

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

Optimizing System Design for Rapid Development, Fast Execution and Re-use

Application Note 1481

CONTENTS

Overview 1
Introduction 2
Architecture 2
Composition of a Functional Test System 3
- Computing 3
- Instrumentation 4
- Switching 9
- DUT Power 9
- Loads 10
- Mass Interconnect 10

Test System Example 11
- Architectural Choices 11
- System Design 11
- Write the Software 14

Conclusion 15
References 15

When developing a test system from scratch, the test engineer has many choices of instrumentation and software available. LAN- and USB-based rack and stack instruments are making strides versus their GPIB cousins. VXI and other cardcage-based platforms remain viable too. Test Executives and Microsoft's® Visual Studio.NET development environment, along with "helper" toolkits, are making software development easier than ever. But there are design choices that should be made up front that can improve performance and make it easier to adapt to new applications as they arise.

This application note explains instrumentation speed/performance tradeoffs, test development environments and architectural differences that the test engineer needs to know about in order to make the right decisions. To demonstrate the thinking process for system design, a system is designed from the ground up that can test an Electronic Throttle Module (ETM) that responds to a brake input and a PWM signal from an accelerator and controls an electric motor which in turn operates a butterfly valve.

Microsoft is a U.S. registered trademark of Microsoft Corporation.

Agilent Technologies

INTRODUCTION

Functional Test Systems are a mixed blessing. The ability to collect data on an electronic module and use that data to improve a process or a design has made it possible to have quality that was undreamed of just a few years ago. But it has also resulted in pressure to do more with less, faster. Today, the test engineer must be able to create new systems fast, optimize their speed of execution and adapt the system to new modules rapidly. These are not trivial tasks. This application note will delve into some concrete ways to accomplish them.

ARCHITECTURE

A test system is essentially a group of subsystems that work together to test a particular device or range of devices. There are many decisions to be made for every subsystem. It therefore pays to think about all these issues before the first piece is ordered. Time spent up-front defining the architecture of a system saves much more time later. But without an understanding of the tradeoffs, the test engineer could end up staring at a blank sheet of paper for a long time.

When an architect designs a house, what factors are taken into account? Esthetics, safety, traffic flow, heat gain, street view, size, snow load, cost, future expansion, drainage, optimal location of rooms, and kitchen use are typical things to be considered. Yet, when test system hardware is designed, how often are things thrown together and then beaten into submission? Instead, many of the same things should be considered: esthetics, safety, I/O, heat, size, cost, future expansion, optimal location of parts, and so on. Once these decisions have been made, test requirements can be used to further narrow down a system design for the expected range of devices.

Unlike hardware, a problem of software is there are too many possible solutions to a given problem. The architect has to work hard to understand and interpret what the software system should do. Sometimes the architect must work through a long process of exploration in order to determine what the precise needs are. This can be the most difficult part of the design to get right. Alan Cooper, a well known software architect, says, “Real software architects do effective planning-they do research, work with stakeholders, distill what the real goals are. Then they begin to synthesize a real solution and hold the hands of the builders as they build.” [1]

A major consideration in test system architecture is the expected use model of the system, and it’s generally different for the four stages of product testing: R&D, Design Validation (DV), Production Validation (PV), and Manufacturing Test. Each stage generally requires more rigor than the last.

When the use model is understood and expressed precisely enough, the system architect switches into a high level design mode. The hardware and software must be partitioned into blocks or layers or components that can work together to accomplish the desired result. The way these parts are partitioned and the way they communicate and interrelate has a huge effect on the cost, performance, maintainability and usability of the system. The architect also establishes important standards and patterns for system-wide behavior, such as how error handling will be done.

The key for both hardware and software design is planning. It is far easier to optimize a well-organized system than one that was thrown together in a hurry to meet a deadline. Ironically, time spent planning usually results in an overall time savings, because less time is spent debugging software, or tracing down the cause of faulty measurements.

COMPOSITION OF A FUNCTIONAL TEST SYSTEM

A functional test system, whether it's used for design validation, production validation or manufacturing test, is typically made up of six major components, or subsystems:

- Computing (Computer, software and I/O)
- Instrumentation (All measuring and stimulus instruments)
- Switching (Relays that interconnect system instrumentation and loads to the DUT)
- DUT Power (Power to the Device Under Test)
- Loads (Parts connected to output pins on the DUT)
- Mass Interconnect (DUT-to-System wiring interface)

The job of the test engineer is to put these subsystems together efficiently. But how?

