How many dynamic (AC) signals do you need to apply simultaneously? This determines the number of channels of arbitrary waveform or function/signal generator you require.

2. DC stimulus. How many static (DC) signals to you need to apply simultaneously? This determines the number of channels of DAC (digital-to-analog converter) you will require.

3. Measurements. What types of measurements do you need to make, and how many simultaneously? If minimizing instrumentation costs is essential, look for ways to minimize the number of instruments you need by paying attention to the ancillary functions of instrument that might perform double duty. For example, you can perform RF power measurement with a spectrum analyzer if accuracy and speed are not critical to your application. If you only need to know the power supply voltage within 0.5 percent, you might be able to use the internal voltmeter inside your power supply, using the read-back mechanism to read voltage on terminals.

4. Protocols. Do you use any special serial data protocols? This determines the need for instruments to handle things such as CAN, ISO-9141, J1850 and many more.

Once you have made your measurements list and answered these initial questions, you can refine your list of instrument possibilities by looking at your budget and time constraints and your requirements around measurement speed.

Development time

When you are choosing instruments for your test system, look for instruments that will minimize your development time. You can save time by using rack-and-stack system-ready instruments that incorporate a high percentage of the measurement solution you need. For example, if you use a source with modulation capability, you don't have to develop your own algorithm or integrate additional hardware to generate the required modulation.

If you want to minimize hardware costs, you can investigate auxiliary capabilities. However, if your goal is to minimize development time, buy instruments that are specifically designed to do the jobs you need done. Using instruments with IVI-COM drivers can save you development time. If the instrument has an IVI-COM driver, you can interchange hardware without rewriting your software, as long as you adhere to the functionality that is specific to the instrument class. See Chapter 3, Understanding Drivers and Direct I/O, for to learn how decisions about drivers affect development time.

Test instruments with downloadable personalities also can save you development time. You download the measurement personalities for a specific application directly into the test instrument's internal memory. Then you can simply choose from a menu of tests, and the personality's "intelligence" automatically performs the tests, from capturing signals to displaying results. Agilent spectrum analyzers, for example, have measurement personalities for testing cable TV, phase noise, cable fault, Bluetooth™, cdmaOne, GSM/GPRS, as well as a variety of other wireless protocols and modulation.

New LXI instruments from Agilent allow instrument monitoring from the instrument web page. This allows monitoring of the instrument state from the same computer screen as your test program. The web page is also a useful debugging tool.

You typically spend a large percentage of total development time on debugging your system, particularly if you are building a new test system. You can reduce your debug time significantly by writing a diagnostic test routine that loops outputs back to inputs through a large part of the switching path. This exercise will help you quickly identify the cause of problems– whether it is a source, a measurement instrument or a switch path.

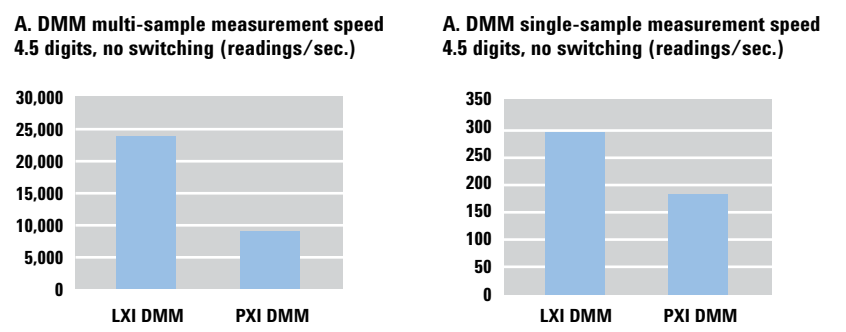
For more ideas on minimizing your development time, see Chapter 4, Choosing Your Test-System Software Architecture.

Measurement speed

If you are building a manufacturing test system (and to a lesser extent in design validation applications), the time it takes to execute each test can be critical. But figuring out how fast your system will perform measurements is harder than it appears. For example, a digitizer may be able to sample 1000 readings very fast, but if those readings are transferred to the PC over GPIB, it could take a long time. A digitizer that can have a decision-making algorithm downloaded into it could allow a simple go/no-go result to be sent back to the PC, which would make GPIB a reasonable option and may save money over a cardcage-based solution. However, it takes extra effort to create and download a decision algorithm into an instrument, which may increase development time as well as "first-run" time of the test program. Also, inside an instrument the readings will be analyzed by a much slower processor than the one in the PC, so this must be factored in as well.

Simply reading the data sheet does not tell the whole story. Maximum reading rate specifications are usually related to burst speed (see Figure 5.6); that is, the speed which you can sample the signal on a single channel. But that is not the typical mode for functional test. In functional test, the system normally makes a single measurement, then changes a parameter like range or function or channel, and then makes another measurement. In this case, the burst rate is meaningless. Take for example, two multimeters—one LXI and one PXI. Note that both multimeters can perform up to 10,000 measurements/second or more in burst mode, but their single-sample measurement speed is much slower due to the transaction overheads of controlling each measurement. Even a high-speed bus such as PXI makes little difference to the readings/second because the total time is dominated by the setup and measurement time.

Figure 5.6. Burst speed can be misleading; since single-sample measurement speeds are usually significantly lower.



At higher resolutions, burst rate again becomes moot, since actual reading rates are a function not only of DMM sampling times, but also of relay switching times. Since such reading times can be generally less than 10/s, these readings tend to be done only when the extra resolution is absolutely necessary.

For a discussion of how data transfer rates over different interfaces affect your system's overall measurement speed, see pages 23-24 in Chapter 2, Computer I/O Considerations. For a detailed look at ways to maximize your system throughput, see Chapter 7, Maximizing System Throughput and Optimizing System Deployment.

Choosing a vendor

The proper design of instrumentation requires attention to minutiae. Choose an instrument manufacturer who has been through the learning process and knows how to minimize system noise and maximize accuracy and throughput.

Simple systems are one thing, but when you put several instruments together, strange things sometimes happen. That's when it's nice to have local support and service. Choose a vendor who can help you with issues like repeatability, system noise, calibration and drift.

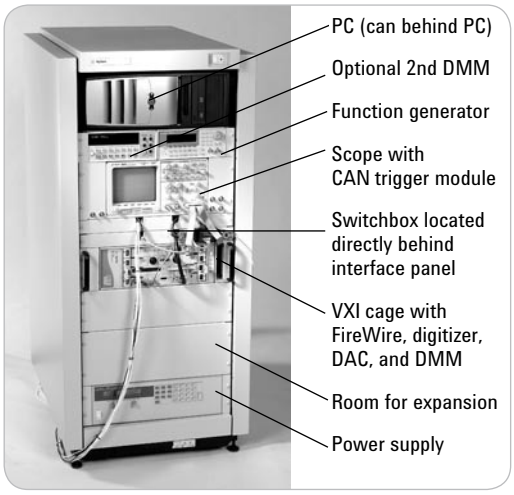
If your vendor can supply specifications that apply to a whole subsystem—like a central switch— it will save you the time and trouble of trying to add all the specifications of a multitude of vendors together to divine what the true accuracy of your system might be.

Calibration can be an expensive and time-consuming part of building a system. Make sure you don't have to ship your system halfway across the world to get it calibrated. Calibration is especially important in the world of RF and microwave, so make sure your vendor's support organization can handle your needs.

Example test system

To illustrate the concepts and issues discussed in this chapter, we will design a test system (see Figure 5.7) from scratch that can be used to test low-frequency, low/medium-pin-count, low/medium power electronic modules. These devices are typical of the automotive and aerospace/defense industries.

Figure 5.7. Functional test system



Make architectural choices

Table 5.1 shows the architectural choices we made for this test system.

Design the system

Now, we will apply the architectural decisions to a system for testing an electronic throttle module for an automotive throttle body. According to the test specification, the following equipment is required to run the tests:

- Programmable volt/ohm/ammeter
- Programmable power supply—0-13.5 V/0-10 A
- Waveform generator capable of pulse-width modulation, 0-10 VDC, 0-3 KHz
- Low current DC voltage source (0-5 VDC)
- Waveform analyzer
- CAN interface
- Simulated or actual stepper motor load

The DUT has 14 pins total on 3 connectors. Looking at various catalogs, and adopting the architecture specified earlier, we chose the instruments shown in Figure 5.7.

There are three LXI instruments—the power supply, switchbox, and oscilloscope. We will use an 8-port LAN hub providing extra ports, thus "future-proofing" the system. Table 5.2 lists the instrumentation used in this sytem.

Our system uses many I/O interfaces: LXI (LAN), RS-232C, FireWire and GPIB. Using Visual Studio.NET with IVI-COM and VXI*plug&play* instrument drivers along with VISA I/O libraries, the control program can communicate easily with instruments on all of these interfaces. In fact, should an instrument's I/O interface ever change (say from FireWire to LAN), all that will have to change in the program is the initialization string. It is also possible to specify use of an aliased name to eliminate the hard-coding of I/O addresses.

Figure 5.8 shows how the instruments will be connected to the switching subsystem. We are using a matrix, so any instrument can be connected to any DUT pin, and we

Table 5.1. Architectural decisions for sample test system

Subsystem	Decision	Reason
Instrumentation (measuring and stimulus instruments)	Mix card-based and rack-and-stack instrumentation	Most cost-effective solution; helps optimize system
	• Use VXI for higher-speed DMM, multi-channel DACs, and digitizer	Maximize system speed; digitizer not available as rack-and-stack instrument
	• Use rack-and-stack for other test instruments	Accuracy, ability to prototype system before writing code
	Allow about 20%-30% extra rack space for rack-and-stack instruments	Allow for future expansion
	For card-based instruments, leave either 20% expansion room in the cage, or room in the rack for a bigger cage	Allow for future expansion (expected need for bigger switchbox and/or more power supplies)
	Use a rack with a top-exhaust cooling fan access anywhere in rack	Hot air rises, and top fan does not interfere with
Computing (computer, software and I/O)	Use an external PC, not an embedded PC	Lower cost, standard interfaces
	Use only industry-standard interfaces	Easier support
	Use FireWire interface to control VXI instruments	For speed
	Use Microsoft Visual Studio.NET software	Rapid development
Switching (relays that interconnect system instrumentation and loads to the device under test, or DUT)	Place switching into a separate subsystem	Separate cardcage-based switchbox houses low-data-rate instruments more cost effectively
	Use a matrix switching architecture for measurement instruments and low-current stimulus	Ease of expandability, more flexibility in where instruments can be connected
Mass interconnect (DUT-to-system wiring interface)	Place the DUT interface panel (mass interconnect or feedthrough panels) in front of the switching subsystem	Minimize cable length, save rack space
Power sources (power to the DUT)	Use high-current power supply and allow room for more than one in the rack	DUT requires high current. Bigger DUTs are expected from R&D in the future
	Consider a modular power source	Greater flexibility
DUT-specific connections (loads, serial interfaces, etc.)	Connect high-current DUT pins to general-purpose relays that can be wired to power supplies and loads	Ability to disconnect loads from DUT to allow other measurements to be made on those pins

can add new instruments easily by expanding the number of rows and columns. All connections to the DUT except for the CAN bus are switched, making it possible to measure continuity from pin to pin. We are using a star ground to avoid ground loops.

A mass interconnect is an option for this system. This particular DUT only has 14 pins, so in an R&D or design validation environment you may not require the flexibility provided by such an interface. If the number of pins is small, simply bringing them directly out of the switchbox to DUT connectors may be sufficient. In the future, if the modules you are testing have more pins, or if you need a place to put other things between the system and the DUT, you may need a commercial mass interconnect solution. Therefore, we will provide a space directly in front of the switchbox for such an interface.

We chose a 5-wire measurement bus because it allows all four leads of the DMM to be connected to different pins on the DUT, making 4-wire ohms measurements possible. We routed two matrix points to the same pin on the DUT (as shown in Figure 5.8 on the Pot1 and Pot2 Gnd pins), to make the resistance measurement very accurate, since the remote sense location is made right at the DUT. If you don't use two wires, you can still make a 4-wire ohms measurement inside the relay matrix, which in some cases may be good enough. The fifth bus wire is connected permanently to the star ground, and so it serves as a common reference for any single-ended devices, such as the oscilloscope, or for floating devices that can be connected to ground, such as the function generator, digitizer, DAC and DMM.

When you use a matrix, you can connect multiple signal sources to the same pin. It is important not to accidentally short such sources together. Switching routines should be carefully written to either eliminate this possibility or to offer warnings when improper conditions occur.

If you need to power up and run the DUT in full-functional mode, you may need to modify the test system either with more instrument busses or with more devices connected directly to the DUT. You must carefully analyze the type of testing that is required and plan accordingly.

Figure 5.8. Block diagram of system

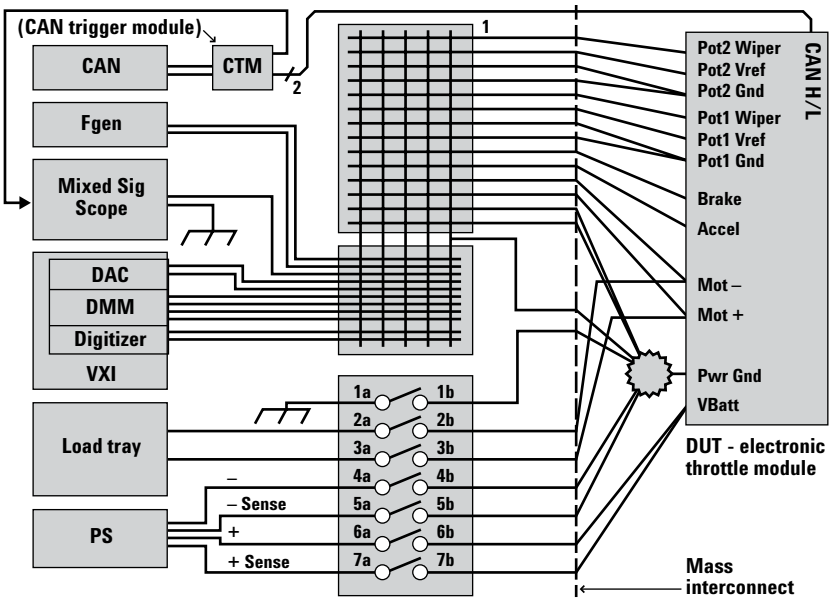


Table 5.2. Instrumentation decisions for sample test system

Instrument	Reason
Rack-mountable arbitrary waveform/ function generator	Need to generate PWM signals inexpensively
Heavy-duty power supply	Module requires 10A of inrush current
Optional DMM	Debug
Oscilloscope with CAN trigger module	Monitors signals including CAN traffic
Dedicated switching cardcage ("switchbox")	Separate cardcage-based switchbox houses low-data-rate instruments more cost effectively
4-slot VXI cage containing:	Provides the most channels in a reasonable form factor; space for future expansion
• Digitizer	For high-resolution sampling
• 16-channel DAC	Need a DAC for generation of a brake signal
• High-speed DMM	Actual measurements are fastest with this one
• An RS- 232C-based CAN interface is located on a shelf behind the PC	Module requires CAN interface for putting module in test mode

It is helpful to make a wiring map that shows how the DUT will connect to your system. Table 5.3 shows how to make one using a spreadsheet. In the future, when it becomes necessary to test a different DUT, all you need to do is to create a new spreadsheet and wire the new DUT accordingly.

Since the system has many resources available and they can be expanded without changing the basic system architecture, new DUTs are easily accommodated. The spreadsheet is constructed with DUT pin names and numbers in the rows and system resources in the columns. Since star ground is physically located outside of both the system and the DUT, it shows up in both a row and a column. Wires are connected from the DUT pin number to the relevant system resource. For example, the battery input, Vbatt (J1-1), has two wires attached to it—one to general-purpose relay 7b and one to general-purpose relay 6b, which puts remote sense of the power supply right at the DUT. In addition to DUT pins, there are other internal system connections that must be made, and they are shown in a separate section of the spreadsheet.

Table 5.3. DUT wiring spreadsheet

		System Resource Name				
DUT Pin Name	**Pin Nr**	**Matrix Col**	**GP Relay**	**CAN H**	**CAN L**	**Star Ground**
Vbatt	J1-1		7b (PS+sense), 6b (PS+)			
Power Gnd	J1-2					X
Brake	J1-3	9				
Accelerator	J1-4	10				
CAN H	J1-5			X		
CAN L	J1-6				X	
Pot1 Vref	J2-1	6				
Pot1 Wiper	J2-2	5				
Pot1 Ground	J2-3	7,8				
Pot2 Vref	J3-1	2				
Pot2 Wiper	J3-2	1				
Pot2 Ground	J3-3	3,4				
Motor +	J3-4	12	3b (load 1)			
Motor –	J3-5	11	2b (load 2)			
Other connections						
PS+Sense		7a				
PS+		6a				
PS-Sense		5a				
PS –		4a				
Motor Load +		3a				
Motor Load –		2a				
Earth Ground		1a				
Switched Earth Ground		1b				X
DUT Common						X
Star Ground		13,14	5b (PS-sense), 4b (PS-) X			X

Conclusion

Before you begin choosing test instruments for your test system, you need to make a series of high-level decisions about your system architecture. The architecture you choose for your test system will depend on whether you plan to use it for R&D, design validation, or manufacturing test and on your budget and development-time constraints, your existing expertise and your measurement throughput requirements.

Important questions to consider include the following:

1. Should you use a rack-and-stack, cardcage or hybrid (combination) architecture?
2. If you decide on card-based instruments, should you use an embedded PC (one that fits inside an instrumentation cardcage) or an external PC?
3. Which switch topology—simple relay configurations, multiplexers or matrices—and which switch types (reed relays, FETS or armature relays) should you use?
4. Does a mass interconnect make sense for your system?
5. Which power supplies and loads should you choose?
6. Which measurement and stimulus instruments should you choose?
7. What should you do to minimize your hardware costs?
8. What should you do to minimize development time?
9. What should you do to maximize system throughput?
10. Which hardware vendor should you use?

If you answer these questions carefully, you will help you ensure that your test system produces reliable results, meets your throughput requirements, and does so within your budget.

6. Understanding the Effects of Racking and System Interconnections

Introduction

This chapter walks you through important considerations for arranging your test equipment in a rack, including weight distribution, heat dissipation, instrument accessibility and operator ease of use. It also explores ways to minimize magnetic interference and conducted and radiated noise to maximize measurement accuracy.

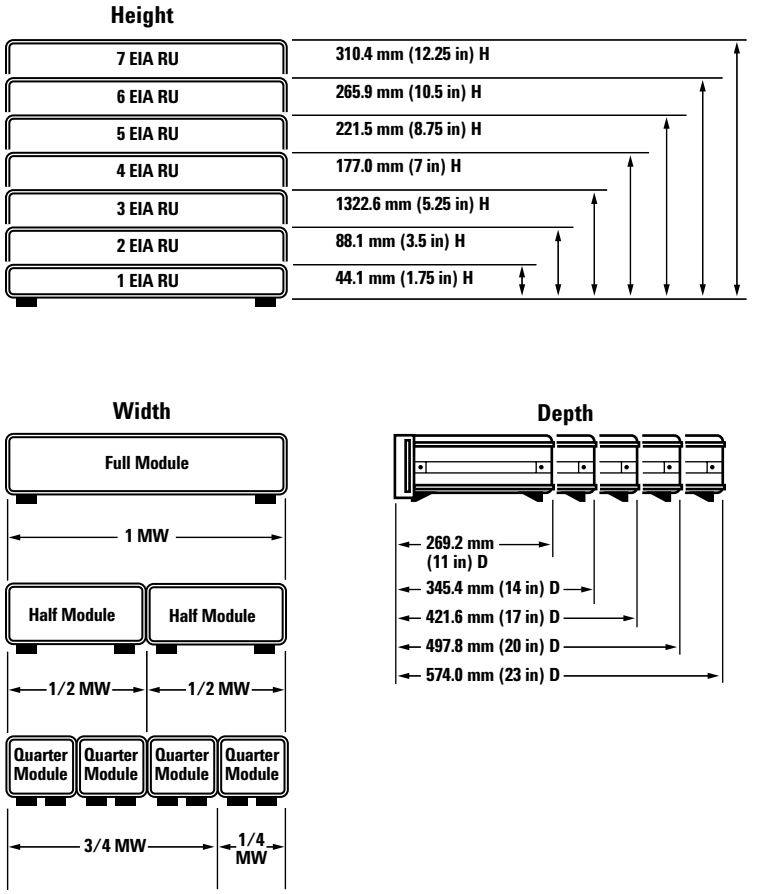
How you arrange test-system components can affect measurement accuracy, equipment longevity and operator ease of use and safety. This chapter focuses on the important decisions you'll make if you are building a system from rack-and-stack test instruments or a mixture of rack-and-stack instruments and cardcage components, and you are using a racking cabinet to hold your system components. However, many of the concepts we discuss are applicable to bench-top systems that are not racked.

Choosing racks and accessories

Before you choose your rack cabinet and accessories, you need to clearly define the quantity and size of the components your rack will house. It is also important to be aware of how users will interact with the equipment, how the equipment will be maintained and any special needs such as environmental or security considerations or the need to transport your system after it is built.

To facilitate racking, most test equipment manufacturers build test equipment according to size standards established by the Electronic Industries Alliance (EIA). The standard heights, widths and depths are illustrated in Figure 6.1. Instrument widths are usually specified as full module width (MW) or half or quarter MW.

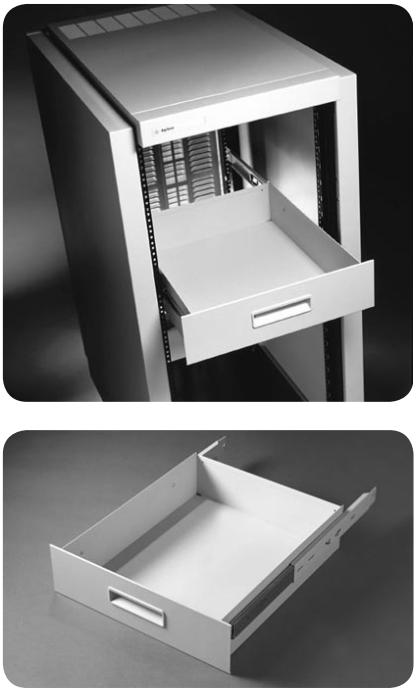
Figure 6.1. Most test instruments are a whole number of standard rack units (RUs) high and either a full, half or quarter module wide. A full module is typically 482.6 cm (19 inches) wide.



When you calculate rack size, you need to decide whether the system controller (typically a computer) and monitor also will be installed in the rack to display test procedures and results. If you are incorporating a computer and monitor, will you also need a keyboard or mouse for operator inputs? If so, be sure to add space for these items into your calculations, along with space for a work surface. If there is a work surface, consider the fact that it may prevent the user from easily accessing any instrument in the space directly below the surface.

You may also want to consider including space for accessory drawers to provide convenient storage for manuals, spare connectors and other small accessories (see Figure 6.2). Slide-out shelves are useful for attaching loads and other custom equipment, and they make access easy.

Figure 6.2. Adding an accessory drawer to your rack provides convenient storage for manuals, spare connectors and other small accessories.



To maximize the re-usability of your test system, keep your future needs in mind when you choose your rack. In the future, you may want to add more instruments and more switches and accommodate bigger devices under test (DUTs) that require more power. To maximize your long-term flexibility, allow at least 20 percent extra room in your rack to accommodate instrument additions.

Other questions to consider:

- What are the physical constraints of the location where your rack will be situated? Will the floor support your system's weight? Are doorways into the facility tall and wide enough for the rack you are considering? Is there adequate power, and does the room have adequate cooling to support the additional heat created by the system?
- Will your system need to be moved to its final destination? If so consider using multiple smaller racks and limiting total rack weight. If you need to ship the system to another location, also consider using ruggedized rack furniture with strain relief fittings and keep shipping concerns in mind (shipping company or airline size and weight requirements, etc.).
- Do you need to be able to prevent or limit access to your system? If so, consider a rack with lockable doors.
- Will you need rear access to your equipment? If the only way to gain rear access to your equipment is to move your rack, you may want to consider installing sliding shelves instead. A sliding shelf allows you to pull the instrument out of the front of the rack for easier access to the backside of equipment.

Instrument layout

When you plan the layout of equipment in your rack, you will attempt to achieve a number of objectives simultaneously:

- Ensure rack stability by carefully distributing the weight of system components in the cabinet
- Make it easy for operators to use the system and be productive
- Minimize magnetic interference
- Provide adequate power and heat dissipation
- Route power and measurement and stimulus signals to the right place as efficiently as possible
- Minimize conducted and radiated noise
- Ensure operator safety

Plan your instrument layout on paper before you start installing instruments in your rack, since you will probably change your layout multiple times before you determine the optimal layout.

Proper weight distribution

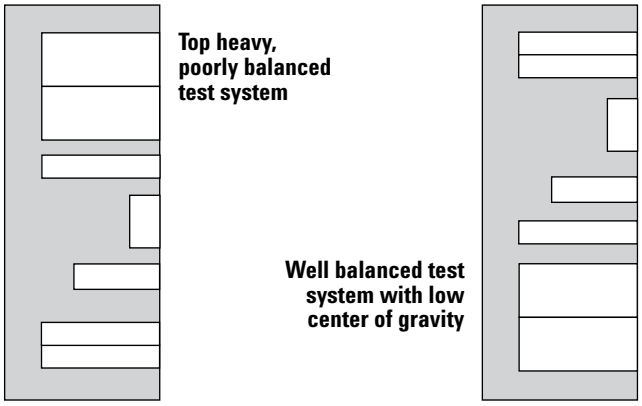
It is important to minimize the risk of your rack tipping over to prevent injury to operators and damage to expensive equipment. To achieve the greatest stability for your rack, keep the center of gravity low by placing the heaviest objects—typically power supplies and signal generators—near the bottom of the rack (see Figure 6.3). You will have to balance this need with the need to make frequently adjusted equipment easily accessible to operators.

In addition to keeping the center of gravity low, make sure the weight of your system is centered (front-to-back and side-to-side) as much as possible. You may need to mount some system components in the back of the rack, rather than the front, to achieve this balance.

When you calculate your system's center of gravity, be sure to factor in the weight of the heaviest DUT you will be testing. Your system needs to be stable with and without the DUT in place. Also consider how weight distribution would change if you were to remove an instrument from the rack for maintenance, if the operator were to lean on the work surface or place heavy manuals on it, or if heavy instruments on slide-out rails were fully extended.

If you have allowed room in the rack for future expansion, you will have empty spaces in the rack. To improve weight distribution, leave some empty spaces near the top of the rack for future addition of lightweight instruments and some at the bottom to allow for future addition of heavy instruments. Use a filler panel to cover the front of the rack to keep dust out of your system and help manage airflow. Filler panels come in the same standard heights as test instruments (see Figure 6.1).

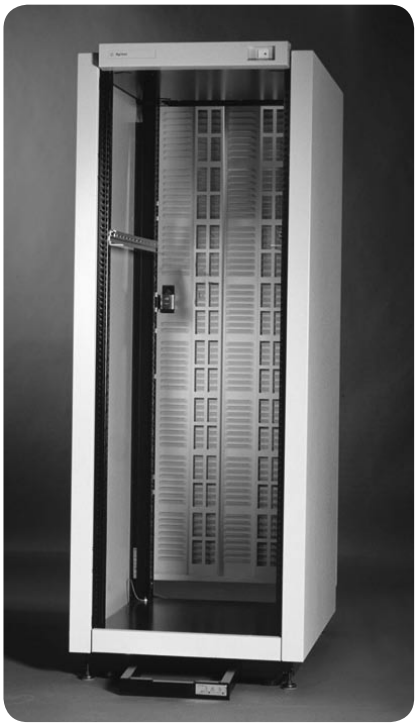
Figure 6.3. Well balanced and poorly balanced test systems



Keeping the center of gravity low is especially important if you will be moving the rack to another location after it is assembled, because the risk of tipping increases when you move it. Of course, the forces acting on your system's center of gravity will change if the system is tilted, so be sure to take this into consideration if you intend to move your system up a ramp as you move it to its final location. When you design your rack, keep in mind that ramps in industrial facilities can be angled at up to 15 degrees, so make sure the rack cannot tip over at that angle. When you push the rack up the ramp, turn the rack so the heaviest part (typically the front if your equipment is front-mounted) faces uphill, if possible.

Once your system is in its final location, you can improve its stability several ways. You can bolt it to the floor, to a wall or to another test rack. If you bolt it to another rack or to a wall, make sure you do not disturb the airflow and cooling and that you leave enough room at the back of the rack for servicing equipment. Some racks are equipped with retractable stabilization feet that you can pull out of the bottom front of the cabinet to prevent it from tipping forward (see Figure 6.4).

Figure 6.4. This rack cabinet features a retractable anti-tip foot that improves the rack's stability when it is loaded.



You also can use ballast, or weights that fasten to the bottom of the rack, to improve rack stability. Most racking systems offer ballast as an option. Ballast mounted at the back of the rack cabinet helps keep the cabinet from tipping forward if you extend heavy, slide-mounted devices from the rack or if you place a heavy object on a work surface that extends from the rack.

Adding ballast, using retractable stabilization feet and bolting rack cabinets to the wall or floor provide an extra margin of safety, but you should not rely on these measures to compensate for poor weight balance in your rack. Always make sure the center of gravity of your system is as low as possible and the weight of your system is centered as much as possible.

Instrument accessibility and operator ease of use

If your system is fully automated, you may be concerned about instrument accessibility only during system development or troubleshooting. If your system is operated manually or semi-automatically, an operator's ability to access instruments and use them easily during testing will be an important consideration as you decide how to rack your equipment.

Instrument access during development and/or troubleshooting

When they are low on rack space, system designers sometimes "bury" instruments inside the rack behind other instruments or mount them backwards or sideways in the rack. Before you choose this tactic, determine if you will need to access the instrument during system development to verify operation or for troubleshooting, repair or calibration. If you perform periodic system self tests to verify operation, you may need access to the front panel of an instrument, making "buried" installation impractical.

In some situations, reverse-mounting (or rear-mounting) instruments in a rack makes sense. For example, if you place the DUT interface panel (mass interconnect or feedthrough panels) in front of a switching subsystem that has the plug-in cards facing the interface panel, you minimize rack space, because the switchbox and mass interconnect are in the same plane, and you reduce wire length from the switching to the DUT. If the switch box you choose has cards in the rear, you can simply reverse-mount the switchbox using the rails on the rear of the rack, as illustrated in Figure 6.5. If you choose to mount an instrument in a non-standard manner, be sure the cooling airflow is not disturbed.

You may be able to rear mount shallow instruments behind front-mounted instruments to save rack space. This space-saving technique can be a practical way to reduce rack height if you have a problem with low doors or you need to meet airline size requirements. However, mounting instruments in both the front and back of a rack can make servicing the instruments in your rack more difficult.

Figure 6.5. Rear-mounting the switching subsystem reduces rack space and minimizes cable lengths.

Instrument access and ease of use during testing

If you are designing a manual or semi-automated system, you need to ensure that the operator can reach the necessary equipment controls and connectors/patch panels without straining. Decide whether operators will sit or stand during testing and position the work-surface height accordingly. If a test instrument has a display the operator needs to see, place it at eye level or above, and if appropriate, provide the ability to tilt the display to reduce glare and eyestrain.

If the operator will interact with a computer, place the monitor where the operator can see it easily. If the operator needs to use a mouse or keyboard, avoid placing these items on the same work surface as the DUT. Provide for left-handed and right-handed operators by allowing a mouse to be placed on either side of the keyboard.

When you are planning the operator work surface, make sure operators sit or stand far enough away from the rack that they do not inadvertently hit controls with their feet.

If you plan to ship the rack to another country, consider operator height and local safety rules, and make sure adequate preparations are made for power, cooling and so on before the rack is shipped. Obviously, local-language instructions may be necessary in some cases. Inadequate preparation can sometimes cause long delays in system deployment.

Minimizing magnetic interference

Magnetic fields generated by test-equipment transformers can interfere with the cathode ray tube (CRT) displays found in many computers and oscilloscopes (newer display types such as LCDs are far less susceptible to magnetic interference). If you put a power supply directly below a scope, the magnetic field from the transformer in the power supply can cause the scope CRT to waver to the point where it may not be usable. To alleviate the problem, move the receiving instrument away from the transmitting instrument. The intensity of the magnetic field decreases as the distance from the source of the field increases; the amount by which it decreases depends upon the configuration of the source of the field and the proximity to the source, but clearly, the greater the separation between the source and the receiving instrument, the lesser the effect.

In some cases, magnetic fields also can affect performance and accuracy of instruments that don't have CRTs. For example, a voltmeter's circuitry could be susceptible to a large magnetic field produced by a transformer. If you are having measurement problems with an instrument, keep in mind that magnetic interference could be one of the causes. Try moving the affected instrument away from likely sources of magnetic fields. Power supplies, fans and high-power-consuming instruments have a higher potential for producing large magnetic fields.

If moving the instruments is not an option, try adding magnetic shielding between the different rack layers or between the instruments. High-permeability metal (Mu metal) is sold for this purpose.

Vibration, especially in the presence of a magnetic field, is a difficult problem for system designers to solve. Cables moving in a magnetic field can generate current, and charge-related noise can be caused by internal stresses in vibrating cables connected to a charge amplifier or DMM. This issue is one of the big reasons for installing a mass interconnect in the system. It minimizes the relative motion between cables, and the chance of charge movement due to pinched cables.

Power dissipation and thermal management

All test instruments produce heat during operation. If you have multiple instruments producing heat in an enclosed rack, the temperature can easily exceed environmental conditions specified for your test instruments. When this happens, your instruments can fail prematurely and your measurement results may be jeopardized. Temperature gradients are also an issue. If one end of the rack is ten degrees hotter than the other end, even if the overall temperature is within instrument specifications, the resulting gradient can cause unwanted thermocouple effects or slow drift errors.

The best way to dissipate the heat inside a rack is to increase airflow. Installing extractor fans in the top of the rack, as shown in Figure 6.6, improves natural convection cooling by increasing the airflow in the rack. The fan moves warm air from the bottom of the rack up and out through the vented top cap, providing cooling to the entire length of the rack. It is a good idea to use a fan when internal rack temperatures are 15°C (27°F) above ambient temperature.

Figure 6.6. Extractor fan installed in rack

If you cannot create enough airflow to remove the heat with a fan, you may need to consider air conditioning your rack. There are standard NEMA enclosures that can be used for this purpose.

When you install equipment in your rack, do not block instrument fans or side air holes and be sure to follow instrument manufacturers' recommendations regarding air flow and clearance around instruments. In general, place your deepest instruments at the bottom of your rack. If you place a full-depth, full-width instrument in the middle of the rack, you block airflow to the instruments below it.

Typical top-mounted extractor fans will move about 200 CFM (cubic feet per minute) of air, which is sufficient for dissipating up to 2500 W of power inside a rack. If your system uses more than 2500 W, you could install additional top-mounted fans or use a 600 CFM fan in the rear rack door to increase air flow.

If your system includes high-power instruments like AC sources or electronic loads with their own fans, use ductwork to vent them directly out the back of the rack. You can make the ductwork out of sheet metal.

The amount of power an instrument dissipates typically is specified by the instrument manufacturer. If that specification is not available, you can estimate power dissipation requirements from the maximum current specification using the equation

```
Worst case power (VA) =
Voltage x Amperage
```

This calculation provides a conservative estimate of power dissipation requirements because power in watts, the true source of heat, is always less than or equal to power in VA. It is a good idea to use conservative figures to safeguard against worst-case situations.

Many test instruments draw a fixed amount of current. However, a power supply draws variable current depending on how much power it is providing to the device it is powering. When you calculate heat dissipation requirements, plan around a power supply's maximum draw.

Routing power and signals

Once you have resolved the weight and balance issues, calculated your airflow and power needs and planned for operator accessibility, you are ready to turn your attention to how you will get power and signals to your instruments and your DUT. Your goal is to route power and measurement and stimulus signals to the right place as efficiently as possible while keeping noise to a minimum.

Multiplexing and matrix switching

Switches, or relays that route power and interconnect system instrumentation and loads to your DUT, are an integral part of most automated and semi-automated test systems and some manually operated systems. Multiplexers and matrix switches make it possible to minimize the number of test instruments in your system instead of using separate instruments for each test point. Switches deliver power and stimulus signals to the DUT when they are needed and route the measurement signals back to your test instruments.

Choosing the proper switch type and topology will impact the cost, speed, safety and overall functionality of your test system. For a thorough examination of switching in test systems, see *Application Note 1441-1, Test System Signal Switching.*

Wiring your system

Power wires radiate electronic noise and both stimulus and measurement signal-carrying wires are susceptible to this noise, so to minimize interference, separate power wires from signal-carrying cables. Proper shielding and grounding techniques can help alleviate noise problems (see "Grounding and shielding" on page 72). Selecting the proper type of cable is also important. A double-shielded or triaxial cable with insulation between the two shields provides the maximum protection against noise coupling.

In some cases, you may need to separate signal measurement cables (which can be sensitive to noise) from signal stimulus cables (which can generate noise). For example, if your stimulus signal is a high-frequency square wave with rapidly changing transitions (fast edges) produced by a function generator, it will radiate more noise than a square wave with slow edges or a high-frequency sine wave, and it would be more likely to interfere with the accuracy of a low-level measurement signal. If possible, keep wires carrying high-frequency square waves and other noise-generating signals away from your measurement paths to minimize interference.

For a detailed discussion of ways to reduce noise in switch systems, see the *Application Note 1441-2, Reducing Noise in Switching for Test Systems.*

Wiring dress and termination—Good-quality cabling is expensive, but you will get the best results if you buy the best cabling your budget will allow. Make sure the cable you select is designed for the task you have in mind and be careful not to exceed the manufacturer's ampacity rating of the wires you choose.

It is a good idea to adopt a systematic approach to arranging and managing your system's cables. For a large system, you may want to consider using cable harnesses or looms. For a smaller system, cable ties may be adequate for bundling cables. Be sure not to wrap power cables in the same bundle as signal cables. For all systems, decide on a consistent method for labeling cables, as it will simplify troubleshooting, maintenance and future replacements. On the label, include either a reference to a look-up table or a full description of the cable's signal type, connectors and purpose. It is also a good idea to document the type and supplier for each cable you use and retain copies of datasheets for all cables and connectors.

Keep your cabling as short as possible to minimize voltage drop and interference, leaving just enough slack to allow you to keep it out of the way. If your instruments are mounted on sliding shelves or rack slides, make sure you allow enough slack to allow the equipment to slide all the way out.

Wire termination devices may be already mounted on the wires you purchase, or you may build your own wire terminations. If you build your own, use gold-plated pins and match the current rating of the pins to your application. Gold-plated pins cost more, but they last longer because they do not oxidize. Ensure that the pin and the wire both can withstand the maximum current you plan to use on that signal path or power path.

For RF applications, typically you will use coaxial cable (to match the characteristic impedance of the application and to minimize radiated noise) and terminate the cables with coaxial connectors (to maintain the integrity of the connection between the inside of the rack and the outside of the rack). Of course, the coaxial signal path should also be terminated with the proper characteristic impedance to minimize signal reflections.

Strain relief—When you wire your system, be sure to protect your investment and minimize system downtime by minimizing sources of cable stress and damage, such as vibration, extreme bending and cutting and fraying caused by sharp edges. If your cable needs to pass through the rack cabinet wall, use a gasket in the hole and support the wire adequately along its path.

If you bend a wire back and forth repeatedly, it will eventually break. For wires in your system that need to be able to move, it is important to minimize the strain on the wires. For example, fixturing wires tend to move often as you connect and disconnect your DUT. If your system is designed for high-throughput manufacturing test, you will need to replace the fixturing wires regularly and pay careful attention to strain relief. Building strain relief into your system cabling helps protect both the cables and the connectors on the test equipment. Make sure that you support cables at regular intervals inside the rack cabinet, so the connectors do not bear the full weight of the cable.

Minimizing noise

We have already discussed some design considerations for reducing noise, but an understanding of where noise might originate is also helpful. In systems designed for testing electronic modules, the most significant causes of noise are conductive coupling, common-impedance coupling, and electric and magnetic fields. In addition, some systems are sensitive to noise from galvanic action, thermocouple noise, electrolytic action, triboelectric effect, and conductor motion.

Conducted and radiated noise

One of the easiest paths for noise to couple into a circuit is a conductor leading into it, resulting in conductively coupled noise. A wire running through a noisy environment has an excellent chance of picking up unwanted noise via radiation and then conducting it directly into another circuit. The power-supply leads connected to a circuit are often the cause of conductively coupled noise. Common-impedance coupling occurs when currents from two different circuits flow through a common impedance. The ground voltage of each is affected by the other. As far as each circuit is concerned, its ground potential is modulated by the ground current flowing from the other circuit in the common ground impedance, leading to noise coupling.

Radiated magnetic and electric fields occur whenever an electric charge is moved or a potential difference exists, and can also be a cause of noise coupling. In a circuit, high-frequency interference may be unintentionally rectified and appear as a DC error. Switch-system circuitry is also susceptible to electromagnetic radiation from radio, TV, and other wireless broadcasts, and it is important to shield sensitive circuitry from these fields. If you want to make accurate measurements of low-level signals in a test-system environment, you need to pay careful attention to the details of grounding and shielding.

It is always a good idea to have a line filter and surge protector in the main power distribution unit (PDU) of the rack. Also, each instrument usually has its own line filter, to reduce conducted interference from the instrument and reduce conducted susceptibility to the instrument. But remember, there is still some residual noise that each instrument can inject into the power grid. Sometimes it becomes necessary to put an additional power filter on an individual instrument to reduce its conducted noise.

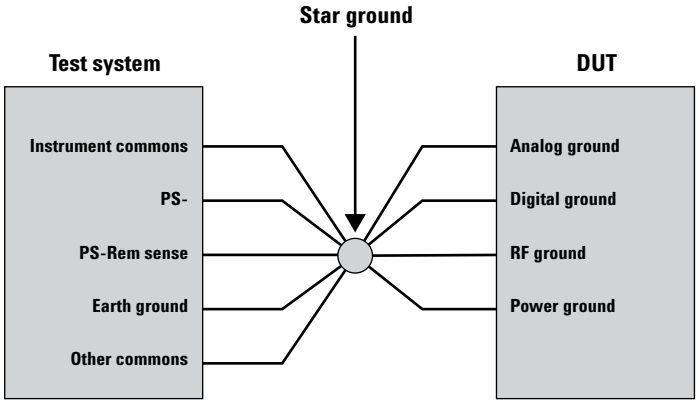
Grounding and shielding

Grounding and shielding are the two primary methods for reducing unwanted noise in a test system. They often work together, such as when the shielding of a cable is connected to ground. In such cases it is important to understand where to ground the cable shield in order to maximize the shield's effectiveness. In some cases, the solution to one noise problem may reduce the effectiveness of the solution to a different noise problem, making it imperative that you thoroughly understand the noise source, method of coupling, and noise receiver so you can make the appropriate tradeoffs.

When you design a grounding system, your goal is to minimize the noise voltage generated by currents from two or more circuits flowing through a common ground impedance, and to avoid creating ground loops that are susceptible to magnetic fields and differences in ground potential.

To accomplish these goals, instrument, power and safety grounds should all be connected as close as possible to the DUT's power ground via a "star" mechanism as shown in Figure 6.7. This eliminates ground loops and contributes to quiet readings.

Figure 6.7. A star ground minimizes noise and eliminates ground loops.



For a detailed discussion of grounding and shielding issues, see *Application Note 1441-2, Reducing Noise in Switching for Test Systems and the white paper Considerations for Instrument Grounding.*

In high-frequency systems, radio frequency interference (RFI) also can cause problems. To minimize RFI, make sure your cable diameter is suitable for the signal wavelengths you are transmitting, terminate all cables in their characteristic impedances, keep cable lengths as short as possible and use only high-quality cables and connectors. For more information, see the white paper *Proper Cable Shielding Avoids RF Interference Problems in Precision Data Acquisition Systems.*

Safety and interlocks

It is important to protect the safety of test-system operators, as well as safeguarding your DUT and the equipment in the rack itself. You need to plan for system safety as part of your overall system design, and you need to comply with company, local, national and international safety standards and regulations that may apply.

Install a system cutoff mechanism that is activated by any action that exposes the operator to potential harm. Make sure you document safety procedures and thoroughly train operators to use them.

Mechanical safety

Fans are a potential source of danger in a test system. Make sure that any fans you use are covered with fan guards that make them inaccessible to human fingers. Positioning fans on top of rack cabinets, instead of in the cabinet wall, reduces the chance that someone's long hair could get sucked in unintentionally.

If the rack is only waist high, be careful to consider what might happen if a liquid is spilled on top of the rack. To safeguard against a rack tipping over, use the guidelines discussed in the "Proper weight distribution" section of this paper (see page 67).

Electrical safety

Install a system cutoff switch (often called an emergency off switch, or EMO) where operators can reach it easily. The switch should cut power to the entire system, not just the DUT. If the cutoff switch is used, make sure operating conditions are safe before you restart the system. Label all high-voltage, high-current and high-power devices in red, and make it clear they are hazardous. Devices carrying more than 42 volts AC or 60 volts DC are hazardous. After a power outage, latching relays may or may not return to a safe state. Consider what they will be controlling and what equipment they will be connecting.

One key to electrical safety is making high voltages inaccessible to operators. If your DUT requires high voltages or high-bias current, use an interlock mechanism to cut power to the DUT when the operator is able to contact it. For example, you can use a special fixture with a see-through cover fitted with an interlock mechanism that cuts power to the device when the cover is opened. Look for a power supply with a "remote inhibit" feature that lets you remotely inhibit the output by simply making the connection between two points.

AC power distribution

In a big system with 10 to 14 instruments, you typically plug each of the instruments into terminal strips inside the rack itself. The terminal strips may get their power from a large power distribution unit (PDU), which is usually located in the bottom of the rack. The PDU typically has a single line that exits the rack and connects to a power source on the wall, floor or ceiling. When you plan your system, check the AC input current rating of individual instruments and make sure the total does not exceed the maximum current you can draw from the terminal strips or from your AC mains supply. Using maximum current figures for each instrument will help you plan for a worst-case situation and avoid tripping circuit breakers. The disadvantage of planning around maximum current draws is that you have the potential for overdesigning your system and wasting capacity.

If a single-phase power line cannot handle your needs, you will need to move to a 3-phase AC input scheme. If you do not know what power types are available at your site, get that information from your facility engineers.

If you use 3-phase equipment in your system, make sure the instruments in your rack share power evenly across all three phases. For line-to-neutral loads, you can accomplish that by designing the rack with three terminal strips, such that each strip runs off one of the three phases. Connect your test instruments so they draw current fairly equally from the three strips. To make this task easier, create a list of the instruments in the rack and the current they draw, keeping in mind which instruments consume fixed power and which draw variable current. For variable-draw instruments, use the maximum current for your calculation.

To calculate power draw for line-to-line loading in a perfectly balanced system, take the sum of the loads and divide by the square root of three to determine the current that is actually being drawn by the phase feeding the system.

If you have no 3-phase equipment in your system, you do not necessarily need to balance power evenly across all three phases. You can just size your cabling for the largest phase load. You also will want to know the actual current draw on each phase (even if they are not balanced) so you can balance correctly in your facility. You could find this number by measuring the current on each AC line with a true RMS meter.

It's a good idea to assess the quality of the mains power before installing any system. Use a power line monitor to check for voltage spikes (surge conditions) caused by motors, RF spikes, dropouts and brownout conditions (sag conditions). This simple test can save you headaches from non-repeatable results and also save damage to the test equipment itself.

Conclusion

It's one thing to connect a PC to one instrument, but when a rack might contain $100,000 or more worth of equipment, it pays to do some planning. Arranging your test equipment in a rack to maximize measurement accuracy, equipment longevity and operator ease of use and safety also takes careful planning. Whether you are using your test system for R&D, design verification or manufacturing test, you need to consider a variety of issues, including weight distribution, heat dissipation, instrument accessibility and operator ease of use, and you need to pay close attention to minimizing magnetic interference and conducted and radiated noise.

7. Maximizing System Throughput and Optimizing System Deployment

Introduction

This chapter discusses hardware and software design decisions that affect throughput, including instrument and switch selection, as well as test-plan optimization and I/O and data transfer issues. It also discusses ways to optimize your system as you prepare to deploy it.

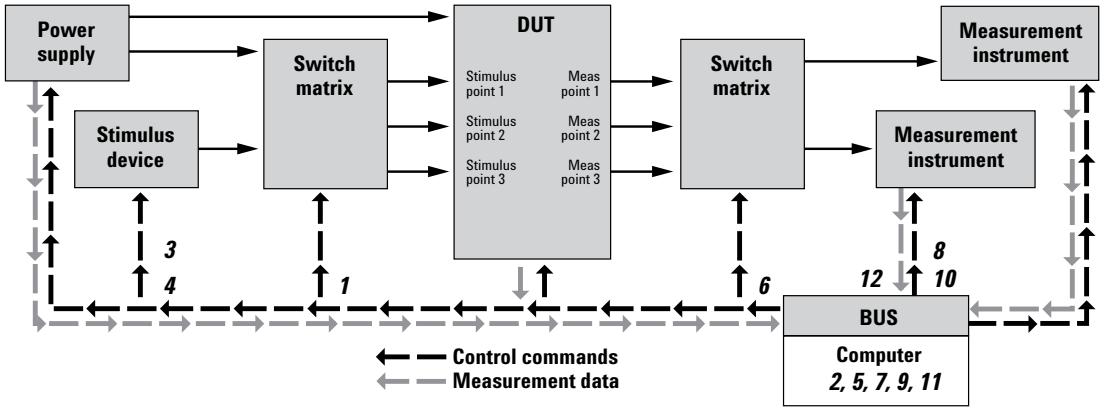
Throughput is a measure of the time it takes to test a device or product. Maximizing throughput is most critical in high-volume manufacturing, where you have thousands of products to test and you want to test them as fast as possible. In high-volume manufacturing, you measure throughput in terms of devices per unit time. The faster you test your devices, the lower your manufacturing costs. In design validation testing, the speed of each individual test is not as critical, but test setup time is important because you need to be able to adapt to pinouts that change often. In design validation, you measure throughput in terms of tests per unit time. The faster you can validate your designs, the faster you can get your new products to market. In R&D, throughput is seldom an issue because you are not likely to repeat tests on large numbers of devices or to perform the same test repeatedly on a single device.

Taking the time to optimize system throughput may require some additional investment up front, but the payoff in lower costs and faster time to market makes the investment worthwhile.

As Chapter 5, *Choosing Your Test-System Hardware Architecture and Instrumentation,* pointed out, a test system is essentially a group of subsystems that work together. The hardware you choose for these subsystems and the software you write to make these subsystems communicate and interact have a huge effect on your system throughput. If throughput is critical in your test application, you need to choose equipment with the performance and features required for fast testing and then configure it and program it for optimum speed. After you've built your system, you can tweak instrument setups and operating procedures to further optimize its speed.

In general, your system first needs to set up a test or configure the proper stimulus and send it to your device under test (DUT). Then your system needs to actually make the measurement on the DUT and transfer the measurement data back to the computer. Figure 7.1 shows typical steps a computer-controlled system would take to make a measurement. (The steps do not necessarily have to be executed in the order presented.) Each of these steps takes some amount of time to execute. To optimize throughput, you need to analyze how long the steps take in your system and decide which steps you can speed up. Depending on your application and budget, you may decide to work only on the steps that have the biggest impact on your throughput, or you may decide to invest the time and money to eliminate every unnecessary millisecond in the entire process.

Figure 7.1. Steps involved in making measurements with a typical computer-controlled system



Steps

1. Tell system where to connect the stimulus
2. Wait for switch to settle
3. Tell stimulus instrument what signal to send to DUT (parameter, range)
4. Tell stimulus instrument to send signal
5. Wait for stimulus to settle
6. Tell switch to send DUT signal to measurement instrument
7. Wait for switch to settle
8. Tell measurement instrument what parameter to measure and the range in which that parameter falls
9. Wait for instrument to process command and complete configuration
10. Tell instrument to make the measurement
11. Wait for instrument to process command and make the measurement
12. Transfer measurement information to computer

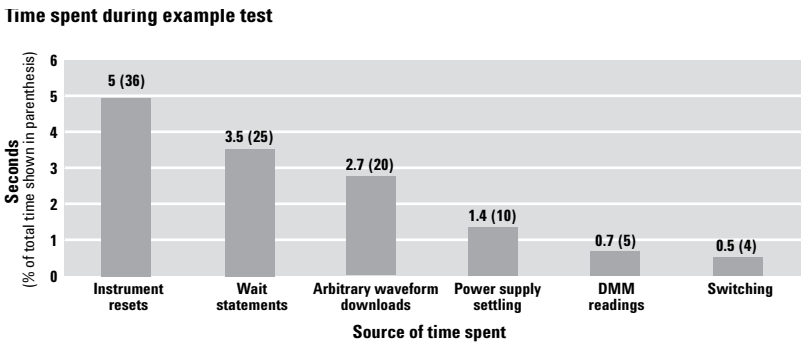
In a typical test system, the steps with the biggest negative impact on throughput include instrument resets, delays (wait statements) programmed into the system software and waveform downloads. Power supply settling time, voltmeter measurements and switching also play a role. Figure 7.2 shows the hierarchy of delays in a typical test plan.

Obviously, if your system stops functioning, your throughput drops to zero. In all phases of product test (R&D, design validation and manufacturing test), therefore, minimizing system downtime is critical to maximizing throughput. To minimize system downtime:

- Select instruments from vendors you trust and choose instruments with high mean time between failures (MTBF) specifications.
- Establish a good spares program: keep backup components for your system so that if an instrument fails, you can quickly swap in a replacement and restore system functionality.
- Perform regular maintenance on your system and its components. Clean fan filters regularly to avoid heat build up (high temperatures contribute to failures). For more information on this topic, see Chapter 8, *Operational Maintenance.*

This chapter focuses on improving throughput for systems designed with rack-and-stack test instruments. However, most of the concepts apply to systems built with card-based instruments (such as VXI and PXI) as well. Card-based systems do have features that lend themselves to optimizing throughput. For example, VXI and PXI backplanes have a built-in triggering bus that makes it easy to implement triggering schemes that can minimize system delays. The new LXI standard (see Chapter 16) also specifies a trigger bus, thus bridging some of the differences between card-based and rack-and-stack instruments. Card-based and rack-and-stack systems are similar in most other regards, and you can use many of the same techniques for optimizing measurement speeds in both types of systems.

Figure 7.2. Hierarchy of delays in a typical test plan



Upfront design decisions affect throughput

If you are designing a new system, rather than optimizing an existing system, you will have a greater opportunity to maximize your system speed. The system hardware and software architectures, instruments, switches, and I/O interfaces you select will have a huge impact on system throughput. For a detailed discussion of system hardware and software architectures, see Chapter 4, *Choosing Your Test-System Software Architecture,* and Chapter 5, *Choosing Your Test-System Hardware Architecture and Instrumentation.*

Making hardware choices

Figuring out how fast your system will perform measurements is harder than it appears. For example, you may decide to use a digitizer instead of an oscilloscope to take advantage of the digitizer's higher resolution. The digitizer may be able to sample 1000 readings very fast, but if those readings are transferred to the PC over GPIB, the total process could take a relatively long time. If you can download a decision-making algorithm into the digitizer, you can send a simple go/no-go result back to the PC, which would make GPIB a reasonable option. However, it takes extra effort to create and download a decision algorithm into an instrument, which may increase development time as well as "first-run" time of the test program. You also need to consider the relative analysis time of a routine computed inside the digitizer compared to the time required to complete it inside the PC.

As you can see, many interdependent factors can affect throughput. If you are looking for test-time reductions amounting to fractions of milliseconds, you must weigh each of these factors carefully. Even if your throughput requirements are not that exacting, the hardware choices you make can significantly affect throughput.

One important factor to consider when you are selecting your instrumentation is command processing time, or the amount of time it takes an instrument to "digest" and interpret a command. Command processing time is usually characterized on an instrument's data sheet. If you cannot find the information, ask the instrument vendor. Command processing times can range from less than a millisecond to dozens of milliseconds. If you send a command just once to an instrument, it may not have a huge impact on your overall test time. But if you are sending the command repeatedly during testing, the time it takes can have a significant impact on your throughput. Note that newer model "smart" instruments tend to have much lower command processing times than older models. Also note that many cardcage-based instruments use the PC for most processing tasks. The time to complete these tasks is highly dependent on the tasks being simultaneously performed on the PC.

As you explore the opportunities for improving your system throughput, keep in mind that when you reduce measurement time, you may sacrifice accuracy and repeatability. If you integrate measurements over a longer period of time you will filter out random noise, and your measurements will be more accurate. Typically, you can improve measurement repeatability by averaging measurements, increasing the number of samples taken per measurement or increasing the measurement sample time, but you will sacrifice measurement speed. If you cannot compromise accuracy and repeatability, it does not mean you will not be able to improve your throughput. Measurement time is just one factor to consider in the overall test plan, as illustrated in Figure 7.1.

In design validation, you typically perform a large number of different tests, so the time you spend setting up the test system is important. To minimize development time, use rack-and-stack system-ready instruments that incorporate a high percentage of the measurement solution you need. For example, if you use a source with modulation capability, you don't have to develop your own algorithm or integrate additional hardware to generate the required modulation. Using instruments with IVI-COM drivers can save you development time. If the instrument has an IVI-COM driver, you can interchange hardware without rewriting your software, as long as you adhere to the functionality that is specific to the instrument class. See Chapter 3, *Understanding Drivers and Direct I/O,* to learn how decisions about drivers affect development time.

Stimulus and measurement instruments

To maximize throughput, consider creating a Pareto diagram of projected delays (see Figure 7.2) in the system and invest your time and money accordingly. If tests A and B are of similar duration but test A is performed much more frequently than test B, then focus your programming efforts, tricks and budget on test A.

When you are choosing instruments, pay close attention to instrument specifications. For example, the Agilent 33120A function/ arbitrary waveform generator is popular for systems applications. But its successor, the 33220A function/ arbitrary waveform generator, downloads arbitrary waveform files 100 times faster than the 33120A, and many of its configuration times are faster (and it also costs less than the 33120A). If you have an existing system that includes 33120A function generators, it is fairly easy to upgrade to the 33220A because the two instruments are programmed similarly, and Agilent provides documentation to help you make the switch.

When you are reading data sheets, pay particular attention to how measurement speeds are specified. Often, measurement speed specifications are related to the speed per reading when thousands of samples are taken, which is a data-acquisition use model. In functional test, it is far more common to close some relays, take a measurement, open those relays and move on to another measurement. In this mode, the measurement instrument's single-point reading speed is most important, and it is dramatically slower than the fastest possible multi-sample reading speeds. In most cases, you will be able to look up the single-point reading speed on the instrument's data sheet.

Look for instruments that have built-in features that will reduce the time needed for communication overhead and post-processing. For example, some test instruments can calculate arithmetic mean, minimum, maximum, and standard deviation. (These capabilities are often called "one-button" measurements.) When you are analyzing multiple data points, these statistical results are much more meaningful than the raw data. Using the system controller to acquire raw measurements can be very time consuming compared to transferring a few measurement results.

Power supplies

Your choice of power supply can dramatically impact system throughput, because waiting for power supplies to settle is typically a time-consuming element in a test plan (see Figure 7.2). Check the settling time specifications of the power supplies you are considering for your system. If you can't find a specific reference to "settling time" on the data sheet, look instead for the "programming speed," "programming response time," or "rise and fall time" specification. Programming speed is defined as the amount of time it takes for the instrument to reach a specified percentage of the voltage setting (typically within 0.1 percent), not including command processing time. Rise and fall times are typically defined as the time it takes to get from 10 percent of the final value to 90 percent of the final value for the rise time, or vice versa for the fall time. Because of the different terminology and definitions, you must be careful when comparing settling times in power supplies from different vendors.

When you are trying to boost throughput in time-critical production test systems, look for a multiple-output supply that can set multiple outputs with a single command, like the Agilent N6700 series. Otherwise, consider using multiple single-output power supplies instead of one multiple-output supply. With multiple-output power supplies, the instrument takes extra time to parse commands, because you are sending an additional parameter to indicate which of the multiple outputs it should use. Also, with most multiple-output supplies, commands sent to the various outputs are processed sequentially, one output at a time (this can be avoided with the Agilent N6700 series). With multiple supplies, one supply can be processing a command while the next is receiving a command, so you avoid delays. For details on using this technique and other techniques, see *10 Hints for Using Your Power Supply to Decrease Test Time*, publication number 5968-6359E.

Another way to reduce test time is to choose power supplies and electronic loads that have built-in measurement features. With power supplies, these capabilities let you measure the supply's output voltage and current. With loads, you can measure load input voltage and current.

A good example is testing a DC-to-DC converter with four outputs, where you need to measure the input voltage to the converter and each of the four outputs in order to fully test the device. If you have a single DMM to measure the voltages, you'll need a multiplexer to sequence through the measurements (see Figure 7.3). In addition to the complexity of this setup, your test program needs to wait for the multiplexer's switches to move and settle for each measurement.

Figure 7.3. Testing a four-output DC-to-DC converter with a single DMM requires a complex multiplexing scheme and can involve significant delays.

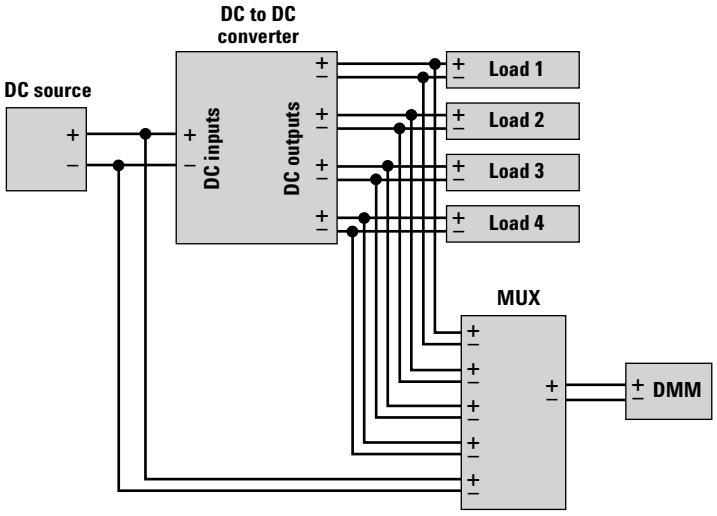
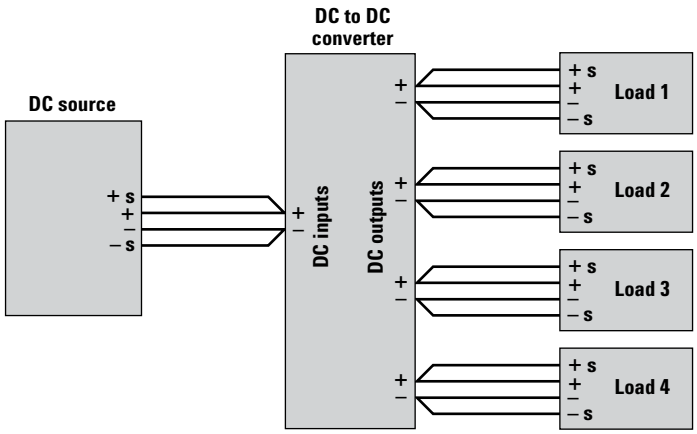


Figure 7.4. By using the built-in measurements in your DC power source and electronic loads, you can eliminate the DMM and MUX and significantly increase your test speed.you can eliminate the DMM and MUX and significantly increase your test speed.



Using DC source and loads with built-in measurement functions (Figure 7.4) can save significant amounts of time. They're already connected to the DUT, and there are no switching delays, so both the setup and test phases are much faster. Note the use of remote sensing here. Although it isn't required, using remote sense is generally a good idea because it provides regulation and measurement at the DUT rather than at the loads or the DC source.

With no need for switching, you'll benefit from faster tests, greater reliability and simpler configurations. This same approach works well for measuring current, and it eliminates the current shunts you'd otherwise need.

Using power supplies that incorporate a feature known as downprogramming can significantly reduce test time, particularly when you need to set multiple voltage level settings. Without downprogramming, the capacitor in the supply's output filter (or any load capacitance) can take seconds or even minutes to discharge when you reduce the output voltage level (the lighter the load, the longer it takes).

Downprogramming uses an active circuit to force the output down to the new level within a matter of milliseconds in most cases. This circuit kicks in automatically whenever the voltage level you set (either manually or programmatically) is below the present output level. The downprogramming rate is fixed in most supplies, but some offer programmable downprogramming.

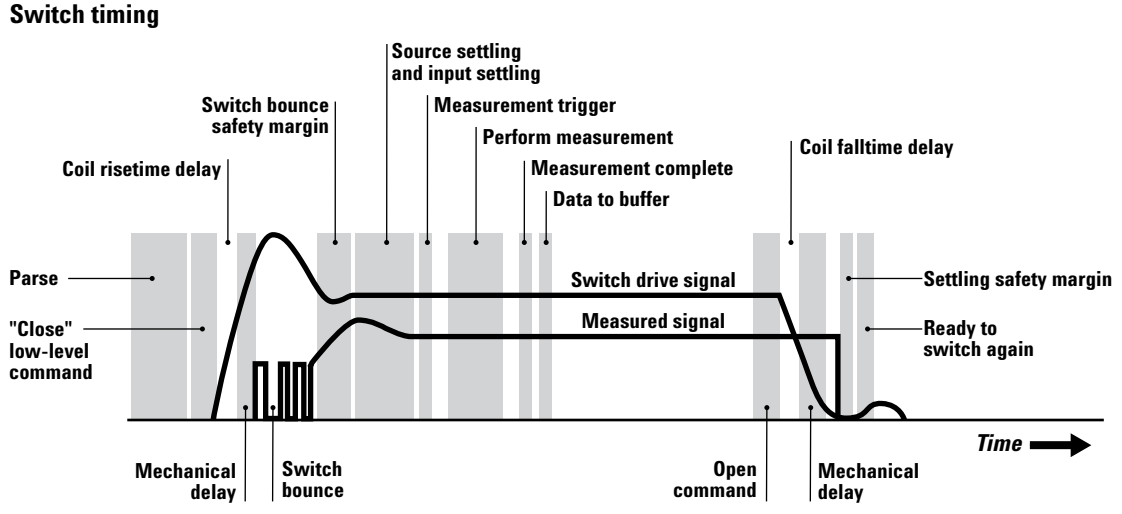
In time-critical tests, it's a good idea to watch out for downprogramming delays. Because programming up is typically faster than programming down, try to sequence multiple tests in such a way that each consecutive test is at the same or higher voltage level as the previous test. See page 81 for more information on test sequencing.

Switches

Switches, or relays that interconnect system instrumentation and loads to your DUT, are an integral part of most test systems because they allow you to use a minimum number of stimulus and measurement instruments to test multiple points on your DUT. If your test plan involves lots of switching, switch speed will have a big impact on your system's throughput, so the type of switches and the switch topology you choose are important. For a thorough examination of switching in test systems, see Application Note 1441-1, *Test System Signal Switching*.

From a system throughput standpoint, the most important switch parameter is settling time, or the time it takes to change states from open to closed and vice versa. Figure 7.5 shows the different actions and the relative times required for a relay to be closed, a measurement to be performed and for the switch to reopen and be ready for the next measurement.

Figure 7.5. This diagram shows what happens when you tell a switch to close, take a measurement, and then reopen. The "switch drive signal" represents the actual voltage that causes the switch to change states. The resulting "measured signal" is connected from the DUT to the measurement instrument.



Electromechanical switches such as reed and armature relays are common in low-speed applications. They are capable of switching high voltage and current levels, but they are limited to switching rates of dozens of channels per second for armature relays up to hundreds of channels per second for reed relays. Reed relays are excellent choices to connect measurement instruments and low-current stimulus to your DUT. They are relatively fast (see Table 7.1), although they can have a higher thermal offset voltage than armature relays. Armature relays are slower, but you can use them for higher current loads. When you use armature relays, group your tests so the relays stay connected to perform as many readings as possible at one time.

Electronic switches, such as field-effect transistor (FET) and solid-state relays, are frequently used in high-speed applications. (Typically for voltage or temperature measurements). However, some FET electronic switches cannot handle high voltage or current, and they must be carefully protected from input spikes and transients. Check the electronic switch ratings carefully.

Switching topologies can be divided into three categories based on their complexity: simple relay configurations, multiplexers and matrices. The best one to use depends on the number of instruments and test points, whether connections must be simultaneous or not, cost considerations and other factors. Typically, the type of relay you choose has a bigger impact on speed than the switch topology you choose, unless you factor in the time required for reconfiguring a switching system (which, as we noted earlier, is more critical in design validation applications.) If you use a switch matrix, you will be able to quickly and easily expand and reconfigure your system as your test needs change. Expanding and reconfiguring systems that use multiplexers typically is more time consuming.

A matrix arrangement of reed relays provides an excellent way to allow any instrument to be connected to any pin on your DUT, and it permits easy expansion as you add new instruments to your system or more pins appear on your DUT. Matrices use more relays than multiplexers, so they tend to cost more. If you don't need to connect multiple instruments to any pin, a multiplexer is a suitable solution. If you have a 1 x 20 multiplexer for example, you can take a test instrument and connect it to 20 pins, but you can't hook anything else to those 20 pins. With those same 20 relays in a matrix, you can connect four instruments to five pins in any combination.

If you want the ultimate in throughput and your budget is not limited, you can use multiple test instruments instead of a switching scheme for making measurements on multiple test points. With multiple instruments, you can set each to the needed range and eliminate the time spent on configuring the test instrument range, as well as the time required for switches to open and close. In some cases it is worth the extra money for the test time you save.

Table 7.1. Relay comparison chart

	Armature relay	Reed relay	Solid-state relay
Switch speed	50/s	1000/s	1000/s
Contact resistance	Low	Very low	High
Life	1 million	10 million	>10 million
Typical failure mode	Fails open	Fails open	Fails shorted
Typical max input	250 V/2 A	100 V/100 mA	250 V/10 A

Controller issues

Unless your PC is ancient, its processor speed is not likely to be a significant factor in your test throughput. Typically, issues associated with stimulus and measurement instruments, power supplies, switches and test software play a much bigger role in determining system speed. Your PC is not in control of data collection speed, and faster PCs don't necessarily collect data any faster. The PC's interface to your test system (GPIB, LAN, USB, FireWire, VXI or PXI) will certainly impact data transfer time, but that is not dependent on PC processor speed. If you are using a LAN or USB interface, we recommend using the highest speed interface and switches/hubs available.

Processor speed is a factor only if you are relying on your PC for analyzing data and if you are using it for your software development. You want to use the fastest PC available when you are compiling programs, but of course you do not have to do your development work on the same computer you use to run your system.

Designing your test plan for speed

Many test programs spend most of the time waiting. Even if you have selected the fastest-available hardware for your system, software issues can slow your test-system throughput significantly. While you can tweak your test-system programming after your system is complete (see *"Fine-tuning your system for speed"* on page 84), you will achieve better throughput if you design your test plan up front to optimize test sequencing and minimize delays.

Optimizing test sequencing

In most test systems, single-instrument measurement times have a smaller impact on overall test time than the test flow (execution sequence) you choose when you are designing your test plan.

In a production environment, first arrange your test plan so the system can find DUTs that are destined to fail as soon as possible. If a particular DUT frequently fails a certain test, move that test to the front of your test program. Ideally, of course, you should feed reports of persistent DUT failures back into R&D or production engineering so they can be resolved permanently. Agilent offers a toolset, Fault Detective Diagnostic Solutions, to help with this process. Fault Detective helps you optimize throughput by quickly diagnosing functional failures in manufacturing and by finding redundancies in your tests. This toolset also helps you maximize quality by identifying gaps in your test process.

Next, when you are ordering your tests, minimize the number of times the stimulus, DUT and measuring instrument change states—particularly those that take a long time—by organizing the program's execution sequence. Start by looking for tests that leave the DUT in the desired state for the next test. If the DUT needs to be turned off for the start of a test, for instance, try to sequence a preceding test that leaves it off. If a particular test requires that the DUT is warmed up, place it later in the sequence and use a system timer to guarantee the DUT has been on long enough. Although they are not always feasible, these techniques can yield big improvements when you can use them.

The program sequence shown in Table 7.2 measures voltage or current on three different DUT test points under three different sets of input conditions. In this case, the ambient temperature setting is used as an example of a stimulus to the DUT. The temperature changes for each test point, and the measurement setup must also change to make the required voltage and current measurements. Each change adds time to the test program, reducing system throughput. For example, if you are using a DMM and you change the measurement function, the DMM reconfigures the hardware and retrieves different calibration constants before making a measurement.

Table 7.2. Typical test sequence

Program step	Input conditions(stimulus to DUT)	Measurement setup (to measure signal out of DUT)	DUT measurements taken
1	Set input condition 1 (e.g., amb. temp. = 0 degrees C)		
2		Prepare measurement setup 1 (e.g., voltage)	
3			Test point 1 voltage
4	Set input condition 2 (e.g., amb. temp. = 25 degrees C)		
5			Test point 1 voltage
6	Set input condition 3 (e.g., amb. temp. = 55 degrees C)		
7		Prepare measurement setup 2 (e.g., current)	
8			Test point 1 current
9	Set input condition 1 (0 degrees C)		
10		Prepare measurement setup 1 (voltage)	
11			Test point 2 voltage
12	Set input condition 2 (25 degrees C)		
13			Test point 2 voltage
14	Set input condition 3 (55 degrees C)		
15		Prepare measurement setup 2 (current)	
16			Test point 2 current
17	Set input condition 1 (0 degrees C)		
18		Prepare measurement setup 1 (voltage)	
19			Test point 3 voltage
20	Set input condition 2 (25 degrees C)		
21			Test point 3 voltage
22	Set input condition 3 (55 degrees C)		
23		Prepare measurement setup 2 (current)	
24			Test point 3 current

If you organize the program to minimize changes to the stimulus conditions and measurement setups, overall test time is reduced. Note that the sequence shown in Table 7.3 provides exactly the same number and type of DUT measurements under exactly the same set of input conditions as the previous sequence, but the overall number of programming steps has been reduced from 24 to 14. Also, the number of stimulus changes has been reduced from 8 to 2, while the measurement setup has gone from changing back and forth 5 times to changing just once.

Organizing nested loops

Structure the basic test flow so that slow operations like setup, DUT connections and temperature settings are in the outermost loop. Nest faster operations like one-button measurements in lower-level loops. Place your fastest operations in the lowest-level loop. You can use a test flow diagram, as shown in Figure 7.6, to get a better conceptual understanding of the test plan and prevent wasted time in nested loops and poor use of DUT connects and re-connects.

Figure 7.6. To minimize overall test time, structure test loops so that the most time-consuming operations are performed the fewest number of times.

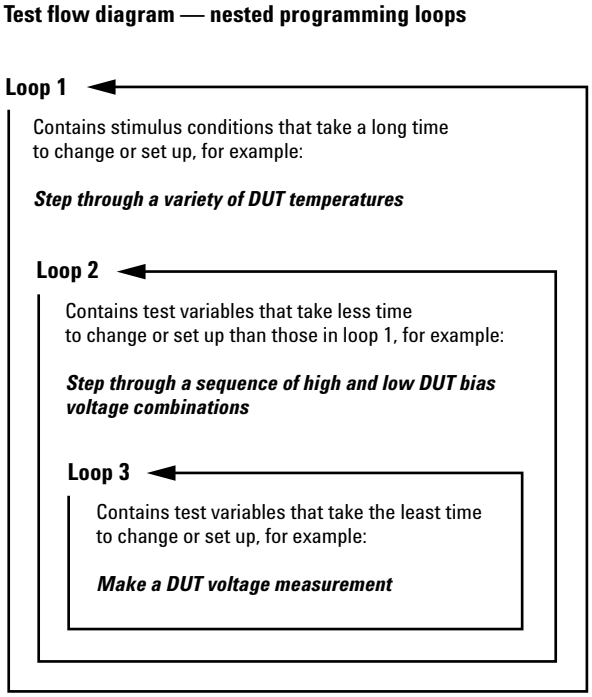


Table 7.3. Test sequence optimized for speed

Program step	Input conditions (stimulus to DUT)	Measurement setup (to measure signal out of DUT)	DUT measurements taken
1	Set input condition 1 (0 degrees C)		
2		Prepare measurement setup 1 (voltage)	
3			Test point 1 voltage
4			Test point 2 voltage
5			Test point 3 voltage
6	Set input condition 2 (25 degrees C)		
7			Test point 1 voltage
8			Test point 2 voltage
9			Test point 3 voltage
10	Set input condition 3 (55 degrees C)		
11		Prepare measurement setup 2 (current)	
12			Test point 1 current
13			Test point 2 current
14			Test point 3 current

Using triggering

In typical test routines, it is common to apply a stimulus to a DUT, insert a delay (wait statement) in the system software to give the stimulus instrument and DUT time to stabilize and then instruct a test instrument to take a measurement on the DUT. However, the length of the required delay is typically a guess. Instead of adding delays to a test routine to assure that enough time has elapsed for the stimulus and DUT to stabilize, use triggering from the stimulus instrumentation (and sometimes from the DUT itself) to initiate a reading as soon as possible, especially if wait time delays comprise a significant proportion of your test time. Also, once a triggered sequence has been started, it is possible to make other measurements while waiting for the triggered measurement to finish.

You can use triggering built into a VXI or PXI backplane or with point-to-point wiring in a rack-and-stack system. In a rack-and-stack system, you need the right cables, the right connectors and a strategy for what is going to trigger what. In a VXI or PXI system, triggering is easier to implement because you don't have to do any special wiring. In LXI systems, the LXI Class A trigger bus provides equivalent triggering capability to LXI/PXI.

Managing wait times

When you are writing your test-system software, you can minimize delays by overlapping wait periods within specific tests. Here's a typical sequence:

- Apply a load to the DUT or set up its programmed state and wait for DUT output to settle
- Connect relays to engage measurement equipment and wait for relays to close
- Set up measurement instrument and wait for setup to complete
- Initiate measurement and wait for measurement to complete
- Disconnect relays
- Turn off power source
- Wait for DUT output to settle

Each step usually involves a wait while the action completes. In addition, most DUTs need time to stabilize after power is applied or a load condition has changed. By separating the programming and wait stages, you can rearrange the test to program one instrument while waiting for another:

- Apply load to the DUT
- Connect relays to engage measurement equipment
- Set up measurement instrument
- Wait for the longest of all previous actions to complete:
  — Relays to close
  — Measurement instrument to settle
  — DUT output to settle
- Initiate measurement
- Wait for measurement to complete
- Disconnect relays
- Turn off power source
- Wait for DUT output to settle

Overlapping the wait periods minimizes overall delays. While the DUT is settling, the test program is busy programming the relays and setting up the measurement instrument.

To implement an overlapped wait, use a common or global timer. Each programming routine that sets up an instrument or DUT can tell a global timer how long each action will take; this identifies which action requires the longest wait. Then, when a measurement or other test requires that the previous commands be completed, a call to a single wait function will wait until the global timer expires before continuing:

- Apply load to the DUT
- Connect relays to engage measurement equipment
- Set up measurement instrument
- Wait for global timer
- Initiate measurement
- Wait for global timer
- Disconnect relays
- Turn off power source

With this approach, the test does not have to wait any more than is absolutely necessary for instrument setup, and the programming is simpler, too. Other techniques for reducing software delays are discussed in *"Fine-tuning your system for speed"* on page 84.

Programming tips for fastest throughput

- Graphical languages are not optimized for speed, so use a textual programming language. For fastest throughput times, write your test program in Visual C++ or C#.
- Avoid the indiscriminate use of the reset command (*RST) to return test instruments to a known state after a measurement. It is best to place resets at the beginning of a test program to initialize the hardware the first time the program is run, then to manage the instrument states carefully so that they are in a benign state (equivalent to the reset state) at the end of the program.
- Use binary data format when transferring large amounts of measurement data.
- Do not use SLEEP statements for instrument-specific timing (consider the operation complete command, *OPC?, the wait command *WAI, and READ statements instead).

In some test systems, I/O speed is not a major determining factor in overall throughput. This is especially true in RF systems, where the network analyzer or spectrum analyzer may take some time to complete a measurement. However, in systems that rely on unprocessed data, or when real-time control is important, your choice of I/O for the connection between your computer and your test system hardware can have a big impact on the overall test time.

While high-speed LAN and USB have much higher throughput than GPIB, the serial nature of these interfaces results in performance that may be similar to GPIB for highly transactional operations in which you are not waiting for the instruments. The extra cost to use gigabit LAN and Hi-Speed USB is relatively low and will result in noticeable speed improvements. Note that a LAN will run at its fastest if you make a direct socket connection.

Connection to a card-based system such as VXI or PXI should usually be done with a fast interface such as FireWire or MXI, as the register-based cards generally have minimal processing capability on-board and count on a fast interface for good performance.

Table 2.1 in Chapter 2 showed the relative speeds for various operations for a stimulus instrument having GPIB, USB and LAN interfaces. As you can see from that table, the instrument's internal speed clearly dominates setup changes, making I/O choices seem moot, but download speeds are much better with LAN and USB when large amounts of data are involved.

For more information about I/O and its effect on system throughput, see Chapter 2, *Computer I/O Connectivity Considerations,* and Application Note 1475-1, *Modern Connectivity—Using USB and LAN Converters*.

Keep in mind that if your instrument's throughput is slow, you are not going to get greater throughput by changing to a faster I/O interface. You can improve your throughput by minimizing the number of GPIB transactions you send. When possible, send multiple GPIB commands at one time. This reduces bus turnaround times and allows the instrument, in some cases, to operate on the commands as quickly as possible.

The character format you use to transfer data can also affect the data transfer rate. You can choose from a variety of general formats, including character string, ASCII, or binary. Binary code is handled as bit streams, typically in block-length message units. These message units are more compact than those made up of string and ASCII characters and therefore they can be transferred more quickly.

For example, when you are downloading a data file for an arbitrary waveform to a function generator, downloading floating-point values (a character string) is slower than downloading binary values, but using floating-point values is more convenient when creating the arbitrary waveform. Here, you need to decide which is a higher priority, faster data transfer (binary), or ease of use (floating-point values in the form of a character string).

Fine-tuning your system for speed

Whether you are turning on a new system or fine-tuning an existing system, there are a number of techniques you can use to improve throughput. Relatively small adjustments to system software, instrument setups and operating procedures can help you optimize your system speed.

Minimize delays

As Figure 7.2 noted, delays (wait statements) programmed into system software typically cause systems to run at suboptimal speeds. When you run a test program there are some operations — such as measuring a complex signal or moving data to an array — that take additional time to complete before the next command can be executed. If these operations do not complete before the next command in your program is executed, errors can occur and the program may halt. When debugging test routines, programmers frequently "fix" the problem by programming in a delay after the operation and before the next command. This is fine as a temporary fix for correcting an error, but it is important to remove the delays, or at least to make them as short as possible, once you find the real cause of the measurement problem. Leaving unnecessary delays in a program slows down the overall system throughput.

An alternative to using a delay is to use system-level control commands such as *OPC? (operation complete) to inform the control software that an operation is complete, which is especially useful for variable-length operations. Many instruments are IEEE-command compliant which means they are able to use the

*OPC and *OPC? commands. Using *OPC? at the end of a command tells the instrument to return a +1 in response to the query as soon as the instrument command has finished executing. The next command in the program sequence can execute without any unnecessary delay.

You also can use SRQs (GPIB service requests) and IRQs (Windows interrupt requests) to minimize delays in your test software. The interrupt structure eliminates the necessity to conduct a poll or a loop waiting for something to happen. Such loops are time-consuming to write and slow to execute. With an SRQ or an IRQ, the hardware tells the control software when it is ready to have its data read (similar to a trigger).

Minimize state changes

"Designing your test plan for speed" on page 80 discussed ordering tests to minimize state changes. If you optimized the order of your tests during the design phase, you may not need to tweak it after your system is up and running. If you are fine-tuning existing system software that was not written with speed in mind, you may find many opportunities to improve your throughput by reordering tests. Range, frequency and function changes are relatively slow and can interfere with fast tests. To compensate, arrange your tests such that tests involving different parameters or different ranges are grouped rather than intermixed. It is also helpful to pick a range that gives the needed resolution for most measurements and then keep it there. If you need to test multiple ranges or multiple parameters and your budget allows, you can use multiple test instruments and set each to a specific range or parameter.

Instrument-specific tips

To maximize throughput, make sure your test instruments are configured for speed. The following suggestions apply to many of today's instruments:

- Make sure you are using the latest version of the instrument's firmware. Firmware upgrades sometimes include significant measurement speed enhancements.
- Turn off the display if it isn't needed. Updating the display slows the reading time.
- Turn off all math functions or other data processing, unless using it allows the instrument to send a single pass/fail result instead of a stream of data.
- Set autozero to "once" or "off," as this feature can double measurement time. However, do this only if the temperature drift in the system is minimal. Otherwise, an autozero should be performed periodically.
- Use the lowest-level commands you can. Instead of using "measure?," use "config" "init" and "fetch?." You do have to pay attention to where and how your readings are stored when you use these commands. For example, the Agilent 34401A multimeter treats "read?" and "init" followed by "fetch?" exactly the same except for where it stores the readings. INIT/FETCH buffers the readings, whereas READ places them immediately to the output buffer. By omitting this extra buffering step, you can get your reading to your computer faster.
- Use the fewest digits of resolution needed for the required accuracy.
- Avoid using auto-range. Define the expected value of a measurement so the instrument spends less time searching for the proper range. Bear in mind, though, that a malfunctioning DUT could result in a reading outside of the selected range. Your program must be able to react to overload readings correctly.
- Whenever possible, use preset states that can be used to recall instrument state setups.

In addition to the general techniques listed above, here are specific techniques you can try with different types of test instruments.

Function generators

- Configure your setups in advance and store them into memory locations. Instead of sending multiple commands to configure the instrument, you can recall the instrument state with a single command.
- When downloading arbitrary waveform data, send it in binary format rather than ASCII. Download the smallest number of arbitrary waveform points you can.
- Consider using modulation to respond to your system (AM, FM, PWM, PM or FSK). If you need the generator to respond to something else in your system, rather than reading a value and reconfiguring the function generator, see if you can use a control signal or even a conditioned signal as an external modulation signal.

Counters

- Use ASCII format for fastest throughput (note: this is different from other instruments)
- Select the trigger level instead of using auto level
- Use the auto arming mode
- Disable printing operation
- Define the trigger command so the fetch command does not need to be sent for every measurement
- For some measurements, a counter may produce readings in which the last few digits are not stable. This can slow a test if a human operator needs to discern the difference in readings. Truncating the last digits will produce a more understandable display, but some tests require that extra resolution. Have the counter calculate the arithmetic mean if you require high resolution and a stable reading or use a limit-testing mode.

Digital multimeters

- When using a scanning meter such as the Agilent 34970A, wire adjacent channels so that the DMM doesn't have to switch modes or ranges
- Select the shortest channel delay (zero)
- Turn off scaling
- Turn off alarms
- Use the fast filter
- Turn off T/C (thermocouple) check. Some scanning meters will check for the existence of a thermocouple by looking for a short circuit before attempting to read the thermocouple voltage.
- Shield the measurement setup to reduce noise pick-up from the operating environment. Shielding may allow you to make measurements with shorter measurement times (aperture) or with less filtering and still achieve sufficient noise rejections to obtain the required accuracy.
- Try to make all readings with the DMM "LO" terminal connected to circuit low. DMMs have fairly large values of capacitance between "LO" and earth which must be charged (increases settling time) when you make floating measurements.

Scopes and digitizers

- If you are importing raw data, use binary transfer mode. Specifically, use byte or word formats. Word format is more accurate but requires twice as much data to be sent over the bus. Some scopes produce more than 8-bit resolution, but many acquisition modes produce only 8-bit data. In these cases, transferring word versus byte data will take twice as long and not provide any additional resolution. It is important to know how and when the instrument produces extra resolution.
- Capture only as much data as you need to analyze.
- Turn off special features such as mask test, jitter analysis and FFT functions if they aren't needed.
- Make sure you have an adequate trigger rate, and use the fastest sweep speed (timebase scale) that is consistent with your application. Long acquisition times and/or slow trigger rates can limit your throughput if your analysis program is very fast.

Power supplies

- If your power supply has list mode, use it to store complete instrument setup states and recall them with a single command, rather than sending a long series of configuration steps.
- Use the built-in measurement capabilities.
- Use power supplies with downprogramming capability.

RF/microwave sources and analyzers

- Agilent application notes offer many tips and tricks can be used to speed up measurements with RF/microwave sources and analyzers. See **www.agilent.com/find/open.**

Conclusion

To maximize system throughput, you need to choose the right equipment and program it for optimum speed. The system hardware and software architectures, instruments, switches, and I/O interfaces you select have a huge impact on system throughput. If you carefully evaluate the complex interplay of the hardware and software elements of your test system, you will find many opportunities for improving the speed with which your system performs measurements. After you've built your system, you can tweak instrument setups and operating procedures to optimize speed. The time you spend doing so will help lower your costs and accelerate your time to market.

8. Operational Maintenance

Introduction

This chapter examines important tasks and decision to consider as your system is put to use. It covers issues related to worldwide deployment, calibration, diagnostics and repair, cleaning, upgrades and expansion.

Once you've created and debugged your test system, you will be putting it to use. But even the best-designed system requires routine calibration and maintenance, and will occasionally fail. Planning for such eventualities will help to reduce the system's downtime.

The issues most often encountered:

- Worldwide deployment considerations
- Calibration
- Diagnostics and repair
- Cleaning
- Upgrades and expansion

Worldwide considerations

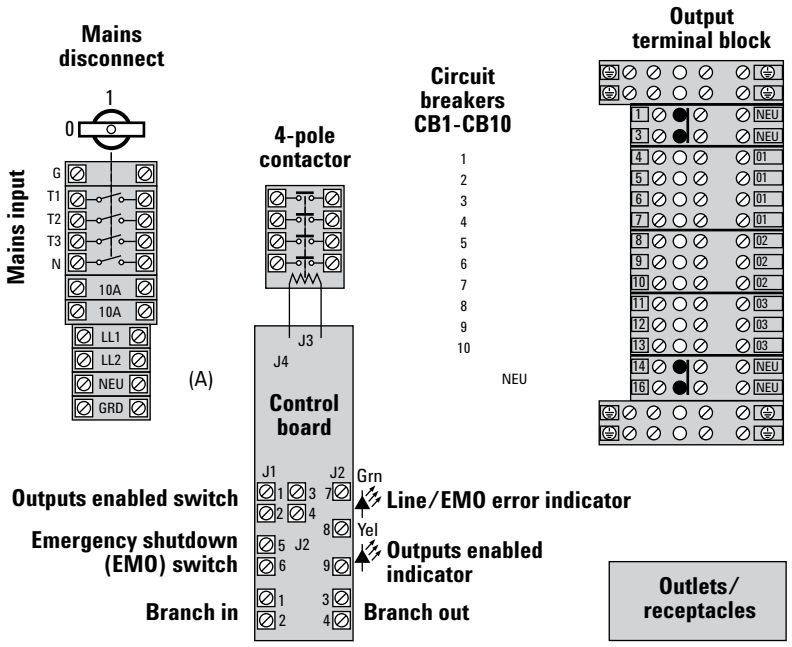
Systems are sometimes shipped from country to country as needs change or manufacturing lines are moved. If you are building a system that might be transported elsewhere, you need to account for the difference in line voltage and line frequency, both from the standpoint of equipment power input and changes to cooling fans that may be required. In addition, there are ergonomic considerations you should think about because of differences in culture or physical characteristics of the operators who will use the system.

Power

Your system is composed of instruments, power supplies and computing equipment that could all be required to run on different line voltages and frequencies. If your system will travel from country to country, you must plan for the changes in voltage or it will be a tedious job changing the equipment's fuses and input switches. Some older equipment must be removed from the system and have its top covers removed in order to reach the internal switches. If possible, choose equipment that runs from 90-252 V (to handle Japan's 100 V lines at low-line and Europe's 240 V lines at high-line) without requiring changes to switch settings or fuses. Information on the most common line voltages, power plug styles and other useful data for various parts of the world is available in *Electric Current Abroad*, a free publication from the U.S. Department of Commerce at **www.ita.doc.gov/media/Publications/pdf/current2002FINAL.pdf**.

Another useful item to consider when shipping systems from country to country is a power distribution unit (PDU). These devices can convert 3-phase inputs into line-to-neutral or line-to-line voltages, and they also can detect low- or high-line conditions. They sometimes can be connected to uninterruptible power supplies, too. A good PDU will also have an emergency off (EMO) switch input, allowing the operator to shut off all or some of the power in an emergency. Figure 8.1 shows typical wiring for a PDU that is used in many Agilent systems.

Figure 8.1. A typical AC power distribution unit.



Cooling

Fans are another problem area when line voltage varies. A 240 V fan may work when operated at 120 V, but at a much lower speed. Thus, the airflow may no longer be sufficient to cool the system. Conversely, a 120 V fan may burn up when connected to 240 V. It can be a nuisance to replace the fans every time the system is shipped from country to country. But fans that can be operated from any line voltage are produced in smaller quantities and are thus much more expensive than single-voltage AC fans.

DC fans, though, can be an excellent choice for systems that must be moved often. A small, fixed 12 V or 24 V DC power supply with universal AC input (i.e., 100-240 VAC) can be installed in the system and connected to the DC fan(s). Other advantages of DC fans are:

- More control over airflow and noise. The speed of the fan is directly related to the input voltage. A 24-volt DC fan can typically be operated between 12 and 28 volts DC. At 12 volts DC, the fan will operate at half speed, producing less air and less noise.
- The life expectancy of a DC fan is higher than that of a comparable AC fan, since DC fans are many times more efficient. The correspondingly low heat dissipation reduces the thermal load on the bearings, thereby increasing lifetime.

In non-air-conditioned factories, temperatures sometimes may exceed the ability of simple fans to keep the instruments operating within their specifications. In this case, consider a dedicated air conditioner for the system. NEMA enclosures are available for a wide variety of rack sizes. These completely enclose the system, and provide a way to attach air conditioner intake and exhaust. Appropriate ductwork must also be added to the factory. See www.nema.org.

Line frequency

The frequency of AC line voltage varies in different parts of the world. In the U.S., 60 Hz is standard. In many other countries, it is 50 Hz. While this won't affect most modern power supplies, it can certainly affect signal measurements. It is common to take low-noise DMM readings with a "1-line-cycle" integration time. At 60 Hz, this is 16.667 ms. At 50 Hz, it is 20 ms. Some DMMs, such as Agilent's 34401A, automatically adjust their integration time based on internally measured line frequency. Others must have this information programmed into them. It is important to set your DMM correctly based on line frequency.

At lower frequencies, the magnetizing current of transformers and motors can go up, even to the point of saturating the core. This can cause nonlinear magnetic fields and overheating of the core, especially at 47 Hz, creating a situation where products designed in a 60 Hz environment can cause problems in other parts of the world.

Logistics and ergonomics

The doorways in many older European buildings are short, and a 2-meter rack may not fit through the doors. Taller racks also require larger aircraft to transport them. If you build tall systems, your shipping costs may be significantly higher over the life of the system if it is moved or shipped frequently.

In some Asian countries where real estate is in scarce supply and space is at a premium, facility aisles and hallways are extremely narrow. It may be difficult or impossible to move a deeper- or wider-than-normal system to its intended location. Once positioned, it could be difficult to open front or rear doors.

The average population height varies country-to-country, too. Use care to place keyboards and monitors at an elevation that is not too high for shorter operators. It is also a good idea to provide keyboard/mouse trays with adjustable heights and provisions for left or right-handed operators. Your safety department can provide you with up-to-date guidelines for ergonomic standards.

Calibration

Most electronic instruments require periodic calibration that is traceable to a government standards agency such as NIST (National Institute of Standards and Technology) in the U.S. This requirement guarantees that measurements meet their published accuracy specifications. Calibration is not the same thing as diagnostics, which are simple tests to verify that the instrument is operating and taking measurements that are at least close to what they should be. Diagnostic tests and fixtures are discussed in the next section.

It may seem logical to build calibration fixtures that would allow your system to be automatically calibrated without having to remove equipment. Unfortunately, such fixtures would be prohibitively expensive. Calibration requires use of components that meet stringent specifications under closely controlled conditions of temperature and humidity. Oil-baths containing "standard" resistors at controlled temperatures, frequency-measuring equipment that connects to the NIST cesium-beam frequency standard and the like are not easily contained in a removable fixture.

There are three ways of assuring that a test system is calibrated:

- Have an in-house calibration lab perform calibration either in the system or by removing instruments, calibrating them and returning them to the system
- Hire a firm that provides calibration services at the location of your system
- Swap instruments with calibrated spares, then send the replaced units out for calibration

Whichever plan you use, it is essential to track the date of each instrument's last calibration, and to set up a method for notifying appropriate personnel when the next calibration due date arrives. You could simply place a dated sticker in a conspicuous place on the instrument whenever it is calibrated (see Figure 8.2), and have someone check dates periodically, or you could program the system with "due" software that notifies appropriate personnel automatically.

Figure 8.2. Cal sticker

Agilent LOVELAND STDS LAB 17025

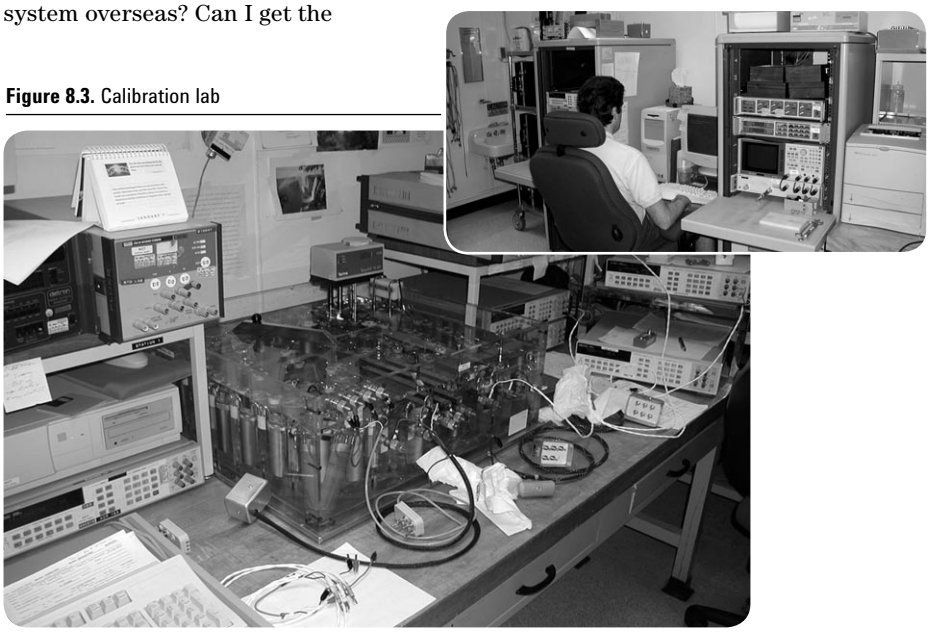
	DD	MM	YY	BY/NO.
CAL				
DUE				

In addition to regular calibration, keeping a log is a good practice. It's helpful to be able to correlate manufacturing anomalies to the particular operator, time of day, calibration period, run number and to many other manufacturing variables. Before you build that test system in Germany for shipment to Thailand, for example, try to answer these questions: Do I have the same calibration system in both places? If not, can I guarantee the measurements made by my test system here will be the same after I ship the test system overseas? Can I get the accuracy I need in both places, and are the calibration services adequate?

In-house calibration lab

If you do not already have an in-house calibration department (see Figure 8.3), you might consider setting one up, although the cost and time to do so can be considerable. If you intend to offer calibration services to others outside your company, your customers may require you to have international accreditation. A good place to start is the International Laboratory Accreditation Cooperation (**www.ilac.org**). Members of ILAC, such as the American Association for Laboratory Accreditation (A2LA—**www.a2la2.net**), will certify your lab after you have met their requirements, a process that can take from four to nine months once the lab is fully operational. If you desire to have your lab accredited, the international standard ISO 17025 will apply. It is not necessary to become accredited, but at the least, you may wish to become ISO 9000 certified.

Figure 8.3. Calibration lab



Contract services

For a broad range of calibration services covering many types of instruments, professional instrument calibration services are available from Agilent. See **www.agilent.com/find/calibration** for details. Non-Agilent equipment is included. Contracted services can be arranged in various levels, from single instruments on an as-needed basis to scheduled volume on-site calibration (VOSCAL).

Swap and return

The third method of calibrating equipment is to simply replace units when they need calibration with others that are still within their calibration period. This requires keeping one or two spares on hand, which can be expensive. However, it is a good idea to keep some spares handy anyway if system uptime is critical, as the next section discusses. There is one caveat in swapping instruments: A replacement may be completely within its calibration specifications, but if it is operating at the opposite end of its calibration range from the original instrument and the production device being tested is already near its limit, a statistical variation could result that is large enough to cause a yield problem. The solution is to run a statistical analysis on the results. This analysis is called a "Gage R&R" study, and it is covered in the next section.