Computing

Here are some of the issues one faces in the computing subsystem:

- Embedded or external PC
- If external PC:
  - Choice of control interface(s) – GPIB, LAN, USB, IEEE-1394 (FireWire), MXI-2, MXI-3, RS-232C
  - How many PCI slots to use for instrumentation, if any? What about serviceability of cards that are plugged into the PCI bus?
- Graphical or Text-based software development and runtime environment
- For manufacturing uses, buy a commercial test executive or design one in-house
- Instrument Driver layers – VISA, SICL, Tulip Drivers, Passport Drivers, VISA-COM, IVI-C, IVI-COM, VXI*Plug&Play* (VXIPNP), Visual Studio.NET Wrappers, C DLLs
- Operating System (O/S) versions and upgrades
- Application software upgrades
- Licensing
- IT support
- Data logging software – Excel, Access, XML, proprietary
- Enterprise connectivity
- And many more

How does the test engineer choose reasonable solutions to these problems? This paper will not attempt to address all of these. Suffice it to say that there are many things to think about that may not be immediately obvious. However, a few of the most pressing issues are worthy of a closer look.

A major concern with the use of a PC in a test system is whether or not to use an embedded PC, i.e., one that fits inside an instrumentation cardcage, or one that is external and is cabled to the instrumentation. At first glance, the embedded PC seems like a good choice. It fits inside an existing cage, so rack space is used 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 do not have the room necessary to hold some modern peripherals. They also do not tend to keep up with the latest PC technology, so they are often a generation behind in processor type and speed. Additional money can be saved by using instruments that do not require special interface cards, since a PC that has industry standard interfaces like USB, LAN and FireWire built-in can be purchased from many sources. This can also save money when it comes to support, since the PC will not have to be opened to service vendor-specific interface cards.

Another major consideration is the choice of software. As software evolves, old code breaks. That's a fact of life. A simple O/S upgrade could shut a line down while the test engineer figures out why the application software isn't working right anymore. Even within the Microsoft Windows environment, one has a choice of Windows NT, Windows 2000 or Windows XP. Patches for these O/Ss are issued fairly often. One must be sure that the O/S, the Application Development Environment, the hardware drivers and other related software continue to work together as they evolve. It is wise to have a strategy, worked out with the IT department, for evaluation of software upgrades.

Finally, the application development and runtime environments are critical choices. Graphical languages are great for R&D and Design Validation (DV), but they can become cumbersome in manufacturing because of the unavoidable complexity of the test plan. In manufacturing, it is easiest to use a Test Executive. This is a program that helps create and

sequence tests, run Graphical User Interfaces (GUIs) for operators, log results, create failure and statistical reports, interface to automation, and re-use previously written measurement routines.

Microsoft's new .NET framework provides a widely supported environment for test and GUI development that is rich in features and that integrates well with instrumentation. The .NET Framework is basically a very web-friendly set of services that allow applications to share information via a language called XML (Extended Markup Language), and it can be used with Test Executives or used standalone via the Visual Studio.NET (VS.NET) development environment. VS.NET offers users a common development environment for Visual Basic, Visual C++ and the new Visual C# (C-sharp) that can improve productivity markedly. For example, the programming for the example test system discussed later in this application note was developed by a Visual Studio.NET novice (the author) in about a week.

Instrumentation

Probably the hardest part of getting a test system to work right is getting the instruments to take the readings that are desired. There are five facets to this problem, listed here in the order they are typically encountered:

- Picking the right instruments
- Connecting the instrument to the PC and installing drivers
- Making sure the right signal is actually getting to the instrument
- Understanding the nuances of the instrument to get it to take a good reading
- Getting the best speed out of the measurement

Picking the right instruments

The types of instruments needed vary widely depending on the application. RF and optical testing is generally very different from the low frequency testing described in this application note. However, there are several universal questions that must be answered in order to select measurement and stimulus instrumentation properly:

1. **AC Stimulus** — How many dynamic (AC) signals need to be applied simultaneously? This determines the number of channels of arbitrary waveform or function/signal generator that are required. For applications needing more than about four channels, an instrumentation cardcage is the best solution. For applications needing low cost, few channels or isolated outputs, rack and stack instruments are a better solution.
2. **DC Stimulus** — How many static (DC) signals need to be applied simultaneously? This determines the number of channels of DAC (digital-to-analog converter) that will be required.
3. **Measurements** — What types of measurements need to be made, and how many simultaneously? For measurement of Volts, Ohms or Amps, a digital multimeter (DMM) is needed. The accuracy and precision that is required over the foreseen operating temperature range of the system, as well as the desired measurement speed helps determine the type of DMM. Glitch detection requires a logic analyzer, oscilloscope, digitizer or event detector. Waveform analysis requires an oscilloscope or digitizer.
4. **Protocols** — Any special serial data protocols? This determines the need for instruments to handle things like CAN, ISO-9141, J1850, and many more.
5. **Power Supplies** — What power supply levels need to be applied, and how many at once? Are leads long enough that remote sense should be used? Choice of supply can dramatically impact system throughput, since waiting for power supplies to settle can be one of the most time-consuming things in a typical testplan. If possible, supplies should be chosen that offer fast down-programming and accurate output current measurement.