Diagnostics and Repair

Perhaps the hardest thing to do once you have a test system finished is to spend some extra time designing a diagnostics test program that can help locate the source of problems when they arise. But it is time well spent. Here's what to do:

- Execute a self-test on every instrument that has this capability.
- Measure the output of every stimulus device with an appropriate measurement device to verify that all instruments are working and taking readings that are nominally correct. This is not sufficient to guarantee that they are in calibration, but it is good enough for a diagnostic tool.
- Feed a small DC voltage from a stimulus device (digital-to-analog converter, power supply, etc.) successively through all internally available switching paths and back to a DMM. This verifies the switching subsystem.
- Create a special diagnostic fixture that loops signals that cannot be automatically connected internally back into the system. Use the same procedure previously described to measure continuity of these paths.
- Read switch cycle count information from any switch box that has this capability. This data can give you early warning of relays that are nearing the end of their specified lives.
- Some instruments can do limited internal automatic calibration (sometimes called "auto-adjustment"). This automatic procedure should be done periodically, but not necessarily every time diagnostic test programs are run. Keep a programmatic calendar to remind the operator to run such programs when the due date occurs (usually about every 30 days).
- Attach a known good device under test (DUT) to the system and run a full suite of tests on it. This technique is not foolproof, since characteristics of such a "golden DUT" can change over time as components age. A useful way to counter this effect is to periodically run a "Gage Reliability and Reproducibility" (Gage R&R) test on the system. There are two sources of variation in any system: the variation of the product and the variation of the measurement system. The purpose of conducting the Gage R&R is to be able to distinguish between the two so as to reduce the measurement system variation if it is excessive. This means running a large quantity of known good boards on the system periodically to obtain a statistical sampling that can be compared to reference data to see if there is any long-term drift in the measurements. Such a study can also be used initially to study the measurement statistical parameters, which can be used to set acceptable upper and lower limits on each test. Look for statistical process control (SPC) and statistical quality control (SQC) software tools that can help you create such data.

In a production environment, diagnostics can be run daily or at the beginning of a shift. In a design validation or R&D environment, running the test once a week or less may be adequate. Once a problem is identified, the next step is to fix it. There are several things you can do to ensure fast repair:

- Make it easy to replace instruments. Make sure that mounting screws are not hidden, that cables are easily removed from the instrument (and labeled so they are replaced correctly), and that instruments are not hidden inside a rack, necessitating removal of other instruments in order to get to them.
- Although PCI slots in a rack-mounted computer are tempting spots to put instruments (since they do not take up additional rack space), remember that removing the computer from the rack to get to them is tedious and time-consuming.
- Use a limited set of custom cables and keep spares on-hand in case they need to be replaced. Use standard, easily available cables whenever possible.
- Fixture connectors can wear out over time. Have a good stock of replacement connectors available.
- Computers are a frequent source of problems. Hard disks fail, monitors quit, and keyboards and mice get dirty. Have spares available. Most importantly, keep important files somewhere else or back up the computer regularly to guard against loss of data.
- Maintain an inventory of spare instruments. This can be expensive, but so is a down production line. Remember, too, that the cost of many plug-in cards for PXI and VXI is greater than an equivalent rack-and-stack instrument because rack-and-stack instruments typically are produced in higher volumes. Thus, it is less expensive to inventory spares of box instruments, and they can double as debug tools when not in use inside a system.
- Place more than one of a key instrument in your system when you design it. For example, an inexpensive DMM could be integrated into the system for use during manual debug, but pressed into service should the main, high-speed DMM require service. With IVI drivers, such interchangeability should not require a change to the software.
- Heat and thermal gradients are enemies of any test system. Provide adequate airflow to minimize heat rise, and avoid a situation where you are continually changing the thermal environment of the test equipment.

Cleaning

Maintaining good airflow through your system is essential, because it keeps the temperature under control, assuring that instruments are operated within their temperature specifications. Many instruments have removable air filters, so be sure to inspect these regularly and clean or replace them when necessary. Some racks are also available with air filters. These should also be inspected regularly. Keep cables away from the filters. If cables must be moved in order to reach the filters, the flexing can make the cables eventually break, causing reliability problems unrelated to dirty air.

If many operators will be using the system, it is a good idea to periodically clean the keyboard, mouse, barcode reader and touchscreen, as applicable. You generally can use simple household cleansers. Disease can be spread easily from one person to the next via these devices. Trained operators may be hard to find, so keep them healthy!

Upgrades and expansion

If you've designed your system well, using the concepts highlighted in earlier chapters, it will be able to handle new instruments easily. You've left extra space in the rack for additional or bigger instruments, and you've allowed expansion room in your switching or instrumentation cardcage if present. You've also designed the switching system in such a way as to allow instruments to be added to the system by simply plugging the new inputs and outputs into a place you've reserved for future instruments (such as the unused rows of a switching matrix, as described in Chapter 5, *Choosing Your Test-System Hardware Architecture and Instrumentation*). You've got room in your fixturing system for more pins, and you've developed a small set of reusable cables to connect those into your instruments and switches.

In the software realm, you've planned for upgrades by doing regression testing every time a major piece of software is changed. This means allowing time to re-run the Gage R&R, diagnostic test plan and/or known good DUT when the operating system, test executive, drivers or other support routines are modified. You've also documented the software and allowed for code changes to be easily tracked. You've written the software in an environment standard to the PC industry so anyone familiar with languages such as Visual Basic or C can take over the system software and make necessary changes as the years go by.

Conclusion

Test systems have made the task of repetitive testing both faster and more reliable, but there's much to consider to keep them running. You must factor in worldwide power issues, calibration, diagnostics, repair, cleaning, upgrades and expansion. At Agilent, we appreciate the talent and effort required to design, build and implement exceptional test systems. If you are creating a test system or need help with one you already use, you can find lots of advice at **www.agilent.com/find/open**.

Section 2. Networking Choices

Overview

The seven chapters in this section explore the range of networking options available for test system automation:

9. **Using LAN in Test Systems: The Basics,** provides an introduction to the essential elements of local-area networking (LAN), the basic attributes of test systems, and the benefits of using a LAN interface for control and data transfer in a system.

10. **Using LAN in Test Systems: Network Configuration,** describes the potential risks of networking a test system, suggests two secure topologies for LAN-based test systems, and outlines the essential aspects of system configuration.

11. **Using LAN in Test Systems: PC Configuration,** describes the steps required to enable communication between a PC and LAN-enabled instrumentation, including network settings in Windows XP and IP address assignments.

12. **Using USB in the Test and Measurement Environment,** offers a closer look at the universal serial bus (USB) as a test system connectivity option, including USB connectivity and data rate options.

3. **Using SCPI and Direct IO vs. Drivers,** outlines the relationship between input/output (I/O) software, application software and the ability to maximize instrument interchange and software reuse in present and future systems.

14. **Using LAN in Test Systems: Applications,** offers advice on balancing cost, convenience and security in three common LAN scenarios: sharing instruments, remote monitoring and data acquisition, and functional test systems.

15. **Using LAN in Test Systems: Setting Up System I/O,** describes the components of the Agilent IO Libraries Suite and presents a quick, six-step process that will make LAN-based instrument connections as simple as using GPIB.

9. Using LAN in Test Systems: The Basics

Introduction

This chapter provides an introduction to the essential elements of local-area networking (LAN), the basic attributes of test systems, and the benefits of using a LAN interface for control and data transfer in a system.

Coping with complexity

The basic purpose of any test system is to characterize and validate the performance of electronic components, assemblies or products. The complexity of this task depends on variables such as the physical nature of the device under test (DUT), the number of tests to be performed, the number of signals to be measured and the desired time per test.

The number of instruments used in the system can further complicate the task—and put a heavy burden on the digital input and output (I/O) between the system computer (usually a PC) and the test equipment (Figure 9.1). One of the best ways to cope with a high volume of I/O traffic—commands, status messages, test data—is LAN technology, a fast, open and low-cost alternative for system I/O.

Figure 9.1. PC and test instruments in a rack

Setting the standard

Today's most pervasive computer networking standard goes by a few well-known names: IEEE 802.3, Ethernet, 100Base-T, 1000Base-T, or Gigabit Ethernet. Some variant of this standard is almost always used when PCs share files, exchange e-mail, access the Internet and so on. With steady improvements in cost, speed and functionality, Ethernet has achieved virtually universal adoption for local-area networking (to the extent that LAN and Ethernet are sometimes used as synonyms).

Devices are often described as being 100Base-T, 1000Base-T and 100/1000Base-T. The number indicates the data rate in megabits per second: 100Base-T is 100 Mbps and 1000Base-T is 1000 Mbps; 100/1000Base-T devices are compatible with both standards. The T indicates unshielded twisted pair (UTP) wiring to differentiate it from older standards that used coaxial cable.

Today, 100 Mbps technology is the most widely deployed standard and provides ample performance for most uses. Consistent with its history, the standard continues to evolve: Gigabit Ethernet was standardized in 1998 and is now widely deployed and 10 Gigabit Ethernet has been more recently standardized.

Tremendous competition among vendors of Ethernet-based LAN devices and cables has driven down prices and driven up the volume of products sold. The net result is a wide selection of high-quality, low-cost solutions for local-area networking.

Defining key attributes and elements

Wired LAN connections are made with UTP cables called Category 5e, commonly referred to as Cat 5e, which is the name of a wiring standard defined by the Telecommunications Industry Association and Electronics Industries Association (TIA/EIA). (Cat 5e replaces the Cat 5 standard that has been in use with LANs for a number of years.) A CAT5 LAN cable contains four pairs of copper wire and uses locking RJ-45 connectors at both ends (Figure 9.2). It is largely immune to interference and crosstalk and can support data rates of up to 1000 Mbps.

The other essential elements of a LAN are the hardware devices that control, manage, direct and amplify the data being sent between other devices on the network.

- **Adapter.** This refers to the LAN card and connector in a PC (and some new-generation test equipment) that provides an electrical interface to the network.
- **Hub.** A small, standalone unit that connects multiple devices. Hubs use a broadcast model to transmit data, a method that reduces the effective bandwidth (or data rate) when network traffic is heavy.

Assessing wireless LAN alternatives

In many companies, an increase in workforce mobility has led to greater demand for flexible networking solutions, most notably wireless LAN (WLAN). As with wired LAN, there is an evolutionary series of standards that go by various names, but all are generally known as Wi-Fi (short for "wireless fidelity"). The four main standards are described below in order of commercial introduction:

- **IEEE 802.11b:** Uses radio transmissions at 2.4 GHz to send data at up to 11 Mbps and with an indoor range of 100-150 feet.
- **IEEE 802.11g:** Uses 2.4 GHz transmissions to send data at up to 54 Mbps and with an indoor range of 100-150 feet. It interoperates with 802.11b. (Some vendors also offer proprietary extensions that can provide higher performance.)
- **IEEE 802.11a:** Uses 5.0 GHz transmissions to send data at up to 54 Mbps and with an indoor range of 25-75 feet. 802.11a is not compatible with 802.11b and g because it uses a different modulation method.
- **IEEE 802.11n:** An emerging standard, backwards compatible with 802.11b and g, which should offer data rates up to 10 times faster than 802.11a or 802.11g and up to 50 times faster than 802.11b.

For test systems, WLAN can enable measurements in remote or hazardous settings and provide an alternative to costly cable runs. However, none of the current standards can match the combination of speed, reach and noise immunity possible with a 100Base-T wired LAN. What's more, WLAN signals are susceptible to interference from other devices that operate in the same frequency range, including cordless phones and microwave ovens. Wi-Fi signals may also interfere with the testing of wireless DUTs.

- **Switch.** Another standalone unit that connects multiple devices on a LAN, usually in what's called a "star" topology (Figure 9.3). In this configuration, any device can discover and talk to any other device on the LAN. Because switches contain more intelligence than hubs, including the ability to send data to a specific destination and to devote the entire bandwidth of the network to any segment of the network, they typically provide better performance than hubs.
- **Bridge**. Similar to a switch but with just one input and one output. Used to break networks into segments, which can improve the performance within each segment.
- **Repeater.** Similar to a bridge with one input and one output but contains active circuitry that reads and regenerates the incoming signal. Used to extend the length of a network segment.
- **Router**. A standalone box that joins multiple networks (wired or wireless) through its ability to handle high-level protocols such as TCP/IP (see "Connecting Ethernet and Internet"). Routers allow one- and two-way communication between devices and enable "awareness" among the devices on a network. They also allow devices to hide their presence, enabling the creation of small, private networks. A router performs the functions of a switch, but also joins other networks. This is most useful when you need a local network for your test system, but also need to connect to a corporate network.

We recommend that every standard test system use either a switch or a router. Also, note that the maximum cable length for any segment of a LAN is 100 meters (about 328 feet). Hubs or switches can extend that distance to roughly 1,600 meters (about one mile) and the use of routers, switches, bridges or repeaters between LAN segments can yield a network of virtually unlimited reach.

Connecting ethernet and internet

TCP/IP stands for "transfer control protocol" and "Internet protocol," two separate standards that work together to provide the foundation of data communication on the Internet. For example, Web browsers use TCP/IP to communicate with Web servers. TCP/IP also enables seamless connections between local Ethernet networks (also called intranets) and the Internet, and between different types of computers (e.g., Windows, UNIX and Linux).

Technically speaking, Ethernet is just one type of network technology that can carry TCP and IP traffic. Other examples include Token Ring (IEEE 802.5), DOCSIS (cable modem), xDSL and ISDN.

Figure 9.2. A CAT5 LAN cable with RJ-45 connector.

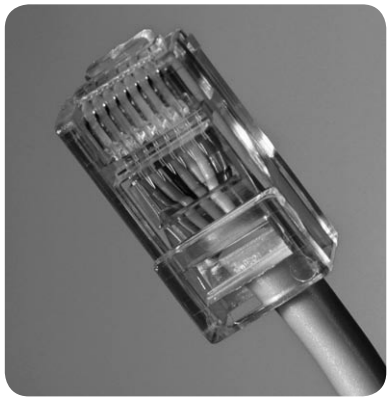
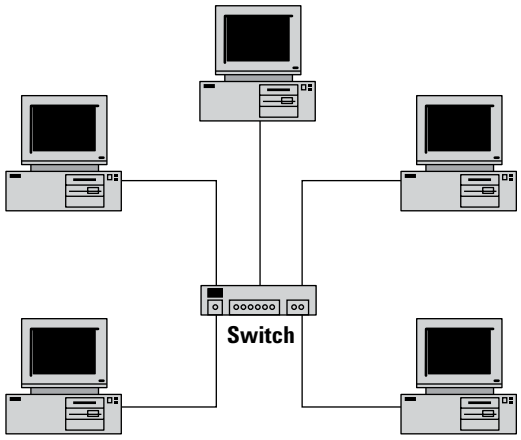


Figure 9.3. A simple network with one switch and multiple PCs in a star topology.



Using LAN in test systems

The computer in a test system plays three important roles, two of which rely on the I/O connection to the test equipment:

- It provides fast, reliable control by sending commands, configuring instruments, reading status messages and initiating measurements.
- It gathers test data—raw or preprocessed—from the instruments and, if necessary, stores it for postprocessing and archival purposes.
- After testing one or more products, the computer (and its test software) may also analyze the results and provide reports for further evaluation by engineering staff, manufacturing management, contract manufacturers and others.

As each task becomes more data intensive, the choice of I/O interface becomes more significant (Figure 9.4). Speed is an important factor, but a test-system connection must also be rugged, noise-tolerant and able to handle multiple instruments.

As Chapter 2 discusses in detail, the advantages of LAN technology make it a good choice for meeting the I/O needs of test systems. With LAN adapters built into most current-generation PCs, the computing portion of the system requires minimal physical configuration to support test system deployment.

This situation is driving the addition of LAN connectors and adapters to current- and next-generation test equipment. The LXI Consortium has chosen LAN as the interface for test equipment (for more information on LXI, refer to Chapter 16). The inclusion of both LAN and GPIB may be quite common for the next few years. One recent example is the Agilent 33220A function generator (Figure 9.5), which has LAN, USB and GPIB interfaces built into its rear panel. Agilent recognizes the pervasiveness of GPIB in existing test systems and will continue to support it. At the same time, we are also committed to providing LXI compatibility in new instruments, making it the most prevalent interface in the near future.

Figure 9.4. An example test system that utilizes a PC, several instruments, a LAN router, a LAN/GPIB gateway and three types of I/O

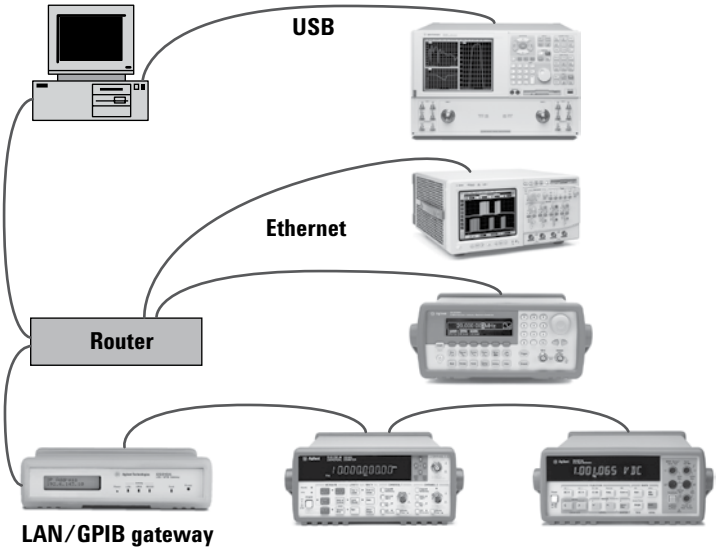


Figure 9.5. The rear panel of the 33220A function generator includes USB, LAN and GPIB interfaces.

As a near-term solution, standalone gateway devices make it possible to connect current-generation PCs to older test equipment and systems. The Agilent E5810A LAN/GPIB gateway (see Chapter 2, Figure 2.5) has a LAN port for the PC and a GPIB port that can control up to 14 instruments. By providing LAN-based access to a test system, the gateway enables useful capabilities such as remote monitoring of test progress and collaboration and consultation with distant colleagues. It also has a built-in Web server, which lets you use a browser to set up, configure and use the gateway—and control instruments—from a remote computer.

The E5810A gateway is fully supported by the Agilent IO Libraries Suite, which enables automatic control of instrumentation from a variety of programming languages. The gateway also has Universal Plug&Play (UPnP) support, making it appear as a network device in Windows XP and Vista.

Communicating with instruments

LAN support in a test instrument usually means three things. First is a 100/1000Base-T adapter, which is compatible with today's most commonly deployed LAN equipment. Next is the locking RJ-45 connector that ensures a dependable connection to the instrument as well as the hub, switch or PC at the other end. The third element is the test and measurement communication protocol called VXI-11.

VXI refers to both a test-and-measurement standards body and its well-known multi-vendor standard for modular, cardcage-based test systems. VXI-11 is a more recent standard that defines LAN-based connectivity for all types of test equipment, not just VXI.

The VXI-11 protocol makes the I/O connection appear to PC applications as though the instruments were connected via GPIB. In practice, this means applications written for GPIB are likely to work on VXI-11 instruments, especially if they use the VISA I/O API—the Virtual Instrument Software Architecture's input/output application programming interface. (VISA is also a multi-vendor standard.)[1]

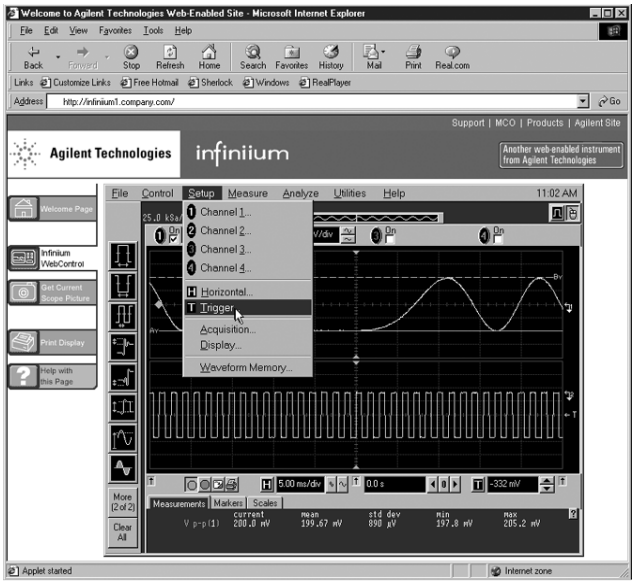
1 To learn more, visit www.vxibus.org/specs.html (VXI-11) and www.vxipnp.org (VISA).

Enabling additional capabilities

Through the many available applications of Ethernet and TCP/IP, LAN-enabled instruments can do more than just support VXI-11. For example, they can be equipped with built-in Web servers. A good example of this capability is the Agilent Infiniium family, which includes digital sampling oscilloscopes, mixed-signal oscilloscopes and digital communication analyzers. By pointing a Web browser at the instrument's IP address, the user can view the instrument configuration, change its settings, start a measurement and see the results (Figure 9.6).

Some LAN-equipped instruments provide even greater functionality: inside every Infiniium product is a PC running custom software on a version of Microsoft Windows. Windows has several LAN services built in, enabling capabilities such as sharing of files, folders, drives and printers.

Figure 9.6. The virtual front panel of the Infiniium oscilloscope enables browser-based interaction with the instrument.



Conclusion

Fast and inexpensive LAN technology has achieved widespread adoption in the computer world and is now shaping the future of test system development and operation. LAN-based systems provide several advantages for test-and-measurement applications: lower-cost hardware and cabling; pervasive availability throughout most enterprises; remote or shared system control; fast data transfers; file, drive and printer sharing; and browser-based interaction with individual instruments. The LXI standard standardizes the LAN protocols and connectors as well as a host of other items for the test and measurement industry, thus ensuring easy interoperability of instrumentation.

For decades, the robust GPIB interface has been the dominant I/O for test systems. Agilent is committed to supporting GPIB well into the future—and we are also committed to developing new-generation test equipment that includes both GPIB and LAN interfaces.

To learn more about I/O connections and other ways to simplify system integration and apply the advantages of open connectivity, please visit **www.agilent.com/find/open.**

10. Using LAN in Test Systems: Network Configuration and Basic Security

Introduction

This chapter describes the potential risks of using LAN in test systems, suggests two secure topologies for LAN-based test systems and outlines the essential aspects of system configuration.

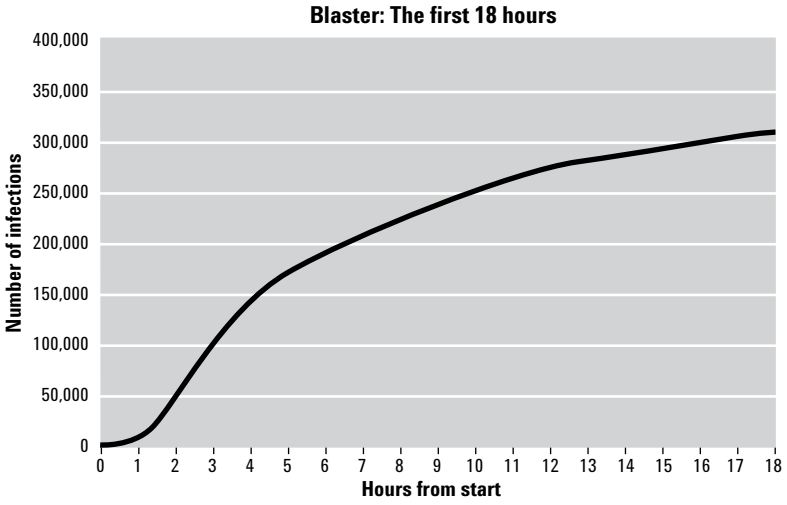
Creating a safe haven

The decision to use LAN in a test system delivers important benefits to your company and your team. From a business perspective, intense competition among equipment vendors has produced a wide selection of high quality, low-cost solutions for local area networking. From an organizational view, widespread use of LAN technology simplifies connectivity and enables new levels of communication and collaboration between team members, wherever they may be in the world.

Of course, the use of any pervasive computing technology also carries risks. Adding a LAN connection can open the door to inadvertent threats carried on a company's intranet, and may expose a test system to a variety of malicious threats from the Internet (Figure 10.1).

Fortunately, there are effective, practical solutions that can protect your system from internal and external risks. Our recommended starting point is to create a protected, private LAN for the test system. The standard capabilities of most Microsoft Windows PCs and many low-cost networking products enable two viable approaches, one router-based and the other PC-based. Several factors will influence your choice, and your decision has implications for the selection and configuration of the PC, the network and the test instrumentation.

Figure 10.1. The Blaster worm infected more than 300,000 computers in less than 18 hours



Understanding the pitfalls

Test systems that aren't connected to enterprise networks are sometimes labeled as "islands of automation." However, their isolation provides an unintended benefit: standalone test systems are insulated from the viruses, worms and Trojans that might strike a company's network.

For a system on an island, the biggest risks come from human interference. System errors might arise if a configuration change is made via the front panel or if two instruments are set to the same GPIB address. These problems are easy to fix and the integrity of the system remains intact.

Recognizing potential threats

Connecting the system's host PC to the company network builds a bridge to the island. It also opens the door to a wider range of threats—including some that may compromise system security and integrity.

Inadvertent threats may reach the system via the company intranet. Some are programmatic, as when another PC on the network causes a configuration change in one or more instruments. Others are systematic, such as configuring the test instruments for dynamic rather than static IP addresses may cause unexpected operation. As an example, if the IP addresses of two power supplies are reversed, the device under test (DUT) could receive the wrong voltages at the wrong points and suffer severe damage.

Malicious threats from the Internet may breach the company's firewall, spread via the intranet and infect the system's host PC. These threats also pose a potential risk to any instruments that contain a Windows PC. One answer is to include a hardware or software firewall in each instrument—a solution Agilent is enabling in next-generation instruments.

Examining other issues

A LAN-based system is also subject to the quirks and limitations of the deployed hardware. As an example, the simplest way to connect a system to the corporate network is through a hub. However, hubs let all network traffic flow in both directions; all intranet traffic would be present within the test system and all test system traffic would appear on the intranet. Excess network traffic could degrade system throughput and the broadcasting of test results on the intranet could be a security risk. Using a switch or router is a better choice because both are specific and selective about filtering and forwarding network traffic.

Some older LAN-enabled instruments also have two weaknesses that must be addressed or acknowledged: the inability to lock connections and authenticate devices. For example, those that don't support the VXI-11 communication protocol (or provide partial support) probably can't create a locked LAN I/O session between the instrument and a PC.

Locking ensures a stable PC-to-instrument connection and also blocks other attempts to access the instrument for the duration of a session. Instruments that do support VXI-11 have an important shortcoming when not locked into a session: they have no authentication capabilities (e.g., password protection) to block unauthorized access. In this case, any PC on the network that supports VXI-11 can access the instrument and easily disrupt its behavior. The solution is a private LAN that limits access to only those devices you trust.

Designing the private, protected LAN

Our basic prescription for any LAN-based test system is to create a private, protected network that includes the host PC and the test equipment. Fortunately, there are two practical, effective ways to set up this type of network. One approach is built around a LAN router, which provides a buffer between the test system and the corporate intranet. The other approach uses the host PC as the buffer by configuring it with two LAN cards and the Internet Connection Sharing (ICS) feature of Windows XP and Vista.

The router-based approach

A router is a standalone box with multiple LAN connectors, one for the external or "public" network and four (or more) for the internal or "private" network. The router links these networks through its ability to handle high-level communication protocols such as TCP/IP. Routers allow one- and two-way communication between devices and also enable "awareness" among devices on a network.

Routers also utilize a feature called network address translation (NAT) that allows devices to hide their presence from public networks. It does this by using a private set of IP addresses that are not revealed to devices on the public side. This is the key attribute that enables the creation of a private LAN for a test system.

As shown in Figure 10.2, the router is the focal point of the network. In the simplest router-based system, its "external" port, usually labeled Internet or WAN (wide area network), is connected to the corporate intranet. Its other ports, usually labeled LAN, are connected to the host PC and a few LAN-enabled instruments. Additional instruments can be added by connecting a switch or hub to one or more LAN ports on the router (Figure 10.3).

Figure 10.2. A test system that uses a router-based private, protected LAN

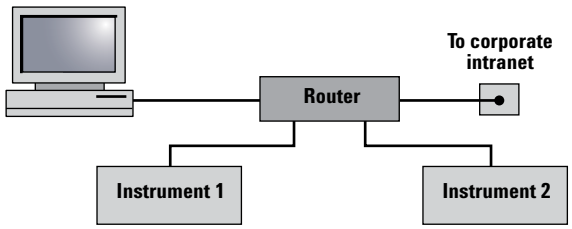
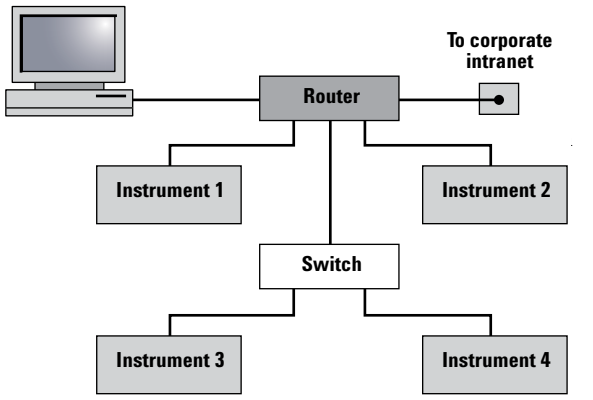


Figure 10.3. An expanded network that uses a switch to connect additional instruments to the test system



shields the system from intranet congestion by isolating all but local traffic. What's more, the router safeguards system operation from the effects of administrative activity or hardware problems on the local intranet because it provides all of the network services needed by the instruments and the host PC. At the same time, the router gives the PC unhindered access to the system network as well as the corporate intranet and the Internet. It also gives all LAN-enabled instruments access to TCP/IP resources on the intranet and the Internet.

The configuration process is relatively simple for the router and the PC—to the extent that you usually won't have to burden corporate IT personnel with the task. For example, a host PC that's already equipped with a LAN card doesn't require any hardware modifications. The only PC configuration change, made after the router is installed and enabled, is to activate dynamic host configuration protocol (DHCP), which is a method of automatically assigning an IP address to any device connected to a LAN. (DHCP may be turned on by default, but it's best to verify this setting.)

Instrument configuration is also quite simple. The only changes are deactivating DHCP then setting the IP address, subnet mask and default gateway. These tasks are easy to complete via the front panel or web interface of most LAN-enabled instruments. A more detailed description of the configuration process is presented in Appendix 10A.

Defining router and PC features

Successful implementation of the router-based private network requires a few essential capabilities. Must-have router features

- **Network address translation (NAT).** NAT allows the router to act as an agent between the public and private networks, mapping private IP addresses to public IP addresses and enabling communication between networkss.
- **De-militarized zone (DMZ)** . The DMZ feature makes it possible to give a PC (or instrument) complete access to the Internet, effectively putting it "outside the firewall." With DMZ, other computers outside the private network can connect to the host PC and use its public services (e.g., shared file folders or a Web server). Because DMZ is implemented differently on various router models, you should verify that you are able to achieve essential tasks such as communication with a manufacturing database Should-have router features:
- **Sufficient ports**. Each device should have its own LAN po**.** Each device should have its own LAN port, either in the router or via one or more LAN switches or hubs connected to router ports.
- **Adequate port speed.** The router should support at least 100 Mbps (100Base-T) on each LAN port (the private side) as well as 100 Mbps on the WAN port (the public side). 1000 Mbps (Gigabit or 1000Base-T) routers and PCs are widely available and recommended for best performance.
- **Built-in DHCP server.** The router assigns IP addresses to the LAN devices attached to its private LAN ports. Some routers keep a table in non-volatile memory that provides a mapping between the assigned IP addresses and the associated Ethernet devices on the private network. Vendors call this capability by many names, including "static DHCP," "DHCP client reservation," "fixed mapping," and "MAC address to IP mapping." (MAC stands for media access control.)

The router-based approach also requires a host PC that uses TCP/IP rather than NetBEUI, IPX or SPX[1] as its network communication protocol. The PC may use DHCP to ensure assignment of a unique IP address, or it can be configured with a static, internal IP address that is compatible with the router's configuration.

1 NetworkBIOS Extensions User Interface, Internetwork Packet eXchange and Sequenced Packet eXchange are alternatives to TCP/IP for network communications. If installed in the host PC, their presence can create problems within the network.

The PC-based approach

By adding a second LAN card and activating ICS in Windows XP or Vista, the host PC can serve as the router in the network (Figure 10.4). ICS routes traffic from one LAN card to the other and provides NAT capabilities for the private addresses.

This method has several advantages in common with the router-based solution: it provides access control, blocks Trojans and worms, and gives the host PC unhindered access to the system network, the intranet and the Internet. LAN-enabled instruments can also access the intranet and the Internet. However, if the host PC is configured to use DHCP rather than a static address then it will have to rely on the corporate intranet being functional and able to provide an IP address.

Although it probably isn't a major obstacle, this approach requires that you are comfortable with the prospect of opening up the PC, installing the second LAN card, and configuring the PC to ensure the peaceful coexistence of two LAN cards.[1]

1 It is also possible to use a USB-to-Ethernet adapter as the second LAN port, but there would be some latency in this connection—and the configuration process is slightly more complex.

The most important step is the configuration of ICS within the host PC, which must be running Windows Vista, Windows XP with Service Pack 1 (SP1), Service Pack 1a (SP1a) or Service Pack 2 (SP2) (Microsoft service packs are cumulative).[2]

2 Other operating system configurations may work but this note focuses on the most recent versions of Windows.

Through the Network Connections control panel, both LAN cards can be enabled and the one connected to the public network can be shared. You then use the Local Area Connection Properties window to enable ICS (Figure 10.5).

Figure 10.4. The PC-based solution, with two LAN cards in the PC and a switch to connect the instruments

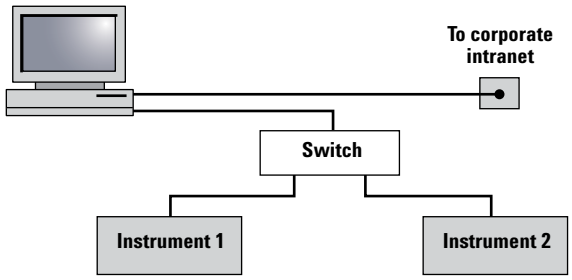
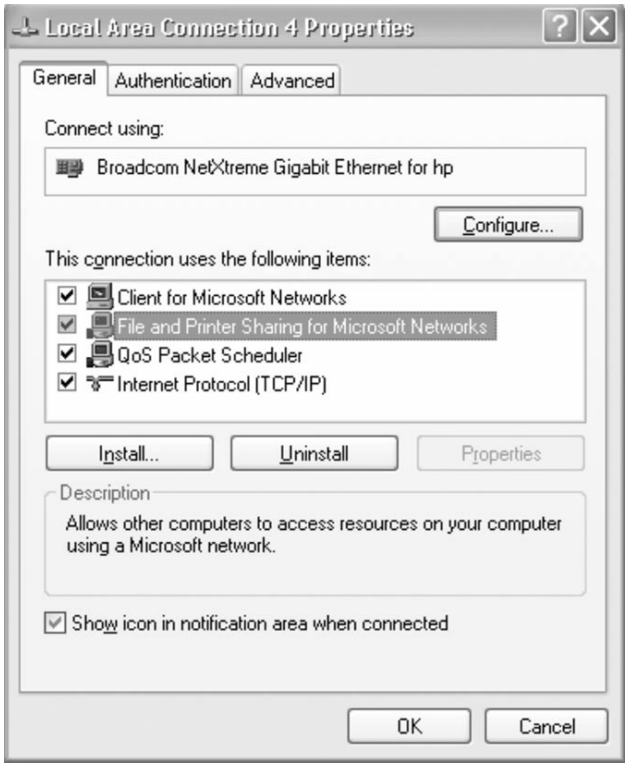


Figure 10.5. Use the Local Area Connection properties window in Windows XP to activate Internet Connection Sharing.



Instrument configuration

Keep one important caveat in mind when using either approach: the default mode of both ICE and a router is to dynamically assign an IP address to every device that joins the network. This is done via DHCP, which prevents addressing conflicts but also creates the possibility of assigning a different address to each test instrument every time it is powered up or reconnected to the network. As described earlier, unwanted address changes can result in improper operation and damaged DUTs.

The easiest way to prevent address changes is to disable DHCP in each instrument and then enter a static (fixed) IP address. Though this will be easy to accomplish via the front panel or web interface of most newer LAN-enabled instruments, it may not be possible with some older equipment. In those cases, the easiest solution is to use the GPIB interface on the instrument and add a LAN/GPIB gateway such as the Agilent E5810A to the network.

The IP addresses you assign to the instruments should only differ from the IP address of the router by the last of the four numbers in the IP address (e.g., 192.168.0.x). You may want to use numbers higher than 200, reserving the first few digits for any DHCP-enabled devices for which the router will typically assign an address in that low range. Applying these ideas to a system that includes a router at the IP address 192.168.0.1, the instruments could use numbers in the range of 192.168.0.200 to 192.168.0.255.

It is also necessary to configure the instruments with the proper subnet mask (usually 255.255.255.0) and default gateway, which is the IP address of the router itself (typically 192.168.0.1, 192.168.1.1 or similar, depending on the router maker).

Once you've saved these settings, you may have to cycle power on each instrument for the changes to take effect. After each instrument has completed its boot-up operations you can then connect it to a LAN port on the router.

Conclusion

The decision to use LAN for system I/O delivers valuable benefits to your company and your team. However, it also opens the door to malicious threats and inadvertent risks that can affect system performance and integrity. The creation of a private LAN can protect the test system from those risks and ensure maximum throughput. Using the standard networking capabilities of today's PCs and the low-cost networking products now available, you can chose either a router-based or PC-based approach.

Both approaches protect the test system from the potential hazards carried on the intranet or Internet, prevent any type of unauthorized outside access, and shield the system from intranet congestion by isolating all but local traffic. The router-based approach has the additional benefits of safeguarding system operation from the effects of administrative activity or hardware problems on the local intranet because the router provides all of the network services needed by the instruments and the host PC.

Appendix 10A: Configuring the router-based system

Of the two solutions described in this chapter, the router-based system is more flexible and therefore more likely to be widely used. The specifics of the configuration process depend on the actual products used to assemble the system. However, three essential steps provide a framework for the implementation of any router-based solution: capturing network information, configuring the router and setting up the test instruments.

Capturing network information

You'll need to record some information about the network and use it to set up the router, which will be inserted between the PC and the intranet. That way, the PC is already programmed with everything you need to know about its network configuration. You'll need to record that information and use it to set up the router.

What you need

Configuration requires the host PC, powered up and connected to the intranet; the router; one LAN cable for the PC and one LAN cable for each instrument.

The process

1. Power up the router.

2. Disconnect the intranet LAN cable from the PC. Use another LAN cable to connect the PC's LAN port to any LAN port on the router. Wait a minute or so to ensure the PC and router are synched.

3. From the PC's Start menu, open a DOS or Command window and type in *ipconfig/all.* This will display several items including "Host Name" and "Physical Address." The PC's host name is registered with the corporate DNS services. The physical address is the unique MAC or Ethernet address of the LAN card in the PC. Write down the host name, the physical address, and the IP address of your computer: you'll use that information later when configuring the router.

To create a new, private network that consists of just the PC and the router, return to the DOS or Command window and type in *ipconfig/renew*.

Configuring the router

The router must be configured to mimic the test system PC on the corporate intranet. Most routers provide a browser-based interface that lets you use any Web browser to log in and modify the configuration. Consult the router's manual for its URL and the default login values for user name and password. Launch your Web browser, type in the proper URL and log in to the router's configuration page. At this point, the details vary by vendor and product. There might be a built-in wizard function, or you may have to navigate through various configuration screens and enter values manually. Either way, you need to accomplish five tasks:

1. Enter the PC's host name.
2. Enable cloning of the PC's MAC address.
3. Modify the security settings to disable blocking of anonymous ping requests. (Allowing other computers to ping the host PC may be a requirement of some corporate intranets.)
4. Enable the DMZ capability and set the DMZ host IP address to 192.168.x.100 (the x must match the value used by your router). This is the default first address assigned by the router's DHCP server and must be used as the IP address for the host PC. Some routers may use different initial addresses: type in the *ipconfig/all* command to find out what address the router assigned the PC after they were connected.
5. Save all of these settings. Locate the intranet cable that was originally connected to the PC and plug it into the router's WAN or Internet port. To verify proper operation, open a DOS or Command window and type *ipconfig/release*. Next, type *ipconfig/renew*: the host PC should now be able to access the corporate intranet via the router.

Setting up the instruments

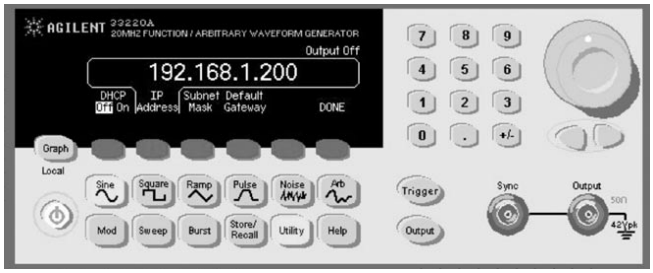
The final step is to configure the test instruments with static IP addresses. Use the front panel keys of each LAN-enabled instrument to access the I/O, Utility or IP Setup configuration menu and disable DHCP. Next, give each instrument a unique IP address in the range of 192.168.x.200 to 192.168.x.255 (Figure 10.6). These values are outside the range of IP addresses routers typically assign to network devices (192.168.x.100 to 192.168.x.149).

You'll also need to navigate the configuration menu and set the subnet mask to 255.255.255.0 and the default gateway to the router's IP address (192.168.0.1, 192.168.1.1 or similar, depending on which brand of router you're using).

Once you've saved these settings, you'll have to cycle power on each instrument for the changes to take effect. After each instrument has completed its boot-up operations, use a LAN cable to connect each one to a LAN port on the router.

To verify proper configuration, open a DOS or Command window and type *ping 192.168.1.200* or any other valid IP address you assigned to an instrument. To verify access to the intranet, launch a Web browser and try a few internal URLs. If these load as expected, this verifies proper communication with the intranet.

Figure 10.6. The IP Setup menu of the Agilent 33220A function/arbitrary waveform generator makes it easy to set the IP address, subnet mask and default gateway.



11. Using LAN in Test Systems: PC Configuration

Introduction

This chapter builds on the information presented in Chapters 9 and 10, describing the additional capabilities required to enable communication between a PC and LAN-enabled instrumentation.

Creating the right environment

The evolution of LAN technology continues to drive improvements in cost, speed, functionality and ease of use. This has created at least two noteworthy trends. One is the widespread use of LANs within most businesses. The second is the inclusion of LAN as a standard feature of most new PCs. A third trend is emerging in test-system development: the advantages of LAN technology are making it an attractive alternative to GPIB for system input/output (I/O).

Chapter 16 offers a closer look at LXI, the new LAN-based standard for computer-instrument communication. Backed by the LXI Consortium, which includes every major test and measurement company, the LXI standard ensures interoperability among vendors' LAN-based instruments and simplifies configuration and programming of LAN-based systems. Given the advantages of LAN and LXI, LAN interfaces are becoming more common in test equipment—though LAN ports will likely coexist with GPIB for years to come.

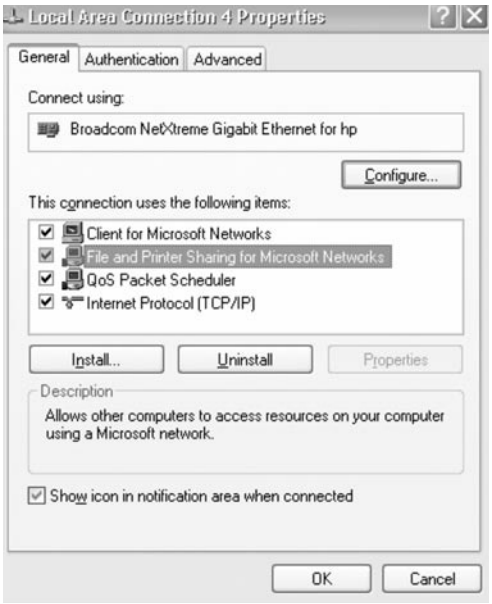
On the surface, the use of LAN to communicate with instruments seems like it should be as simple as connecting a printer to a PC: just grab a network cable and make the connection. Unfortunately, it requires a bit more effort to create the right environment within a PC for transparent communication with LAN-equipped instruments. Making it work depends on the LAN services of Windows XP and Vista, and the additional capabilities provided by a suite of I/O libraries from Agilent make it nearly "printer easy."

Exploring network settings in Windows XP and Vista

A standard Windows XP or Vista installation includes three software components that enable networking: Client for Microsoft Networks, File and Printer Sharing for Microsoft Networks, and Internet Protocol (TCP/IP).1 TCP/IP is the network protocol that enables data communication with the Internet, your corporate intranet and other computers.

You access the PC's TCP/IP settings through the Windows Control Panel and its Network Connections icon (or Network and Internet Connections category). The Network Connections window will present every installed LAN card under the default name "Local Area Connection" (you can change this name). Each entry will indicate the device's connection status (enabled or disabled) and manufacturer and model. Right clicking on a Local Area Connection entry will bring up a menu that includes a Properties selection that opens the Local Area Connection Properties window (Figure 11.1). This window includes a list of check-box items, and "Internet Protocol (TCP/IP)" is the last item in the list. Clicking on that item (not the check box) and then clicking the "Properties" button will bring up the TCP/IP configuration window. This is where you can change IP address settings (static or dynamic) and other networking parameters.

Figure 11.1. Use the Local Area Connection Properties window to set networking parameters such as the IP address.



1 Transfer Control Protocol, Internet Protocol. Please refer to the glossary for more on networking terms.

Using multiple network connections

It is possible to install multiple network cards in a Windows XP or Vista machine. One reason to use two LAN interfaces in one PC is described in Chapter 10: the PC can do double duty as the host computer for a LAN-based test system and as a router that links a private network (for the test system) to the corporate intranet. The Interconnection Connection Sharing (ICS) feature of Windows XP and Vista, accessed through the Local Area Connection Properties window, makes this configuration possible.

Managing IP addresses

In the dual LAN card configuration, the PC acts as the network controller for the private network, enabling access to the public network and providing network address translation (NAT) and dynamic host configuration protocol (DHCP) services to connected devices. NAT is the key capability that enables the private, protected LAN by shielding private IP addresses from the public network.

DHCP automatically assigns an IP address to any device connected to the network and also ensures that no two devices receive the same IP address. This is the default setting in Windows XP and most LAN routers. It's very simple and it works well 99 percent of the time—as long as the default DHCP server on the network is up and running.

Of course, the "dynamic" part of DHCP means that a device may receive a different IP address if it is disconnected and later reconnected to the network. This can cause problems in a LAN-based test system. For example, if the system contains two power supplies and DHCP reverses their IP addresses, the device under test (DUT) could receive the wrong voltages at the wrong points and suffer severe damage. There are two alternatives. One is to assign permanent (static) IP addresses to every device on the network. This lets you fine tune the network and its settings; it also isolates the network from any failure of the corporate DHCP server. On the downside, this approach can become overwhelming in a large network, increasing the chances of configuration errors or bad settings that could cause problems across the network.

The other alternative is to use a dynamic domain name server (dynamic DNS or DDNS) on the local network. DDNS lets an instrument, PC or other network device establish a specific host name when it connects to the network. (The host name can typically be entered via the front panel of a LAN-enabled instrument.) Large corporate intranets usually have such a server, which allows other devices to use the host name with DNS to find the device's IP address and connect to it. If an instrument's IP address changes, DDNS ensures a quick update of the DNS table of addresses. This approach has one major caveat: DDNS will not typically be available on the small, secure networks we recommend for test-system applications.

Running a network without DHCP

If a device running TCP/IP is set to obtain its IP address automatically but there is no DHCP server available, the device will use a capability called Automatic Private IP Addressing (APIPA or "auto IP") to assign itself an address about two minutes after boot-up. This feature is built into Windows PCs and most Agilent instruments. Auto IP creates addresses that are designed to be compatible with each other, enabling the establishment of a network without DHCP or the configuration of static IP addresses. Also, most of Agilent's LAN-enabled instruments use the NetBIOS (Network Basic Input/ Output System) protocol. In a network configuration that lacks a DHCP server, the PC can connect to the instrument by using the host name that is configured from the instrument front panel. This makes it easy to create a direct connection from a PC to a single instrument, and all it takes is a LAN crossover cable.

Configuring LAN with Agilent IO Libraries Suite

The preceding section describes a tedious process that could take hours (perhaps days) to complete for a large test system. It all became much easier with the Agilent IO Libraries Suite.2 This is an enhanced version of the Agilent IO libraries that simplifies and accelerates the process of connecting test equipment to a PC. One of its greatest contributions is a set of automated tools that detect connected instruments, configure the interfaces and verify the connections—even in test systems that mix multiple interfaces and instruments from multiple vendors.

Using the Connection Expert

The Agilent Connection Expert is one of the most powerful tools in the IO Libraries. Not only does it automatically discover connected instruments and configures the PC's interfaces for communication, but it also manages GPIB, RS-232, VXI, USB and LAN interfaces simultaneously.

The Connection Expert includes an on-screen task guide that helps both occasional and expert users perform connection tasks. In most cases, you should be able to establish error-free connections in less than 15 minutes.

The Connection Expert makes programming easier, too. It can help you find information relevant to your instruments in popular development environments such as C, C++, Visual Basic, Visual Basic .NET, Agilent VEE Pro and NI LabVIEW. The Connection Expert can also point you to numerous example programs written in various languages. Another nice touch: it lets you create an alias name for each instrument so you don't have to change the source code if you change an instrument's IP address. You can even switch from GPIB to LAN connectivity and use an alias that looks like the old GPIB address. Using this feature, you can often make older programs work via LAN without reconfiguring or recompiling the code.

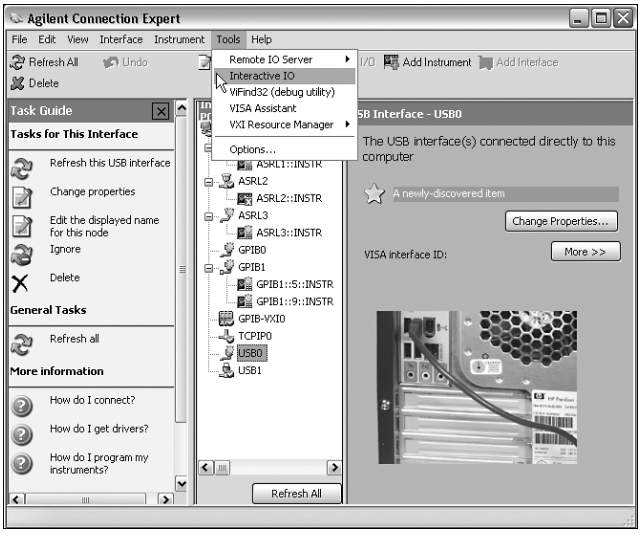
Debugging with I/O utilities

The Agilent IO Libraries Suite also includes a set of utilities that will help you perform various debugging tasks from the PC (Figure 11.2):

- **Interactive I/O.** Allows you to query instruments one command at a time, sending commands and reading the responses.
- **Remote I/O Server.** Lets you connect to instruments that are attached to a different PC that resides on the same network.
- **VXI Resource Manager.** Helps you configure the Agilent E8491 IEEE-1394 PC link to the VXI interface.
- **ViFind32 Debug Utility.** Uses VISA functions to find resources and lists them in a console window.

These utilities are just one more way Agilent can help you streamline your test-system development activities.

Figure 11.2. The Agilent Connection Expert provides quick, easy access to connection utilities.



2 We use "IO" rather than "I/O" in the product name because most operating systems don't allow the slash character in file names.

Enabling communication with instruments

With connectivity completed and verified, the next step is to ensure and enable communication between the PC and the instruments. As one example, the VISA I/O API can use two different methods to communicate with LAN devices: the VXI-11 communication protocol and raw TCP/IP socket communication. VXI-11 is the preferred choice when moving existing (GPIB-based) code or when you want to keep a consistent programming style with GPIB instruments. Sockets should be used when the host PC does not support VXI-11. Sockets also provide the highest level of performance, but at the cost of some programming complexity.

From the PC's point of view, the VXI-11 protocol creates an I/O connection that looks and behaves like GPIB. In practice, this means software written for GPIB is likely to work on VXI-11 instruments. Almost all of Agilent's LAN-enabled instruments support the VXI-11 protocol.

Conclusion

With LAN ports in most current generation PCs and many new-generation test instruments, connecting the two is almost as simple as plugging in a network cable—if you apply the capabilities and tools that are part of the Agilent IO Libraries Suite.

To discover more ways to simplify system integration, accelerate system development and apply the advantages of open connectivity, please visit **www.agilent.com/find/open.**

12. Using USB in the Test and Measurement Environment

Introduction

This chapter offers a closer look at the universal serial bus (USB) as a test-system connectivity option. For a review of the comparative advantages of LAN, GPIB and USB, please refer to Chapter 2, Computer I/O Considerations.

USB in the PC universe

Chances are you're already familiar with USB, thanks to its wide use in PC printers, scanners, cameras and other digital devices. However, some background on USB's place in the PC universe might help you decide how and when to use USB for test and measurement.

The USB story

On the timescale of computing technology, USB has been around for quite some time: the original version of USB was introduced concurrently with Microsoft Windows 95. USB's original goals included replacing the multiple types of interfaces then in use in PCs and eliminating the complex configuration steps they sometimes required. Computers with USB 1.0 first appeared in 1996, and Windows has supported USB ever since.

The USB standard has gone through two major revisions since version 1.0. USB 1.1, introduced alongside Windows 98, took advantage of the new Plug and Play connectivity in the operating system. All you needed to do in most cases is attach the connector and you're ready to go. (Nearly all PCs today, both desktop and laptop, come with built-in USB ports. You can also add USB cards to older PCs.)

This simplicity spurred rapid growth in the number of PCs and peripheral devices that offer built-in USB. However, as digital devices began to demand more bandwidth, the 12 Mbits/second top speed of USB 1.1 became a growing concern in some applications. USB 2.0, introduced in 2001, dramatically expanded bandwidth with speeds up to 480 Mbits/second. USB 2.0 is backwards compatible with USB 1.1, although this can lead to some confusion about data rates, as you'll see below.

USB connections

With its intended use in consumer applications, USB is not only inexpensive but also easy to use. Connections are hot pluggable (sometimes called hot swappable), so there's no need to power down before you add or change connections, and the PC auto discovers new devices as soon as you plug them in. And unlike GPIB, where you must assign a unique address identifier to every device in the system (and keep track of which device is where if you reconfigure the system), every USB device has an embedded serial number that the PC reads as soon as you connect it.

From a mechanical standpoint, USB 1.1 and 2.0 are identical; both use the same four-wire scheme (two power wires and a twisted pair for data), and any fully compliant USB cable will work in any USB system, regardless of speed.

The Benefits of USB

- Near-universal availability in today's PCs
- Hot-plugging with auto discovery for true plug-and-play
- Low cost
- Simple connection with no configuration required
- Flexible speed levels to accommodate a variety of devices
- Simultaneous connection of up to 128 devices

The theoretical maximum number of devices in a single USB system is 128 (the PC plus 127 other devices). However, you can't daisy-chain devices together as you can with GPIB. Rather, you can expand by using a hub; typical hubs provide ports for four to eight devices. To add more devices, you can daisy-chain additional hubs. Hubs can be either self-powered or bus-powered; devices that require a significant amount of power often require a self-powered hub to ensure adequate power levels.

USB speeds

The USB 2.0 Specification encompasses all USB data transfer speeds: Hi-Speed (480 Mb/s), full-speed (12 Mb/s) and low-speed (a 1.5 Mb/s alternative designed for keyboards, mice, and other low data-rate devices). Just because a device is USB 2.0 compatible doesn't automatically mean it can operate at 480 Mb/s. The best way to verify the speed of USB devices is to look for the official USB logo. Devices that are certified to run at one of the two original USB rates, 1.5 Mb/s or 12 Mb/s, should carry a white and blue Certified USB logo (Figure 12.1). Devices certified to run at 480 Mb/s rates carry the red, white, and blue Certified Hi-Speed USB logo.[1]

Figure 12.1 USB logos identify the device's speed rating

The speed rating of hubs in a USB system determines the operating speed of the system. For instance, if you connect Hi-Speed USB devices through a full-speed hub, the maximum speed you can expect from any of the devices is 12 Mb/s, not 480 Mb/s. To take advantage of Hi-Speed data rates, you must connect these devices through a Hi-Speed hub.

1 USB Implementers Forum Web site, **www.usb.org.**

Agilent support for USB instrument connectivity

To provide users with the utmost in convenience, Agilent has committed to providing USB connectivity as a standard feature in nearly every new test and measurement instrument. In most cases, new instruments support 480 Mb/s Hi-Speed USB, which delivers greater bandwidth and lower latency (the time required to respond to programming commands) than GPIB. Those few instruments that support full-speed USB (12 Mb/s) offer bandwidth similar to GPIB with somewhat higher latency.

You can also take advantage of USB with your existing GPIB instruments. The Agilent 82357B USB/GPIB Interface (Figure 12.2) connects GPIB instruments to a USB port on your computer, giving you a way to control up to 14 GPIB instruments from a laptop or other PC for each 82357B.

Figure 12.2. Agilent 82357B USB/GPIB Interface

The 82357B is a Hi-Speed, fully compliant, hot pluggable USB device, so you can connect it whenever you need it, without rebooting your PC. Instruments connected via the 82357B have GPIB-style VISA and SICL addresses, just like Agilent's PCI- or older ISA-based GPIB cards, so legacy programs in systems that use these cards require no reconfiguration or code changes.

You're not limited to locally available instruments, either. With the Agilent E5813A networked 5-port USB hub, you can access remote USB devices or instruments through your standard LAN. With the E5813A connected to your PC and properly configured, those remote instruments and devices will function as though they were locally attached.

Agilent also provides a USB solution for RS-232 instruments. The Agilent E5805A USB/4-port RS232 interface provides a direct connection from the USB port on your notebook or desktop PC to up to four RS232 instruments or devices.

To simplify programming of instruments over USB connections, Agilent and other test equipment vendors co-developed the industry standard USB Test and Measurement Class (USBTMC) and USB488 I/O protocols. These protocols, along with Agilent's IO Libraries Suite, allow you to easily switch from GPIB to USB connections without making big investments in new PC software or rewriting existing programs. Aside from address conventions, your USB instruments will look and act just as they would under GPIB control.

Setting up USB instruments with the Agilent IO Libraries

The Agilent IO Libraries (which is now included with most Agilent instruments, T&M software products such as Agilent VEE Pro and connectivity products such as the 82357B, E5805A and E5813A) make USB measurement setups even simpler by automating the connection and configuration process for you. The IO Libraries include three separate direct I/O Application Programmer Interface (API) libraries so you can choose the library that works best with your development environment:

- VISA (Virtual Instrument Software Architecture, an industry standard application programming interface)
- VISA COM (a version of VISA that conforms to Microsoft's Common Object Model and IVI Foundation standards)
- Agilent SICL (included primarily to support existing test systems; VISA or VISA COM is the recommended direct I/O API for new system development)

Connecting instruments via USB

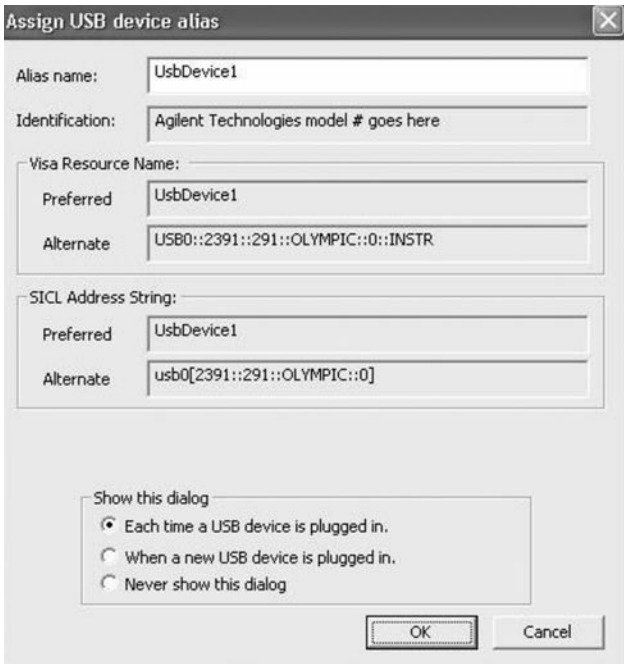
The IO Libraries include the drivers for USBTMC/USB488 devices as well as the 82357B USB/GPIB Interface, so you're ready to go as soon as you install the software. As you start plugging in instruments (or the 82357B interface), you'll be presented with a dialog box that lets you name each device with a human-readable USB alias (Figure 12.3). The alias capability is a helpful way to manage device names, since the standard VISA resource naming convention for USB devices can be rather cumbersome (USB0::2391::1031::MY43000786::0::INSTR, for example).

The alias capability also enables the same test system software to work on multiple automated test systems, provided the same alias names are used, such as the alias "DMM" for a voltmeter. And if you have an existing program that communicates with an instrument over a GPIB or other non-USB interface, you can create a VISA alias that looks like a GPIB address, such as "GPIB1::23::INSTR" and the program will function as though it were still communicating over a GPIB interface.

Additional software included with the Agilent E5805 and E5813A works with the Agilent IO Libraries to provide the same flexibility for RS-232 instruments and remote USB instruments. These instruments and devices appear to be local to the PC and can be aliased as well.

To verify the connection with each instrument, simply launch the Agilent Connection Expert, the configuration utility in the Agilent IO Libraries. Refresh the list of instruments if your instrument is not already displayed, then choose "Verify This Instrument." You can also launch an interactive I/O session with the instrument and send the *IDN? command, the standard instrument identification query in the Standard Commands for Programmable Instruments (SCPI) command set. The instrument will respond by identifying its manufacturer, model number, serial number, and firmware revision.

Figure 12.3. Example of the connection dialog in the Agilent IO Libraries



Communicating with USB-connected devices

You don't need to worry about the details of a USB connection, so most programs written to talk to GPIB devices work with USB-connected devices without modification. However, if your program uses low-level commands that affect the entire GPIB bus (such as through a VISA session such as GPIB::INTFC), you may encounter some unexpected results. USB devices are optimized for modern instrument communication, which discourages lower-level, error-prone interface manipulation operations. Consult the documentation for the instrument or I/O adapter for any limitations.

As mentioned earlier, an instrument connected via a USB cable acts like an instrument connected over a GPIB bus, aside from a different I/O address. Here's some example C code that communicates with a USB-connected instrument, either natively or with the E5813A networked 5-port USB hub:

Similarly, the 82357B USB/GPIB interface looks and acts like a PCI/GPIB adapter for typical instrument communication, so instruments connected to it have GPIB-style address names and act like any other GPIB-connected instruments. The source code is exactly the same as above, with the exception that the instrument address would be something like "GPIB0::23::INSTR" instead of "FuncGen."

```
#include <iostream>
#include <tchar.h>
#include <stdio.h>
#include "visa.h"

#pragma comment(lib, "visa32.lib") /* include the visa32.lib
import library */

/* Error-checking routine */
void CHECKERROR(ViSession vi, ViStatus status)
{
      char desc[256];
      ViStatus err = 0;
      if (status < 0)
      {
            err = viStatusDesc(vi, status, desc);
            fprintf(stderr, desc);
            viClose(vi);
            _exit(status);
      }
}

int _tmain(int argc, _TCHAR* argv[])
{
      char idnResult[256];
      ViSession rm = 0, funcGen = 0;
      ViStatus err = 0;
      viOpenDefaultRM(&rm);
      err = viOpen(rm, "FuncGen", VI_NO_LOCK, 0, &funcGen);
      CHECKERROR(rm, err);
      err = viPrintf(funcGen, "*IDN?\n");
      CHECKERROR(funcGen, err);
      err = viScanf(funcGen, "%t", idnResult);
      CHECKERROR(funcGen, err);
      printf("The *IDN? string is %s", idnResult);
      viClose(funcGen);
      viClose(rm);
      return 0;
}
```

In an RS-232 scenario, the E5805A USB/4-port RS-232 interface will look like a standard RS-232 port on your PC, and instruments connected to it will have RS-232-style address names and act like other RS-232-connected instruments:

```
/* Same header and error-handling code as above... */

/* Do a simple *IDN? Instrument Identification Query */
int _tmain(int argc, _TCHAR* argv[])
{
      char idnResult[256];
      ViSession rm = 0, dmm = 0;
      ViStatus err = 0;
      viOpenDefaultRM(&rm);
      err = viOpen(rm, "ASRL1::INSTR", VI_NO_LOCK, 0, &dmm);
      CHECKERROR(rm, err);
      /* don't bother checking errors for these, nothing will happen until communication is attempted
*/
      err = viSetAttribute(dmm, VI_ATTR_ASRL_PARITY, VI_ASRL_PAR_NONE);
      err = viSetAttribute(dmm, VI_ATTR_ASRL_BAUD, 9600);
      err = viSetAttribute(dmm, VI_ATTR_ASRL_DATA_BITS, 8);
      err = viSetAttribute(dmm, VI_ATTR_ASRL_STOP_BITS, VI_ASRL_STOP_ONE);
      err = viSetAttribute(dmm, VI_ATTR_ASRL_FLOW_CNTRL, VI_ASRL_FLOW_DTR_DSR);
      /* clear out any old data and prepare the instrument */
      err = viFlush(dmm, VI_IO_IN_BUF_DISCARD | VI_IO_OUT_BUF_DISCARD);
      CHECKERROR(dmm, err);
      err = viPrintf(dmm, "*CLS\n");
      CHECKERROR(dmm, err);
      /* do the identification query */
      err = viPrintf(dmm, "*IDN?\n");
      CHECKERROR(dmm, err);
      err = viScanf(dmm, "%T", idnResult);
      CHECKERROR(dmm, err);
      printf("The *IDN? string is %s", idnResult);

      viClose(rm);
      return 0;
}
```

Conclusion

The low cost and simplicity of USB—and the dramatic speed improvements of Hi-Speed USB make—make USB an ideal connectivity option for simple, ad hoc test systems. Achieving expected data rates does require some attention to the hubs used in a USB system; the system will operate only as fast as the fastest hub, so make you're your hubs are also Hi-Speed rated. Agilent offers comprehensive support for USB-based test systems, including built-in USB ports in many instruments and support for USB in the Agilent IO Libraries.

13. Using SCPI and Direct I/O vs. Drivers

Introduction

This chapter outlines the relationship between input/output (I/O) software, application software and the ability to maximize instrument interchange and software reuse in present and future systems. It builds on the information presented in Chapters 9 through 12, which provide essential background on the use of LAN in test systems.

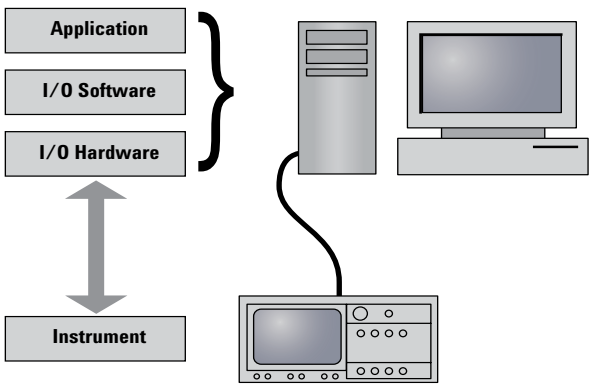
Deciding how to communicate

Once you've chosen an I/O interface for your system—GPIB, LAN, USB or a combination—the next step is deciding how to enable connectivity and achieve communication between the host computer and the instruments in the system. Recently, the alternatives for connectivity and communication have been shifting; vendor-specific commands, libraries and interfaces are giving way to industry-standard command sets, application programming interfaces (APIs) and instrument drivers.

In system development, the use of standards offers two key benefits: it accelerates development by maximizing software reuse and it enhances system flexibility by making it easier to use different instruments. Standards also help you achieve your goals for system functionality and performance by letting you combine methods such as direct I/O with Standard Commands for Programmable Instruments, or SCPI (see Chapter 3), and instrument driver-based communication within a single application.

The best choice of I/O software depends on factors such as the number and type of instruments in the system, the functionality to be used within each instrument, the system's throughput requirements and the number of systems to be deployed. It also depends on which application development environment you're using and the current level of your programming skills.

Figure 13.1. Three essential elements of instrument commu nication reside within the system's host PC



Sketching the big picture

The diagram in Figure 13.1 is our starting point. It connects the conceptual side of the discussion—layers of software and hardware—with the actual test system, which includes the computer, I/O cable and test equipment. Within this model, commands and information flow from the application through the software and hardware layers, down the cable, to the instrumentation and back again.

Focusing on the upper-left of the diagram, the application is the program—purchased, downloaded or written by you and running on a programming language—that controls the test system. The I/O software layer is the translator that enables communication between the application and the physical I/O hardware—the GPIB, LAN, USB, VXI, PXI or RS-232 interface in the PC. These three elements reside within the host PC and enable connectivity with the test equipment.

That's all necessary to enable connectivity, but it isn't sufficient to achieve communication. It's similar to the story of placing a phone call to a friend in another country: you pick up the handset, hear the dial tone and dial the number—and then your friend's mother answers the phone. Your inability to speak each other's language prevents meaningful conversation. You have a connection but you haven't achieved communication.

It's the same with test systems. Even if an application has connectivity with an instrument, it must use the right commands and protocols to achieve communication, control, data transfer and so on.

Enabling connectivity

In the early days of automated testing, system controllers—called desktop calculators or instrument controllers— had limited processing and sparse memory. To keep the syntax as simple as possible, equipment vendors used short commands, initially in binary and later in ASCII.

Different manufacturers defined their own command strings and these were typically unique to the specific capabilities of each instrument. In a system, replacing an instrument with one from another manufacturer, or even a new-generation product from the original maker, could mean completely rewriting the system software.[1]

Instrument commands aren't the whole story, however. Enabling connectivity between a controller and the system instruments requires additional layers of software. Historically, the I/O software layer contained libraries such as the Standard Instrument Control Library (SICL) or NI-488. The application used these libraries to achieve direct communication with an instrument. Each vendor had a proprietary application programming interface (API) that communicated exclusively with its own I/O interfaces. This made it difficult for system developers who were building mixed-vendor test systems—and, of course, many systems used (and continue to use) equipment from multiple vendors.

1 For more about the evolution of instrument control, see Chapter 3, Understanding Drivers and Direct I/O.

Standardizing the API

To make it easier to create mixed-vendor test systems, a group of instrument vendors created the Virtual Instrument Software Architecture (VISA). This provided a standardized API that allowed control of instruments through a common interface—directly or with drivers. From the application's point of view, every vendor's VISA interface looks the same.

One important caveat goes with VISA: Although the VISA API is standard, each vendor employs different layers beneath the VISA layer to control the hardware. In addition, each vendor may have made enhancements to enable unique features in its application layers. To make it all work, the version of VISA installed on the host computer must be compatible with the I/O hardware. (In contrast, this points to another advantage of PC-standard I/O such as LAN and USB: any version of VISA that supports those interfaces will work because the low-level drivers are standardized.)

Expanding freedom of choice

As I/O development was proceeding in the test and measurement industry, the PC industry was pursuing independence in both I/O and programming languages. Microsoft created the Component Object Model (COM), which is a software architecture that allows components made by different software vendors to be combined into a variety of applications. COM is not dependent on any particular programming language.

To incorporate the advantages of language independence, Agilent initiated the creation of VISA COM as a companion to the VISA standard. VISA COM is an object-oriented representation of the VISA API; it exposes the VISA API to the application layer through use of the Component Object Model.

The result: VISA COM gives you the freedom to pick from the most popular I/O configurations and also choose from a wealth of "COM friendly" languages such as C#, Visual Basic 6 and Visual Basic .NET. As we'll discuss later, the application development environment (ADE) you choose will influence the best choice of library and API for your application.

Achieving communication

Once you've enabled connectivity, it's time to decide how to achieve communication between the host computer and the system instrumentation. The two alternatives are direct I/O and instrument drivers. Direct I/O creates an explicit connection to each instrument, which makes it faster but limits instrument interchange and software reuse. Most instrument drivers utilize direct I/O and SCPI but sometimes hide that connection. Generally speaking, drivers trade decreased flexibility (and possibly speed) for improved interchange and reusability. However, in most situations you can use both instrument drivers and direct I/O to achieve the best balance of speed, flexibility and measurement functionality.

Standardizing direct I/O

An early attempt at improving consistency and ease of use came in 1989 when Hewlett-Packard2 introduced an instrument communication language called the Test & Measurement Systems Language (TMSL). HP and eight other manu-

2 HP spun off its test & measurement businesses as part of Agilent Technologies in 1999.

facturers joined forces to generate a universal approach to instrument control, using TMSL as the starting point. SCPI was the result of that collaboration.

The implementation of SCPI within an instrument's firmware has made the programming syntax for direct I/O much more robust and predictable. The syntax defines a strict hierarchy that specifies consistent commands, responses and data formats across instrument models. These commands and responses are defined for source, sense and switch devices. Today, SCPI is still the most-used form of instrument control (Figure 13.2).

Improving interchange and reuse

SCPI was a big improvement, but the subsequent development of instrument drivers has taken interchange and reuse to new levels. An instrument driver (or just "driver") is a high-level, instrument-specific (or instrument class-specific) piece of software that enables communication between a PC and an instrument. For software developers, drivers often simplify programming and shorten development time by guiding the programmer through the necessary steps and describing the capabilities of the instrument within the programming environment (rather than in a manual, as would be the case with SCPI and direct I/O).

First-generation drivers were vendor-specific and typically worked only with a specific ADE. (Numerous legacy application programs still use these proprietary drivers.) Today, however, three types of standardized instrument drivers are available. These work with multiple ADEs and enable communication with an instrument through any vendor's I/O hardware.

- **VXI*plug&play*.** Originally developed for modular VXI instruments, these were later expanded to address non-VXI instruments. Conforming drivers always perform I/O through the VISA library. The VXI*plug&play* WIN32 driver specification works in all popular languages and is today's most widely used driver architecture.
- **IVI-C.** IVI-C has two distinct drivers. The term is generally applied to drivers based on proprietary tools from National Instruments (NI). With the advent of the IVI standards, NI updated its tools to conform with the standards, but many systems based on the proprietary tools are still in use. To enable reuse and interchangeability, IVI-C requires additional software to patch around its core DLL technology, which does not directly support software interchangeability. An application must call an intermediate driver (an "IVI-C class driver") which then calls the specific instrument driver to accomplish the function.

Figure 13.2. This block of Visual Basic 6 code uses SCPI and VISA COM I/O to communicate with a function generator.

```
Dim Fgen As VisaComLib.FormattedIO488
' Code removed: Set up the connection to the instrument
With Fgen

      WriteString "*RST"                          ' Reset the function generator
      IO.Clear                                    ' Clear errors and status registers
      WriteString "FUNCtion PULSe"                ' Select pulse waveshape

      WriteString "OUTPut:LOAD 50"                ' Set the load impedance to 50 Ohms (default)
      WriteString "VOLTage:LOW 0"                 ' Low level = 0 V
      WriteString "VOLTage:HIGH 0.75"             ' High level = .75 V

      WriteString "PULSe:PERiod 1e-3"             ' 1 ms intervals
      WriteString "PULSe:WIDTh 100e-6"            ' Pulse width is 100 us
      WriteString "PULSe:TRANsition 10e-9"        ' Edge time is 10 ns (rise time = fall time)
      WriteString "OUTPut ON"                     ' Turn on the instrument output

For I = 0 To 18                                   ' Vary edge by 5 nsec steps
      WriteString "PULSe:TRANsition " & (0.00000001 + I * 0.000000005)
      Sleep 300                                   ' Wait 300 msec

Next I

End With
```

- **IVI-COM.** This standard does the most to enable interchangeability and reuse by leveraging the COM computer standard. IVI-COM drivers integrate with standard PC component architecture software and enable control of instruments from familiar, conventional ADEs that provide major productivity improvements. IVI-COM drivers that control VXI or GPIB instruments use VISA (either VISA COM or VISA-C). Because many new instruments include computer-standard I/O such as LAN and USB, IVI-COM drivers for non-GPIB instruments are not required to use VISA, although many do.

If you are unsure of which I/O technology an application or driver is using, take a look at the connection string or "instrument address" used for instrument communication. VISA-type strings look like "TCPIP: 34980A.tm.agilent.com::inst0::INSTR" while SICL-based strings are similar to "lan[34980A.tm.agilent.com]:inst0."

Exploring the application alternatives

Shrink-wrapped software often provides convenience in measurement and analysis at the expense of performance and flexibility. Such products are often a good fit with the small or one-off systems used during product development. In contrast, custom-built software is often the best answer for applications such as design verification or manufacturing test that require high performance and maximum flexibility.

Simplifying basic analysis tasks

There are alternatives to general-purpose development environments. One example is "targeted applications," which address specific measurement or technology domains, or specific phases or tasks in the product development lifecycle. These applications include software designed to make the infrequent measurements (manual or semi-automated) that are typically performed during the early phases of product development or during design verification.

Applications such as the free Agilent IntuiLink connectivity software and low-cost Agilent BenchLink make it easy to perform semi-automated measurements, collect data and analyze results from a wide variety of instruments. Both applications utilize either drivers or direct I/O—transparently—to enable instrument communication, control and data transfer.

- **IntuiLink.** This connectivity application simplifies data transfers by adding a toolbar to popular PC applications such as Microsoft Word and Excel. IntuiLink enables direct retrieval of data and images from a measurement instrument, letting you remain in the PC application and use its familiar interface. IntuiLink also eliminates barriers between instruments and PCs by supporting GPIB, USB, LAN and FireWire interfaces.

- **BenchLink.** This application is available in versions that support numerous instruments. BenchLink is a Windows-based application (Figure 13.3) that uses a familiar spreadsheet format to streamline data collection, presentation and analysis. It can communicate with measurement instruments via LAN, USB or GPIB using the included I/O software.

There are higher-cost alternatives to BenchLink, including instrument-control software for functional testing and domain-specific applications. These range from general test executives to application-specific programs such as cell phone regulatory testing tools. All serve to further reduce the burden of instrument programming, connectivity and communication.

Figure 13.3. Agilent BenchLink Data Logger provides spreadsheet-like test set up and real-time display and analysis of measurements.

Comparing development environments

The software environment you choose will have a significant impact on the time, effort and cost required to create and maintain a test system. Development environments are either graphical or textual. Graphical environments such as Agilent VEE Pro and NI LabVIEW use a schematic approach, which is regarded as being easy for engineers to learn. You manipulate icons or objects that represent commands or functions and connect them with program-flow lines. This makes it easier to visualize the paths of execution and interaction; it also shields you from the underlying syntax. What's more, T&M-specific graphical environments have extensive I/O and instrument drivers as well as measurement-related math and graphing capabilities. Graphical programming is best suited to small- and medium-sized applications—the visual interface tends to become difficult to understand with large programs.

In contrast, textual programming has a steeper learning curve because it requires detailed knowledge of a language's commands and syntax. However, because most textual languages are based on open standards, they offer a greater selection of development environments, software tools and training opportunities. There also tends to be a wider variety of available third-party drivers, tools and add-ons. Textual programming is often the best choice for large, comprehensive programs because it is easier to navigate and comprehend.

In the past, textual programming produced applications that had pronounced speed advantages—at runtime—over those created with graphical programming. Today, however, there is less difference in runtime speeds between applications created with either approach.

Maximizing performance and flexibility

You can pick from a wide variety of alternatives that support the creation of custom measurement software. These range from test automation applications to full-featured development environments that utilize either graphical or textual programming. Your preferred approach will determine the best choice for instrument communication.

Microsoft Visual Studio

Visual Studio is a textual programming solution that offers an extensive range of developer tools and built-in help capabilities that can accelerate development of Windows-based applications. Its integrated development environment provides a consistent interface for all supported languages, including Visual Basic, C++ and C#. As a standardized, mainstream development product, Visual Studio offers several advantages:

- Open. Because Visual Studio is based on open, pervasive standards, it can communicate with practically any other programming technology. As a result, thousands of third-party tools—software, drivers and so on—are available to support your development efforts.
- COM-friendly. Visual Studio works very well with programming technologies that are based on Microsoft's COM technology. This includes VISA COM and IVI-COM.
- On-screen help. The IntelliSense feature and the "F1 help" capability work with COM- and .NET-based third-party drivers and software. As an example, the IntelliSense window for a driver will show all available operations, a brief description of each, and a summary and description of all allowed parameters. Depending on the driver or component, pressing the F1 key may open a new window that presents an online help manual for the driver. Using this type of on-screen, context-sensitive help is much faster than thumbing through a printed programming manual.

There is one downside in test-system applications: it can be difficult to use C APIs with the new .NET-based languages in Visual Studio. The latest releases of Microsoft programming languages utilize .NET technology to communicate with drivers and third-party software—and .NET is rapidly phasing out C API technology. This affects the C API version of the VISA I/O library as well as IVI-C and VXI*plug&play* drivers. To get around this problem, Agilent provides a .NET wrapper for the VISA API. The wrapper is available as a free download from **www.agilent.com/find/iolib**; it is also included in the Agilent IO Libraries.

Visual Studio with Agilent T&M Toolkit

The Agilent T&M Toolkit 2.0 with test automation extends the .NET-enabled versions of Visual Studio with a suite of integrated, easy-to-use software tools and components—project wizards, APIs, class libraries, widgets, graphs, drivers and more. This creates an environment that simplifies the process of incorporating tests and measurements into custom applications. Using T&M Toolkit 2.0 within the Visual Studio environment lets you use your preferred textual programming language and integrate your new code with existing code written in other languages.

T&M Toolkit 2.0 offers several other capabilities that speed and simplify system development:

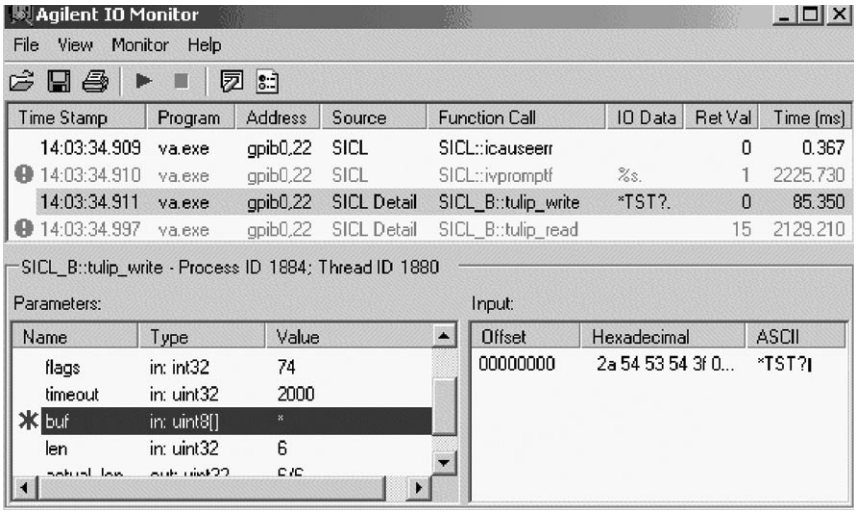
- **DirectIO class.** This is the easiest way to send commands directly to an instrument.
- **Wrapped VXI*plug&play* drivers.** This integrates the drivers into .NET with full IntelliSense and F1 help capabilities. T&M Toolkit also recognizes and uses IVI-COM drivers, which have IntelliSense built-in.
- **Instrument Explorer.** This tool makes it easy to see and edit the instrument I/O configuration and initiate communication with instruments.
- **IO Monitor.** This utility makes it much easier to use instrument-control software and instrument drivers—IVI-COM and VXI*plug&play*—and diagnose problems by letting you watch both the underlying direct I/O commands that are sent to the instrument and the resulting data that is returned (Figure 13.4).

In all, the combination of Visual Studio and T&M Toolkit eliminates many of the difficulties often associated with connecting to and controlling test equipment from a custom application.

Agilent VEE Pro

For those who want an alternative to textual programming, Agilent VEE Pro is a powerful, easy-to-use graphical programming environment that accelerates the process of building and programming test systems (Figure 13.5). To create a program, you choose high-level graphical objects from a huge library and connect them with lines or "wires." The wire connections specify functionality and sequences within intuitive block diagrams. Because VEE Pro is an open, standards-friendly environment, it also offers several advantages in test system development:

Figure 13.4. T&M Toolkit's IO Monitor traces I/O layers for Agilent's VISA, VISA COM, SICL and SICL Detail, helping you find bottlenecks in your source code.



- **Direct I/O.** Through its easy and powerful Direct I/O capability, VEE Pro provides excellent support of direct I/O for control of any standard instrument and many vendors' PC plug-in cards.
- **Instrument drivers.** VEE Pro supports industry-standard drivers such as IVI-COM and VXI*plug&play*. It includes support for nearly one thousand drivers, supporting popular instruments from more than 70 manufacturers.
- **COM and .NET.** No familiarity with .NET programming languages is required to utilize these capabilities. VEE Pro takes care of the details, ensuring successful interaction with both COM and .NET software.

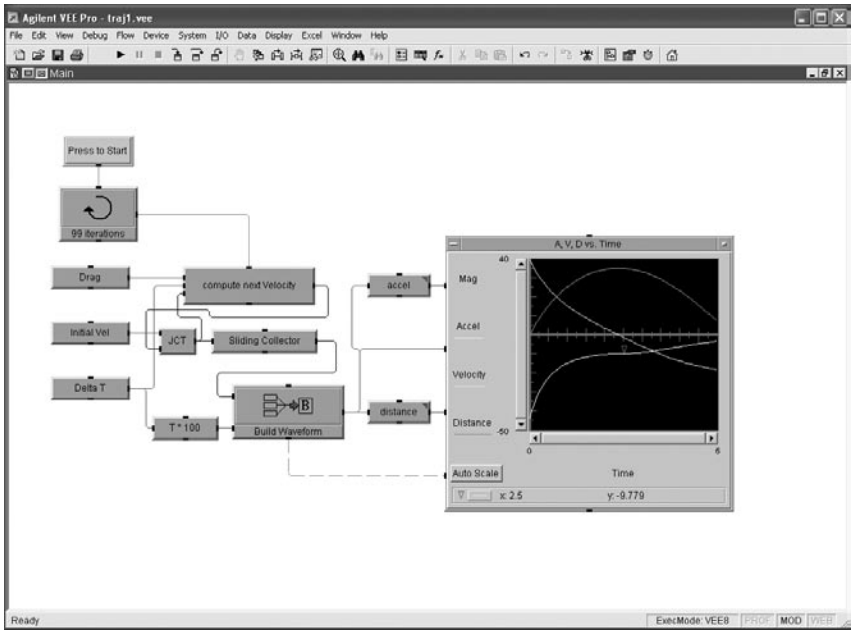
Assessing I/O software alternatives

Our ultimate goal is to minimize the amount of time you have to spend sorting out which I/O libraries or drivers to use in your test systems. Today, however, that effort is unavoidable—but we can offer a few suggestions that will simplify the process.

Instrument drivers vs. direct I/O

When comparing drivers and direct I/O, there are two key factors to consider. One is a tradeoff between speed of development and speed of execution: drivers contribute to faster development while direct I/O enables faster execution.

Figure 13.5. With its intuitive graphical programming and extensive support for both direct I/O and instrument drivers, Agilent VEE Pro simplifies and accelerates test system programming.



The other factor is access to instrument functionality. Drivers typically cover a subset of an instrument's total feature set–and this is often limited to the most commonly used functions. In contrast, the combination of direct I/O and SCPI commands can typically access 100 percent of an instrument's programmable functions, no matter how arcane. If you prefer the advantages of drivers but need to access unsupported features, it is possible to use both methods within an application.

ADE vs. I/O API

The ADE you select will affect the best choice of I/O library and API for your application. Table 13.1 shows the various I/O APIs that Agilent supports and, for each ADE, highlights the recommended library as well as the preferred and historical alternatives.

As one noteworthy example, we recommend VISA COM over the VISA API when using Visual Basic 6 because VISA COM is an object-oriented, hierarchical view of the VISA API. Using the COM version means you don't have to add the .bas file to the VB project (though the reference is needed) and VISA COM allows for the use of context-sensitive IntelliSense help.

ADE vs. instrument driver

As mentioned earlier, three types of standardized instrument drivers are available: VXI*plug&play*, IVI-C and IVI-COM. These work with multiple ADEs and enable communication with an instrument through any vendor's I/O hardware.

Reading from left to right, Table 13.2 shows a continuum that ranges from least to most standardized across three generations of drivers—proprietary, T&M standard and PC-industry standard. These represent the past, present and future of driver technology.

To accelerate test-system development, we recommend using the latest IVI-COM drivers and VXI*plug&play* WIN32 drivers for instrument control. The IVI-COM driver technology is the only one built on a PC-standard architecture. A component driver built on COM works in all popular PC languages and most T&M languages. What's more, it utilizes the most popular types of I/O and can be used in the latest .NET technologies.

Table 13.1. ADEs and recommended I/O libraries

Programming language	Recommended I/O library	Supported alternatives	
		Preferred	Historical
Visual Basic .NET, C# and other .NET languages	T&M Toolkit DirectIO[1]	VISA COM VISA	—
Visual C/C++	VISA	VISA COM	SICL
Visual Basic 6	VISA COM	VISA	SICL

1 Agilent T&M Toolkit 2.0 is a set of measurement and test tools and components for the Microsoft Visual Studio .NET development environment. The T&M Toolkit DirectIO class enables instrument connections from within Visual Studio.

Table 13.2. ADEs and their recommended instrument drivers

Proprietary T&M (specific to one language)	**Test & Measurement** (based on T&M standards)	**Component PC** (based on PC standards)
LabVIEW Plug & Play (VXI*plug&play* GWIN) VEE Panel Drivers	LabWindows/CVI Plug & Play WIN VXI*plug&play*	IVI-COM

Conclusion

Open standards such as COM and LAN have achieved widespread adoption in the computer world and are now shaping the future of test-system development. Standards accelerate system development by maximizing software reuse and enhance system flexibility by making it easier to swap out instruments—different models and even different brands. Standards also enhance system functionality and performance by letting you utilize direct I/O, SCPI and drivers within a single application.

Your choice of development environment can make it easier to incorporate tests and measurements into custom applications. If you prefer textual programming, Visual Studio with Agilent T&M Toolkit eliminates many of the problems associated with connecting to and controlling test equipment. If you prefer graphical programming, Agilent VEE Pro is an open, standards-friendly environment that supports direct I/O and instrument drivers as well as COM and .NET technologies.

14. Using LAN in Test Systems: Applications

Introduction

This chapter offers advice on balancing cost, convenience and security in three common LAN scenarios: sharing instruments, remote monitoring and data acquisition, and functional test systems. These discussions include a look at the issues of public versus private networks and wired versus wireless networks. In addition, advice on configuring a virtual private network is provided, along with a comparison of data rates over various network and protocol combinations.

Scenario 1: Sharing instruments

Sharing access to instruments or devices under test is one of the most obvious benefits of connecting test equipment over a LAN and, by extension, the Internet. However, you need to consider the security implications before exposing instruments and test data to any network and the public Internet in particular.

Sharing instruments over an unprotected LAN

The quickest and easiest way to share test equipment over a network is to simply plug a LAN-enabled instrument into the local intranet. Most intranets will auto-configure Agilent's LAN instruments so that they can be accessed from PCs by their host name with a VISA VXI-11 address (such as "TCPIP::Jeffs_34980a.sanfran.tmresearch.com::inst0::INSTR"). If the host name is unknown or the intranet doesn't support auto-naming of computers, instruments can be reached via the IP address they are assigned by the local intranet, and local LAN instruments can be automatically found using the Agilent Connection Expert utility provided with the Agilent IO Libraries Suite connectivity software package.

As noted earlier, most Agilent LAN-enabled instruments have web servers built in that allow access to and control of the instrument from a remote client via a web page. The only information required to connect is the host name or IP address of the instrument and the only software required is a web browser. Moreover, many of these instrument web pages support simple cut-and-paste operations for sending data to or retrieving data from the instrument.

Sharing instruments over a VPN

Although sharing instruments over a regular LAN is fairly simple, it's not secure—and exposing the instruments directly on the Internet is strongly discouraged. Most modern LAN instruments have some protection against the common viruses and Internet worms, however it is prudent to protect your equipment from attack.

You can make a local network secure through a variety of methods that isolate it from outside access (see Chapter 10, Using LAN in Test Systems: Network Configuration), although this of course eliminates the possibility of sharing physically separated resources.

To accomplish secure, long-distance sharing, you can deploy a LAN router that supports virtual private network (VPN) end-points with roaming clients. A VPN end-point feature means that the router can terminate one end of a secure, virtual "tunnel" between two points on the Internet or intranet. These endpoints are often used to connect geographically separated offices of an organization into one larger, virtual local area network. In addition, roaming client capability means that the VPN end-point is also optimized to allow remote PCs to create a direct connection to the router's VPN end-point, rather than just having two VPN-capable routers configured to talk to each other.

Figure 14.1 shows the physical layout of such a setup, and here are the basic configuration steps:

1. Physically connect the router and instruments as illustrated in Figure 14.1.

2. Via the web interface or other means, configure the VPN router's basic DHCP (Dynamic Host Configuration Protocol) and other network settings to create a simple, private network on which the instruments (and PC clients connected via VPN) can communicate. See Chapter 10 for an example configuration.

3. Configure the router to have a VPN end-point to give roaming users a connection point. Windows XP and Vista provide a basic VPN client that supports the L2TP/IPsec and the PPTP VPN protocols, so pick a VPN configuration using one of those protocols unless you have a different preferred VPN client. (See "Configuring a VPN" for more on choosing a protocol.)

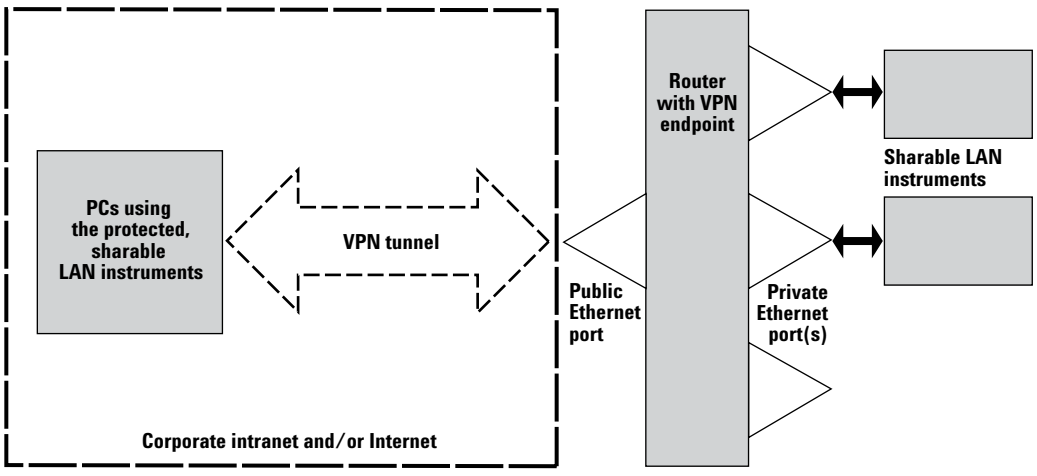
4. Configure your PC. For Windows XP and Vista, use the "Create New Connection" task in the task pane of the Network Connections utility that is accessible from the Windows Control Panel. When prompted by the wizard, use the public IP address or host name of the router.

5. After creating the connection, right-click the new VPN connection and configure the VPN type, tunnel name and password/key you configured on the router.

A VPN router configuration for instrument sharing protects the instruments on the private side of the router from public intranet/Internet access but gives PCs configured with the VPN's parameters and passwords unlimited access to those instruments. (Note that for the duration of the VPN session, the external client PC has two IP addresses—one for the VPN connection and one for the standard intranet/Internet connection.)

By default, the VPN client PC can access only the virtual network behind the VPN router when the VPN connection is active. However, it is possible to configure Windows XP/Vista so that both networks can be accessed at the same time, with Windows deciding which connection to use based on the IP address of the remote device. VPN routers typically use non-routable, private network addresses in the range 192.168.x.x or 10.0.x.x for the private networks they create. If the local intranet also uses private addresses, care must be taken to configure the subnet of the VPN router's private network so that it doesn't conflict with the intranet, otherwise the PC client won't be able to route traffic to the proper network interface (either the real network interface or the virtual network interface created by the VPN connection).

Figure 14.1. Using a VPN router to share LAN instruments securely



If you plan to expose instruments on the Internet via a VPN router, you'll need to work with your network administrators to configure the firewalls to allow such direct Internet connections. Your organization may have specific acceptable hardware lists or other policies for such configurations.

Also, when selecting a VPN router, bear in mind that capabilities and performance vary. For instance, most routers support multiple simultaneous VPN connections, depending on the model and the VPN protocol. However, performance can suffer if you initiate multiple connections through a lower-cost router, some of which have data rates less than a megabit/second for VPN connections. Higher-end models have hardware co-processors that handle the encryption necessary to make the VPN secure, which provides better VPN throughput.

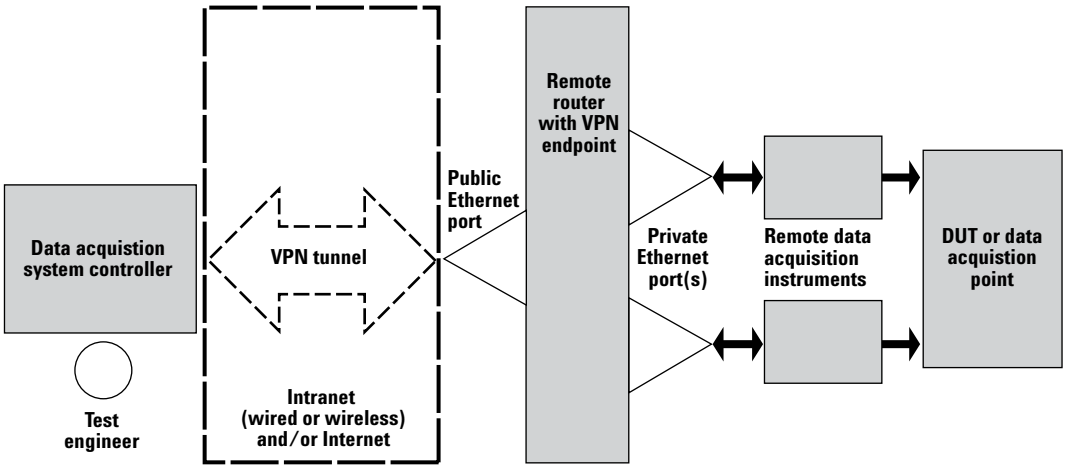
Scenario 2: Remote monitoring and data acquisition

The marriage of LAN technology and LAN-enabled instruments presents an ideal solution for many applications in data acquisition and remote monitoring. For example, the Agilent 34980A multifunction switch/measurement unit combines a built-in digital multimeter, a modular mainframe that can be reconfigured for an endless variety of switching or data acquisition needs and a LAN port for complete remote control of the instrument.

Before you deploy a remote monitoring or data acquisition solution, it's important to temporarily co-locate the controller PC, the remote instruments, system wiring, sensors and any devices under test in order to complete the initial configuration tasks. Once these major configuration steps are complete, you can usually make most minor changes via the software in the controller PC.

By keeping the test system controller nearer to the engineer responsible for maintaining the data acquisition system (see Figure 14.2), you can dramatically shorten the turnaround time for follow-on configuration changes. Although this arrangement results in all the acquired data being transferred over the network, TCP/IP and Ethernet are optimized for such data transfers and there is little or no performance penalty for keeping the data acquisition system controller remote from the data acquisition instruments.

Figure 14.2. Recommended network design for remote monitoring and data acquisition applications



Remote monitoring and acquisition over wired connections

If the point of measurement is physically located near or in the local corporate intranet, a wired LAN connection is the best choice. In most situations, using a VPN router to provide security is desirable to prevent unauthorized access to the measurement equipment and to prevent infection by viruses. See Figure 14.1 for the recommended VPN configuration for such a data acquisition system.

Remote monitoring and acquisition over wireless connections

For a data acquisition point that is not co-located with your LAN, you might be able to use a relatively low-cost wireless LAN (WLAN) solution. If the data acquisition point is within 10 miles of the nearest line-of-sight point of the corporate intranet, a commonly available wireless solution may be possible. Off-the-shelf, high gain antennae are available for the common 802.11b/g wireless Ethernet standards that can, when properly paired and aligned, create a long distance wireless Ethernet bridge between two points, connecting two Ethernet networks as though they were local (see Figure 14.3).

It's important to note that long distance WLAN installations require specialized knowledge and equipment, and they are highly dependent on terrain and other environmental factors. Due to FCC restrictions on unlicensed equipment in the 2.4 GHz radio band in the United States, the signal cannot be amplified to achieve greater transmission range, but effective range can be increased by using a pair of highly directional (high-gain) antennae.

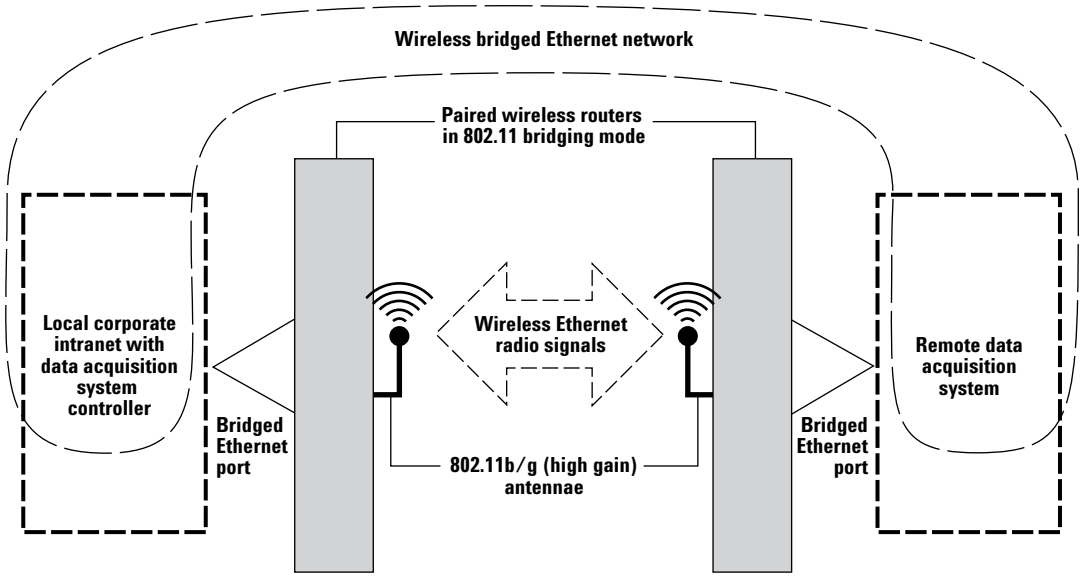
Because radio signals in the low gigahertz range can be impeded by water, such signals are prone to degradation by changes in atmospheric conditions and terrain. Consequently, it is not safe to assume an "always-on" connection for a long-distance installation. This might require keeping the system controller computer local to the instruments and point of measurement so that information and control is not lost for periods of time, rather than keeping the controller local to the test engineer for easier maintenance.