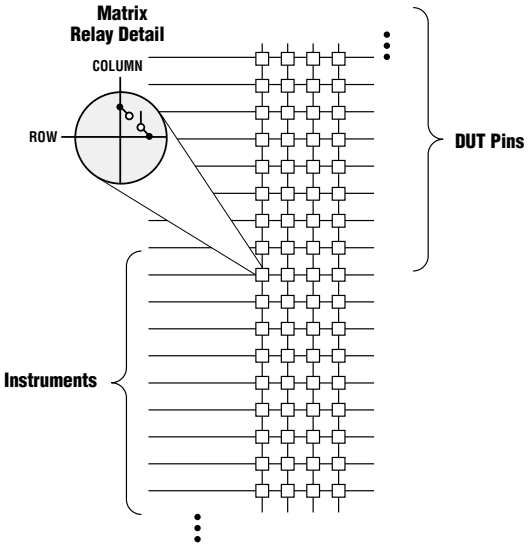


Figure 1: *A 4-wire instrument bus is a useful way to route instruments to each other and to DUT pins*

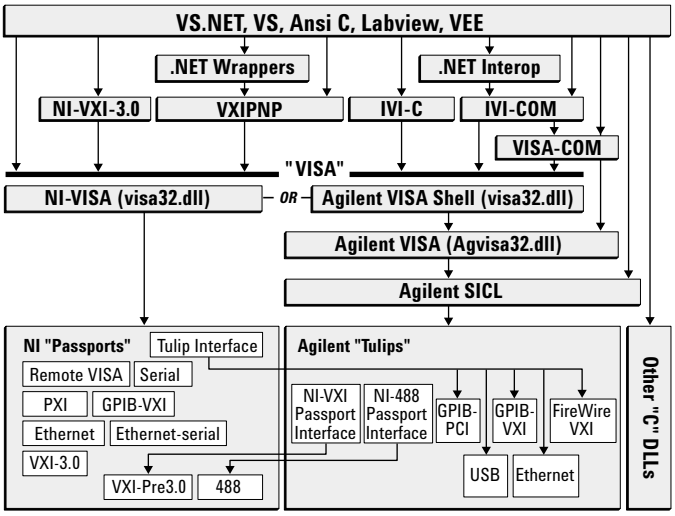


Figure 2: *I/O layers in a typical test system with a mixture of hardware and software vendors*

The number of measurements or stimulus that must be applied at the same time helps to determine the switching structure that must be used. Typically, a 4-wire bus is a good solution, as it allows the four terminals of a DMM to be connected to the DUT for 4-wire Ohms measurements. It is also seldom necessary to have more than two isolated instruments or three single-ended instruments active at once, since electronic modules usually contain built-in test routines that can be activated programmatically to allow tests on one function of the module at a time (called "DUT-Assisted Test"). **Figure 1** shows a switching structure of this type. A fifth bus is also sometimes desirable to allow four single-ended instruments to be used with a common ground. Note too that "rows" can be the vertical lines and "columns" the horizontal lines or vice versa. Matrices can be configured in a variety of ways. Relays are covered in more detail in the "Switching" section.

Connecting the instrument to the PC and installing drivers

Assuming the cabling from the PC to the instruments works as planned, the next problem is usually finding a match between operating system, application software and instrument driver. Although standards such as VISA, VXI*Plug & Play*, IVI and IVI-COM have been adopted, the resulting proliferation of acronyms has become bewildering. Two of the largest providers of instrumentation libraries, Agilent Technologies and National Instruments (NI), worked together to try to make their instruments work with each other's software. **Figure 2** shows the resulting cross-connects between NI's "Passport" hardware drivers and Agilent's "Tulip" hardware drivers. The result is not perfect, but it does indicate that it is possible to intermix many types of interfaces – LAN, USB, RS-232C, MXI, FireWire, GPIB – and still get the system to work. As a result, it is not necessary to focus on just one style of interface. The test engineer can now choose the instrument that is right for the task at hand.

Making sure the right signal is actually getting to the instrument

Common problems in test system design are ground loops, sneak current paths, shorts, opens, signal loss and stray capacitance. Two solutions to these problems, both of which should be used, are:

1. Create a diagnostic test plan. If the system is designed with a relay matrix arrangement as shown in **Figure 1**, it is fairly easy to route stimulus signals to measurement devices, which verifies the relays, the cables and the instruments. It is worth the time to provide a way to run a good self-test on the system. This can help find problem areas that prevent the right signal from getting to the instrument, both during development and after a system is put into regular use.