Choosing a wireless technology

Unfortunately, choosing a WLAN technology is not a simple matter, since there is a confusing variety of WLAN standards, both implemented and under development. Although all of these stem from the IEEE 802.11 base standard, you can see from Table 14.1 that the various single-letter suffixes represent a variety of technologies and protocols within the 802.11 framework.

As you plan a wireless implementation, keep in mind that the data rates you'll achieve in a real-world system are likely to be, at best, half of the rate of the physical layer speed (5 Mbps for 802.11b, for example). In addition, signal loss can limit the speed negotiated between two WLAN radios, and error correction can further reduce effective bandwidth.

Figure 14.3. Establishing a remote data acquisition connection via wireless LAN



Addressing wireless security

A quick perusal of Table 14.1 should convince you that security is an important–and complex–issue in wireless networking. *Authentication* (are all talking parties who they say they are), *encryption* (can any unauthorized listener understand the communication), and *data integrity* (can any unauthorized party interject or change data in a communication session) are all concerns when anyone can listen on or talk through a public medium such as the airwaves.

The first attempt at a security mechanism for WLAN was wireless equivalent privacy (WEP). At its simplest, it merely describes an encryption and data integrity solution through a private, pre-shared encryption key of 64 or 128 bits (actually 40 or 104 bits when the initialization vector is factored out). Add-ons such as 802.1x permit scalable, enterprise-level authentication. Unfortunately, a flaw in the WEP design allows it to be reliably broken if enough data encrypted with the same encryption key is intercepted. This means that any wireless channel encrypted with WEP could eventually be compromised if enough data passes across the channel.

Microsoft recommends[1] deploying either WEP plus 802.1x or WPA (or, assumedly, 802.11i when available) for secure, scalable solutions. Of the two, only WPA and its successors can be used securely in non-enterprise-level installations because 802.1x relies on a RADIUS server, such as Microsoft Windows Active Directory (the technology that centrally manages Windows passwords and identities for many corporate intranets).

1 Joseph Davies, *Deploying Secure 802.11 Wireless Networks with Microsoft Windows,* Microsoft Press © 2004

Table 14.1. Wireless networking standards

Standard	Type	Description
802.11b	11 Mbps Ethernet in the 2.4 GHz radio band	The most common WLAN standard; being replaced by the faster 802.11g.
802.11g	54 Mbps Ethernet in the 2.4 GHz radio band	The fastest-growing WLAN standard; still operates in the crowded 2.4 GHz radio band.
802.11a	54 Mbps Ethernet in the 5 GHz radio band	A standard that was approved before 802.11 b/g but has taken longer to roll out. Its primary benefit is operation in the less crowded 5 GHz radio band, but it is not capable of the range of 802.11b for the same power level and is more readily absorbed.
802.11 WEP	Weak wireless security protocol	Wired Equivalent Privacy encryption/authentication standard for 802.11 security; has been found to be inherently insecure. If an unauthorized listener captures enough encrypted data, the encryption key can be broken and the security compromised. For example, only a few gigabytes are required to break 128-bit (actually 104-bit) WEP keys. Automated tools are available for compromising WEP encryption.
WPA	Strong wireless security protocol	A stronger (as-yet unbroken) encryption and authentication standard for 802.11; an interim specification until 802.11i is approved.
802.1x	Wired or wireless port-based authentication	Applies to all Ethernet configurations but is particularly useful for 802.11a/b/g networks. Forces users to authenticate themselves before being given access to the network, with centralized authentication such as Microsoft Windows Domain servers allowing for an enterprise-wide solution. In wireless access points that support it, 802.1x can be used with WEP encryption to auto-generate WEP encryption keys so that there is a limited period of time that each WEP key is used, preventing listeners from discovering the WEP encryption key and thereby compromising security.
802.11i	Strong wireless security protocol	Sometimes called WPA2, the 802.11 wireless security protocol that will eventually replace WEP; believed to be secure and unbreakable.
802.11n	Up to 540 Mbps Ethernet in the 2.4 GHz radio band	An emerging standard that will have increased throughput from 802.11g and have greater range than comparable standards.

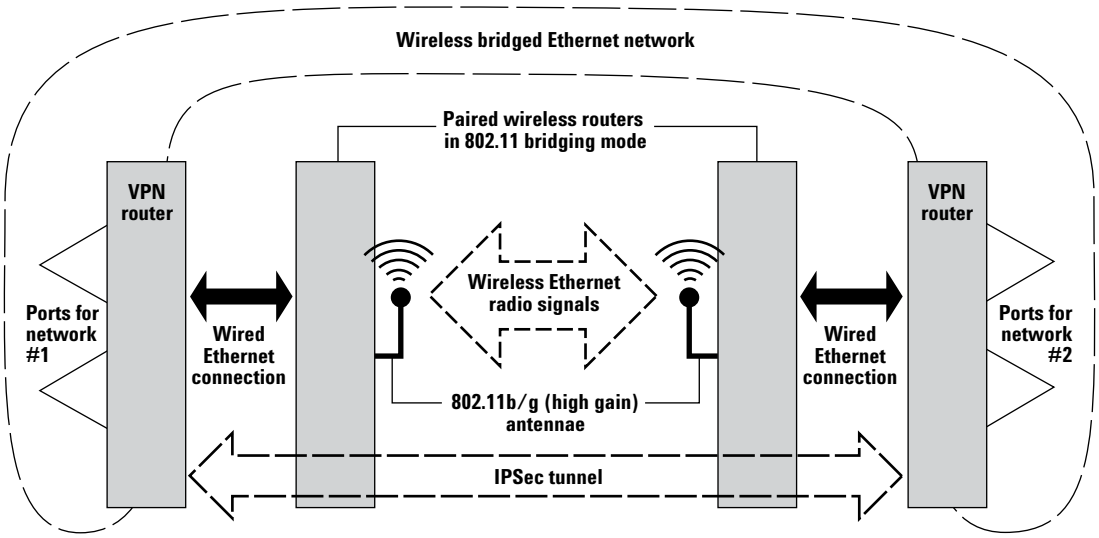
However, WPA may not be available in all possible configuration modes for wireless access points. For instance, in tests at Agilent, we were unable to use WPA with a D-Link DWL-2100AP access point in wireless bridging mode, where two access points seamlessly bridge the two networks they connect into one, larger network. Only WEP security was available in this mode, and we have found no commercial products that claim to implement WPA for bridging mode. If this remains true for 802.11i, we recommend either not using bridging mode for bridging from your intranet or putting VPN routers on either side of the wireless access points to use secure IPSec communications (see "Peer-to-peer IPSec tunneling/bridging") to guarantee wireless security will not be broken (see Figure 14.4 for a sketch of such a configuration).

Scenario 3: Functional test systems

Functional test systems represent a third potential for LAN connectivity. In addition to the benefits discussed in earlier chapters, several points need to be considered when applying LAN technologies to functional test systems: security and independence from network infrastructure, timing and deployment.

- Chapter 10 discusses security and independence from network infrastructure through the use of static network configurations and inexpensive never systems as required, since the GPIB primary address is relative to the computer and the GPIB adapter (as opposed to being a globally unique number). The Agilent IO Libraries Suite brings back some of this simplicity by letting you assign a friendly name to each instrument or resource in the form of an alias. For instance, you can assign your function generator any name you like even though it resides at a specific IP address.

Figure 14.4. Wirelessly bridged Ethernet network with VPN tunnel for additional security



Configuring a VPN

Although an endless variety of VPN implementations are available, only a few criteria need to be considered when choosing a configuration for distributed test and measurement applications: (1) Does the application require an always-on connection, or will a temporary connection suffice? (2) What level of security is required? Does the system need to be protected only from inadvertent access attempts, or is powerful encryption and security necessary to prevent deliberate, malicious attacks? (3) What are the available options in the desired price and performance range that meet your other criteria?

This section explores two configurations that can address virtually any combination of these criteria: client/server tunneling and peer-to-peer IPSec tunneling.

Client/server tunneling

Client/server VPN tunnels have become a popular means to allow remote access to enterprise networks. A key advantage of this method is that it can be used over the Internet and through routers and firewalls. VPN tunnels were designed to minimize the impact of firewalls, and updated firewalls can be configured to allow VPN tunnels through, making them the best choice for exposing instruments securely over the Internet. Another advantage of client/server VPN tunnels is multiple, noncontinuous connections from clients. Server hardware or software designed to support such tunnels allows multiple clients to connect and disconnect from the VPN server at random times for random durations, making these tunnels the best choice for ad hoc sharing (such as is common in R&D labs, for instance).

Approaches

Two common approaches to implementing these tunnels are known as L2TP/IPSec (IP Security with Layer-2 Tunneling Protocol) and PPTP/MPPE with MS-CHAPv2 (Microsoft Point-to-Point Encryption over Point-to-Point Tunneling Protocol with Microsoft Challenge-Handshake Authentication Protocol version 2). Each of these technologies is a combination of a transport layer and a security layer. The transport layer packages up network communication so that it can be successfully transmitted over the secured, virtual tunnel between the client and server. The security layer provides protection from deliberate or inadvertent deception or attacks.

The transport layers (L2TP and PPTP) provide similar capabilities, although the newer L2TP is becoming the more common choice. However, the security layers present distinct differences. Of the two, IPSec provides the best support for encryption, authentication, and data integrity; MPPE with MS-CHAPv2 is considered less secure. In general, MPPE with MS-CHAPv2 is good enough for home use and for use over secured intranets, but IPSec is the only truly secure choice for use on or over the Internet.

Implementation notes

Windows XP, Vista, and 2000 offer built-in IPSec/L2TP and MPPE/PPTP with MS-CHAPv2 clients as part of their dial-up networking support, and all VPN routers have some combination of IPSec, L2TP, PPTP, and MS-CHAP protocol support. The speed of the connection can vary greatly based on the router's implementation, especially if the router has an encryption co-processor built in to offload the computational burden of encrypting the tunneling data. As you would expect, cost typically increases with performance and capabilities.

By default, such VPN configurations turn off any other Internet connection when the VPN connection is active by configuring the default internet gateway to go through the VPN connection. If you don't want this to happen, configure the VPN tunnel to manually create the necessary TCP/IP routing information so that only information addressed for the private network across the VPN tunnel is sent across that tunnel, and all other network connections are sent through the primary network connection.

To provide a working example of client/server VPN tunnel configuration, we've posted detailed instructions and open-source, contributed utilities for establishing an MPPE/PPTP with MS-CHAPv2 VPN Tunnel between a Windows 2000/XP client and a D-Link™ DI-804HV VPN server at **www.agilent.com/find/adn_vpn_examples**.

Peer-to-peer IPSec tunneling/bridging

IPSec provides its own tunneling mode where two networks are virtually joined by establishing a secure tunnel between the network endpoints on a larger network, such as a corporate intranet or the Internet. The networks that the tunnel endpoints connect can be anything from a single computer to a large corporate LAN. This tunneling mode is designed for permanent network configurations between two points with known IP addresses or host names (making it peer-to-peer, rather than a client/server architecture). An example application of IPSec tunneling would be to virtually connect two campuses of an organization via an IPSec tunnel over the Internet so that the two ends of the tunnel are combined into one large, secure virtual LAN.

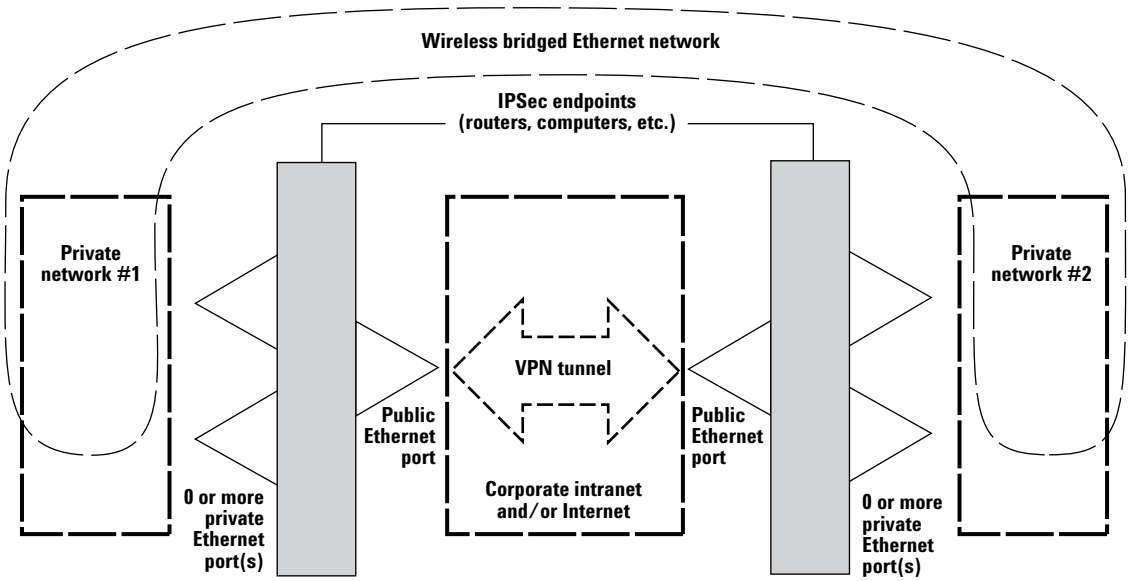
There are a few situations where IPSec tunneling is the preferred choice. Because of the always-on configuration of this tunnel, IPSec tunneling is a good choice for virtually connecting a set of measurement hardware and a controller/monitor, such as in a test system with a remote controller or a remote data acquisition application (see Figure 14.5). Because the endpoints of the tunnel can be networks of devices, IPSec tunneling is a good choice for connecting two separate test systems or permanently connecting a test system controller.

IPSec tunnels may not work across corporate firewalls, so the tunnel end-point hardware may have to be exposed to the Internet to allow such tunnels to be connected over the Internet.

As part of its IPSec feature set, Windows provides the capability for IPSec tunneling, although their configuration tools are too complex to use without instructions or experienced help. However, the Windows implementation is very flexible and powerful, allowing traffic destined for the private network behind the remote IPSec endpoint to be automatically encrypted and sent over the tunnel and all other TCP/IP traffic to continue to its destination unimpeded.

Many VPN routers also have IPSec tunneling support, with varying degrees of configuration help. VPN routers without a hardware encryption co-processor can be an order of magnitude slower than the most powerful, more expensive routers with an encryption co-processor built-in. Configuration information and contributed utilities for configuring an IPSec tunnel between a Windows 2000/XP client and a D-Link DI-804HV router are also available at **www.agilent.com/find/adn_vpn_examples**.

Figure 14.5. IPSec VPN combining two private networks

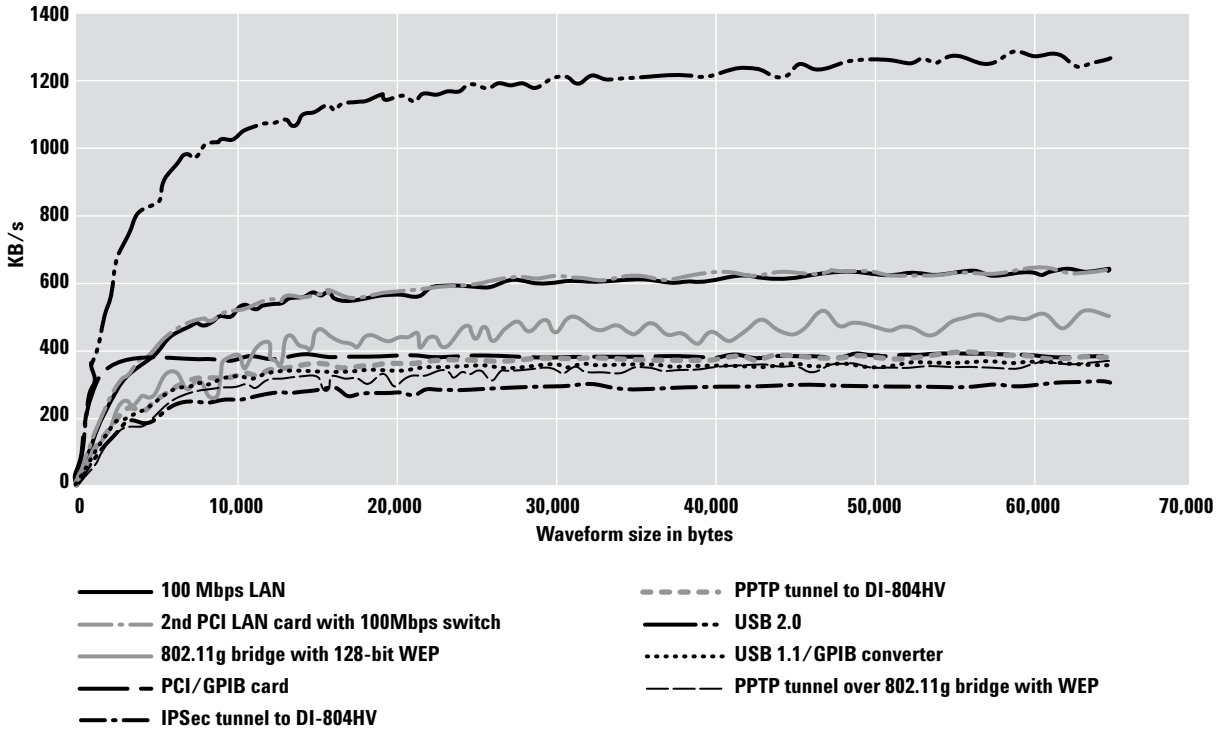


Comparing network performance

As you would expect, various networks have different performance characteristics, based on a combination of the underlying technology and the specific details of each vendor's implementation. Moreover, various instruments behave differently depending on the I/O connection type as well. Figure 14.6 compares the data rates measured while uploading a waveform from a PC to an Agilent 33220A function generator over a variety of transports/ networks. (The results include the time necessary to upload the waveform, plus the time it took to receive a response from the instrument that it had successfully received the data.)

As you can see from the chart, using a second PCI LAN card to create a private LAN incurs no extra cost compared to using the primary LAN card and the corporate intranet's infrastructure to connect to the instrument. Using an 802.11g wireless Ethernet bridge did incur a small performance penalty, as it is limited to 54 Megabits per second, less than the 100 Mbps LAN. In contrast, using the encryption features of the D-Link DI-804HV VPN router caused some of the slowest results because this model does not have a data encryption co-processor, meaning that the encryption of the data packets has to be done by the router's primary CPU.

Figure 14.6. Comparative data rates over vrious network/transport schemes



Conclusion

LAN is a powerful, compelling choice for many test and measurement tasks and systems, but engineers need to be aware of its limitations and complexities to create reliable, long-lasting configurations. The huge variety of LAN media, devices, protocols and technologies mean that a large body of complete tools and solutions is available to choose from when designing test systems. Picking the right technologies to use and deploy is essential to developing the best system in the least amount of time.

15. Using LAN in Test Systems: Setting Up System I/O

Introduction

This chapter describes the components of the Agilent IO Libraries Suite and presents a quick, six-step process that will make LAN-based instrument connections as simple as using GPIB.

Simplifying LAN-based instrument connections

The advantages of LAN technology are making it an attractive alternative to GPIB for system input/output (I/O). As a result, LAN interfaces are becoming more common in test equipment—though LAN ports will likely coexist with GPIB for years to come.

On the surface, the presence of LAN ports in most current-generation PCs and many new-generation test instruments may make connections seem as simple as finding a network cable and plugging it into both devices. Today, making the connection work depends on the LAN services of Microsoft Windows and the additional capabilities provided by the Agilent IO Libraries Suite. A quick, one-time configuration process will make LAN-based instrument connections as easy as using GPIB.

Once the IO Libraries Suite is installed and configured, it accelerates the connection process with software libraries and utilities that let you quickly connect instruments to a PC, configure and verify the connections, and get on with your job—whether it entails the creation of instrument-control software or the use of pre-existing application software.

Assessing the Agilent IO Libraries Suite

Agilent IO Libraries Suite is a collection of libraries and utilities that make LAN, USB, VXI and GPIB connections equally easy to use within your test system. The *libraries* provide the ability to access instruments from software programs that perform test and measurement (T&M) operations. The *utilities* enable quick, easy connections of instruments to a PC by helping you debug test programs and diagnose problems in a test system.

The IO Libraries Suite is included with more than 150 Agilent instruments as well as the Agilent VEE and Agilent T&M Toolkit software products. The IO Libraries Suite also works with instruments and software from other vendors.

As an introduction to the IO Libraries Suite, let's take a closer look at its three major components: Agilent IO Libraries, Agilent Connection Expert and the I/O utilities.

Agilent IO Libraries

The suite includes three separate I/O libraries. Each provides similar functionality that lets you programmatically control instruments, send commands to them and receive responses and data.

Agilent VISA

Agilent's implementation of the Virtual Instrument Software Architecture (VISA) is an industry-standard I/O application programming interface (API). You can use it to develop I/O applications and instrument drivers that will be interoperable with applications from many vendors. These applications and drivers will also comply with IVI Foundation standards for instrument communication and control. The current version of Agilent VISA is backwards compatible with previous versions.

Agilent provides both the C API version of VISA and VISA COM.

Agilent VISA COM

This is a Microsoft Component Object Model (COM) implementation of the VISA standard; it is based on the Agilent VISA architecture. Agilent VISA COM also conforms to IVI Foundation standards.

Agilent SICL

Many test systems still rely on the Standard Instrument Control Library (SICL), which Hewlett-Packard (now Agilent) developed to make software as I/O-independent as possible. This modular library for instrument communication works with a variety of computer architectures, I/O interfaces and operating systems. We include it in the IO Libraries Suite to enable compatibility with customer's legacy programs.

In most cases, we now recommend VISA over SICL. The exceptions are applications that require capabilities such as parallel polling that are unique to SICL.

Suggested approach

If you are developing new test and measurement applications, we generally recommend VISA and VISA COM as the most effective solutions for instrument I/O. The best choice of I/O library depends on your preferred programming language, refer to Table 13.1 on page 124 for recommendations.

Agilent Connection Expert

The Agilent Connection Expert (Figure 15.1) is a software utility that helps you connect instruments to a PC—via GPIB, LAN, USB, RS-232 or the VXI interface—in just a few minutes.

You can use Connection Expert to speed and simplify several essential configuration tasks: configure instrument I/O interfaces; connect to instruments over the LAN; automatically discover instruments connected directly to the PC; and browse the test system's structure and connections (PC, interfaces and instruments). Connection Expert also helps you detect and troubleshoot connectivity problems, either during the configuration process or later when the system is in use.

To enhance program portability and readability, Connection Expert also lets you create programming aliases for each instrument in the system. The ability to update an alias with a new IP address or hostname can make it easier to handle system migration and changing network settings.

Connection Expert also improves user productivity by including an on-screen task guide that provides shortcuts to common tasks and frequently needed information.

I/O Utilities

A set of six software utilities enhances your ability to quickly configure and debug instrument-to-PC connections:

- **Interactive IO.** Lets you query instruments one command at a time and view the response to each command. This utility can help you learn an instrument's command set or prototype commands and check the instrument's responses before you write any code.
- **Remote IO Server.** Enables connections to instruments that are physically connected to another PC on the network. When the Remote IO Server is running on one PC (the server) you can use instruments connected to that server from separate client PCs by using Connection Expert to configure remote interfaces on the clients.
- **ViFind32 Debug Utility.** When called from a script, viFind32 uses VISA functions to find available resources and then list them in a console window. This utility is useful for verifying that Connection Expert has configured all expected interfaces, and that the expected devices are all attached.
- **VISA Assistant.** Included with older versions of the IO Libraries and provided in the latest version as a convenience. Most of its capabilities are replaced by either Connection Expert or Interactive IO.
- **VXI Resource Manager.** Used to configure the Agilent E8491 IEEE-1394 PC link to VXI interface in systems that include modular VXI hardware.
- **IO Control.** Provides easy access to all of the IO Libraries Suite utilities and the associated documentation. The IO Control icon appears in the Windows notification area (Figure 15.2), enabling a quick launch whenever you want to use the utilities.

Figure 15.1. Agilent Connection Expert's easy to follow tree view helps you see connected devices instantly

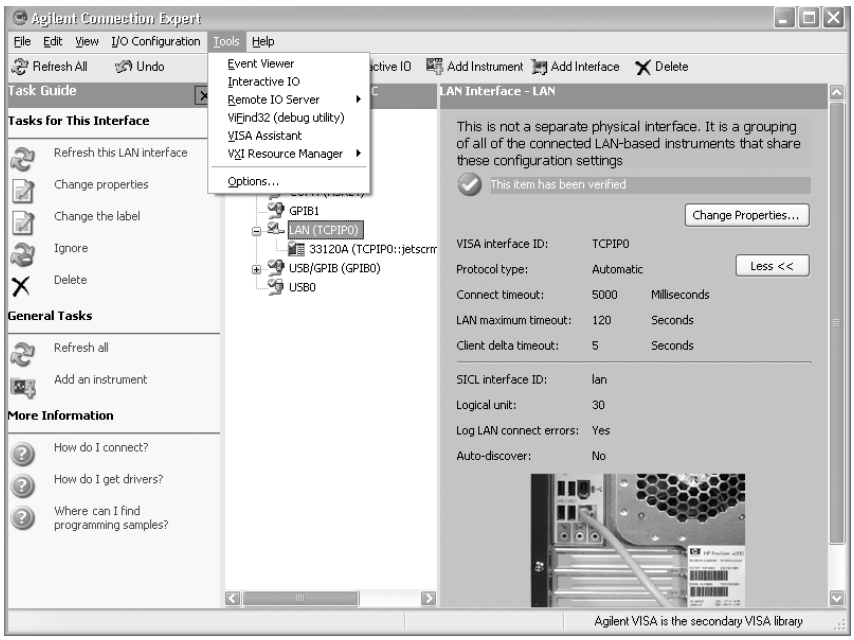


Figure 15.2. The IO Control icon resides in the Windows notification area.

Connecting instruments to LAN

We suggest a six-step process that will help you quickly connect and configure your LAN-enabled instruments. The specific tasks in Steps 2 through 4 depend on whether the instruments are connected to a site LAN or a local, private LAN (these are defined in Step 2). The procedures presented here focus on the private LAN case, which is our recommended approach. If your situation dictates the site LAN approach, please see Chapter 2 of the Agilent *Connectivity Guide*, which is included with the IO Libraries Suite and is also available at **www.agilent.com/find/iolib.**

Step 1: Install I/O software

The first step is to install the Agilent IO Libraries Suite on a PC that meets the minimum system requirements and is running a supported version of Microsoft Windows: 98 Second Edition; 2000 with SP4 or later; Millennium Edition (Me); or XP with SP1 or later. For the latest information about PC requirements, please visit the IO Libraries Web page at **www.agilent.com/find/iosuite** and download the latest version of the product data sheet.

If you don't have the software, you can get a copy from **www.agilent.com/find/iosuite** by downloading it or requesting a CD (see Web site for licensing restrictions). Once you have a copy of the software on or in your PC, the InstallShield Wizard will guide you through the installation process. (For a detailed description of the software installation process, please see Chapter 6 of the Agilent *Connectivity Guide*.)

Step 2: Select network type

Once you've installed the I/O software, the next step is to decide if you will connect the system instruments and PC to a site LAN or a private LAN.

In this chapter we define a site LAN to be an enterprise LAN, a corporate intranet or a workgroup LAN. This type of LAN may carry a tremendous amount of data traffic—and network congestion can severely hinder test system performance. A site LAN may also be more susceptible to viruses, Trojans, worms and other threats that pose a security risk to sensitive data. Physically, this type of connection may involve the direct connection of the PC and instruments to the site LAN or a switched connection through a hub, router or switch.

A private LAN is a standalone network reserved exclusively for use by the test system. It is protected from site LAN traffic (and security risks) by either a router or a PC configured with two LAN cards. The latter configuration will also include at least one hub, switch or router if the system uses multiple instruments. A private LAN implemented with either configuration is our recommended approach. For a detailed description of both approaches, please see Chapter 10, Using LAN in Test Systems: Network Configuration.

A single instrument can also be connected directly to the PC with a crossover Ethernet cable. These cables are usually bright yellow to distinguish them from regular Ethernet cables. This method requires a second LAN card in the PC if the primary LAN card is already connected to the enterprise LAN.

Step 3: Gather network information

With a private LAN configuration you, as the network designer, are effectively the system administrator and can define network parameters that best suit your test system.

One of the key details is support for Dynamic Host Configuration Protocol (DHCP), which automatically assigns IP addresses to devices on the network.[1] In general, a router-based private LAN will support DHCP but not Dynamic Domain Name Server (Dynamic DNS), a service that maps specific names to IP addresses and enables use of names in place of IP addresses in test-system programs.[2]

The key information to be gathered or defined is summarized in Table 15.1. You'll want to make a copy of the card for each instrument that will be connected to the test system's private LAN.

1 Most PCs and most Agilent instruments will use a function called auto-IP to automatically assign an IP address if they are configured for DHCP but cannot connect to a DHCP server.

2 Some of Agilent's LAN-enabled instruments may be addressable by hostname via NetBIOS, an older networked PC protocol that is still supported by Windows and works with most LANs.

Table 15.1. Private LAN information card

Network	DHCP enabled	Yes ____ No ____
	Dynamic DNS enabled	Yes ____ No ____
	UPnP enabled OK	Yes ____ No ____
	Subnet mask	_____.____.____.____
	DNS server IP address	_____.____.____.____
PC	Ethernet (MAC) hardware address	____ ____ ____ ____ ____ ____
	IP address	_____.____.____.____
	Subnet mask	_____.____.____.____
	DNS server	_____.____.____.____
	Hostname	_____
	Domain name	_____
Instrument	Instrument serial number	_____
	Ethernet (MAC) hardware address	____ ____ ____ ____ ____ ____
	IP address	_____.____.____.____
	Subnet mask	_____.____.____.____
	DNS server	_____.____.____.____
	Hostname	_____
	Domain name	_____
	DHCP	On ____ Off ____
	Auto IP	On ____ Off ____ N/A ____
	UPnP	Enabled ____ Disabled ____ N/A ____

Step 4: Connect your instruments

This step assumes that the PC's hardware and software are properly configured for LAN operation and the PC is connected to the private LAN.

As mentioned above, this configuration uses either a router or a PC configured with two LAN cards and a LAN device such as a hub, switch or router. Before connecting any instruments, turn off the power to all instruments. Next, connect each instrument to the LAN device with a standard CAT5e Ethernet cable then turn on power to each instrument. Verify the completion of the power-on sequence for each instrument and, if anything seems unusual, refer to the instrument's user's guide for detailed information.

Hostname versus IP address

We recommend that you configure LAN instruments with a hostname and use it to connect to them whenever possible. IP addresses assigned by DHCP can change without warning, breaking established connections to your instruments. If your network doesn't support connecting by hostname, we recommend the use of a statically configured IP address for each instrument. If any of your instruments are connected to the site LAN, you will need to ask your network administrator to provide static IP addresses for those instruments.

Step 5: Configure your instruments

Configuration may not be necessary for the latest generation of LAN-enabled instruments. In general, the default configuration of these instruments is compatible with the procedures presented here.

For current-generation instruments with modified LAN configurations or older instruments that require manual configuration, you will need to enter a few quick changes via the front panel or a Web browser. Most LAN-based instruments have a built-in Web server, making it possible to access an instrument's internal Web pages (Figure 15.3) and, in most cases, view and modify the network configuration parameters as needed.

Making changes via the built-in Web server begins by entering the instrument's IP address into the browser's address box (e.g., http://192.168.1.200) and pressing Enter. This should display the instrument's welcome page, which may provide links to other pages. Access the page that displays the current LAN configuration and then modify the instrument's TCP/IP parameters as necessary: this may require changes to the IP address, subnet mask and default gateway.[3] To complete the process, you may have to save the changes then reboot the instrument by turning power off then on again.

Step 6: Run Agilent Connection Expert

With Steps 1 through 5 completed, you're ready to launch Agilent Connection Expert. Click on the Agilent IO Control icon in the Windows notification area then select Agilent Connection Expert (or click "Refresh All" if Connection Expert is already running). You can now perform additional tasks related to LAN-enabled instruments: add a LAN instrument to the system (mandatory), configure a LAN interface (optional) or communicate with any connected instrument via Interactive IO (optional).

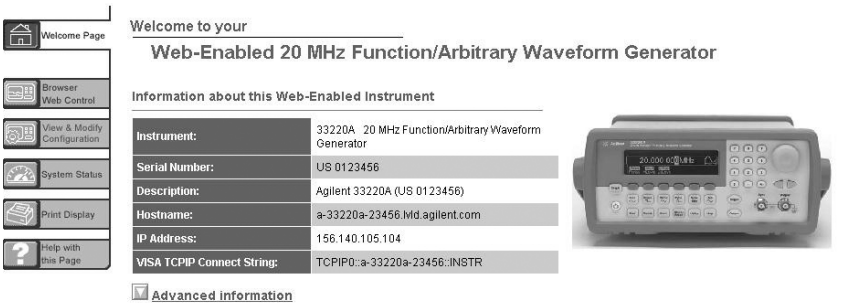
Add a LAN instrument. Whether an instrument resides on a private or site LAN, there are several ways to add the instrument to your test system. You may also use any of these methods to make a change to any instrument that is already part of the system.

When using Connection Expert, the key concept is the "local subnet," which is typically the portion of your intranet connected to the private or non-enterprise side of the nearest router. To prevent disruptions of network traffic beyond the router—and avoid excessively long searches across the entire LAN—Connection Expert can only discover instruments automatically within the local subnet. To communicate with remote instruments (those on the other side of the router) you must be able to specify a hostname or IP address in the LAN Instrument dialog box.

- Local subnet instrument. When an instrument is on the local subnet, the easiest way to add the instrument to the test system is to ask that Connection Expert discover it. This eliminates the need to specify the instrument's hostname or IP address.
- Non-local subnet instrument. After you enter an instrument's hostname or IP address, Connection Expert will try to open a connection to that instrument on the LAN. If it succeeds, it will perform the same bus addressing procedures as it would for a local instrument.

Configure a LAN interface. If you select the LAN interface in the explorer pane, the properties pane will display the current properties for that interface. The properties pane displays the most commonly accessed properties at the top and provides a "More" button that will display additional properties in the lower portion of the pane. Clicking on the Change Properties button will display a dialog box that lets you modify configuration parameters such as interface ID (used by VISA and SICL) and timeout values. The online Help, available from the Help button in each dialog box, gives information on each configuration parameter.

Figure 15.3. Browser-accessible welcome page for the Agilent 33220A function/arbitrary waveform generator



3 Individual instruments will respond differently to these changes. In most cases the new settings will not take effect until you cycle the instrument's power off then on. Some instruments may change the IP address immediately and you will have to enter the new address into the browser's address box before making additional changes.

Coexisting with other versions of VISA

Although the VISA API is a standard, each vendor employs different layers beneath the VISA layer to control their hardware. For proper operation, the version of VISA installed on the system's host PC must be compatible with the I/O hardware.

Fortunately, it's possible to perform a side-by-side installation of Agilent VISA with other VISA libraries and achieve proper I/O operation. For example, you can use Agilent VISA and National Instruments' NI-VISA in the same PC.

The VISA standard requires that visa32.dll, the dynamic-link library that implements the VISA interface, be installed in specific locations. Thus, only one installed implementation can fully comply with the standard at one time.

To avoid conflicts, the Agilent IO Libraries Suite gives you the option to install Agilent VISA in side-by-side mode. In this case, Agilent VISA is installed in a different location and does not overwrite any other VISA already present on the PC.

After installing Agilent VISA in side-by-side mode you can use either the Agilent or NI VISA library in your programs. The side-by-side installation allows you to choose which VISA library you link your program against so you can take advantage of the support and features of each vendor's VISA implementation.

If you choose to use Agilent interface hardware in a program linked against NI-VISA, IO Libraries Suite online Help can guide you through the configuration of NI-VISA. This is the easiest way to use your VISA code without rebuilding your program. (NI-VISA does not allow side-by-side mode by default and must be manually enabled.)

If you use NI GPIB cards and devices, you don't need to install NI-VISA. Instead, you can simply install NI-488.2 as a driver for these devices and avoid the complications of side-by-side operation. You could then use Connection Expert to configure Agilent VISA to use the NI GPIB card.

Whenever you mix interface hardware from different vendors, it's best to configure each device using the configuration tools from its manufacturer. NI interfaces should be configured with the NI MAX utility prior to use. Agilent interfaces are automatically configured when you install the IO Libraries Suite, but if you add a new interface or decide to change interface properties, use Agilent Connection Expert.

Communicate via Interactive IO. You can use this software utility from within Connection Expert (or from IO Control) to verify communication with instruments via LAN:

- Interactively send commands and view responses without writing program code
- Quickly verify connectivity to an instrument
- Troubleshoot communication problems
- Learn the instrument's command set
- Rapidly prototype commands and check the instrument's responses before you begin writing code

You can start Interactive IO from within Connection Expert in one of three ways: by clicking the *Send commands to this instrument* task in the task guide; by clicking Tools then Interactive IO on the Connection Expert menu bar; or by right-clicking the instrument in the explorer panel and selecting *Send Commands To This Instrument.*

Conclusion

Connecting instruments to LAN is almost as simple as plugging in a network cable if you take advantage of the libraries and utilities that are part of the Agilent IO Libraries Suite and follow the quick, six-step configuration process described in this chapter. Once that process is completed, GPIB, LAN, USB and other standard interfaces are equally easy to use within a test system. What's more, included tools such as Agilent Connection Expert and a set of six I/O utilities make it easy to configure, debug and troubleshoot those connections.

Section 3. LXI: The Future of Test

Overview

The four chapters in this section introduce the features and benefits of LXI (LAN eXtensions for Instrumentation), a new standard that combines the functionality and PC-standard connectivity of stand-alone instruments with the modularity and compact size of plug-in cards—but without the size or cost of a cardcage.

16. **Value, Performance and Flexibility: the Promise of LXI,** provides an introduction to LXI, presents its advantages, and outlines LXI usage models that expand the reach, capabilities and definition of test systems.

17. **Transitioning from GPIB to LXI,** compares GPIB and LXI, sketches hybrid system architectures, outlines a step-by-step approach to system set-up, and describes how to easily modify existing system software to work with LXI devices.

18. **Creating Hybrid Test Systems with PXI, VXI and LXI,** compares PXI and VXI with LXI, sketches hybrid system architectures that incorporate your existing test assets and describes what will be possible in the future as you migrate to fully LXI-based systems.

19. **Assessing Synthetic Instruments,** presents a brief history of synthetic instruments (SIs), compares a rack-and-stack system to an SI-based system, describes the initial applications of SIs and illustrates the emulation of conventional instruments with SIs.

16. Value, Performance and Flexibility: The Promise of LXI

Introduction

This chapter provides an introduction to LAN eXtensions for Instrumentation (LXI), reviews why test managers are looking for an alternative to conventional test architectures, presents the advantages of LXI, offers a closer look at the LXI standard, and outlines usage scenarios that expand the reach and capabilities—and perhaps the definition—of test systems. Two appendixes discuss the concept of synthetic instruments and LXI's role within the Agilent Open development strategy.

Why test managers are asking for a new approach

Test managers across many industries face several of the same issues: shorter launch windows, reduced staffing, dwindling software expertise, smaller development budgets, and outsourced (or offshore) manufacturing. Most are looking for the same solution: a more cost-effective way to develop test systems.

An obvious first step is to reduce the cost of instrumentation. The overhead costs of current modular systems—cardcages, slot-0 controllers, proprietary interfaces, and so on—shrink the budget available for actual measurement hardware. Also, if the cardcage is filled, the addition of just one more device to the system requires an additional cardcage. Similarly, because most PCs now include USB and LAN interfaces, it seems wasteful to require the additional cost and complexity of a measurement-specific interface.

In the big picture of system development, however, cost effectiveness goes far beyond simply lowering the cost of test instrumentation. Even if GPIB, PXI and VXI hardware were free, developers would still face six challenges that affect the cost-effectiveness of system creation: reuse, set-up time, system throughput, system size, consistency and future-proofing.

- **Reuse.** Developers seldom have the luxury of building a test system with all-new hardware and software. As a result, many systems include a collection of instruments that use different I/O interfaces and command sets. It can be difficult to reuse existing instrumentation and test-system code without tools that simplify instrument connectivity and control in the PC environment. The challenge of reuse extends to software too, of course. It's difficult to leverage software and ensure measurement integrity across the product lifecycle if different types of instruments are used in each phase. This is especially true if testing shifts from benchtop instruments in R&D to modular instruments in manufacturing.
- **Set-up time.** System set-up can be time consuming, especially when you're trying to get the PC to communicate with the instruments or get the instruments to work with the system software. It's even more time consuming with systems that include multiple interfaces: GPIB, RS-232C, VXI, PXI, MXI, FireWire, USB or LAN. Add to that multiple I/O libraries and instrument drivers from multiple manufacturers, and it may take days or weeks to troubleshoot the system and get it to work as expected.
- **System throughput.** In time-critical applications, every millisecond counts. However, improving overall system throughput requires more than just a fast backplane. Bottlenecks may occur in test routines, measurement algorithms, data transfers, the sequencing of system tasks and more.

- **System size.** Whenever systems must be shipped elsewhere or deployed where floor space is at a premium, system size matters. Unfortunately, with existing approaches, this may also mean sacrificing functionality, performance and accuracy as the system shrinks.
- **Consistency.** In systems that require source, measure, power and RF/microwave capabilities, developers often need to mix two or more of the current instrumentation standards. This type of inefficiency also affects cost effectiveness.
- **Future-proofing.** With limited versatility, existing test architectures make it difficult to meet future needs—higher frequencies, greater accuracy, faster throughput and so on. As more systems are deployed to remote locations, they become increasingly difficult to manage and troubleshoot without onsite expertise. In addition, test systems often remain in service longer than the lifetime of most backplanes and interfaces. Computer backplanes—ISA, EISA, VME, PCI, and Compact PCI—change every few years but usually offer little or no backward compatibility. The instrumentation versions (VXI and PXI) have the same drawback. To compound the problem, standardized test and measurement interfaces such as GPIB and MXI have fallen short of both the increased speed and widespread adoption of LAN and USB. Instead, new GPIB or MXI cards must be developed and purchased whenever computer architectures change.

There are also a few issues specific to each of today's three major legacy test-system architectures.

- **GPIB.** Although this remains the current instrumentation standard, it has slower data transfer rates than other architectures, forces you to install an interface card in your PC, requires expensive cables, and allows only 14 devices on the bus.
- **VXI.** This architecture requires an expensive cardcage, a slot-0 controller and an expensive, proprietary interface (MXI).
- **PXI.** In addition to the VXI overhead costs mentioned above, PXI has issues with size, power and EMI that limit the range of solutions to those normally covered by PC plug-in cards. PXI is also transitioning to PXI Express, limiting the longevity of PXI modules or requiring expensive hybrid mainframes.

Figure 16.1. Current-generation instruments such as the Agilent 34980A switch/measure unit include LAN, USB and GPIB interfaces.

Addressing the challenges with LXI

To help test system engineers overcome all these drawbacks, Agilent is leading the way to a new vision for test systems.

Building on the widespread LAN foundation

Many current-generation instruments include LAN ports (see Figure 16.1), and LXI is the next logical step in the evolution of LAN-based instrumentation. LAN is gaining momentum in T&M because it has several inherent advantages over most parallel and serial interfaces. For example, LAN can handle an unlimited number of nodes and provides long distance inter-device connectivity. It also includes TCP/IP error checking and fault detection—and these functions minimally impact throughput rates.[1] Better still, LAN enables automatic device discovery, addressing, asset management and network management. LAN also has a cost advantage: the prices of cables, interface cards, hubs, routers, switches, wireless access points and so on are low and continue to fall.

1 This is especially true if the test system uses a dedicated network. To learn more about creating private networks for test systems, please see Chapter 10.

These advantages come from the computer industry's substantial investment in networking technology. A big part of that investment is in brainpower: the computer industry employs far more design engineers than the T&M industry. Since 1980, their efforts have boosted Ethernet speeds by three orders of magnitude, from 10 Mb/s to 10 Gb/s. Even more impressive, they have preserved backward compatibility as speeds have increased. In comparison, T&M standard interfaces such as GPIB and MXI have not kept pace with the speed, capabilities or compatibility of LAN.

Rather than inventing yet another proprietary standard, it makes far more sense to ride the wave of LAN innovation. By leveraging PC-standard technologies, T&M equipment makers can focus on what they do best, which is provide great measurements.

The LAN interface becomes "GPIB easy" when used with innovative software products such as the Agilent IO Libraries Suite, which simplifies connections between PCs and LAN-enabled instruments. Looking to the future, adding the enhancements defined by the LXI standard will let test engineers take even greater advantage of this powerful I/O connection.

Extending LAN for instrumentation

The LXI vision starts with full-fledged instruments packaged in easy-to-integrate modules that utilize PC-standard I/O. LXI can be packaged in larger sizes with front panels and displays, or integrated into smaller, faceless modules. It also includes hardware and software building blocks that enable rapid arrangement and rearrangement of functional building-block modules known as *synthetic instruments* (see Appendix 16A) that increase a system's flexibility while reducing its size and cost. By specifying the interaction of proven, widely used standards such as Ethernet LAN, Web browsers and IVI drivers, LXI enables fast, efficient and cost-effective creation and reconfiguration of test systems. LXI combines the measurement functionality and PC-standard input/ output (I/O) connectivity of standalone instruments with the modularity and compact size of plug-in cards—but without the size or cost of a cardcage.

The promise of system longevity has inspired over 50 companies to join the LXI Consortium including all the largest test & measurement companies (see sidebar). Agilent and others introduced the first wave of LXI-compliant products in September 2005. As the number of available LXI devices continues to grow, you will be well-equipped to take the next step beyond GPIB, PXI and VXI.

The LXI standard enables long-lived instrument and system implementations by relying on the stability of computer and networking standards, and by freeing system developers from proprietary standards that often fall behind in performance and functionality.

The LXI Consortium

The LXI Consortium is a not-for-profit corporation that coordinates the efforts of leading companies in the T&M industry. Its driving goal is to ensure a consistent, positive user experience through hardware and software interoperability. The consortium aims to achieve this goal by developing, supporting and promoting the LXI standard. The formation of the consortium was driven by the realization that several companies were developing LAN-based measurement modules.

Ultimately, many agreed that it made sense to abandon multiple incompatible approaches and instead combine their efforts into an industry standard that will better serve the present and future needs of T&M customers. To learn more about the LXI Consortium, visit **www.lxistandard.org.**

The advantages of LXI

LXI's visionary approach delivers numerous benefits in virtually every aspect of system design, implementation, operation, and maintenance. By addressing all of the shortcomings described earlier with other architectures, LXI does more to help you reduce the expense and effort required to create cost-effective test systems.

Ease of use

The LXI standard address ease of use for system developers in a variety of ways, including harnessing the Ethernet standard, enabling easy interaction via Web browsers, making programming more efficient with standard drivers and simplifying physical integration.

Harnessing the Ethernet standard

LXI includes some key elements that simplify the use of LAN in test systems:

- **Physical layer.** To help ensure successful instrument interaction, the LXI standard specifies automatic negotiation of LAN transmission speed and duplex communication. The standard also recommends Auto MDIX, a feature that enables the use of either straight-through or crossover LAN cables in direct controller-to-instrument or peer-to-peer connections. The instrument automatically adjusts to the existing cable and its communication counterpart.
- **Network (IP) layer.** LXI instruments support automatic IP configuration through a DHCP server (often available in managed corporate networks and in cable/DSL routers) or through dynamic configuration of local addresses (typically used in small or ad-hoc networks). LXI also recommends support for DNS, which instruments can use to publish their host name through a DNS server (another feature usually available in corporate networks).
- **Application layer.** LXI-compliant instruments support the VXI-11 protocol (based on remote procedure calls) for automatic discovery of new instruments and identification through the *IDN? query.

Enabling easy interaction

A few noteworthy instrument control features help LXI surpass GPIB in ease-of-use. For example, LXI devices include a built-in Web server to enable configuration and troubleshooting. In many Agilent LXI products, the instrument page also allows interactive instrument control and monitoring, a capability that can be very useful during system configuration and deployment. It also simplifies remote troubleshooting.

Making programming more efficient

For programmatic control, you can use IVI instrument drivers (see Chapter 3). Recommended by the LXI standard, IVI-COM drivers are based on the widely used Microsoft COM architecture and work with today's most popular test software environments. These object-oriented drivers use a hierarchical API, making it easy to utilize the advanced features of modern, object-oriented environments. One key example is easy navigation through a driver's hierarchy of functions and simplified coding via autosuggestion and autocompletion. Agilent LXI instruments also support ASCII-based SCPI commands for programming flexibility and to support non-Windows environments.

Simplifying physical integration

The LXI standard also includes an optional mechanical specification that simplifies the integration of modules within system racks. Compliant modules are full- or half-rack wide and standard EIA heights (e.g. 1U, 2U). The standard also specifies that signal input and output connectors (and status lights) are placed on the front of the module while power, Ethernet, triggering and other control connectors are placed on the rear.

Reducing set-up time

Through proven standards such as Ethernet and IVI drivers, LXI ensures that everything is compatible—and setup will take less time. Because Web pages built into every LXI instrument, a standard Web browser is all you need to view device information, change its configuration and even monitor results and control measurements (in many Agilent LXI devices). You can also use proven tools such as LAN hardware, LAN cables and ping servers to communicate via LAN and troubleshoot local or remote systems.

Flexibility

LXI's modular approach makes it easy to mix and match modules that provide the exact functionality required for each system or application. LXI-compliant instruments provide new levels of flexibility in hardware selection, product testing, software reuse, instrument communication and even organizational responsiveness.

Addressing multiple testing needs

The LXI standard spans classic instruments, faceless modular instruments, and functional building block modules (synthetic instruments or SIs). Broad-based support from major instrumentation vendors means you will be able to address your full range of testing needs—source, measure, RF/microwave, switching and power—with just one architecture. Even when space is at a premium, you don't have to sacrifice functionality, accuracy or performance.

Even better, you won't have to sacrifice your existing test assets. To help you create hybrid systems that use LXI-based devices alongside GPIB, PXI and VXI hardware, Agilent offers a range of I/O gateways and converters. Bringing your system software forward to work with LAN requires nothing more than simple address changes.

Testing all along the lifecycle

The various forms of LXI devices also make it easier for you to test your product across its entire lifecycle. In many cases, a classic instrument can be used on the bench while an equivalent faceless instrument can be used in a rack in the final test system— without rewriting the system software. This concept can be extended with synthetic instruments: through the necessary SI hardware and software modules, a few functional building blocks can do the work of multiple RF/microwave instruments.

Working independently

With their embedded processors, today's test instruments have enough computing power to carry out measurement tasks on their own, freeing the system controller for other tasks. LXI uses this power to provide greater flexibility in communication, too: instruments can communicate without arbitration through the system controller. Instead, they can use TCP for peer-to-peer communication and UDP for multicast (one-to-many) messages.

Boosting team efficiency

LXI also helps you address future organizational needs. Test-system experts are becoming scarce in many organizations and can't be everywhere at once—onsite, offshore or anywhere in between. With LXI, you can place test systems virtually anywhere on your intranet, enabling your team to perform centralized troubleshooting, remote monitoring and more.

Modularity and scalability

Scalability means buying just what you need when you need it—and being able to easily expand the system in the future. With LXI, scalability follows from modularity. This truly modular architecture lets you freely mix and match different types of measurement resources and add measurement channels, digital I/O lines, switches and signal sources as you go.

In PXI and VXI, if a cardcage is filled, the addition of just one more device to the system requires the addition— and additional cost—of another cardcage. Because LXI modules don't require a cardcage, there is no hard limit to the number of devices you can add to a system. In practice, you will instead be limited by factors such as rack space and the number of ports available on a hub or router.

Performance

Test-and-measurement interfaces such as GPIB and MXI are challenged by the need for increasing bandwidths and faster data-transfer rates. One key advantage of LXI is its ability to leverage ongoing innovations in LAN that satisfy the need for speed.

LXI makes it possible to build high-speed distributed systems that utilize intelligent instruments communicating with each other—without PC intervention—and operating in parallel. Everything will stay synchronized through the use of the IEEE 1588 timing and synchronization standard (see "Precise Synchronization" on page 149), LAN-based triggers, peer-to-peer and multicast messaging, and the hardware trigger bus. These capabilities offer new ways to build highly efficient test systems that deliver dramatic improvements in overall system throughput.

Moving more megabytes

With a Fast Ethernet connection (IEEE 803.2u, 100 Mb/s), the maximum payload data rate is approximately 12.5 MB/s. Gigabit Ethernet (IEEE 802.3z), which is recommended by the LXI specification, boosts top-end performance by a factor of ten to approximately 125 MB/s.[2] Looking ahead to 10 Gb Ethernet, LXI will be able to surpass the performance of VXI 3.0 (160 MB/s). The backward compatibility of the various Ethernet standards is an added bonus that contributes to system longevity.

Raw network speed isn't the only consideration: simultaneous communication on any network can cause degradation in performance due to collisions and retransmission. To avoid or limit this effect, we recommend the creation of a local subnet dedicated to the test system.

2 Assuming IPv4 and maximum frame size, the bandwidth remaining for application data is about 95 percent of the transmission rate.

Accelerating system throughput

Other aspects of LXI enhance performance by enabling faster system throughput. For example, LXI makes it possible to build high-speed distributed systems comprised of intelligent instruments that can communicate with each other and operate in parallel. Devices stay synchronized through the IEEE 1588 timing standard, LAN-based triggers, peer-to-peer and multicast messaging, and a hardware trigger bus. IEEE 1588 also accelerates throughput via time-based triggering, which initiates instrument operations at a specific time rather than after a trigger or command.

Distributed applications

Unlike cardcage-based systems, LXI modules can be easily distributed in a test rack, across a lab or throughout a building. This allows you to place instruments where they can best meet the needs of each measurement or application. Examples include the monitoring systems used in environmental applications, power generation and the process-control industry. Another intriguing example is testing of wireless base stations: protocol test equipment can be placed near or inside base stations located many miles apart.

With LXI, these solutions can be designed using the same instruments you would use for local applications and rack-based systems. There is no need to create custom gateways—remote access comes without extra effort. Using your corporate intranet or the public Internet, large distances can be bridged easily and the connection is transparent to the end user.

Of course, security is a concern for any application that requires a connection outside your secure, well-controlled corporate network. Rest assured that solutions designed for the IT world also work with LXI. You can utilize routers that include security features such as access filtering based on MAC or IP addresses, WLAN encryption and so on. If a distributed application needs to access the public Internet, you can use a virtual private network (VPN) to send IP packets securely, encrypted through IPsec or other encryption protocols.

Leverage and longevity

In general, test systems address two large classes of devices: long-lived and short-lived products. Many devices developed for aerospace and defense applications require test systems that are easy to maintain and update far into the future. In contrast, rapidly evolving commercial wireless products require test systems that can be developed rapidly and within budget—and be easily reused as the products evolve. The ability to meet the needs of either long- or short-lived devices improves with LXI, which is designed to fulfill the promise of long-lived measurement hardware, I/O and software.

This need for stability is in sharp contrast to the rapid innovation cycles in today's computer buses. For example, in just a few years instruments based on computer buses have had to change from ISA to EISA to PCI and now to PCI Express (a serial bus not compatible with previous parallel implementations).

In comparison, Ethernet is an extremely stable standard. Like GPIB, it's more than 30 years old—and Ethernet is clearly here to stay. With its stability and other virtues, Ethernet has been adopted in many industries, including corporate communications, consumer electronics, industrial automation and now test equipment.

Ethernet is also a living, evolving standard. It has accommodated the addition of higher-layer protocols as well as enhancements such as Gigabit Ethernet at the physical layer and IPv6 at the network layer. Amazingly, these enhancements have retained backward compatibility, protecting investments in previous versions of the standard.

Extending the life of LXI systems

In addition to the continued evolution and assured compatibility of LAN technology, two additional ideas extend the life of LXI systems in particular: the ability to download new capabilities or personalities into intelligent instruments and the possibility of injecting updated or upgraded technology into SI-based systems. These capabilities simplify the task—and reduce the cost—of keeping pace with evolving measurement standards, wider frequency ranges and tighter accuracy requirements. Taking a wider view, LXI enables new levels of versatility by making it possible to configure or reconfigure a system through software changes to IEEE 1588 clocking and LAN triggering.

LXI also helps you address future organizational needs. Test-system experts are becoming scarce in many organizations and can't be everywhere at once—onsite, offshore or anywhere in between. With LXI, you can place test systems virtually anywhere on your intranet, enabling your team to perform centralized troubleshooting, remote monitoring and more.

Reusing existing instruments and software

The various forms of LXI devices make it easier for you to test your product across its entire lifecycle. In some cases, a classic instrument can be used on the bench or in a rack to develop and refine test routines that can then be used with an equivalent faceless instrument in the final test system. This concept can be extended with synthetic instruments: through the necessary SI software modules, a few functional building blocks can do the work of multiple RF/microwave instruments.

Agilent also offers a range of I/O gateways and converters that make it easy to create hybrid systems that include LXI-based devices and your existing test assets. Bringing your system software forward to work with LAN requires nothing more than simple address changes.

Cost

LXI offers potential cost savings throughout the lifecycle of your systems. Those savings start with the ability to incorporate, rather than replace, much of your existing instrumentation. Unlike other architectures, LXI isn't an "all or nothing" proposition. You can manage the cost of transition by using devices such as the Agilent E5810A LAN/GPIB gateway to create hybrid systems that include existing GPIB-only equipment alongside LXI-based instrumentation.

When you're ready for an all-LXI test system, it is likely to be less expensive than a system based solely on GPIB, VXI or PXI. This is especially true when compared to VXI and PXI because LXI doesn't require costly cardcages, slot-0 controllers or proprietary interfaces and cables.

The LAN interface required for LXI is a standard, no-cost feature of most PCs. Also, LAN infrastructure such as hubs, switches and routers is either already available or can be purchased at very moderate cost. For example, Fast Ethernet routers are available for less than US$75 at consumer electronics stores.

LXI also lets software developers leverage their existing investments because test routines written for standalone instruments will also work with faceless modular equivalents. System integration is also faster because LXI utilizes the host PC's LAN interface and Web browser; no time is spent installing and configuring a GPIB or MXI interface or installing software instrument front panels.

LXI-based remote devices provide a low-cost, portable way to deploy sensors, cameras, microphones and more. Benchtop implementations provide accurate, cost-effective instruments with built-in LAN connections (Figure 16.2). Intelligent instruments can receive new measurement capabilities and personalities via download, enabling reuse for a variety of applications.

In addition to these initial savings, LXI can help reduce support and maintenance cost through its enhanced ease-of-use, flexibility and stability.

Precise synchronization

Measurement accuracy depends on precise synchronization of every device in a test system. While LAN technologies are excellent for communication and control, their timing specifications are not stringent enough for measurement applications—especially in distributed systems. The IEEE 1588 standard, through its precision time protocol (PTP), addresses this shortcoming.

The underlying technique—developed by Agilent Labs—relies on system devices that contain real-time clocks and, via the PTP, enables system-wide synchronization

In a typical LXI-enabled distributed application, the system will include intelligent instruments capable of performing measurement tasks on their own, independent of the system controller. To make this approach practical, the instruments will typically include a local clock that enables them to time-stamp measurements and events.

When the synchronization process begins, those devices identify the most accurate clock in the system and assign it the role of master clock. Figure 16.3 illustrates the rest of this elegantly simple process.

Figure 16.2. Classic instruments such as the Agilent 34980A multifunction switch/measure unit offer LXI compliance

1. The master clock sends a sync pulse and the current time to every other device on the network. All slaves set their clocks to the master time.

2. Each slave sends a time-stamped reply to the master. The master calculates the offset between the original transmission time of the sync pulse and the various reception times.

3. The master sends an offset value to each slave, which adjusts its clock to compensate for the difference between the master's sync pulse and its reception time at the slave. After this initial alignment, periodic sync pulses are enough to keep the slaves precisely synchronized to the master clock. The result is a test system that can address the most demanding distributed measurement applications.

What's especially appealing about IEEE 1588 is that it works across Ethernet—the same Ethernet being used for instrument control. No additional cables are required. Depending on the size of the network and its variation in latency times, it is possible to achieve precise synchronization of LXI devices located anywhere on a network—local or remote.

To learn more about IEEE 1588, visit the National Institute of Science and Technology (NIST) Web site at **http://ieee1588.nist.gov**.

Security

Security risks can be minimized through simple precautions such as creating a private, protected LAN. The standard capabilities of most Windows PCs and many low-cost networking products enable two viable approaches to security: one is built around a router (with built-in firewall) and the other is based on a PC equipped with two LAN cards. For a detailed description of both approaches, please see Chapter 10.

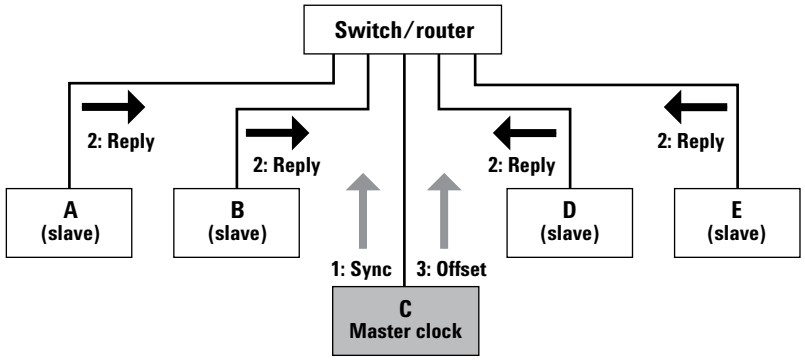
A closer look at LXI

The LXI standard defines instrument-specific requirements needed to ensure reliable, accurate operation in a system environment:

- Cooling
- Triggering
- Interrupt handling
- Mechanical interfaces
- Software interfaces
- Electromagnetic and radio frequency interference
- Network routing and switching
- Discovery
- Synchronization across multiple devices

LAN is at the heart of LXI. However, instead of modifying existing standards, LXI clearly specifies the interaction of proven standards in five areas: physical implementation, Ethernet, programmatic interface, instrument pages and synchronization.

Figure 16.3. In an IEEE 1588-enabled network, a simple process ensures precise synchronization between all devices



Physical implementation

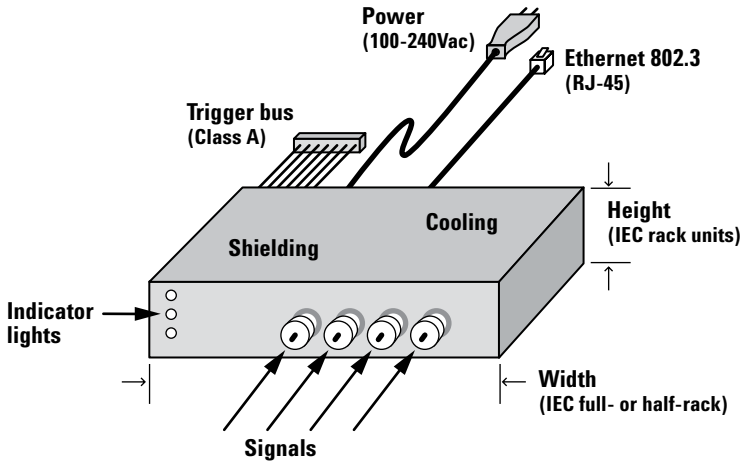
To achieve physical consistency, the LXI standard begins with IEC-standard rack dimensions. To help simplify system integration and implementation, it also recommends the placement of various connections (Figure 16.4). For example, compliant instruments use the front panel for signal inputs and outputs plus indicator lights for LAN, power and IEEE 1588 (synchronization). The rear panel is used for hardware triggering, power input and Ethernet communication. Each LXI module must meet worldwide standard cooling and EMI shielding specifications.

The LXI standard defines three types of instruments that can be readily mixed and matched within a test system.

- **Class C.** This is the base class and includes all of the requirements for the LAN interface and protocols, LAN discovery, IVI driver interface and instrument pages plus recommendations for power, cooling, size, indicators and a reset button. Class C devices are standalone or bench-type instruments that replace GPIB with LAN and harness the full breadth of LAN's capabilities. They also utilize a Web interface (with XML) for instrument set-up and data access. To simplify programming, Class C instruments provide an IVI driver API (application programming interface). Today, most instruments meet the class C specification. Over time, more instruments will implement Class B and A capabilities.
- **Class B.** These devices are designed to enable simple synchronization and distributed measurement systems. They meet Class C requirements and add IEEE 1588 synchronization. Class B also adds peer-to-peer and multicast LAN messaging (required in Class B and A, permitted in Class C).
- **Class A.** Devices in this category satisfy Class C and B requirements and add a hardware trigger bus. This bus enables triggering of LXI instruments in close proximity. Similar to the VXI backplane bus, the trigger bus is an eight-line, differential-voltage bus that enables precise timing accuracy and dynamic trigger reconfiguration for co-located instruments.

Physically, standalone LXI instruments may be full- or half-rack width and tall enough to accommodate the front-panel display and keypad. Modular LXI instruments (without a display or keyboard) are typically half- or full-rack width and just 1U or 2U high.

Figure 16.4. The LXI standard strives for physical consistency that simplifies system integration and implementation



Although not mentioned in the standard, LXI enables leverage from classic instruments into faceless modular instruments and synthetic instruments. Agilent is already moving in this direction with the introduction of synthetic instruments based on popular benchtop RF and microwave products. By using the same measurement hardware in both classic and modular instruments, we're boosting your ability to leverage test software as the system evolves.

Ethernet

LXI uses the IEEE 802.3 networking standard to define the appropriate connections, protocols, speeds, addresses, configuration and default conditions that must be implemented to ensure a consistent—yet easy-to-use—test system.

- **Connections.** LXI devices use standard RJ-45 connectors and implement Auto-MDIX to sense the polarity of LAN cables (through or crossover).
- **Protocols.** Compliant devices are required to implement TCP (transmission control protocol), UDP (user datagram protocol) and IPv4 (Internet protocol version 4). TCP is the standard Internet protocol that will be used most often in peer-to-peer messaging while UDP is a low-overhead protocol that will be typically used for multicast messaging when high speed delivery is critical.
- **Speeds.** The standard recommends use of 1 Gb Ethernet (and permits 100 Mb) with auto-negotiation to ensure that devices use their optimum speed.

- **Addresses.** LXI devices must support IP addresses (assigned by the server), MAC addresses (assigned by the manufacturer) and hostnames (assigned by the user).
- **Configuration.** Compliant devices must support ICMP (ping server), DHCP-based assignment of IP addresses, manual Domain Name Server (DNS) and Dynamic DNS. Because DNS can translate domain names into IP addresses, it can contribute to the longevity of system software: IP addresses may change but domain names will not.
- **Default conditions.** As a safeguard, LXI defines a set of default LAN conditions and requires a "LAN configuration initialize" (LCI) switch that will reset a device to this set of known conditions.

Programmatic interface

Because the LXI standard requires that all devices have an Interchangeable Virtual Instrument (IVI) driver, it allows you to use whichever programming language or development environment you prefer. IVI-COM and IVI-C are well-established industry standard drivers that instrument makers supply with their products.

The LXI standard also mandates that compliant devices implement LAN discovery, which enables the host PC to identify connected instruments. Currently, the LXI standard requires use of the VXI-11 protocol, which defines LAN-based connectivity for all types of test equipment, not just VXI. Going forward, future revisions to the LXI standard may include other proven discovery mechanisms such as Universal Plug&Play (UPnP).

Instrument pages

Every LXI-compliant device must be able to serve its own Web page. This page provides key information about the device, including its manufacturer, model number, serial number, description, hostname, MAC address and IP address. The standard also requires a browser-accessible configuration page that allows the user to change parameters such as hostname, description, IP address, subnet mask and TCP/IP configuration mode. Accessing these Web pages is as simple as typing the instrument IP address into the address line of any web browser.

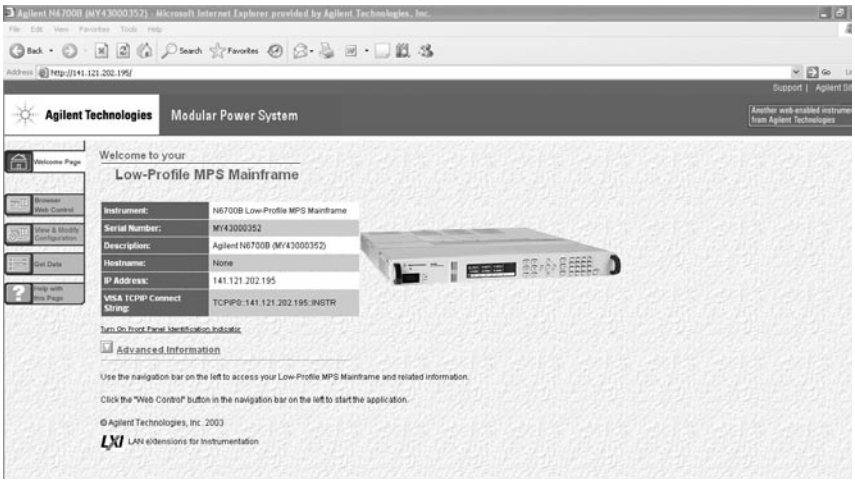
Agilent's LXI-compliant instruments go beyond the LXI requirements, providing monitor and control capabilities (see Figure 16.5). For example, you can set up a DMM, command it to start making measurements and then read the results. Some of our LXI devices even allow you to download complete measurement personalities (for CDMA, GSM, or Wi-Fi, for example) into the instrument and perform specific measurements with one command. The ability to control an instrument through its browser interface opens up a realm of new possibilities for test engineers who need a simple way to access test systems from virtually anywhere in the world.

Trigging and synchronization

One especially intriguing aspect of LXI is its triggering and synchronization capabilities. LXI provides a variety of optional triggering modes that are not available in GPIB, PXI or VXI.

The three classes of LXI devices implement these capabilities to an increasing degree. As an example of what is possible with LXI, all Class B and A LXI instruments (optional in Class C) can utilize triggers embedded in LAN packets that can originate from any device on the network—a PC or another instrument. One device can send a multicast message that triggers all instruments on the network without the need for a real-time computer. Peer-to-peer messages can enable measurement scripts or cause data to be passed from one LXI device to another without involving the system's host computer (a potential communication bottleneck).

Figure 16.5. LXI specifies an informative instrument page that can be accessed with a standard Web browser



Exploring new possibilities with LXI

LXI-compliant devices open up a number of useful new possibilities that are difficult—and in some cases impossible—to implement with traditional rack-and-stack or cardcage systems. The following examples are not meant to define the complete set of possibilities: they are simply an initial set of concepts that will grow as the use of LXI spreads.

Easier transitions

One of the biggest challenges in a new product's lifecycle is the transition of its test system from development to manufacturing. With LXI, this transition can be achieved much more easily and cost effectively than with cardcage-based systems.

As an example scenario, engineers can utilize standard instruments during the R&D phase, using the display and keypad to quickly access a wealth of measurement and analysis capabilities. In manufacturing, a system containing the same LXI instrument in a faceless, modular form can use the software and test routines developed with the standalone instrument. Unlike VXI or PXI, this ensures instrument-equivalent precision and performance while also eliminating the overhead of a cardcage and proprietary interface.

Enhanced throughput

The flexibility of LXI provides two ways to boost system throughput. In one scenario, software routines can be run within the LXI module, perhaps performing basic analysis functions and simply passing results (rather than data blocks) to the host PC. If necessary, advanced routines can be run in the PC, which will typically have greater computational power than most LXI modules. In the other scenario, peer-to-peer communication between LXI modules can be used to coordinate their activities, eliminating bottlenecks that could occur if all messages were handled by the host PC.

Multi-site collaboration

When a geographically distributed team is working on a one-of-a-kind prototype, LXI makes it possible for team members to make measurements from their desk, wherever it may be. To help ensure system security, standard security procedures can be implemented, such as using a firewall and virtual private network (VPN). Typically, most LXI devices will be part of a system that has a dedicated LAN, but remote users can gain secure access to the system through its host PC.

Synchronized systems

With the timing capabilities of Class A and B LXI devices, it's possible to synchronize multiple systems within a building, between sites or around the world. This is enabled by IEEE 1588, which has the ability to achieve millisecond accuracy among devices located anywhere on the network. Possible applications include trend and cause-and-effect analyses driven by data from multiple instruments or systems. By time stamping all of the data, it can then be correlated and analyzed in one or more computers to identify trends or cause-and-effect relationships.

Distributed testing

Current-generation systems use a PC-centric approach in which the computer controls basic instruments and "dumb" devices. The PC sends commands and uses wait statements or queries to determine when an operation is complete—and all data returns to the PC through a dedicated I/O port. This is fine for small systems but can become slow and inefficient in larger systems that use four or more instruments. While the speed of the I/O connection plays a role, successful operation requires a skillful programmer who can manage the flow of both control and data.

Next-generation systems, as embodied in LXI, make it possible to apply a distributed approach that utilizes the intelligence of the instruments. Much of the analysis and synchronization can be performed in the measurement hardware, offloading these chores from the PC. Data flow is reduced because only the results of the analysis are sent to the PC. Timing is simplified with LXI Class B and A devices that can start their activities at a specific time or based on messages from other instruments. Instruments also can exchange information using peer-to-peer and multicast messaging. With this architecture, the PC and its I/O path are less likely to become bottlenecks in large, complex systems.

Long-distance operations

Through the LAN interface, LXI makes it possible to place instruments far from the PC and from each other. As an example, instruments can be placed near the devices or processes they are monitoring or controlling—and be connected to existing LAN ports in a test lab or near a manufacturing line. LXI devices can even be placed inside a text fixture, minimizing cable runs and enhancing measurement results.

Expert troubleshooting

Whether a system is located in the next room, the building next door or a site halfway around the world, your system developer (or product expert) can check its operation and troubleshoot problems. No travel is required: simply type an instrument's URL or IP address into a standard Web browser and the instrument page will appear.

Intelligent instruments

Without the size restrictions of VXI and PXI, LXI enables use of intelligent instruments within a system. You can download measurement personalities into a spectrum analyzer, sophisticated signals into an arbitrary waveform generator or complex power sequences into a programmable dc supply—and let the instrument handle the details. The capabilities built into these instruments help you save time, too.

You can reduce programming time by taking advantage of the software (and firmware) developed by the vendor rather than writing it yourself. Instrument set-up time can be reduced by creating configurations in advance and recalling them as needed. Data transfers take less time because the instrument can make measurements, perform the required analysis and then send results—not large data blocks—to the host PC.

Rapid reconfiguration

LXI-based synthetic instruments reduce system size and cost by utilizing multi-purpose modules—digitizers, waveform generators, upconverters, downconverters and more—that can be combined to create traditional instruments such as spectrum analyzers, signal generators and oscilloscopes. These fundamental building blocks depend on PC software that dynamically aggregates and "synthesizes" different measurement tasks.

As an example, an RF downconverter LXI module could be used for spectral measurements in one test sequence and then be reconfigured for network measurements in another. To create the stimulus signal for network analysis, simply adding a different LXI upconverter makes it easy to change the output frequency range without having to purchase an entirely new signal generator. Reducing the redundancy—and increasing the utilization—of these fundamental hardware elements helps trim the size and cost of test systems (Figure 16.6).

Synthetic instruments

In addition to the attributes mentioned earlier, SIs create two additional possibilities. SI hardware and software modules can be used to emulate obsolete instruments, removing the burden (and cost) of supporting outdated equipment in long-lived systems. SIs also make it possible to create and perform totally unique measurements that are not currently possible with traditional instruments.

Peer-to-peer triggering

By making it possible for one instrument or device to send triggers and information to another, LXI frees up the PC to perform other, higher-level tasks. Peer-to-peer triggering also eliminates the need for an expensive real-time controller to issue precise triggers to the instruments in a system. Ultimately, overall test time can be reduced because techniques such as wait states and status queries will be used less often in system software.

Time-based triggering

With IEEE 1588, time-based triggering may prove to be a revolutionary way to synchronize measurements within systems and between instruments. For example, this method eliminates the need for trigger-specific external cabling so is not limited by the distance between instruments. All measured data can be time stamped, making post-test analysis easier, more efficient and more meaningful. System throughput also increases because each instrument can start at a specific time rather than waiting for a trigger or command.

Figure 16.6. LXI devices reduce the size and footprint of test systems

Appendix 16A: Defining synthetic instruments

In the mid 1990s, the U.S. Department of Defense (DoD) assigned the U.S. Navy the task of developing new types of automated test systems (ATS) for the testing of avionics and weapons systems in the factory, on the front lines and anywhere in between. The project has six driving goals:

- Reduce the total cost of ownership of ATS
- Reduce the time to develop and deploy new or upgraded ATS
- Reduce the physical footprint of each system
- Reduce the logistics footprint via decreased spares, support systems and training
- Provide greater flexibility through systems that are interoperable among U.S. and allied services
- Improve the overall quality of testing

These are ambitious goals but equipment manufacturers, defense contractors and the DoD believe they can be achieved over time by applying advances in commercial technologies. The greatest progress toward these goals may come from the use of synthetic instruments (SI). According to the Synthetic Instruments Working Group[3], a synthetic instrument is a reconfigurable system that links a series of elemental hardware and software components via standardized interfaces to generate signals or make measurements using numeric processing techniques. The key word is reconfigurable: the elemental blocks can be arranged and rearranged via software commands to emulate one or more traditional pieces of test equipment.

To make it work, an SI contains as many as four major components: signal conditioners, frequency converters, data converters and numeric processors. The basic block diagram shown in Figure 16.7 describes most microwave instruments, including spectrum analyzers, frequency counters, network analyzers and signal generators.

Unlike general purpose instruments, which are optimized to perform one task (e.g., spectrum analysis or signal generation), the synthetic instrument architecture is optimized to provide greater efficiency in ATS by reducing redundant elements such as the digitizers and downconverters found in multiple instruments within a system. The DoD expects these SI modules to come from a variety of vendors, enabling easy mixing and matching as requirements change or modules become obsolete. What's more, any substitution of modules should require only minimal changes to the core system software.

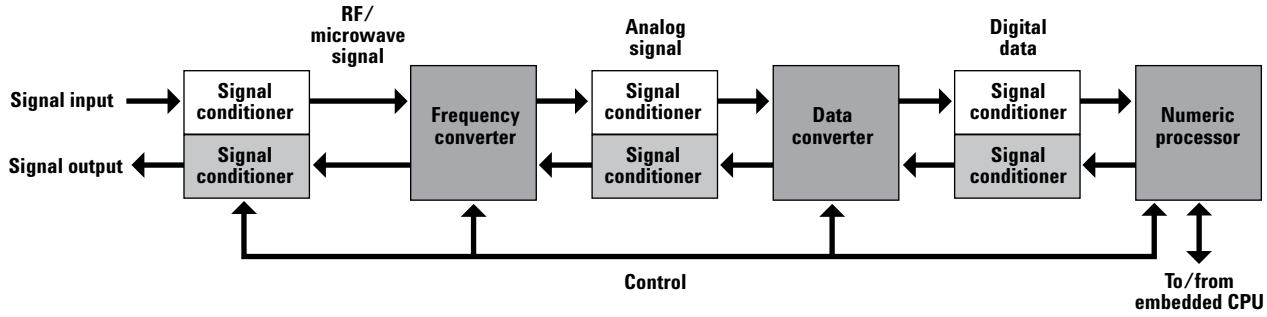
Although this approach can be applied to any type of instrument, it is especially well suited to RF instruments. As an example, an RF vector signal analyzer can be broken down into a downconverter, a digitizer and the associated analysis software. Similarly, an RF signal generator can be reduced down to its elementary building blocks. By creating these building blocks as distinct hardware modules and using software to control their arrangement and configuration, it becomes possible to create the functional equivalent of multiple instruments with a handful of modules.

This approach can also reduce the cost of system updates. Because different types of building blocks are based on different technologies, they have different innovation cycles. For example, downconverters contain relatively stable technology but, in contrast, rapid advances in integrated circuit technology accelerate improvements in digitizer speed and resolution. With SIs, it should be less costly to keep up with the latest advances.

Not surprisingly, LXI is becoming the preferred technology for synthetic instruments. At the building-block level, communication between these components becomes a critical factor. Rather than relying on custom, instrument-internal communication schemes, Ethernet offers the simultaneous benefits of excellent data rates and the flexibility of peer-to-peer and concurrent communication via TCP/IP.

3 Includes joint participation of the DoD, prime contractors and suppliers.

Figure 16.7. Basic architecture of an RF/microwave synthetic instrument

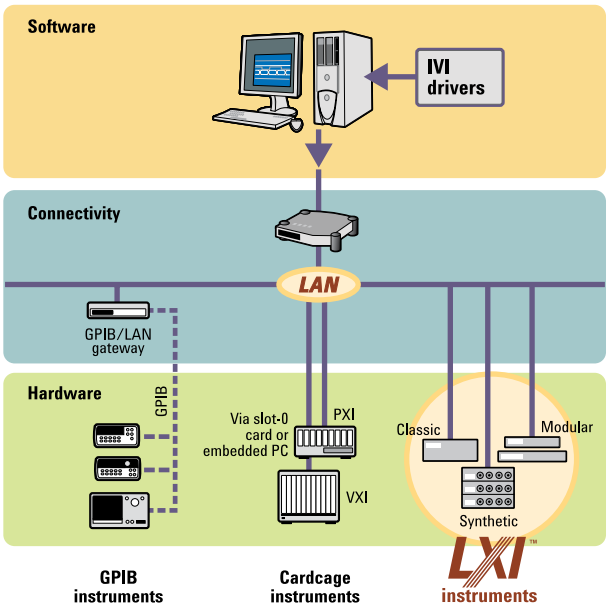


Appendix 16B: Creating cost-effective measurement solutions with Agilent Open to test your way

LXI solves the key problems faced by system developers: it cuts costs, reduces system size, simplifies integration, accelerates throughput and provides more opportunities for reuse of both hardware and software. These benefits make LXI a test architecture for today and into the future.

To help you fully realize these benefits, we've adopted LXI as part of the Agilent Open concept (Figure 16.8). Your test system architecture should give you choices. Its range of possibilities should fit your requirements, your preferences and your existing test assets—hardware, software and I/O. This is the power of Agilent Open, a combination of proven standards and time-saving tools for test automation. By giving you greater flexibility, Agilent Open accelerates the creation of cost-effective measurement solutions—and enables testing, your way.

Figure 16.8. Through Agilent Open and LXI, LAN becomes the backbone of test systems that easily incorporate present and future test assets.



Create versatile solutions with system-ready instrumentation

Agilent Open instruments are designed for faster throughput and easier integration—in test software and system racks. Choose classic benchtop instruments for R&D then use their modular, system-optimized equivalents in manufacturing—and run the same software with few or no changes. To reduce software development time, many instruments let you install measurement personalities that address specific applications, including Mobile WiMAX, jitter, phase noise and more.

Simplify system communication and connectivity

Choose the best connection for your requirements through instruments equipped with GPIB, LAN and USB ports. You can connect those instruments quickly and easily with the Agilent IO Libraries Suite software, which supports the major test-system interfaces—GPIB, LAN, USB, VXI and RS-232C. With support for LXI, you can control instruments and monitor measurements remotely via the Web servers built into Agilent Open instruments.

Achieve efficient development with open software tools

Configure a typical system in less than 15 minutes with the Agilent IO Libraries Suite, which supports literally thousands of instruments from hundreds of vendors. Get your systems up and running sooner with industry-standard IVI drivers that put instrument functionality at your fingertips—and work in the application development environment you prefer.

Develop hybrid systems that combine new and existing assets

Protect your existing assets by easily integrating GPIB instruments into LAN- and USB-based systems with Agilent interface gateways and converters. You can even add VXI and PXI equipment to LAN-based systems via LAN slot-0 cards. Utilize the multiple I/O ports of Agilent Open instruments to connect via GPIB now and LAN or USB in the future. Using VISA, making the transition from GPIB to LAN or USB requires nothing more than simple address changes in your system software.

Agilent is leading the way in migrating test systems to the advanced capabilities of LAN. We're continually introducing new additions to what is currently the industry's largest portfolio of LAN-enabled instruments. At the same time, we're also protecting your investment in GPIB instruments by offering devices such as the Agilent E5810A LAN/GPIB gateway and the 82357A USB/GPIB interface.

To discover more ways to accelerate system development, simplify system integration and apply the advantages of open connectivity, please visit the Agilent Open Web site at **www.agilent.com/find/open**. Once you're there, you can also sign up for early delivery of future application notes in this series.

17. Transitioning from GPIB to LXI

Introduction

More than 30 years after its creation, GPIB remains popular due to its ease-of-use and robustness. However, LXI (see Chapter 16) meets or beats GPIB on both counts—and offers an array of other compelling benefits. From browser-based configuration and troubleshooting to Ethernet's own 30-year history, LXI enables fast, efficient and cost-effective creation and reconfiguration of test systems. By specifying the interaction of proven, widely used standards, LXI helps you conquer the challenges of product testing without overloading your budget or your team.

This chapter compares GPIB and LXI, sketches hybrid system architectures, outlines a step-by-step approach to system set-up, and describes how to easily modify existing system software to work with LXI devices.

Comparing system architectures

Every test system depends on four basic elements: measurement hardware, system software, PC-to-instrument connectivity and cabling to the device under test (DUT). As you consider the transition from GPIB to LXI, it's worthwhile to consider the effects on all four aspects as your preferred system structure evolves from pure GPIB to GPIB/LAN hybrids to all LAN/LXI.

A typical GPIB system

After more than three decades of widespread use, the basic structure of a GPIB-based system is almost second nature to engineers everywhere: it includes a controller (typically a PC) configured with a GPIB card and up to 14 rack-and-stack instruments daisy-chained together with GPIB cables. The controller and instruments are usually located within a few meters of each other due to the length constraints on GPIB communication (although longer distances are possible with GPIB bus extenders)

Advantages. From a hardware perspective, GPIB instruments are readily available from either your internal equipment pool or Agilent and numerous other vendors. These highly specialized devices tend to be long-lived because they are generally immune to changes in the system controller. They also include proven measurement routines that provide accurate, reliable and repeatable results. What's more, many of the latest instruments offer enhanced flexibility through downloadable personalities, which provide specialized measurements for applications such as wireless communications.

Connectivity is simple and well-understood with GPIB—and a 30-year history stands as testament to its proven dependability. Routine programming is also relatively simple, whether you choose to use Standard Commands for Programmable Instruments (SCPI), Interchangeable Virtual Instruments (IVI) drivers or some other type of drivers for communication and control.

Disadvantages. The biggest disadvantage may be the need to add a GPIB card to the host PC, increasing both the cost and complexity of the system. This can be especially problematic with notebook computers that need an adapter for the PC Card slot or an available I/O port (e.g., a USB-to-GPIB adapter). Troubleshooting the GPIB card and the associated I/O libraries may take a considerable amount of time. Once everything is up and running, communication may be slower than is possible with LAN and other alternatives.

Depending on your test requirements, GPIB instruments may consume a lot of rack space and add redundant or unnecessary capabilities (e.g., multiple display screens). In some cases, 14 instruments may not be enough to fully test your product. With large multi-instrument systems, bulky GPIB cables and their large connectors can be difficult to route and dress within the confines of a system rack.

System programming has its own challenges, starting with the basic task of tracking down useful, reliable drivers for every instrument in the system. GPIB systems generally require additional trigger lines that you must connect between instruments and then activate via software commands. Timing and synchronization within a system can also complicate programming because GPIB doesn't provide a common clock or trigger line.

Typical LAN-based systems

Making the transition to LXI doesn't require sweeping changes to your system architecture. Instead, a variety of evolutionary system structures are possible when using LAN communication along with LAN-enabled and LXI-compliant instruments.

In all cases, the starting point is a PC with a built-in LAN port: unlike GPIB-based systems, the PC needs no physical modifications. However, the system structure does require the addition of a switch or gateway (external to the PC) to enable communication with multiple instruments.

Scenario 1—GPIB-to-LAN. The easiest initial transition is to use LAN to communicate with an existing GPIB system. A device such as the Agilent E5810A LAN/GPIB gateway (see Figure 17.1) enables remote access to GPIB instruments via LAN—and eliminates the need to install a GPIB card in the PC. Addressing remains the same: your system software will see the gateway device as a GPIB interface even though it communicates via LAN. Because your instruments still look like GPIB devices, you can transition your system without changing its software. The E5810A gateway can be mounted in the system rack, which, with a LAN connection, is freed from the distance constraints of GPIB.

Scenario 2—GPIB plus LAN. A typical next step in the transition to LXI is the addition of a LAN router between the PC and the LAN/GPIB gateway (see Figure 17.2). This makes it possible to incorporate GPIB, LAN and LXI equipment into a single system by connecting GPIB instruments to a LAN/GPIB gateway and then connecting the gateway and any LAN or LXI instruments to the router.

.One important point is worth remembering: although many test instruments are equipped with LAN ports, not all can be controlled via LAN. Some use the LAN port only to communicate with external peripherals—so it's best to check the product manual or built-in help function to verify LAN-specific capabilities. Of course, if an instrument carries the LXI logo, it is has passed compliance testing and, at a minimum, can be controlled via LAN, has a browser-accessible Web interface, is provided with an IVI driver and meets LXI's physical specifications.

Scenario 3—All LAN. LXI-based products are becoming widely available so it is now possible to evolve to an all-LAN structure. These systems will include one or more LAN routers as needed to accommodate all of the LXI instruments (see Figure 17.3). Every instrument will be able to take advantage of LAN's speed while utilizing low-cost I/O cabling. The browser-based interface within every LXI instrument will help speed and simplify instrument—and system—configuration and troubleshooting. The long reach of LAN and the synchronization made possible by the IEEE 1588 precision time protocol (PTP) will enable a variety of new capabilities and applications. (Please refer to Chapter 10, Using LAN in Test Systems: Network Configuration and Basic Security, for advice on setting up a private, protected measurement network using either a router-based or PC-based approach.)

Please refer to Chapter 16, Value, Performance and Flexibility: the Promise of LXI, for a closer look at the fundamental concepts and advantages of LXI.

Figure 17.1. A LAN/GPIB gateway can connect GPIB instruments to a PC's LAN port

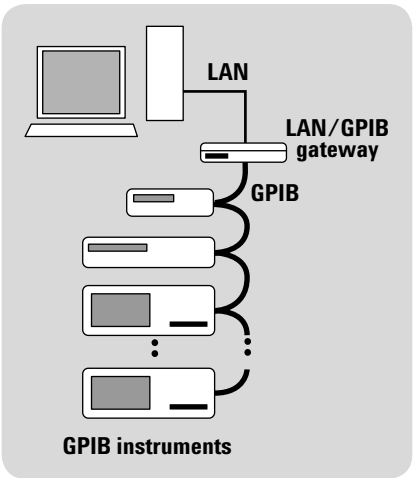


Figure 17.2. A router plus a LAN/GPIB gateway enables connections of GPIB, LAN and LXI instruments to a PC's LAN port

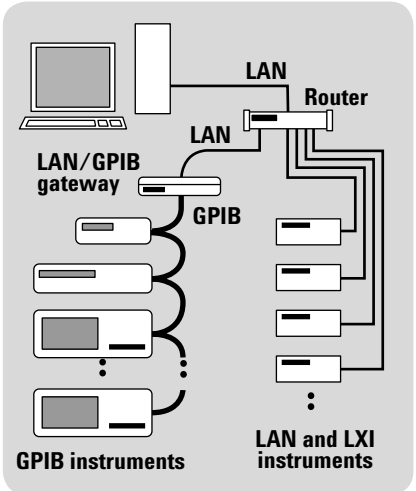
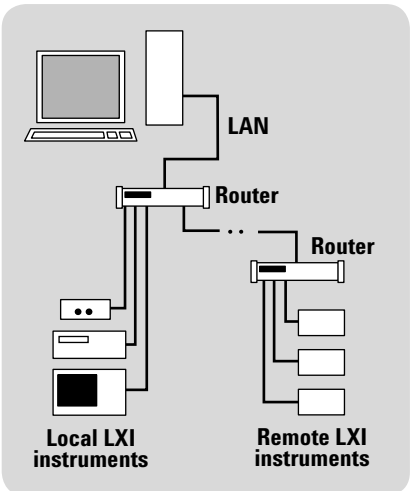


Figure 17.3. Using multiple routers enables connection of local and remote LXI instruments to a PC's LAN port.



Setting up an LXI system

A demonstration is the best way to see how quickly and easily you can configure a LAN-based system. To provide a virtual demo, the following step-by-step procedure outlines the suggested actions and tools that will simplify system configuration during initial setup and future changes.

Step 1: Connect LAN cables

The first step is to connect all of the instruments to the necessary LAN hardware (router, etc.) using standard LAN cables. Next, connect the router to the system's host PC.

Step 2: Insert CD into PC

Install the Agilent IO Libraries Suite onto the host PC. Provided free with Agilent instruments, the IO Libraries Suite works automatically with both Agilent and NI interfaces (it is fully compatible with NI-488).

It typically takes less than 15 minutes to load the IO Libraries Suite and run the configuration tools. To simplify configuration, the software recognizes other installed libraries such as NI VISA and configures itself in a compatible manner.

Step 3: Identify PC interfaces

The Connection Expert, one of the key tools in the IO Libraries Suite (see Figure 17.4), identifies and configures the various interfaces within the PC—LAN, USB, GPIB and serial (COM). It starts by automatically recognizing the manufacturer, model number and serial number of installed or attached interface cards and converters. Connection Expert completes this step by configuring the appropriate I/O libraries for each interface and converter.

Step 4: Identify connected instruments

Connection Expert can find and identify instruments from dozens of vendors then help you configure them appropriately. One click on any instrument reveals information such as manufacturer, model number, serial number and IP address (or URL). The IO libraries communicates with the instrument to find this information.

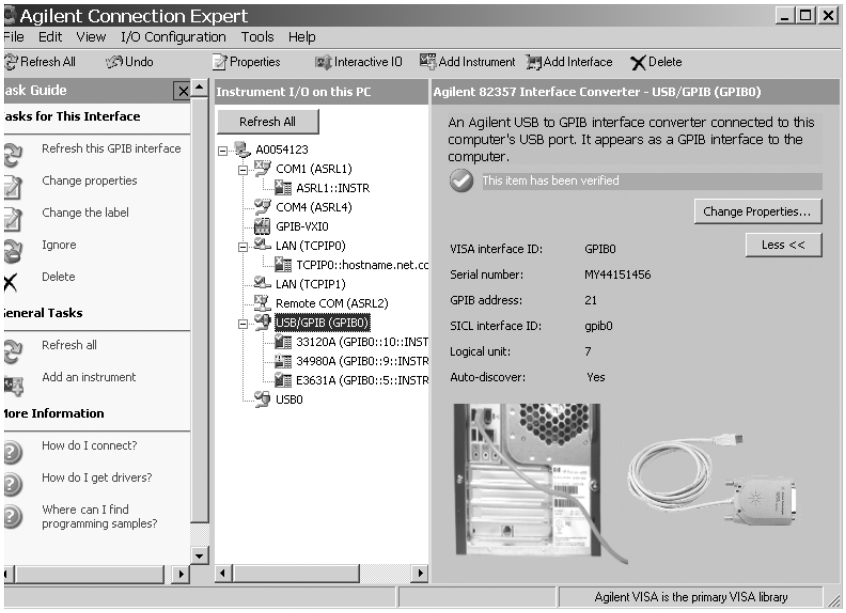
Step 5: Test communication links

If you'd like, Connection Expert can automatically test the communication link—LAN, USB or GPIB—with every connected instrument identified in Step 4.

Step 6: Configure LXI instruments

Start any Web browser, type in an LXI instrument's IP address or URL, and view the built-in instrument page. As defined by the LXI Standard, an instrument page includes information such as manufacturer, model number, serial number, firmware revision code and instrument IP address. LXI instruments also provide a configuration page that lets you adjust LAN settings through the Web interface. Agilent instrument pages generally include a product photo and links to additional information. "Intelligent" instruments can also use this page to download firmware revisions or measurement personalities.

Figure 17.4. The Agilent Connection Expert simplifies the configuration of PC-to-instrument I/O.



Many Agilent instruments provide additional built-in pages that let you interact with the instrument and perform various tasks: make measurements, generate signals, close channels, read values and display results (see Figure 17.5). Some will also let you try program commands or command sequences and verify the instrument's response.

Agilent IO Libraries Suite, Agilent Connection Expert and the LXI browser interface are a powerful combination that can reduce set up time from days to minutes. Best of all, the IO Libraries Suite and Connection Expert are designed to work with instruments from virtually every manufacturer. Agilent customers can download the IO Libraries Suite at no charge from **www.agilent.com/find/open.**

Simplifying software changes

Making the transition to LXI doesn't require sweeping changes to your system architecture or your system software. Four important items can simplify the process of modifying your system software to communicate with LXI-compliant devices.

Dual-interface instruments

Many of Agilent's GPIB instruments that are updated to LXI compliance will have both LAN and GPIB ports. These instruments can be used via the GPIB port without modifying your existing software or through the LAN port with a simple address change in your program. For smaller programs, you can change addresses from GPIB to IP either manually or via search and replace. For larger programs or test suites, you can modify the device declarations within the program. You can also use Agilent IO Libraries Suite, which allows a table of aliases. This approach may cause slightly slower communication but provides a convenient way to get your system up and running.

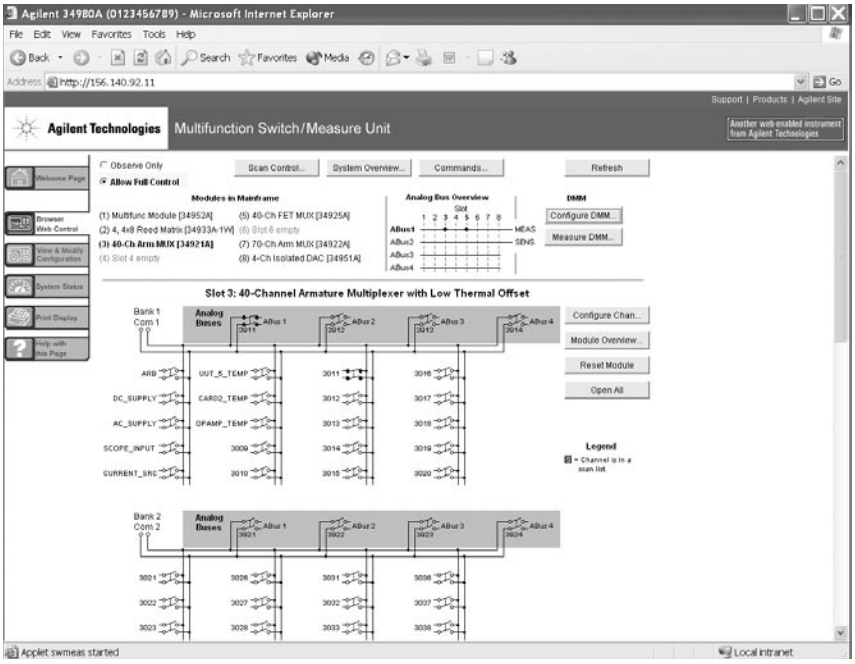
VXI-11

From the perspective of a PC application, many instruments implement the VXI-11 communication protocol that makes LAN I/O look just like a GPIB connection. In practice, this means software written for GPIB is likely to work with identical LAN-enabled instruments that implement VXI-11.

Command-compatible instruments

Instruments that are command-compatible with older instruments can be used with your existing software. As an example, the Agilent 34410A and 34411A digital multimeters have a compatibility mode that mimics the widely used 34401A or E1412A DMMs. The LXI-compliant 34410A and 34411A, equipped with both GPIB and LAN interfaces, can replace a 34401A in a system with either no changes (GPIB) or minor changes (IP addressing for LAN) to the system software.

Figure 17.5. Many of Agilent's LXI instruments include built-in web pages that let you configure the device and make measurements. For example, this page on the 34980A multifunction switch/measure unit makes it easy to configure each module in the system..



Drivers

Many manufacturers are modifying existing instruments to achieve Class C LXI compliance. In most of these cases, the existing driver should work correctly even if you switch to the LXI version of the same model. The IVI drivers required by the LXI standard support another possible solution: “class drivers,” which enable instrument substitution. Instruments within a specific class, such as the DMM class, for example, can be substituted for each other within a test system. The one caveat is the possibility of different results if, for example, you substitute a class-compatible 4½-digit DMM for a 6½-digit model.

Conclusion

GPIB has served the test and measurement community well for decades and will continue to be an important asset for years to come. However, LXI provides not only the ease-of-use and robustness of GPIB but also includes capabilities such as browser-based configuration and troubleshooting that enable fast, efficient and cost-effective creation and reconfiguration of test systems.

With a series of incremental changes to your system architecture and software, you can make the shift to the speed, distance and performance advantages of LAN and LXI.

18. Creating Hybrid Test Systems with PXI, VXI and LXI

Introduction

It's common to associate certain types of instrumentation with the stages of your product's lifecycle. For example, benchtop instruments are often used in R&D because they enable interactive control of specific measurements and provide rapid feedback. As your product moves to manufacturing, modular solutions such as PXI or VXI are sometimes used because they can reduce the size of automated test systems.

Unfortunately, in the transition from benchtop to modular instruments, the lack of leverage—in hardware, software and test strategy—can be costly and time consuming. LXI offers the potential to change this situation by offering related or identical products in multiple form factors (including benchtop, modular, and synthetic instruments) and making it easier to leverage your existing test strategy and system software across your product's lifecycle.

This chapter compares PXI and VXI with LXI, sketches hybrid system architectures that incorporate your existing test assets and describes what will be possible in the future as you migrate to fully LXI-based systems.

Assessing modular systems

Both PXI and VXI require several discrete elements—a mainframe, plug-in cards, I/O, PC and software—to create the functionality of one standalone instrument or perhaps a complete rack-and-stack system (Figure 18.1). Achieving equivalent measurement and analysis capabilities requires that large amounts of data be moved within the mainframe (or chassis), various plug-in cards and the host PC. Some amount of programming, by the end user or a system integrator, is typically required to achieve the needed level of functionality. The resulting software application provides the user interface as well as most (if not all) of the measurement capabilities, data displays and data analysis routines.

Making all of that work with acceptable performance often requires a powerful PC that can process and analyze measured data while also controlling the hardware and providing the user interface. When an external PC is used as the system host, it will require the installation and configuration of an interface card. When an embedded controller is used, this may require a larger mainframe that can accommodate the controller and the various plug-in cards. While this approach eliminates the interface to an external PC, it still requires that a monitor, keyboard and mouse be connected to the embedded PC.

Although this somewhat complicated approach has become popular in certain applications, it is not a universally useful solution, and it has advantages and disadvantages that are worth a closer look.

Figure 18.1. A typical VXI system

Advantages of PXI and VXI

Both PXI and VXI have useful advantages in hardware, connectivity and programming when compared to rack-and-stack systems.

Hardware. One key advantage is the density of switching, sourcing and measuring capabilities that can be packed into a single mainframe. PXI and VXI will usually be smaller than a rack-and-stack system with similar functionality. PXI and VXI also have an edge over rack-and-stack in triggering and synchronization, thanks to their high-speed backplanes and included triggering capabilities.

Connectivity. PXI and VXI offer a variety of I/O alternatives: MXI, GPIB, LAN, USB, FireWire and serial. This allows you to make case-by-case tradeoffs between performance and convenience.

Programming. System creators can use graphical or text-based development environments to create the required measurement and analysis functionality. While it can be difficult to work with register-based PXI and VXI plug-in cards, the use of device drivers can greatly simplify communication and programming (see "Programming register-based devices"). The resulting measurement solution may be smaller and faster than an equivalent rack-and-stack system built with benchtop GPIB instruments.

Disadvantages of PXI and VXI

Both PXI and VXI have shortcomings that can affect your ability to create a system that fully satisfies the budgetary, technical and lifecycle requirements of a test specification.

System host. Unlike a typical rack-and-stack system, a PXI- or VXI-based system can be heavily dependent on the performance of the host PC—and higher performance commands a higher price. What's more, the PC-dependent approach does not scale well for large, complicated systems: as more modules move more data more often, the PC can become a processing bottleneck that slows overall system performance.

Embedded controllers come with their own set of shortcomings. Because these are a specialty item produced in limited quantities, they typically cost three to eight times as much as an equivalent desktop PC. They also tend to lag behind the latest advances in performance and capabilities.

Hardware. In addition to the high entry cost of PXI and VXI, you may need to buy a mainframe that has more slots than needed if you want to allow for future expansion. Once a mainframe is filled, there is also the potential cost of adding another mainframe if the system needs just one more plug-in card.

When the required functionality isn't available in a modular format it will be necessary to add bench-type instruments to the system. Examples include many RF measurements as well as high-wattage power supplies. The inclusion of standalone instruments can increase the complexity of both system integration and programming. It may also negate the size advantages of VXI or PXI.

Programming register-based devices

Because PXI and VXI are leveraged from computer buses (PCI and VME), their plug-in cards usually depend on register-based operations to read or set attributes, initiate measurements, load or unload data, and so on. While this type of low-level programming enables detailed computer control of each module, it can be quite complicated and time consuming.

One popular solution is device drivers, which handle the low-level details and enable programming at a higher level. The best choice of driver depends on the type of hardware or software being used. For example, National Instruments uses IVI-G drivers with LabVIEW and IVI-C drivers with LabWindows. While IVI-C and IVI-G drivers are available for many Agilent instruments, Agilent and others have provided IVI-COM drivers. These are language- and platform-neutral and one version will work in all Microsoft® COM (and compatible) environments, and with Microsoft Excel. Not only does this provide additional flexibility because you can work in your preferred development environment, but it also can enhance your productivity through features such as IntelliSense pop-up menus that provide onscreen command-completion help.

Connectivity. Using either MXI or GPIB as the interface adds hardware cost and configuration complexity to an external host PC.

Programming. Because most PXI and VXI devices lack any sort of built-in user interface—front panel or browser-based—you typically have to purchase, install and configure some type of software to control even the simplest device. Additional programming may be required to perform a measurement, manipulate the data and analyze the results. What's more, T&M-specific software that provides these capabilities tends to be much more expensive than commercial programming environments available from Microsoft and other vendors.

Cost. Both PXI and VXI incur a large overhead cost in the mainframe, controller, connectors and I/O subsystem. PXI and VXI both require a considerable investment before the first module can be used. Additionally, the per-slot overhead costs can be prohibitive when the PXI or VXI mainframe contains low-cost modules such as digital IO, DACs, or simple switching.

Exploring LAN-based hybrid systems

As earlier chapters have noted, LAN is rapidly gaining favor as the interface of choice for automated test systems. While the earliest LAN-enabled instruments offered inconsistent implementations of the interface, the LXI standard now ensures a consistent approach that makes it possible to use compliant instruments from multiple vendors. Chapter 16 offers a closer look at LXI.

In most cases, it is relatively straightforward to create hybrid systems that utilize LXI devices alongside GPIB, PXI and VXI hardware. A hybrid structure lets you harness the advantages of each architecture within a single system. In addition to saving money by protecting your existing investments in test assets, this approach also helps you save time because you can continue using familiar hardware, interfaces and software.

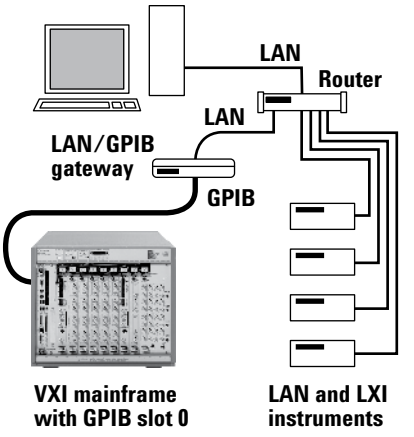
A typical LXI-based system starts with a host PC and its built-in LAN port, which provides a connection to local and remote LXI-based devices through commercially available LAN switches or routers. This is also the starting point for hybrid configurations that include LXI devices working alongside a VXI or PXI mainframe. Today, four possible scenarios are likely and feasible.

Scenario 1: VXI and GPIB

If a VXI mainframe contains a GPIB slot-0 card, it can be connected to the PC via LAN by adding an interface converter such as the Agilent E5810A LAN/GPIB gateway (Figure 18.2). With the gateway connected between the VXI mainframe (GPIB) and the router (LAN), any application running on the PC will be able communicate transparently with the VXI hardware as GPIB devices.

- **Advantages.** This hybrid structure eliminates the need to install a GPIB card in the PC. With the gateway, addressing can be kept the same so no software changes will be required.
- **Disadvantages.** System performance may decrease if the gateway cannot keep pace with the demands of any high performance measurement cards installed in the VXI mainframe.

Figure 18.2. With a router and a LAN/GPIB gateway, test software on the PC can communicate transparently with VXI instruments, as if they were GPIB devices.



Scenario 2: VXI and LAN

When a VXI mainframe is equipped with a LAN slot-0 card, adding it to the system network is as simple as connecting it to the LAN router (Figure 18.3). Even if the LAN-equipped VXI system is not LXI compliant, it can coexist on the network with any LXI devices.

- **Advantages.** Every instrument in the system—LXI or VXI—can utilize LAN's I/O speed. If the system software is already programmed to communicate with the VXI hardware via LAN, addressing should remain the same so few or no software changes will be required. Any required programming changes should be relatively modest, even when you replace an MXI or FireWire slot-0 card with a LAN slot-0 card.1
- **Disadvantages.** Depending on the devices installed, this configuration may provide less performance than a purely backplane-based system (e.g., one that uses an MXI interface) but should be faster than the LAN/GPIB configuration described in Scenario 1.

1 One example is the VXI Technology EX2500 LXI-VXI Gigabit Ethernet Slot-0 Interface.

Scenario 3: Embedded controller

If a PXI- or VXI-based system is using an embedded controller within the mainframe, the controller can be connected to the test-system network through its built-in LAN port. The PXI or VXI portion of the system would still be controlled by the existing software running on the embedded PC. To simplify the overall system structure, the existing software could be modified to control the LXI devices, eliminating the need for an external PC that controls only LXI devices (Figure 18.4).

- **Advantages.** This is a straightforward way to add the advantages of LXI to a PXI- or VXI-based system. If suitable modular LXI devices are available to provide functionality that isn't available in PXI or VXI formats, the resulting system may also be simpler and more compact than one that uses GPIB instruments.
- **Disadvantages.** Modifying the existing software to control the LXI devices could hinder system performance by putting an additional burden on the embedded PC; however, this may have a modest impact given the built-in intelligence of most LXI devices. This system structure also requires the addition of a LAN router, which will cause a slight increase in system cost and complexity.

Figure 18.3. Adding a LAN slot-0 card to a VXI mainframe lets you create a LAN-based hybrid VXI/LXI system..

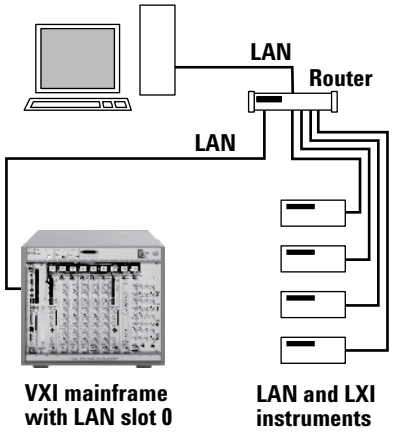
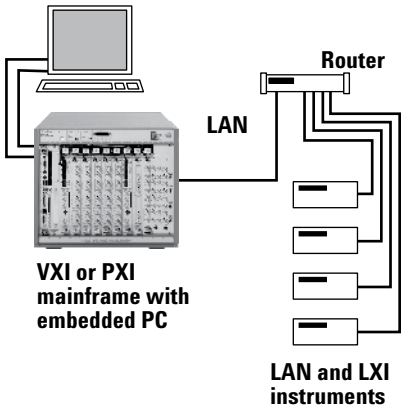


Figure 18.4. An embedded PC with a LAN port can be used as the system controller in a hybrid VXI/LXI or PXI/LXI system.



Scenario 4: LXI-compliant mainframe

Some manufacturers of PXI-based instrumentation are actively supporting the LXI standard. To ease the transition from PXI to LXI, at least one vendor has created Class C LXI-compliant mainframes that support a wide variety of switching modules.2 With this approach, you simply install the switching cards in an LXI-compliant mainframe equipped with a PXI slot-1 interface, which is connected (through its LAN port) to the system router (Figure 18.5).

- **Advantages.** This solution provides the advantages of existing PXI switching cards, including high density and a variety of capabilities, within an LXI-based system. For new systems, this approach is also likely to be less expensive than an all-PXI solution that uses either an embedded controller or a PC-to-PXI interface.
- **Disadvantages.** Currently, this approach is supported only for PXI switching cards. Future developments may make it possible to support the demands of register-based PXI measurement cards.

All four of these scenarios enable a cost-effective transition that protects your current investments in system hardware and software. However, these hybrid structures also entail compromises that may be most easily remedied in the future with a LAN-centric, all-LXI system architecture.

Going beyond hybrid to all-LXI

As an alternative to PXI or VXI, LXI eliminates the overhead and complexity of system development with a backplane. When using benchtop and modular LXI instruments to create a system, the approach is conceptually similar to using GPIB instruments: each device contains built-in measurement functionality (and intelligence) and provides specified measurement accuracy. However, LXI adds triggering and synchronization capabilities that go beyond GPIB—and can rival or exceed PXI or VXI. With these capabilities built into LXI instruments, your programming effort can focus on test management and the management, analysis and reporting of results.3

As more LXI-based products become available, it will be possible to evolve to an all-LAN structure. These systems will include one or more LAN routers as needed to accommodate local and remote LXI instruments (Figure 18.6). Every instrument will be able to take advantage of LAN's speed while utilizing low-cost network cabling. The browser-based interface within every LXI instrument will help speed and simplify instrument or system configuration and troubleshooting. The long reach of LAN and the synchronization made possible by the IEEE 1588 precision timing protocol will enable a variety of new capabilities and applications.

Figure 18.5. An LXI-compliant mainframe brings the benefits of PXI switching to a hybrid PXI/LXI system.

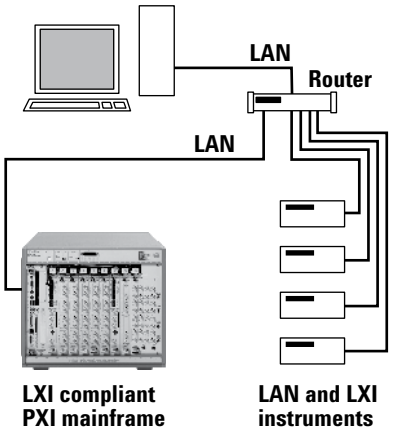
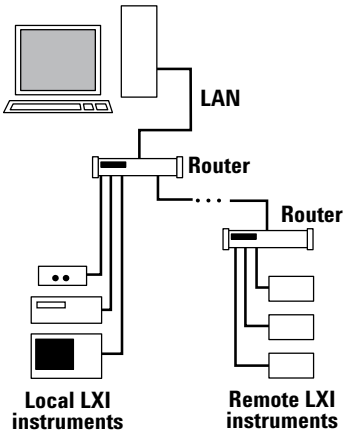


Figure 18.6. Using multiple routers enables easy connection of local and remote LXI devices to a PC's LAN port.



2 The Pickering Interfaces 60-100 and 60-101 are seven-slot chassis that support a variety of 3U PXI modules.

3 The use of LXI-based synthetic instruments is more similar to PXI and VXI in philosophy and approach. This topic is covered in detail in Chapter 19.

Conclusion

The PXI and VXI architectures offer a number of advantages over rack-and-stack approaches, but they also present some disadvantages than can limit a test engineer's ability to meet technical and economic constraints in a test system. LAN-based hybrid systems that incorporate PXI or VXI instruments can be a cost-effective way to leverage equipment and programming efforts across the product life cycle. This chapter explored four hybrid options that deliver the benefits of LAN connectivity while protecting investments in existing instrumentation. As LXI solutions are becoming more pervasive across the T&M spectrum, the transition to an all-LAN LXI approach is becoming increasingly feasible and attractive from both functional and financial perspectives.

19. Assessing Synthetic Instruments

Introduction

For decades, automated test systems (ATS) built around benchtop instruments have been the dominant test-system architecture. In the late 1980s, modular VXI-based systems addressed several shortcomings of the rack-and-stack approach. In particular, card-based instruments mounted in a multi-slot mainframe reduced the size and weight of systems. The speed and capabilities of the VXI backplane also enabled enhanced triggering and faster data transfers. However, all such commercial technologies tend to have lifecycles that are much shorter than a typical aerospace or defense system, possibly affecting long-term maintenance and support of an ATS.

These issues are the driving force behind an approach called synthetic instrumentation (SI). The concept is simple: SIs let you configure and reconfigure modular hardware and software elements to create the functionality of multiple measurement devices. This building-block approach makes it possible to update or upgrade an ATS or a Test Program Set (TPS) by simply replacing a single module such as a digitizer or downconverter. It can also reduce the burden of software updates over the lifetime of an ATS.

This chapter will help you assess the potential value of SI relative to your present or future requirements. It presents a brief history of SI, compares a rack-and-stack system to an SI-based system, describes the initial applications of SIs and illustrates the emulation of conventional instruments with SIs.

Reviewing the roots of SI

In the mid 1990s, the U.S. Department of Defense (DoD) assigned the U.S. Navy the task of developing new types of ATS for the testing of avionics and weapons systems. This ongoing project has six driving goals:

- Reduce the total cost of ownership of ATS
- Reduce the time to develop and deploy new or upgraded ATS
- Reduce the physical footprint of each system
- Reduce the logistics footprint via decreased spares, support systems and training
- Provide greater flexibility through systems that are interoperable among U.S. and allied services
- Improve the overall quality of testing

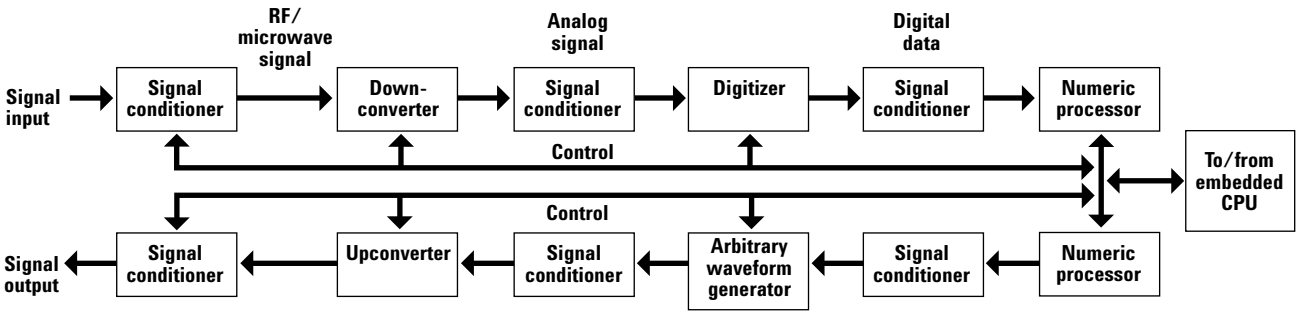
These are ambitious goals, but the DoD, defense contractors and equipment manufacturers believe they can be achieved over time by applying advances in commercial technologies (LXI is one important recent example).

The greatest progress toward these goals is coming from the use of SIs. According to the Synthetic Instruments Working Group (SIWG)[1], a synthetic instrument is a reconfigurable system that links a series of elemental hardware and software components via standardized interfaces to generate signals or make measurements using numeric processing techniques. The key word is reconfigurable: the elemental blocks can be arranged and rearranged via software commands—and the signals rerouted via switching—to emulate one or more types of traditional test equipment.

To achieve this flexibility, an SI will contain as many as four major components: signal conditioners, frequency converters, data converters and numeric processors. For example, the basic block diagram shown in Figure 19.1 describes most microwave instruments, including spectrum analyzers, frequency counters, network analyzers and signal generators.

1 Includes joint participation of the DoD, prime contractors and suppliers.

Figure 19.1. Basic architecture of an RF/microwave synthetic instrument



Unlike general purpose instruments, which are optimized to perform one task (e.g., spectrum analysis or signal generation), the synthetic instrument architecture is optimized to provide greater efficiency in an ATS by reducing redundant elements such as the digitizers and downconverters found in multiple instruments used within current systems.

The DoD expects these SI modules to come from a variety of vendors, enabling easy mixing and matching as requirements change or modules become obsolete. What's more, any substitution of modules—replacement or "technology insertion"—should require only minimal changes to the core system software.

Putting SIs in perspective

SIs are clearly intended to address a specific set of needs that are especially important to the military, but may also be relevant to some commercial organizations. For example, if your company is bidding on a contract that requires or gives preference to the NxTest concept, then SI will be required. Longer term, commercial organizations that utilize outsourcing and offshore manufacturing may benefit from the use of SIs in test systems they define or provide.

Assessing the situation

Two factors will affect the rate of SI adoption in the near term: hardware availability and software effort. Gradually, a wider variety of hardware is becoming available, and Agilent is in the vanguard of both SI and LXI. The LXI standard, which addresses the needs of synthetic instrumentation, is perhaps the most promising platform for SI due to the potential longevity of the LAN interface (see Chapter 16 for more on LXI).

Software is another matter. Currently, substantial effort is required to create the software modules that provide essential functionality such as the measurements and calibration routines needed to replace a stand-alone instrument. There is also the time and effort required to support software written in-house. If you add to that the typical effort required to create the mainline test program or suite of TPS, then the total up-front development cost is acceptable only if SIs are required.

Looking ahead, SI vendors recognize the need for software tools that will reduce effort, accelerate development and ensure accurate, repeatable results. As these tools become readily available and reuse of software modules becomes more commonplace, the development costs for SI-based systems should decrease. However, vendors need to address one key issue: the interchangeability of software components. If vendor substitution is equally viable with both the hardware and software elements of SIs, then the major benefits of the NxTest vision will be within reach.

Weighing commercial applications

If you develop systems within a commercial organization, the business model for most automated test applications probably can't support the higher initial costs of developing SI-based solutions. Of course, this requires a case-by-case assessment, and only you can decide if the potential benefits outweigh the current tradeoffs. Some early adopters may find SI to be very useful in a specific application.

Over the longer term, the promise of SI is well worth watching for many commercial firms. As more hardware and software modules become available, the economic benefits will increase for commercial applications.

Comparing present and future approaches

Whether you view SIs as a near-term requirement, a long-term curiosity or something in between, a comparison with traditional approaches reveals some interesting highlights. Within the context of the DoD's driving goals, it is easy to illustrate the advantages of synthetic instruments over GPIB, VXI or PXI solutions.

Reviewing purpose and usage

The main purpose for a military-related system is to test devices or assemblies in locations such as the flight line, an aircraft hangar or a repair depot. The same system may also be used in the original manufacturer's facility.

When the test system is fielded for military use, the top priority is to identify and replace defective electronic systems or assemblies as quickly as possible to return an aircraft or vehicle to operational service. The second priority is to repair the defective system or assembly and put it into the inventory of spares.

The usage model for such test systems involves rapid deployment, perhaps into areas of conflict. Putting the systems closer to the aircraft or vehicles they support translates into higher levels of operational readiness—and reduced downtime—for those aircraft or vehicles. In this scenario, flexibility and easy maintenance are more important than absolute measurement throughput.

Looking at current solutions

In this context, systems built around benchtop GPIB instruments, modular architectures such as VXI and PXI, or a combination, have noteworthy advantages and disadvantages.

GPIB instruments

The foremost advantage of GPIB devices is the combination of measurement capabilities, performance, accuracy and repeatability contained in one unit. Essentially every type of measurement from DC to low frequency to RF is available in this format. What's more, the cumulative expertise of the vendor, the science behind an accurate measurement, is built into the firmware of each instrument. For system integration, GPIB is well established as the dominant architecture for automated testing.

On the downside, a system built with just GPIB instruments can be so large and heavy that it is difficult to move frequently or across long distances. One obvious reason for this is the number of front-panel displays and keypads that go unused in a computer-controlled system (Figure 19.2). Less obvious in a large test system is the number of redundant digitizers, frequency converters and other block-diagram elements within many of the instruments.

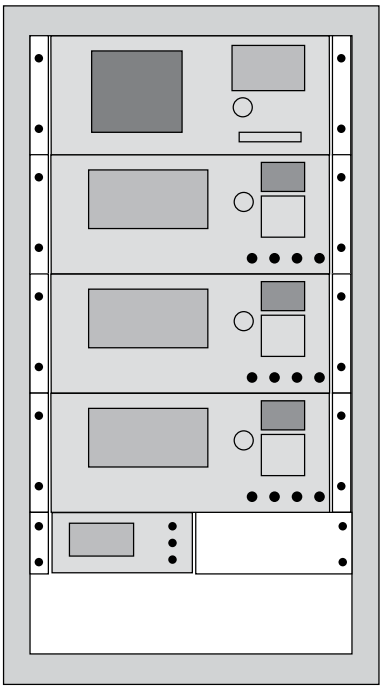
It is also costly to upgrade such a system. For example, when a faster, wider-bandwidth digitizer becomes available, it may take several months before it is available in a GPIB instrument—and it may be necessary to replace the existing instrument to get the benefits of the new digitizer. What's worse, changing an instrument may require software modifications, which entail additional time and expense to make the system software work with the new device.

VXI and PXI

With these modular architectures, the key advantage is the combination of measurement performance and triggering capabilities available in a compact form factor. Also, there is only one display, which is connected to either an external or embedded controller. The ability to embed the controller in the VXI or PXI mainframe also saves space and simplifies system transport.

Because VXI and PXI are based on flexible, reconfigurable modules, the SIWG accepts them as SIs within the DoD NxTest vision. However, some functions or measurements (such as high frequency RF and high-wattage power supplies) are not available in VXI or PXI. The cost of a VXI- or PXI-based solution is also generally higher than an equivalent rack-and-stack system.

Figure 19.2. Redundant or unneeded hardware such as instrument displays, keypads and digitizers add extra volume and weight to rack-and-stack systems.



With regard to system longevity, both modular architectures fall short because they are based on computer backplanes that tend to evolve rapidly then become obsolete. For example, VXI is based on the 1980s-vintage VMEbus, which is gradually disappearing from the computer world. Similarly, PXI is based on the PCI bus, which is being replaced by PCI Express. As time passes, it will become more expensive to support and sustain VXI- and PXI-based systems.

System software

With any of the three major test-hardware architectures, an essential key to success is the ability to reduce the time, effort and expense of software development and support. This depends heavily on development tools and environments that enable greater reuse of software in system creation or modification. Today, text-based programming with the variants of C is most commonly used for high-performance test systems. Other solutions such as Agilent VEE Pro and NI LabVIEW provide graphical tools for system creation.

Whichever tools you prefer, the use of device drivers can simplify the programming task. This is especially true with register-based VXI and PXI devices: drivers allow programming at a higher level by handling low-level operations such as reading and setting card attributes, initiating measurements, and loading or unloading data. Although programming at the register level enables detailed computer control of each module, it can be quite complicated and time consuming.

Understanding the SI approach

With SI, the fundamental elements of multiple instruments are realized through functional modules such as digitizers, upconverters, downconverters and arbitrary waveform generators. By arranging and rearranging the interconnection of these building blocks and the associated software modules, it is possible to emulate the functionality of an oscilloscope, a spectrum analyzer, a power meter and other instruments in much less physical space. Operationally, this is a software-intensive process in which the system could perform a series of tests by configuring the hardware, any needed switching and the associated software module for one type of measurement and then reconfiguring the hardware, switching and software modules for the next type of measurement.

System hardware

As a comparison, a rack-and-stack system containing a spectrum analyzer, three microwave sources and a power meter might occupy 18U of rack space. Using a variety of half-rack SI modules that don't have displays or keypads, the same functionality occupies 11U of rack space, as shown in Figure 19.3. This type of system is smaller, lighter and easier to transport. It also simplifies support by making it easier to replace or upgrade individual instrument modules as needed.

A rear-panel view of the SI system would reveal LAN ports on each module. By creating LXI-compliant SIs, Agilent is providing a PC-to-instrument interface that delivers the stability, longevity and performance of LAN. This simplifies PC connectivity and also helps lower the total cost of ownership for the ATS.

The rear view would also show a hardware trigger bus cable that complements a variety of LAN-based triggering capabilities. In combination, these triggering capabilities equal or surpass the capabilities VXI and PXI.

System software

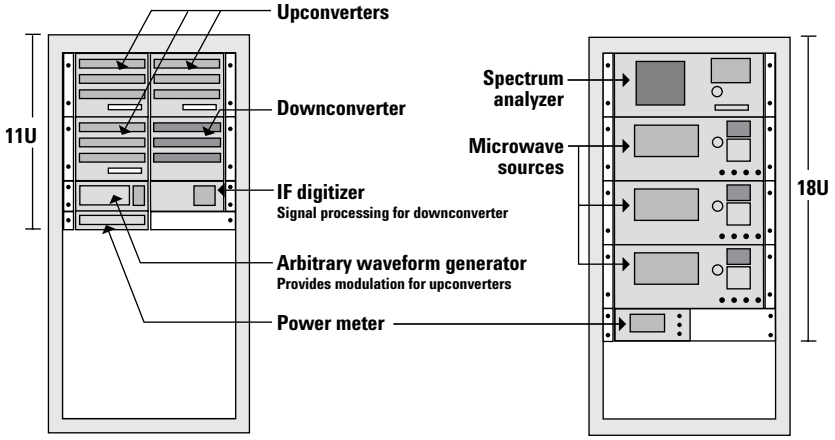
The points mentioned earlier regarding software development and maintenance still apply. Currently, creating the necessary measurement and calibration functionality requires a significant development effort. However, any software modules designed for transportability can be reused with other SI-based systems and potentially with other hardware modules.

Deciding if SIs are right for you

In the near term, SIs offer useful benefits that must be weighed against the tradeoffs. The main overall benefit of the modular, building-block approach is greater flexibility in less space. This approach will also make it easier to replace individual modules or implement technology insertion when new, updated capabilities are available. Longer term, this should also make it easier to replace any modules that become obsolete.

Currently, the major tradeoff is the intensity of the software effort, but as mentioned earlier, this is likely to change as hardware vendors begin to provide the necessary software tools. Looking at the whole system, another possible tradeoff is in the ability of SI-based systems to scale gracefully: larger systems will tend to put greater demands on the host PC. A higher-performance PC may be able to handle the demands of a complex system, but a faster processor (or multiple processors) and more memory also means a higher price for the PC and therefore a higher total cost for the system. The use of intelligent, LXI-based instruments, which can offload many computing tasks from the PC, is another way to head off this potential issue.

Figure 19.3. An SI-based system can provide equivalent (or greater) functionality in less rack space.



Exploring the initial applications

Today, SIs are a good fit with certain problems but are not quite ready for others. For example, SIs are not optimized for single-purpose applications (e.g., just spectrum analysis), one-box testers on the production line, benchtop applications in R&D, or short-lived test systems. As more software tools become available, the situation will become more favorable for these scenarios.

In contrast, SIs are well suited to situations that require multiple identical ATSs or when a system will be in service for many years. The flexibility of SI is also a good match when you need to test a wide variety of similar devices with a limited set of measurement hardware. These are clearly the major issues facing the DoD and its prime contractors when creating, supporting and preserving a TPS.

As mentioned earlier, SIs must address four present and future scenarios: flight-line test, intermediate-level (I-level) test, depot test and at-manufacturer or OEM test. These represent a continuum of testing that includes conscious tradeoffs between size, cost, speed and performance.

The vision is to use a common, scaleable hardware platform complemented by common test software and database management software that will be networked across all levels of service and support, and across all branches of the military. In practice, the flow of information starts in the field when an aircraft or vehicle detects an anomaly in one of its electronic systems. From that, the hardware and the information flows from one stage to the next:

- **Flight-line test.** In this operational, front-line application, the test system receives a message from the aircraft or vehicle and flags it for attention. When it returns to base, the critical need is to quickly identify and swap out the correct *subsystem*. The defective unit is recorded in the central database so it can be tracked through the rest of the process.

- **I-level test.** The key need is to identify the defective *module* within the subsystem. If it can be removed, it is recorded in the central database and then sent to the next stage.

- **Depot test.** At centralized repair centers, the module is tested with the intent of identifying defective *components* at the card level. The repaired unit will be placed into the inventory of spares where it will eventually return to service—and enable the increased availability of aircraft or vehicles.

Typically, OEM testing occurs before the card, module or subsystem is delivered to the military and put into service. If the same test system—both hardware and software—is used by both the manufacturer and the military, there can be greater confidence in the results and potentially lower costs in system development, deployment and support.

Utilizing current SI devices

In May 2006, Agilent's initial offering of six synthetic instruments became the first Class A LXI products to achieve certification from the LXI Consortium. These SIs demonstrate Agilent's ability to leverage proven RF technologies into innovative LXI-based solutions that serve the needs of the DoD, its prime contractors and others who can benefit from the flexibility of modular instrumentation.

Reviewing the original six

N8201A. This high performance 26.5 GHz downconverter provides IF output frequencies of 7.5, 21.4 and 321.4 MHz, enabling three different signal bandwidth capabilities. External mixing can be utilized to downconvert microwave signals as high as 110 GHz. The N8201A is leveraged from the Agilent PSA Series spectrum analyzers.

N8211A. This high performance 20/40 GHz analog upconverter generates a stimulus signal with superior AM, FM and pulse modulation capabilities via external or internal modulation (Figure 19.4). The N8211A leverages the Agilent PSG analog signal generator's high output power, low phase noise and excellent level accuracy. This module is available with a variety of options, including output power and modulation type.

N8212A. This high performance 20 GHz vector upconverter functions as a microwave source with greater than 2 GHz I/Q modulation bandwidth. It features AM, FM and pulse modulation (via external or internal modulation) and multisource coherent carrier capability. The N8212A is based on the Agilent PSG vector signal generator and includes options for greater spectral purity and enhanced phase noise.

N8221A. This 30 MSa/s IF digitizer has a 7.5 MHz IF input and provides 80 dB dynamic range, 14-bit resolution, and 8 MHz modulation bandwidth.

This module was also leveraged from the PSA Series spectrum analyzers.

N8241A. This arbitrary waveform generator (AWG) features 1.25 GSa/s output with 15-bit resolution and is based on the Agilent N6030A AWG. The N8241A offers dual-channel, single-ended and differential outputs, with 500 MHz of instantaneous analog bandwidth per channel (Figure 19.5).

N8242A. This AWG features a choice of either 1.25 GSa/s or 625 MSa/s with 10-bit resolution. It offers dual-channel, single-ended and differential outputs, with 500 MHz or 250 MHz of instantaneous analog bandwidth per channel.

Figure 19.4. Agilent N8211A 20/40 GHz performance analog upconverter

Figure 19.5. Agilent N8241A arbitrary waveform generator

Others. For signal routing, the L4445A microwave switch/attenuator driver module allows control of a broad range of microwave switches and attenuators. These LXI-based modules provide switching bandwidth up to 50 GHz. Agilent also offers the N8262A, a 40 GHz wideband peak and average power meter with 100 MSa/s continuous sampling rate and 30 MHz video bandwidth. This LXI device is based on the Agilent P-Series power meters.

Emulating RF instruments

This versatile set of modules can be quickly and easily reconfigured to make a host of measurements that would ordinarily require a vector signal analyzer, spectrum analyzer and oscilloscope. They can also be used to emulate the capabilities of an obsolete instrument such as the HP 8902A measuring receiver. Two brief examples will illustrate some of the possibilities.

SI stimulus unit

This requires signal generation hardware and software modules to create the required signals and perform scalar or vector signal analysis. Signal generation might utilize the N8241A AWG module (for maximum signal bandwidth and accuracy), the associated signal-creation software, and the N8211A or N8212A upconverter, depending on requirements for modulation, output power and signal purity (Figure 19.6).

SI measurement unit

The input signal would be routed to the N8201A downconverter, which would provide a 7.5-MHz signal to the N8221A digitizer. Through its LAN connection, the host PC would acquire one or more data blocks and apply the appropriate software modules for vector signal analysis or spectrum analysis of signals from devices such as radar systems, cell phones and wireless networking equipment (Figure 19.7).

A caveat

You can use SIs to emulate a legacy instrument up to the point where

Figure 19.6. For signal generation, the SI chain includes numeric processing, data conversion and frequency conversion.

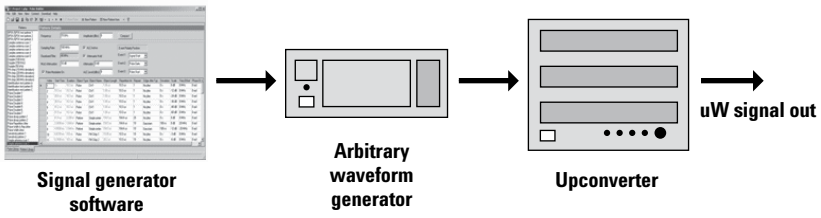
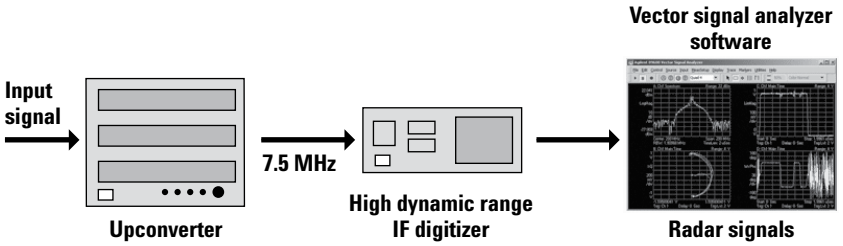


Figure 19.7. Agilent N8241A arbitrary waveform generator



the SI hardware is too different. For example, software can emulate a legacy instrument that has poor noise floor; however, it can't emulate a legacy instrument that has a better noise floor than the SI hardware. Also, most GPIB instruments have a unique set of timing, network and bus issues that are very difficult to reproduce. In other words, SIs can emulate legacy instruments, but no emulation will be a perfect duplication of the original.

Conclusion

The basic premise of synthetic instruments is very appealing: they let you configure and reconfigure building-block modules to create the functionality of multiple measurement devices. With benefits such as smaller test systems, easier transport, single-module updates or upgrades, long-lived I/O and simpler software updates, LXI-based SIs readily support the DoD's vision for NxTest.

As SI vendors address the need for software tools that reduce effort, accelerate development, ensure accurate, repeatable results and enable interchangeability of software components, the development costs of SI-based systems will fall and these solutions will become more viable for commercial applications. Over the long-term, it will be worth watching the growth and development of SIs—and worth monitoring their progress toward a new era of greater flexibility in automated test systems.

Section 4. RF/Microwave Test Systems

Overview

The three chapters in this section explore some of the unique challenges of automating RF/microwave tests, particularly as devices and test requirements become increasingly complex.

20. **Optimizing the Elements of an RF/Microwave Test System,** offers advice on creating flexible, long-lived RF/microwave test systems that will provide accurate, repeatable assessments of the device under test. The focus of this article is making it easier to configure, update and modify your systems now and in the future.

21. **Six Hints for Enhancing Measurement Integrity in RF/Microwave Test Systems,** offers insights into successfully balancing the tradeoffs between performance, speed and repeatability.

22. **Calibrating Signal Paths in RF/Microwave Test Systems,** provides an overview of three approaches that can be used to calibrate RF signal paths and produce accurate, repeatable measurements.

20. Optimizing the Elements of an RF/Microwave Test System

Introduction

Whether you need to test the latest cell phone, a next-generation military radio or an advanced radar system, proving the device's ability to meet customer requirements depends on a test system that can provide accurate, repeatable results. For both parametric and functional testing, the ability to achieve accuracy and repeatability becomes more difficult as devices become more complex. Greater complexity often translates into more tests, which may mean longer development time and a more complicated test system. The challenge grows when you try to create a system that meets budget and schedule constraints but is also flexible enough to meet both current and future testing needs.

This chapter offers ideas and suggestions that can help you create flexible, long-lived RF/microwave test systems that will provide accurate, repeatable assessments of the device under test (DUT). Our focus is on making it easier for you to configure, update and modify your systems now and in the future.

Letting the DUT define "future"

When discussing the future-proofing of a test system, it's important to clarify what "future" means within the context of the DUT and its expected lifetime. For RF/microwave test systems, there are two large classes of DUTs that have specific future requirements.

- **Long-lived DUTs.** Many devices and systems developed for aerospace and defense applications require test systems that are easy to maintain and update far into the future. An example of this is the NxTest program from the U.S. Department of Defense (DoD). Its guiding vision combines a common hardware architecture with software-driven functionality to enable rapid deployment across different programs and facilitate easy updates in the future.
- **Short-lived DUTs.** Fast-cycle aerospace/defense programs and rapidly evolving commercial wireless products require test systems that can be developed rapidly and within budget. For example, creating a new test system from scratch for every new phone model—or new wireless standard—becomes less desirable as introduction cycles become shorter and budgets get tighter. The ability to leverage existing investments in test equipment and software will accelerate system development and deployment while also reducing system cost.

The ability to meet the needs of either long- or short-lived DUTs improves when the test system includes long-lived hardware, input/output (I/O) and software. Careful selection of these three elements will enhance a system's flexibility and its ability to perform accurate, repeatable measurements of multiple DUTs and applications—today and tomorrow.

Reviewing some essential considerations

When it's time to define and assemble an RF/microwave test system, two major factors will affect your decisions about test equipment: the key attributes of the DUT and the various constraints on the test system. A quick review of important attributes and constraints will lay a foundation for the discussions that follow.

Key attributes of the DUT

The attributes of your DUT obviously affect test-system design, and it's helpful to look at them from both general and specific perspectives.

General attributes

At a high level, it's helpful to consider the DUT's complexity, its stage in the product lifecycle, and the nature of the manufacturing process. For example, multi-function devices are often the most difficult to test: cell phones with built-in cameras, military radios that carry voice and data, and LAN devices with both wired and Wi-Fi capabilities may require a much wider range of measurements and a more costly and complex test system.

Whether a product is simple or complex, the early stages of its lifecycle generally require thorough testing of numerous characteristics—parametric and functional—to ensure expected performance and operation. As a product matures, fewer characteristics are tested, and often in less detail.

Within the manufacturing process, the product volume and mix also affect equipment choices. The most difficult case is high volume and high mix, which might require several identical test systems that are able to measure multiple products or product variations.

Specific attributes

The electrical attributes of the DUT often drive the shortlist of viable instrumentation candidates. Most DUTs contain a mix of circuitry that is becoming less analog and more digital while going higher in frequency with every generation. On the analog side, operating parameters such as frequency range, bandwidth and resolution—along with headroom for today's harmonics or tomorrow's enhancements—define the essential specifications for signal analyzers, signal generators, oscilloscopes and so on. The availability of test equipment with the necessary performance or capabilities will have a strong influence on the design of your system.

Greater digital content makes it possible for new devices to support multiple communication standards. This might be CDMA, TDMA and GSM in a cell phone or various protocols in the military's Joint Tactical Radio System. The need to support all relevant standards will demand much greater flexibility from the test system—and perhaps lead to the use of instrumentation that also has greater digital content in the form of advanced digital signal processing (DSP) capabilities.

The physical configuration of the DUT will also affect choices about handling, fixturing, switching, power, loads and test accessories. As an example, the number and kind of ports available for external connections may change as the device moves through the manufacturing process. Once the circuitry is loaded into its enclosure, any built-in test points may become inaccessible and the test interface may have to shift from hard-wired to antenna-based.

Constraints on the test system

A combination of business and technical factors will also influence system decisions. On the business side, budget and timeline are often the primary drivers of tradeoffs when selecting test equipment. At one extreme, for example, you might need to get the system up and running as quickly as possible—and the ideal solution may be a one-box tester, which trades rapid development time and optimized measurements for decreased flexibility. At the other extreme, your contract may require compliance with NxTest, which specifies the use of modular synthetic instruments—an approach that yields tremendous flexibility but at the expense of development time.

Within those constraints and tradeoffs, numerous expectations are placed on the test system. These include its capabilities and performance: inputs, outputs and switching; measurement and analysis; speed, accuracy and repeatability; and data handling and reporting. There are also expectations about cost effectiveness, which may suggest the use of hardware elements that are easy to reconfigure or replace and software that is easy to modify or reuse.

Expectations about system longevity follow from both the length of time the DUT will be manufactured and its estimated service life. Those requirements define how long the test system itself must also be supported and maintained.

Translating requirements into optimized equipment choices

With the essential attributes, constraints and expectations in mind, the next step is translating those requirements into the best combination of hardware, I/O and software for your system. We will look at all three elements separately but will emphasize the selection of system hardware.

Comparing hardware types across a common example

A conventional test system uses a variety of instruments that perform a single function such as spectrum analysis, signal generation or network analysis. These instruments are usually reliable, well understood and easy to use. However, they lead to large and often inflexible test systems that include many redundant elements (such as displays, keypads and mixers) and require complicated switching and fixturing.

In contrast, an ideal test system might use a few well-defined functional modules or building blocks (such as frequency converters, digital-to-analog converters and DSP engines) that could be arranged and programmed via software to perform the required measurements. If this type of "generic" test system were to contain flexible switching, powerful DSP hardware and fast, wideband analog-to-digital and digital-to-analog converters, it could analyze and generate virtually any type of signal.

These two sketches represent the ends of a continuum—and many of today's test instruments are hybrids that reside somewhere in between the conventional and ideal approaches. One popular example is a category called "vector" instruments. These integrate powerful DSP technology with conventional analog components to create versatile, accurate signal analyzers, network analyzers and signal generators that can handle highly complex signals and devices.

If used exclusively in a system, each of these hardware architectures—conventional analog, next-generation modular and modern vector— would produce a very different block diagram. To provide a consistent comparison, the next three sections describe how each approach might be used to create a system that performs multi-tone testing of a communication device.

Example 1: Conventional analog instruments

As shown in Figure 20.1, this is a complex system that includes three signal generators, one spectrum analyzer and a variety of external accessories—amplifiers, low-pass filters and a combiner. The system also includes a PC with software that controls the signal generators and the spectrum analyzer.

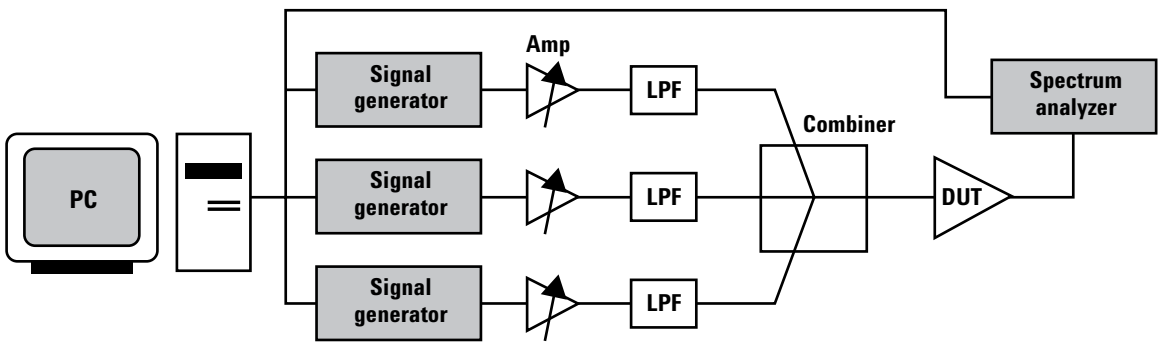
Advantages

In many cases, most of the equipment may be readily available on an engineer's bench, in a central loaner pool or from an instrument manufacturer. It will typically be relatively low cost and, as a result, quite cost effective. Because test engineers have been using this type of equipment for many years, it will likely be familiar and well understood, enabling rapid development.

Disadvantages

The single-purpose nature of conventional analog instruments gives them limited functionality and little versatility. This has three noteworthy drawbacks. First, a complete system will require numerous instruments and consume a lot of rack space. Second, the system will be more complex, requiring myriad interconnections among the various instruments and accessories. Third, this type of system needs frequent calibration to ensure its accuracy and repeatability.

Figure 20.1. A complex multi-tone test system implemented with conventional analog instruments



Example 2: Next-generation modular instruments

Compared to the conventional approach, this type of system requires a somewhat less complex arrangement of hardware (Figure 20.2) that includes four building-block modules: an arbitrary waveform generator (AWG), an upconverter, a downconverter and a high-speed digitizer. The PC provides system-control functions that arrange and rearrange the building blocks as needed to send or measure a variety of signals. The PC also runs user-written software that provides system functionality, ranging from calibration to measurement algorithms to data analysis.

Advantages

The modular approach provides the ultimate in flexibility, enabling a high level of hardware reusability and making it easy to rearrange the building blocks to create functionality that is equivalent to multiple instruments. For example, because the AWG can generate virtually any type of signal, this configuration can handle much more than just the multi-tone test.

Modular hardware also offers the possibility of obtaining better performance by simply replacing an outdated module with a new, higher-performance building block. What's more, this approach can also eliminate redundant hardware elements, which may reduce a system's size and its hardware and support costs. The DoD and others believe the building-block approach offers the greatest potential for enabling longer-lived test systems.

Disadvantages

Initially, this architecture will require a significant investment in software development. The main reason is the need to understand, define and create the individual measurement algorithms and analysis functions that will utilize data from the hardware modules. (This is in sharp contrast to a fully integrated instrument that has a vendor's measurement expertise built into its firmware.)[1] As a result, software development costs will tend to be higher for this type of system.

Another key issue is measurement accuracy. Because manufacturers cannot anticipate every possible combination of modules, developers will have to create routines that, for example, calibrate every on-the-fly rearrangement of the modules. Consequently, traceability may be an issue for the earliest systems built on this foundation.

1 Over time, Agilent expects to provide a broad and deep set of software tools to accompany its building-block hardware modules. Possible software tools include individual measurement routines (e.g., group delay, VSWR), complete measurement modules (e.g., spectrum analysis) and even legacy instrument emulation modules.

Example 3: Modern vector instruments

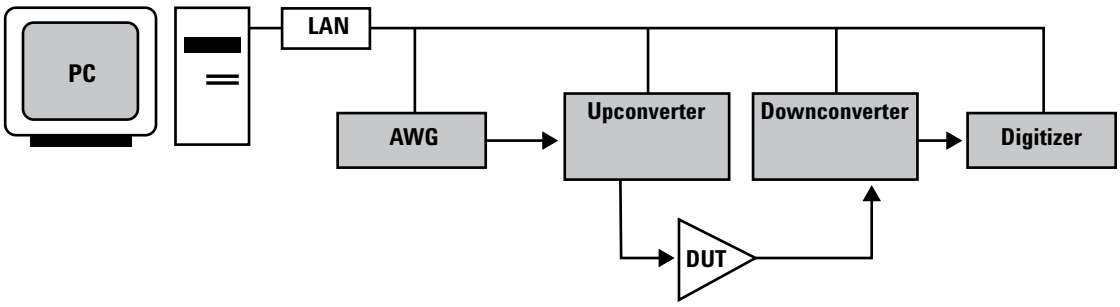
As shown in Figure 20.3, the use of modern vector instruments produces the simplest system, requiring just one vector signal generator and one vector signal analyzer. The PC does more than serve as host and controller: it also adds functionality via the Agilent Signal Studio software, which makes it easy to create the required multi-tone signal and download it into the vector signal generator.

Advantages

The tight integration of analog and DSP technologies delivers exceptional versatility and functionality. Comparing this system to the conventional approach, one vector signal generator replaces three analog signal generators and seven external accessories. On the measurement side, some vector signal analyzers also provide waveform analysis capabilities, possibly replacing a separate digitizer or oscilloscope. These flexible instruments can also be used for a variety of measurements, not just the multi-tone example. In a system, fewer instruments mean fewer connections, less complexity and fewer opportunities to introduce measurement errors.

Vector instruments can also provide better longevity: because they are firmware-based, it is easy to enhance their functionality and add new capabilities. Because so much of their

Figure 20.2. The multi-tone test system implemented with LAN-based building-block instruments



functionality is DSP-based, vector instruments can often provide better accuracy and performance through digital corrections to IF stages, filters and so on. These performance enhancements are traceable and also enable longer intervals between full calibrations.

Disadvantages

Currently, the hybrid approach commands a higher cost per unit but, as shown here, a single unit may replace multiple analog instruments. Also, if greater analog performance is needed, the whole unit must be replaced when that level of performance is available in a new vector instrument.

Comparing the three approaches

Each of the three approaches has something to offer. Conventional analog instruments are very familiar to many system developers and so may enable faster system development. What's more, they are often readily available and may be the first to offer the required level of performance. Next-generation modular instruments will provide tremendous flexibility and potentially greater system longevity than the other two approaches—but with longer development time and higher software costs. Today, modern vector instruments provide the strongest combination of functionality, versatility and accuracy. The ability to expand their capabilities via firmware updates vgives them an advantage when testing devices that include evolving communication standards.

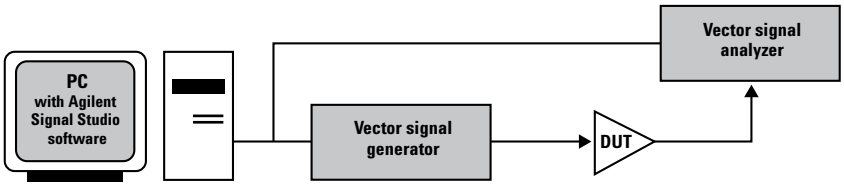
Before deciding which approach is the best fit for your system, it's important to also consider the available choices in connectivity, software and instrument communication. All will affect system development time, performance and longevity.

Assessing the connectivity choices

As discussed in earlier chapters, most current-generation PCs include one high-speed LAN port and multiple USB ports. In the T&M world, an increasing number of measurement instruments—and most new Agilent instruments—now include LAN and USB ports alongside the GPIB connector.

Spurred by the PC industry's steady advances in LAN performance (and commitment to backward compatibility) the trend in test equipment is toward greater use of the future-proof LAN interface while continuing to support GPIB. As an example, vector and modular instruments work well with LAN but you can easily incorporate up to 14 GPIB-only instruments into a LAN-based system via the Agilent E5810A LAN/GPIB gateway.

Figure 20.3. The multi-tone test system implemented with DSP-based vector instruments



Reviewing software and communication alternatives

Your chosen combination of application development environment (ADE) and instrument communication method creates tradeoffs between development time, software reuse and system performance.

ADEs are either textual or graphical. Textual environments such as Microsoft Visual Studio® have a steep learning curve because they require a detailed knowledge of commands and syntax. Graphical environments such as Agilent VEE Pro and National Instruments LabVIEW use a schematic approach, which engineers tend to learn easily. In the past, programs written in textual languages had a speed advantage at runtime but this difference has been reduced with time.

Instrument communication has been evolving, with direct I/O and vendor-specific commands giving way to industry-standard command sets and instrument drivers. Direct I/O has two important advantages: speed and access to an instrument's full feature set. However, because it is instrument-specific, direct I/O hinders software reuse. Instrument drivers are high-level pieces of software that are also instrument- or instrument-class-specific but, in contrast, they simplify programming by letting you substitute one driver for another if you replace an instrument in a system. The tradeoffs are in functionality and speed: drivers typically access only the most commonly used commands and often communicate more slowly than direct I/O.

Pulling it all together

Table 20.1 compares analog, modular and vector instruments based on five essential aspects that affect system performance: measurement capabilities, measurement performance, I/O connectivity, system software (and instrument communication) and potential longevity. Those elements capture the value of each approach, and that overall value provides a broader context for the sixth element, which is hardware cost.[2]

Ultimately, the best answer will depend on the attributes of your DUT and the constraints on your system. However, if you are creating a new test system, we suggest you consider the use of vector instruments, LAN-based I/O and instrument drivers. This combination will provide a highly future-proof system that should be easy—and cost-effective—to modify in the near-term, maintain and update in the future. If you are required to comply with NxTest, then substitute modular instruments for vector instruments in the preceding recommendation.

2 In many cases the lack of software transportability will drive the cost of developing new software far beyond the hardware cost.

Conclusion

Conventional analog instruments, next-generation modular instruments and modern vector instruments each offer compelling benefits for RF/microwave test systems. Choosing the approach for your next system depends on a number of factors: (1) whether your DUTs tend to be long-lived or short-lived, (2) the characteristics of your DUTs—including both general factors such as stage in the product lifecycle and manufacturing volume and specific factors such as degree of digital content and physical configuration—and (3) any financial and technical constraints on your test system.

In addition, be sure to consider your options for connectivity, software development tools and instrument communication.

For new test systems, our baseline recommendation is a combination of vector instruments, LAN-based I/O, graphical programming and instrument drivers. Modify this approach as needed, of course, but in general it will provide a high degree of future-proofing and the ability to modify, maintain and update quickly and cost effectively.

Table 20.1. Comparing key attributes of the three hardware approaches

	Conventional analog instruments	Next-generation modular instruments	Modern vector instruments
Measurement capabilities	Good but limited	User creates individual functions, gets maximum control	Best, very versatile; easy for manufacturer to update via firmware changes
Measurement performance	May offer best raw measurement performance (e.g., frequency range, bandwidth)	Able to mix and match modules to achieve desired combination of speed, range and bandwidth	May offer best speed, resolution and accuracy
Connectivity	GPIB	LAN	Most have GPIB, LAN and USB
Software & communication	Typically used with textual programming and direct I/O (and perhaps SCPI)	Graphical or textual programming with drivers; may require low-level programming of individual modules	Graphical or textual programming with drivers (and direct I/O, if necessary)
Potential longevity	Good, but must eventually replace to achieve latest performance and capabilities	Excellent potential: Update software as needed to create new capabilities; replace single module to obtain latest performance	Very good for commercial programs; may be too short for aerospace and defense programs. Can add capabilities via firmware updates; however, must eventually replace instrument to obtain latest analog performance.
Hardware cost	Moderate for individual instruments but may need more than one of each type	High (initially) for individual modules but may provide lower overall cost due to flexibility and longevity of test system	Somewhat high for individual instruments but each one may replace multiple analog instruments (and provide greater flexibility)

21. Six Hints for Enhancing Measurement Integrity in RF/Microwave Test Systems

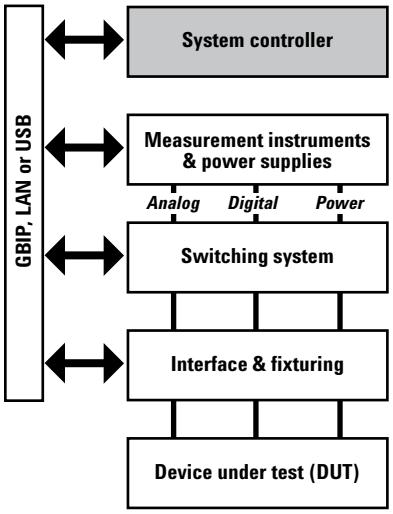
Introduction

Even though most RF and microwave test systems measure devices within a few broad categories such as amplifiers, transmitters and receivers, every individual system faces a unique set of circumstances, requirements and challenges. As unique as each situation may be, three universal factors interact when you define any RF and microwave test system: performance, speed and repeatability. Within the unique situation each system developer faces, the ability to make tradeoffs among these factors is one key to achieving the required level of measurement integrity.

Opportunities to manage these tradeoffs can occur at many points along the pathways between the device under test (DUT) and the measurement instruments (Figure 21.1). This chapter suggests a framework for those tradeoffs and offers six sets of hints that address common problems that may exist along RF signal pathways.

Hint 1 provides a foundation for all six hints. The remaining hints address the three major tradeoffs: Hints 2 through 5 can help you achieve greater performance, Hint 6 suggests several ways to improve measurement speed, and Hints 3 and 4 can help you enhance measurement repeatability. In general, these hints apply to signals in the range of 100 MHz to 26.5 GHz.

Figure 21.1. Within any test-system architecture, there are numerous opportunities to manage measurement integrity by balancing the tradeoffs among performance, speed and repeatability..



Hint 1: Prioritize performance, speed and repeatability

All test scenarios require a balance of performance, speed and repeatability. In most situations, one or two of these will be the dominant factor that drives your test requirements and your equipment choices. In all cases, a closer look at the interactions and tradeoffs among performance, speed and repeatability will help you manage your unique situation.

Building the foundation

To lay the foundation for all six hints, it's essential to clarify our definitions of performance, speed and repeatability in this context.

Performance

In RF and microwave test equipment, Agilent's definition of "performance" focuses on instrument accuracy, measurement range and bandwidth. Instrument accuracy includes the specified absolute accuracy of amplitude and frequency measurements. Measurement range refers to dynamic range, distortion, noise level and phase noise, which are the attributes that enable precise characterization of signal levels. Bandwidth refers to the frequency width or data rate that can be processed and analyzed.

Speed

Test system speed or throughput depends on hardware, input/output (I/O) and software. Our focus is on the hardware and the four factors that influence speed: measurement set-up time, measurement execution time, data processing time and data transfer time. At RF and microwave frequencies, a key aspect of set-up time is the settling time of the DUT or the test system whenever a change is made (such as switch closures and power level).

Repeatability

For any test system, the ability to produce consistent results—test-to-test and day-to-day—is crucial. However, repeatability does not infer a high level of precision, which depends on the performance of individual instruments. Instead, repeatability means a consistent result, whatever the specified accuracy. For any given instrument, repeatability may be different for certain measurements or modes so it's important to check the product specifications or ask the manufacturer. To some extent, repeatability can be improved with more averaging or through modified algorithms that produce an accurate approximation of a standardized measurement. It can be optimized by minimizing changes to measurement settings such as center frequency, span and attenuation level.

Summarizing the interrelationships

The test requirements and business drivers for a DUT will help you assess the relative importance of performance, speed and repeatability. Once you've identified the dominant factor and the intensity of its requirements, sorting through the interactions and their impact on the system becomes easier. Tables 21.1 through 21.3 summarize the implications of these interactions in two cases: when the intensity of the dominant factor is either high or low.

Table 21.1. When performance dominates, the most important interaction is between performance and speed.

Performance Requirements	Implications for speed	Implications for repeatability
Low	Can go faster: Will spend less time on tasks such as instrument calibration and measurement averaging.	Probably lower: This situation suggests low performance equipment, which may yield greater uncertainty and, therefore, less consistency from test to test.
High	Must go slower: Will probably need to spend more time on tasks such as instrument calibration, path correction and error removal to ensure greater precision.	Probably greater: High performance equipment with lower noise floor, fewer distortion products, greater isolation, and so on, will tend to provide less uncertainty and greater measurement consistency.

Table 21.2. When speed dominates, the key relationship is between speed and repeatability.

Speed Requirements	Implications for performance	Implications for repeatability
Low	Greater precision: Can spend more time on calibration, path correction, error removal, etc. However, this situation may suggest lower-cost instruments, which often have fewer performance-enhancing features	Greater consistency: Can increase the number of averages, number of samples or sweep time (with average detectors). May be able to use methods such as long RMS detection, narrow video bandwidth or precise, time-intensive algorithms.
High	Lower precision: The need for speed may lead to compromises such as less accurate measurement techniques, lower measurement resolution, fewer sweep points and faster sweep speeds.	Lower consistency: Less time available for measurement averaging and intricate, precise algorithms may mean greater uncertainty and lower consistency..

Table 21.3. When repeatability dominates, the key relationship is once again between repeatability and speed.

Performance Requirements	Implications for performance	Implications for speed
Low	May be lower: Low repeatability implies a larger error budget, which may also infer lower-performance instruments (less absolute accuracy).	Can go faster: When repeatability has low importance, less time will be spent on improving measurement consistency

Repeatability and performance

In Tables 21.1 and 21.3 there is an important secondary relationship between repeatability and performance. This is an indirect relationship linked by measurement uncertainty. When dealing with uncertainty, some system developers create an "error budget," the size of which depends on the margin between test requirements and system uncertainty. The two major contributors to uncertainty are absolute accuracy (instrument performance) and measurement consistency (repeatability). If the instruments in a system have high absolute accuracy, then there is a wider margin in the error budget for lower repeatability. If the instruments provide consistent results, that leaves more room in the budget for somewhat lower absolute accuracy.

Multiple "high" requirements

Satisfying requirements such as "high speed with high repeatability" or "high performance with high speed" will probably require sophisticated instrumentation that is somewhat more expensive than less-capable equipment. However, many high performance instruments may include hardware accelerators that speed up time-consuming operations such as averaging and calibration. Some models may also include multiple algorithms for calculating parameters such as adjacent channel power (ACP).[1]

If all three requirements rate "high" then every element of the system—test equipment, switching, cabling, connectors, and so on—must be scrutinized. The best solutions will likely demand a high price, but may provide additional capabilities and benefits.

1 As an example, some Agilent PSA series spectrum analyzers include a standard "ACP mode" and a "fast ACP mode." The fast mode provides an accurate approximation of the standard-compliant measurement.

Hint 2: Review the nature and behavior of the DUT

A typical automated test system performs three basic tasks: sourcing, measuring and switching. Decisions about which signal generators, power meters, spectrum analyzers, network analyzers, switch matrices and cables to use depend on the electrical and mechanical attributes of the DUT. At RF and microwave frequencies, a few essential characteristics require special attention.

Electrical parameters

The basic nature of the DUT is a key consideration: Is it passive and linear or active and nonlinear? Passive, linear devices are easier to deal with because they typically have fixed gain and phase shift at any allowed input power level across their operating bandwidth. In contrast, active devices demand greater care because they usually have a nonlinear operating region that is highly sensitive to input power, producing different results at different levels. Within a test system, this may suggest the addition of amplifiers or attenuators to precisely control power levels, and perhaps the addition of couplers to split off and verify the power level being delivered to the DUT. These additions should not be taken lightly: At high frequencies, every system element has a complex-valued impedance (with associated S-parameters), and every additional connection creates the possibility of undesirable interactions with the DUT.

- **Avoid mismatches.** An impedance mismatch at any connection can cause insertion loss, which robs power from any sourced or measured signal. As a truism, power is expensive at high frequencies—and it becomes more expensive if it has to be delivered across a wide frequency range. ***Hint:*** Use high precision cables and accessories, and fully characterize their actual impedance using a vector network analyzer (VNA), especially if the DUT is an active device.
- **Minimize VSWR.** The combination of a switch matrix, its connectors, its internal and external cables, and even the bending radius of any RF cables can induce errors caused by voltage standing waves in the DUT. ***Hint:*** To minimize this error, use a switch matrix with a voltage standing wave ratio (VSWR) specification of 1.2:1 or better.
- **Enhance isolation.** If your test requirements call for simultaneous measurements of high- and low-level signals then the isolation specifications of the switch matrix will affect measurement integrity. ***Hint:*** If there are multiple pathways through the DUT, use a signal generator and spectrum analyzer to characterize the isolation properties to the extent possible. If this can't be done then the system should be configured and programmed to route high- and low-level signals on non-adjacent pathways or through separate switch units.

Mechanical attributes

One set of details to consider is the number and type of connectors for signals and power (AC or DC). This will influence factors such as the required size of the switch matrix and the complexity of system cabling. ***Hint:*** Use a switch matrix with enough ports to let you make all system-to-DUT connections just once. This will minimize delays while waiting for signals to settle, and minimize the chances of damaging the switch matrix or DUT with sudden changes in power level.

Hint 3: Understand, characterize and correct RF signal paths

Without additional correction, product specifications extend only as far as the "calibration plane" that exists at an instrument's input and output connectors. To achieve accurate, repeatable measurements—and corrected DUT results—we suggest that you push the calibration plane out as close as possible to the DUT. There are several ways to achieve this, whether the pathways are passive or active and the DUTs are local or remote.

Handling passive pathways

As just noted, passive devices have fixed gain and phase shift at any allowed input power level across their bandwidth. However, every connection along a passive path may have an impedance mismatch, which will cause insertion loss and phase shifts (or delays). At high frequencies seemingly simple passive elements become complex transmission-line elements, precluding simple algebraic addition of losses and phase shifts along the path. ***Hint:*** Use a VNA to either measure the entire connected path or characterize the S-parameters of each element and use vector math to model the total loss and phase shift of the entire path. These values can be stored in the system PC and applied as needed to correct a measurement, or they can be applied by a network analyzer, for example, to enable real-time adjustment of filters and other variable DUTs.[2]

2 To learn more about S-parameter measurements, please see Application Note 154, S-Parameter Design, and Application Note 1287-3, *Applying Error Correction to Network Analyzer Measurements.*

Correcting active pathways

The performance of active devices may vary with changing input power. The process required to improve measurement accuracy depends on whether the device is operating in the linear or nonlinear portion of its response. If an active device such as an amplifier is operating in its linear region—well below its 1-dB compression point—during both calibration and measurement operations, then corrections can be accurately applied at any power level within that region. ***Hint:*** If the active device is operating in the nonlinear portion of its response then the power level used for a measurement must also be used during calibration to ensure accurate correction. If measurements will be made at multiple power levels in nonlinear mode, then individual calibrations must be made at each of those levels and stored for later use.

Hint: Check the frequency response of the active device over the frequency range of the DUT. Again, you should either measure the entire path at specific power levels or characterize the S-parameters of each interface and use vector math to create a model that can applied after-the-fact or in real time.

Hint: To simplify the process of characterizing and correcting RF signal paths, some system developers minimize the use of active devices. This reduces both the calibration effort and the chance of errors caused by variations in power level when operating in nonlinear mode.

Dealing with DUT distance—near or far

Accurate correction can be difficult whether the DUT is mounted in a fixture at the test system or located several meters away in a test chamber. Fixture-based measurements are challenging because pathways often include transitions from coaxial cables to microstrip-based shorts, opens and loads. ***Hint:*** If high quality microstrip elements are not available it will be necessary to measure the fixture with a network analyzer, model the impedance and remove those effects from the measurements.

When the DUT is remote, the main issues are path attenuation in long cable runs and path variation due to temperature fluctuations and cable flexion. ***Hint:*** Characterize path attenuation either by measuring the entire pathway between the instrument and the DUT (if possible) or by measuring all elements along the path and using vector math to combine their complex-valued responses.

Hint 4: Be aware of everything connected to an instrument

Test equipment manufacturers specify the performance of every instrument up to the front-panel connectors that source and measure signals. From there, everything that comes between the instrument and the DUT can affect instrument performance and measurement repeatability. At RF and microwave frequencies and power levels, the three worst offenders are typically cables, switches and signal conditioners.

Selecting the right type of cable

When specifying a test system you will need to decide what type of cabling to use for device interconnection, and you may be able to specify the type used within the switch matrix. As a general rule, a stable cable will provide lower insertion loss, better VSWR and, therefore, greater measurement repeatability. At high frequencies, the three most commonly used types of cabling are semi-rigid, conformable and flexible.

Semi-rigid

As suggested by the name, these cables do not easily change shape, ensuring excellent performance and repeatability. High quality semi-rigid cables achieve additional stability during the manufacturing process through techniques such as MIL-standard temperature cycling. When applied after the forming process, temperature cycling can eliminate internal stresses that may cause later deformation of the preformed cable.

The quality of the dielectric used in these cables also affects their measurement performance. Solid PTFE on is the most common but contributes to insertion loss. Expanded PTFE is currently the best alternative, providing lower insertion loss and wider frequency range. All of this attention to detail is reflected in the cost of these cables, which is considerably higher than conformable or flexible cabling.

Conformable

These cables offer less stability than semi-rigid cables because they are easily shaped and reshaped. Their flexibility affects measurement repeatability and long-term reliability.

Flexible

Sometimes called “instrument-grade cables,” these typically offer good phase stability and low insertion loss but at a relatively high price. They also tend to be high maintenance, requiring careful handling because severe deformation can alter their electrical properties and cause inaccurate measurement results.

Avoiding switch-related problems

Switching is central to overall system functionality, automating the connection of signals and power supplies between instrumentation and the DUT. Because most sourced and measured signals pass through the switch matrix, any shortcomings in its specifications can affect measurement performance, speed and repeatability. At high frequencies, three specifications are particularly important: isolation, VSWR and insertion loss.[3]

- **Maximize isolation.** Leakage between signal paths can make it very difficult to measure low-power signals in the presence of one or more powerful signals. (This is most likely to occur when high- and low-power signals are routed through a switch matrix simultaneously.) ***Hint:*** Choose a switch with isolation specifications of 90 dB or better. This will reduce leakage and potentially minimize the need to route signals through physically separate switch assemblies.
- **Minimize VSWR.** High VSWR can cause phase errors and therefore affect the accuracy of vector and modulation measurements.[4] VSWR in a switch matrix is directly related to the VSWR of the coaxial switches used within the matrix, and the VSWR of an individual switch depends on its mechanical dimensions and tolerances.

3 For detailed information, please see the Agilent Custom Switch Matrices product note, publication number 5966-2961.

4 Phase repeatability is another important specification to consider when making these measurements.

Hint: You can further minimize VSWR by using cables that are short compared to the required bandwidth. If this is not practical because of wide bandwidth or mechanical requirements, the best alternative is to add insertion loss to the transmission lines via pads or lossy cables. This will reduce the amplitude of VSWR-induced ripples over the frequency range of interest, but at the expense of higher overall insertion loss.

- **Manage insertion loss.** This tends to become a problem at higher frequencies and is typically specified versus frequency in tabular or equation form. ***Hint:*** As a switch ages, its insertion loss may change so look for specifications such as "insertion loss repeatability" or "insertion loss stability" that are valid through the end of the product's expected lifetime. Knowing this type of worst-case value can help you manage your error budget.

Evaluating signal conditioners

As described in Hint 3, the DUT, its test requirements and its location may dictate the insertion of passive or active signal conditioners into the signal paths. These can be standalone devices or may be built into the switch matrix. Amplifiers, attenuators and frequency converters are the most commonly used signal conditioning devices.

Amplifiers

A signal might need additional gain if a precise amplitude measurement is required or if it is being sent over a long cable run. Several key specifications will help you determine an amplifier's suitability for your application.

- VSWR. Amplifiers are notorious for having poor VSWR. ***Hint:*** Alleviate VSWR problems by connecting an attenuator or an isolator (though these have limited bandwidth) to the amplifier output.

- Intermodulation. Amplifier bandwidth is important when measuring intermodulation distortion or spurious signals outside the bandwidth of the DUT. ***Hint:*** Beware of amplifiers with poor dynamic range or a low 1-dB compression point, which can produce enough intermodulation distortion to affect harmonic measurements in the presence of a strong fundamental.

- Spurs. Switching power supplies may cause spurs that are related to the switching frequency, which is typically 100-200 kHz. ***Hint:*** Avoid using amplifiers or any other devices that contain switching power supplies.

Attenuators

Electromechanical and electronic designs provide different levels of flexibility and precision in managing signal levels. Electromechanical attenuators use discrete switches that typically provide stepped resolution of 1 or 10 dB. Electronic attenuators provide virtually continuous settings with 0.1 or 0.25 dB resolution; however, those that use PIN diode-type switches can produce "video leakage" spikes that may contaminate measurement results. ***Hint:*** Cascade electromechanical and electronic attenuators as needed to provide greater control of attenuation.

Hint: Pay attention to the plating material used on attenuator connectors. As an example, nickel becomes nonlinear at high power levels and will cause intermodulation distortion. Instead, choose a higher quality conductor such as gold.

Frequency converters

When the DUT is remote from the test system, you can reduce insertion loss in long cable runs by using a downconverter to shift signals to a lower frequency range. ***Hint:*** At the test system, upconversion can be used to restore the signal to its original frequency, but it may be necessary to also apply filtering to remove unwanted frequency components created during the conversion processes.

Hint: If multiple signals, paths or conversions are used when making vector or modulation measurements, some form of phase locking must be used to ensure accurate results. You can do this by connecting the instruments and frequency converters to a common frequency reference and then measuring the phase of each signal relative to the reference signal.

Hint 5: Examine the operational attributes of switches

When deciding what type of technology to use in a switch matrix, it can be helpful to go beyond the electrical performance and look at operational attributes such as device longevity, power requirements and fail-safe operation.

Electromechanical versus electronic

With numerous moving parts and physical contacts, electromechanical switches tend to suffer from relatively rapid degradation, declining repeatability and limited life. In contrast, electronic switches have no moving parts so offer longer life and greater repeatability. In practice, the best choice depends in part on the actual number of switching cycles a system will require; consider the number of closures per test, the number of tests per day, the expected lifetime of the system and so on.

Another practical consideration is the power level of the routed signals. Switching of high power signals will damage most switches, lowering repeatability and shortening lifetime. ***Hint:*** To prevent the premature demise of either electromechanical or electronic switches, program the system instrumentation to reduce signal levels before opening or closing any switches in the matrix.

Latching versus non-latching

Internally, electromechanical switches use either latching or non-latching relays. Most latching types need a 100-200 msec pulse of DC power to open or close the relay.[5] Non-latching switches require constant powe—typically 24 V at 200 mA—to maintain contact. In a large switch matrix non-latching switches can generate enough heat within a system rack to affect measurement performance. ***Hint:*** If you choose to use non-latching switches, check the actual heat rise and be prepared to include additional cooling in the system rack.

Hint: It's essential to know how either type of switch will behave after a power failure or emergency shutdown. For maximum safety, select a switch matrix that returns to a known condition or configuration when power is restored. Non-latching switches are often the default failsafe choice because they open when power is removed and won't close again until power is applied by the test program. However, latching switches can be made fail-safe if they include hardware and firmware that will latch them into a safe mode at power down.

5 ***Another hint***: To minimize power requirements, some developers program the system to actuate these switches serially or in small batches, though this causes longer total switching time.

Advanced features: Built-in signal conditioning

One advantage of having a switch matrix in a system is that signal conditioning can be built into the matrix by the manufacturer. As an example, Agilent's custom switch matrices can be configured with a variety of devices, including amplifiers and attenuators; filters and isolators; and phase- and frequency-translating devices such as mixers, doublers, and dividers. These devices are permanently connected with semirigid coaxial cables and no additional external cabling is needed. The result is a compact, convenient, one-box solution.

Hint 6: Accelerate measurement set up and execution

Whether you gauge system performance as devices tested per unit of time, tests per unit of time or another time-based metric, measurement speed depends on two essential factors: the time required to set up the system and the time required to perform the measurement. The three major elements of any system– hardware, I/O and software—can help or hinder both processes. Chapter 7, *Maximizing System Throughput and Optimizing System Deployment*, offers several useful tips about software design, system I/O and low-frequency instrumentation. To complement that material, this hint adds new information specific to RF/microwave instruments and systems.

Fine tuning individual instruments

Any configurable device used in a system can become a bottleneck that limits measurement speed. The latest generations of RF/microwave instruments—signal generators, power meters, spectrum analyzers and network analyzers—offer flexible features and capabilities that can minimize bottlenecks and enhance system performance.

Signal generators

Many of these are available with built-in modulation and arbitrary waveform capabilities, potentially reducing the number of instruments in a system, simplifying system cabling and lessening software complexity. ***Hints:*** Instrument configuration may be somewhat complex and time consuming, but you can significantly reduce test time by creating states ahead of time, saving them in memory and then programming the system to recall the saved states as needed. If the system needs to load arbitrary waveform data during a test, download the minimum number of points and use binary format rather than ASCII.

Power meters

The biggest potential time savings come from models that offer built-in calibration capabilities that extend the cal interval from hours to months. ***Hint:*** Use digitizing power meters that offer wide video bandwidths and fast data sampling. Some of these units can generate 1,000 or more corrected readings per second, improving measurement accuracy and repeatability through averaging.

Spectrum analyzers

With any spectrum analyzer, the three key adjustments are frequency span, points per measurement and resolution bandwidth (RBW). ***Hints:*** Using the fewest necessary points and the widest possible RBW is the easiest way to reduce measurement time. Utilize a current-generation spectrum analyzer that automatically speeds things up by switching to Fast Fourier Transform (FFT) mode when measuring narrow spans.

Hint: To gain maximum benefit, use automatic input ranging selectively. When used to measure signals of rapidly varying amplitude, auto ranging may frequently change the input attenuator settings and slow the measurement. However, if signal levels are low and relatively constant, auto ranging can improve the signal-to-noise ratio and also shorten measurement time by allowing use of wider span and RBW settings.

Network analyzers

Calibration of VNAs can be very time consuming, especially the manual connection of shorts and standards. ***Hint:*** Agilent's line of electronic calibration or "ECal" modules automates this process, offering faster and more repeatable calibrations on one to four ports through a single connection. This method also reduces wear on test-port connectors and calibration standards.

Hint: The application of correction data is usually faster when performed inside the analyzer rather than externally in the system controller. With most VNAs you can save the calibration curve for a specific test and recall it as needed. Note that this method is more effective when used over a series of somewhat narrow frequency spans than with one extremely wide measurement span.

Conclusion

Every test system faces a unique set of challenges, but in all cases the ability to manage the direct and indirect tradeoffs among performance, speed and repeatability will help you achieve the required level of measurement integrity. The ability to manage crucial tradeoffs also applies to the selection of instrumentation, I/O connections and software elements for your test system. Agilent is helping reduce the number of compromises you have to make by offering system-ready instrumentation, PC-standard I/O and open software environments. By creating complementary system elements and supporting continually advancing standards such as LXI, Agilent can help you optimize—and even maximize—system performance now and in the future.

22. Calibrating Signal Paths in RF/Microwave Test Systems

Introduction

In any RF test system the ability to achieve instrument-port accuracy at the device under test (DUT) will enhance measurement accuracy and repeatability. Unfortunately, the non-ideal nature of the cables, components and switches in the paths between the instruments and the DUT can degrade measurement accuracy. Vector or scalar calibration is usually required to characterize and correct for this loss of accuracy.

The proper calibration method depends on both the type of measurement and the signal path. For example, measurements of gain and phase require complex-valued vector calibration, which is typically performed with a network analyzer. As another example, measurements of power levels and frequency content may be vector measurements of modulated signals (accurate phase information is essential) or scalar measurements of continuous wave (CW) signals. In these cases, vector measurements are performed with a network analyzer while the scalar measurements are typically performed with a signal generator and a power meter or spectrum analyzer.

This chapter provides an overview of three approaches that can be used to calibrate RF signal paths and produce accurate, repeatable measurements. It's important to note that these calibrations are a complement to—not a substitute for—the calibration of individual instruments within a system.

Understanding vector and scalar calibration

Within any test system, common elements such as fixturing, switching and cabling will introduce offsets and errors that will affect measurement accuracy. The two types of calibrations used to account for and correct these errors are vector and scalar calibration.

Vector calibration

This method requires measurements of both the magnitude and phase characteristics of the RF path. This can be done by either performing a network analyzer calibration at the DUT's input and output ports, or by using a calibrated network analyzer to measure the S-parameters of an RF path (see sidebar). The latter method provides a complete, complex-valued characterization of the signal path.

Scalar calibration

This approach characterizes only the magnitude characteristic of the RF path, which is equivalent to measuring only the magnitude portion of the S21 transmission coefficient in a vector calibration. A common technique involves driving one end of the path with a signal generator and measuring the signal at the other end with a power meter. The magnitude portion of the path response is determined by subtracting the source power level

Reviewing S-parameters

Scattering parameters, commonly referred to as S-parameters, are used to describe the way any device, component or path modifies an applied signal. The computed S-parameter coefficients are ratios of measured and applied signals at the ports of the device.

In S-parameter annotation, subscripts are used to indicate the ports of the device: the first number specifies the port that is measured; the second number specifies the port where the signal is applied. For example, S21 indicates a ratio of the signal measured at port 2 versus the signal applied to port 1. In the case of a two-port device (Figure 22.1) there are four S-parameters, each one describing the reflection or transmission of an applied signal:

- **S_{11}, Reflection Coefficient.** The ratio of the reflected signal measured at port 1 to the signal applied to port 1.
- **S_{21}, Transmission Coefficient.** The ratio of the transmitted signal measured at port 2 to the signal applied to port 1.
- **S_{22}, Reverse Reflection Coefficient.** The ratio of the reflected signal measured at port 2 to the signal applied to port 2.
- **S_{12}, Reverse Transmission Coefficient.** The ratio of the transmitted signal measured at port 1 to the signal applied to port 2.

To learn more, please see Application Notes 1287-3, *Applying Error Correction to Network Analyzer Measurements* (pub. no. 5965-7709E), and 1364-1, *De-embedding and Embedding S-Parameter Networks Using a Vector Network Analyzer* (pub. no. 5980-2784EN).

Figure 22.1. Modeling the RF signal path as a two-port device provides the S-parameters needed for calibration and correction.

setting (in dBm) from the measured power level (also in dBm). This is repeated at multiple frequencies across the band of interest to determine the overall magnitude characteristic.

Scalar calibrations can achieve very good results as long as high quality components, adapters and cables are used in the system. This helps minimize measurement uncertainty and increase measurement repeatability. However, when compared to a full vector calibration, scalar calibration is less likely to detect any changes in impedance match along a signal path.

Comparing the two methods

The best choice of calibration method depends on factors such as the test specification and its measurement and accuracy requirements, the likelihood of inaccuracies internal to the measurement instruments, and the availability of a network analyzer. The advantages and disadvantages of each method are summarized in Table 22.1.

Table 22.1. Comparing the advantages and disadvantages of vector and scalar calibration.

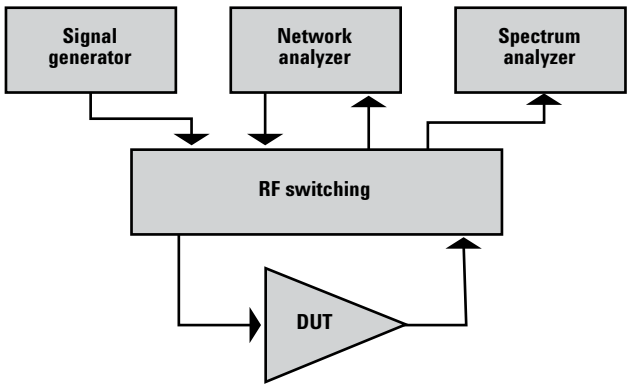
Calibration type	Advantages	Disadvantages
Vector	• Enables complete characterization of the path and, therefore, more accurate measurements • Allows adapter embedding and de-embedding • Provides excellent confidence in path integrity	• Higher cost than scalar because network analyzer is required • Doesn't account for inaccuracies internal to instruments connected to the signal path
Scalar	• Lower cost approach (network analyzer not required) • Can compensate for inaccuracies internal to instruments connected to the path, which may result in better overall accuracy	• Not a complete characterization of the path (magnitude only) • Doesn't support adapter embedding or de-embedding • Provides less confidence in path integrity

Defining our reference point

We will describe the application of vector and scalar calibration to the types of RF signal paths that are present in most systems. The basic system diagram shown in Figure 22.2 will be our reference point as we explore three different methods that can be used to characterize RF paths:

- Vector calibration of a network-analyzer path
- Vector calibration of a non-network-analyzer path
- Scalar calibration of a non-network-analyzer path

Figure 22.2. The essential elements of a simplified RF/microwave test system



Performing vector calibration of network-

analyzer paths

Network-analyzer paths are those that connect a network analyzer to the DUT. A vector calibration enables the network analyzer to precisely measure the complex-valued S-parameters that fully describe changes in magnitude and phase versus frequency. S-parameter measurements of the DUT are made using a swept continuous wave (CW) signal generated by the network analyzer (Figure 22.3).

Network analyzers have built-in routines that allow the instrument to compensate for any cabling and RF components that lie between the instrument and the DUT. Mechanical or electronic standards with known characteristics (e.g., shorts, opens and throughs) are used for this purpose. By substituting the standards for the DUT and measuring the response, the network analyzer can generate and store error terms that are recalled as needed to correct measurements of the DUT. In this case, the path data is retained in a set of error terms stored in the analyzer's memory.

When calibrating network-analyzer paths it is important to use the same conditions that will be used to test the DUT: all switch settings, power levels, frequency ranges and so on should be identical. This is especially important if the DUT is an active device that has linear and nonlinear operating modes.

Removing adapter effects with embedding

The connection of a mechanical standard or electronic calibration module to the DUT cables will often require an adapter on one or both ports of the DUT cables. The addition of these adapters may induce errors such as impedance mismatches, reflections and delays.

You can remove these effects by using a process called *adapter embedding*, which moves the calibration plane towards the network analyzer (Figure 22.4) and ensures characterization of just the signal paths of interest. In this example, the embedding process moves the calibrated reference plane to the end of the DUT cable from the end of the adapter, where the calibration standards are attached during network analyzer calibration.

Performing vector calibration of non-

Figure 22.3. Network-analyzer paths to and from the DUT

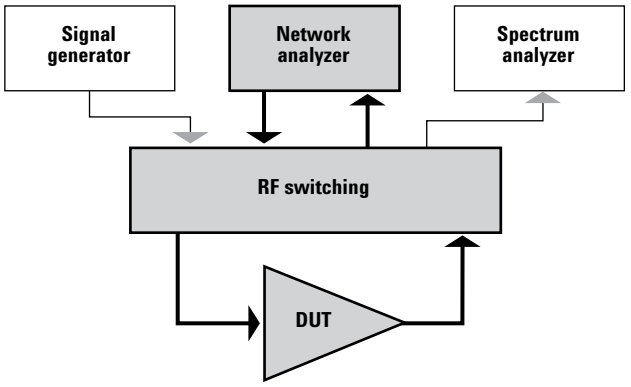
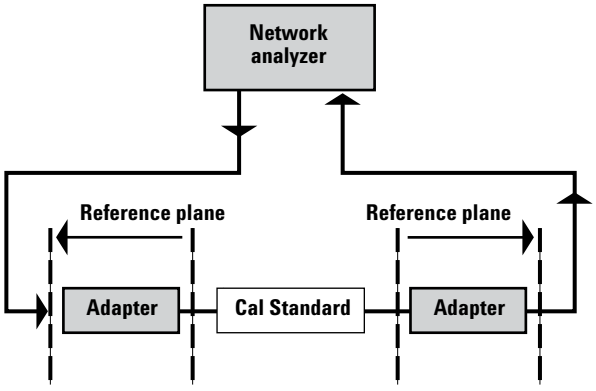


Figure 22.4. Adapter embedding moves the reference plane closer to the network analyzer, ensuring characterization of just the signal paths of interest



network-analyzer paths

Non-network-analyzer paths connect instruments other than a network analyzer to the DUT. The measured signals may be either modulated or CW.

Vector calibration of these paths is accomplished by connecting a calibrated network analyzer to the path and measuring its S-parameters. Prior to measuring the RF path, the network analyzer is calibrated in a standalone configuration with special calibration cables. The results of these path-calibration measurements are stored in the system controller for later recall and application.

Removing adapter effects with de-embedding

Adapters may be required to connect the network analyzer to each system path during the calibration process. The effect of these adapters is usually very small if high quality adapters are used; however, if their effect is significant it can be removed using a process called *adapter de-embedding.* Adapter de-embedding effectively moves the calibration plane away from the network analyzer (Figure 22.5) to ensure characterization of just the signal path of interest. In this example, the de-embedding process moves the calibrated reference plane from the end of the calibration cable (where the calibration standards are attached during network analyzer calibration) to the end of the adapter where the system path is connected.

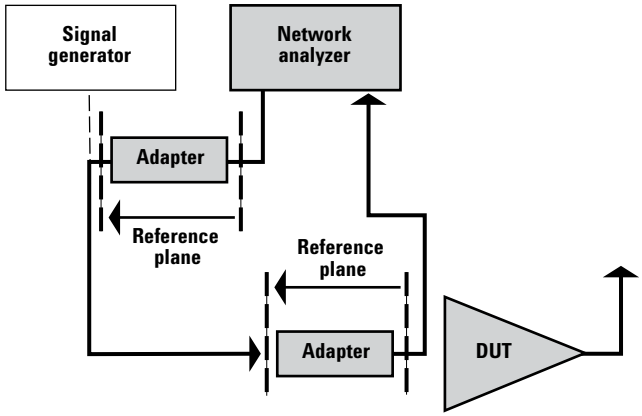
Deriving additional benefits

In addition to high accuracy, two additional benefits come from network-analyzer characterization of the system paths used for modulated DUT measurements. One is greater confidence in path integrity, which comes from the ability to easily measure characteristics such as the return loss of the path (S11 and S22). This allows for a more comprehensive self-test of the system and helps minimize the uncertainties caused by input and output mismatches.

The other noteworthy advantage is the ability to modify the path data after a system calibration is completed. This makes it possible to account for separately characterized adapters such as test fixtures or circuit boards that interface to the DUT. Combining these elements with existing path data requires that all S-parameters be known for both the adapter and the path.

Performing scalar

Figure 22.5. When calibrating paths such as signal-generator-to-DUT-input, adapter de-embedding moves the reference plane away from the network analyzer, ensuring characterization of just the signal path.



calibration of non-network-analyzer paths

While the primary measurement instrument for vector calibration is a network analyzer, the main instrument used for scalar calibration is a power meter, which is the most accurate way to measure absolute power. Scalar calibration also requires a signal generator, which is used to provide signals of known frequency and power. This method typically requires a two-part process that first characterizes the pathway to the DUT input then the signal path from the DUT output.

Characterizing the path to the DUT input

The first path to measure is the one that connects the signal generator output to the DUT input (Figure 22.6). You can characterize the loss through this path using a power meter connected to the end of the DUT cable (in place of the DUT input).

The signal generator is configured to provide signals at the range of frequencies and power levels that will be used when testing the DUT. The power meter measures the power output at each frequency and power level, and the offsets (in dB) are calculated and stored in the system controller for later use. The calculated offset accounts for path loss as well as some inaccuracies internal to the signal generator.

This is a scalar measurement because only the magnitude is calculated; there is no phase information. This is usually acceptable because the absolute phase of the signal incident at the DUT input is not important as long as the magnitude response is relatively flat and the phase response is linear over the frequency band of interest.

Note a key assumption here: The accuracy of this method depends on minimal mismatch between the input impedance of the DUT and the input impedance of the power meter. It is important to verify these impedances because a large difference will cause significant measurement errors.

Characterizing the path from the DUT output

To complete the scalar calibration, we measure the signal path from the DUT output to the spectrum analyzer (Figure 22.7). The loss through this path can be characterized by applying a known signal source, reading the power level measured by the spectrum analyzer then subtracting the path to the DUT input (described in the previous section).

Figure 22.6. By substituting a power meter at the DUT input, you can measure loss through the input path.

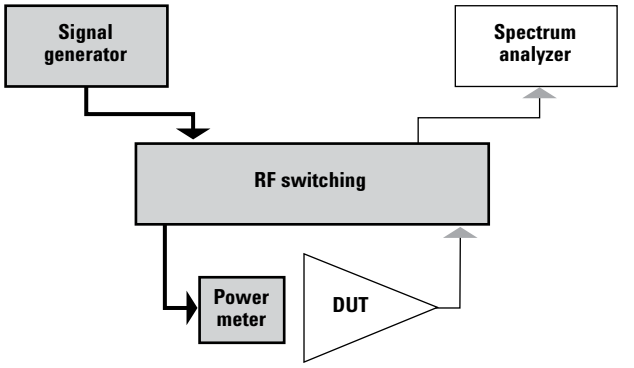
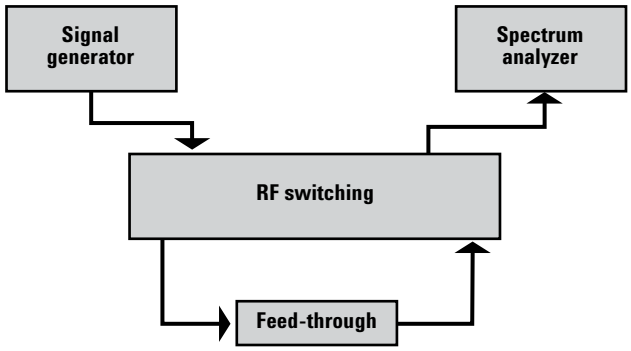


Figure 22.7. By substituting a feed-through for the DUT, you can measure loss from the DUT output to the spectrum analyzer.



You can do this by (1) using a feed-through to connect the DUT input cable directly to the DUT output cable, (2) setting up the signal generator to output the required range of frequencies and power levels and (3) making power measurements with the spectrum analyzer.

The spectrum analyzer should be configured just as it will be for DUT measurements. This is especially true of the input attenuator settings, which often cause wide variations in the spectrum analyzer's input impedance. The resulting calculated offsets will account for path loss as well as some inaccuracies internal to the spectrum analyzer.

Note a key assumption here as well: The accuracy of this calibration depends on the impedance of the DUT output cable being very similar to the input impedance of the power meter. It is important to verify these impedances because a large difference will cause significant measurement errors.

Measuring adapter effects

Accounting for adapters necessary to perform scalar-path calibrations is usually accomplished by estimating or measuring adapter loss at various frequencies of interest and then accounting for those losses in the offset calculations. However, this is much less accurate than the adapter embedding and de-embedding procedures described in the vector calibration sections.

Conclusion

The use of vector and scalar calibration can increase measurement accuracy by helping you correct for errors in the RF signal paths. Each method has advantages and disadvantages, and the choice depends on both the type of measurements you're making and the nature of the signal path.

This chapter reviewed three different to methods of characterizing the RF path: vector calibration of network-analyzer paths, vector calibration of non-network-analyzer paths and scalar calibration of non-network-analyzer paths. When performing vector calibration of network-analyzer paths, the technique of adapter embedding ensures characterization of only the signal paths of interest. Adapter de-embedding provides the same benefits when you're performing vector calibration of non-network-analyzer paths. Within the context of your specific measurement needs, each method provides valuable calibration benefits.

Glossary of Test-System Development Terms

Adapter — the LAN card and connector that provides an electrical interface to the network

ATE — Automated test equipment

ATS — Automated test system

AWG — Arbitrary waveform generator

Bridge — a LAN device that connects segments of a network

CASS — Consolidated Automatic Support System

COTS — Commercial off-the-shelf

DDNS — dynamic domain name server; a service that allows a network device to establish its host name when it connects to the network. This lets other devices use that host name with DNS to find the device's IP address and connect to it.

DHCP — dynamic host configuration protocol; a method of automatically obtaining an IP address for a LAN-connected device (e.g., PC, router, instrument, etc.)

DMZ — De-militarized zone; a firewall configuration that helps secure the private LAN

DNS — domain name server; maps specific names to IP addresses, enabling use of names in place of IP addresses in testprograms

DoD — United States Department of Defense

DUT — device under test; the component, subassembly or product to be measured by the test system

eCASS — The modernized version of CASS

Ethernet — a specific LAN technology that is the dominant implementation of the physical and data link layers; also known as IEEE 802.3

Firewall — a hardware device or software program (or combination) that protects a computer network from unauthorized access

Gateway — a hardware device that connects devices that use different standards and protocols (e.g., LAN to GPIB)

GPIB — General Purpose Interface Bus; the dominant 8-bit parallel I/O connection for test equipment and test systems

Hub — a multi-port LAN device that connects multiple devices together, usually in a star topology

ICS — Internet connection sharing

IF — Intermediate frequency

IP— Internet protocol; requires an address to communicate

IPX — Internetwork Packet eXchange; a communication protocol used in the Novell Netware network operating system

LAN — local area network

LXI — LAN eXtensions for Instrumentation

MAC — media access control; every LAN device has a unique MAC address

NAT — network address translation; maps private addresses to one or more public addresses to enable access to an intranet or the Internet

NetBEUI — NetBios Extended User Interface; a network communication protocol used in many versions of Windows

NxTest — Next-generation Automatic Test Systems

OEM — Original equipment manufacturer

PCI — Peripheral Component Interconnect

PXI — PCI eXtensions for Instrumentation

RF — Radio frequency

Router — a LAN device that joins multiple networks and enables creation of small, private networks

SI — Synthetic instrumentation

SIWG — Synthetic Instruments Working Group

SPX — Sequenced Packet eXchange; a communication protocol used in the Novell Netware network operating system

Subnet — a group of connected network devices; used to partition networks into segments for easier administration

Subnet mask — a setting that accompanies an IP address and defines the boundaries of a subnet

Switch — a LAN device that connects multiple devices to a single LAN line; however, unlike a hub, it preserves full network bandwidth to each device

TCP/IP — Transfer Control Protocol and Internet Protocol; the two standards that provide the data communication foundation of the Internet

Technology insertion — The introduction of new or improved hardware or software capabilities into an existing system

TPS —Test program set

USB — Universal Serial Bus; designed to replace the RS-232 and RS-422 serial buses used in PCs

VME or VMEbus — Versa Module Eurocard

VXI — VME eXtensions for Instrumentation

Agilent Email Updates

www.agilent.com/find/emailupdates
Get the latest information on the products and applications you select.

Agilent Direct

www.agilent.com/find/agilentdirect
Quickly choose and use your test equipment solutions with confidence.

Agilent Open

www.agilent.com/find/open
Agilent Open simplifies the process of connecting and programming test systems to help engineers design, validate and manufacture electronic products. Agilent offers open connectivity for a broad range of system-ready instruments, open industry software, PC-standard I/O and global support, which are combined to more easily integrate test system development.

LXI

www.lxistandard.org
LXI is the LAN-based successor to GPIB, providing faster, more efficient connectivity. Agilent is a founding member of the LXI consortium.

Remove all doubt

Our repair and calibration services will get your equipment back to you, performing like new, when promised. You will get full value out of your Agilent equipment throughout its lifetime. Your equipment will be serviced by Agilent-trained technicians using the latest factory calibration procedures, automated repair diagnostics and genuine parts. You will always have the utmost confidence in your measurements.

Agilent offers a wide range of additional expert test and measurement services for your equipment, including initial start-up assistance onsite education and training, as well as design, system integration, and project management.

For more information on repair and calibration services, go to

www.agilent.com/find/removealldoubt

www.agilent.com

For more information on Agilent Technologies' products, applications or services, please contact your local Agilent office. The complete list is available at:
www.agilent.com/find/contactus

Americas

Canada	877 894 4414
Latin America	305 269 7500
United States	800 829 4444

Asia Pacific

Australia	1 800 629 485
China	800 810 0189
Hong Kong	800 938 693
India	1 800 112 929
Japan	81 426 56 7832
Korea	080 769 0800
Malaysia	1 800 888 848
Singapore	1 800 375 8100
Taiwan	0800 047 866
Thailand	1 800 226 008

Europe

Austria	0820 87 44 11
Belgium	32 (0) 2 404 93 40
Denmark	45 70 13 15 15
Finland	358 (0) 10 855 2100
France	0825 010 700
Germany	01805 24 6333*
	*0.14€/minute
Ireland	1890 924 204
Italy	39 02 92 60 8484
Netherlands	31 (0) 20 547 2111
Spain	34 (91) 631 3300
Sweden	0200-88 22 55
Switzerland (French)	41 (21) 8113811 (Opt 2)
Switzerland (German)	0800 80 53 53 (Opt 1)
United Kingdom	44 (0) 118 9276201

Other European Countries:
www.agilent.com/find/contactus

Revised: May 7, 2007

Printed in USA, May 7, 2012
5989-5367EN

Agilent Technologies