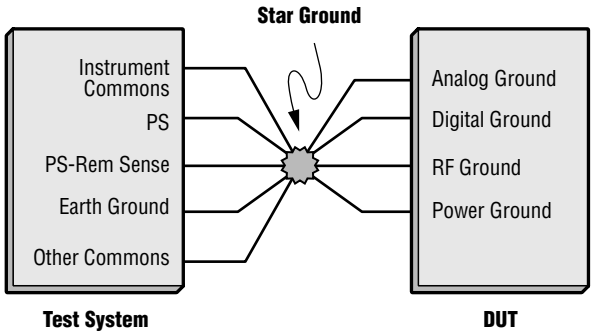


Figure 3: *Star ground minimizes noise and eliminates ground loops.*

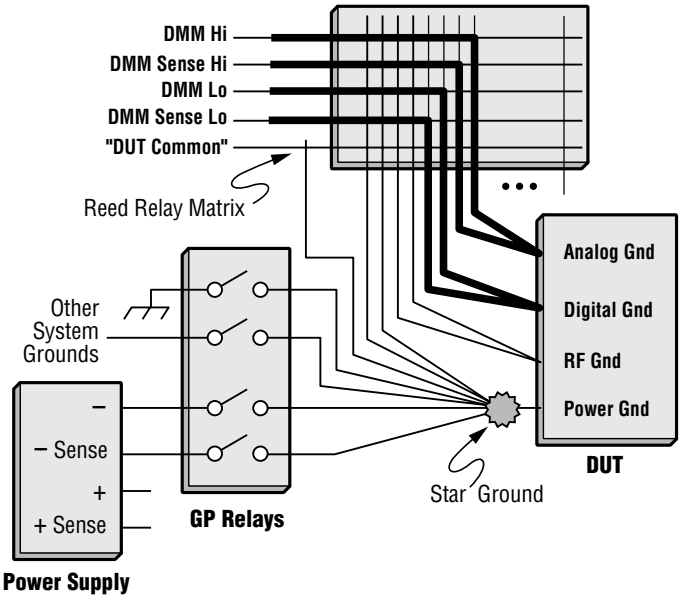


Figure 4: *Star ground with switching through General Purpose (GP) relays to allow continuity tests. DMM path for a 4-wire Ohms measurement between Analog and Digital Ground is shown in bold.*

6

2. Use a "star" ground system. 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 3**. This eliminates ground loops and contributes to quiet readings. Note however, that connecting grounds in this manner does not allow for continuity measurements between ground pins. The solution is to add relays to the ground pins as shown in **Figure 4**. It is usually sufficient to pick the highest current ground pin (Power Gnd in this case), and reference all measurements to it once internal continuity to the other ground pins has been verified.

Understanding the nuances of the instrument to get it to take a good reading

It can take a while to figure out the right set of commands to send to an instrument to get the fastest reading or best source setup consistent with the accuracy and resolution demanded by the test spec. VXIPNP and IVI-COM drivers along with Microsoft's IntelliSense functionality (**Figure 5**) in the Visual Studio.NET environment can make the task of figuring out the right commands fairly easy. IntelliSense is a command completion feature. The programmer types the symbol name of the instrument, such as `"MyHp34401"`, and then types a decimal point ("dot"). At that point, IntelliSense pops up all the available functions that can be used with that instrument along with a description of the function. If any parameters are required, they are shown along with their data types once the function is selected. Use of this feature can make it unnecessary to consult instrument manuals once the programmer is familiar with an instrument.

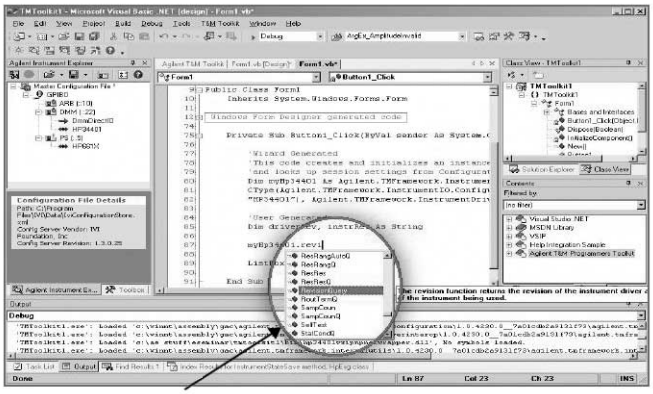


Figure 5: *IntelliSense in the VisualStudio.NET environment helps to quickly discover available instrument commands.*

Getting the best speed out of the measurement

In order to make a test system execute fast, there are several tenets that must be observed:

1. Use fast instruments. This 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,

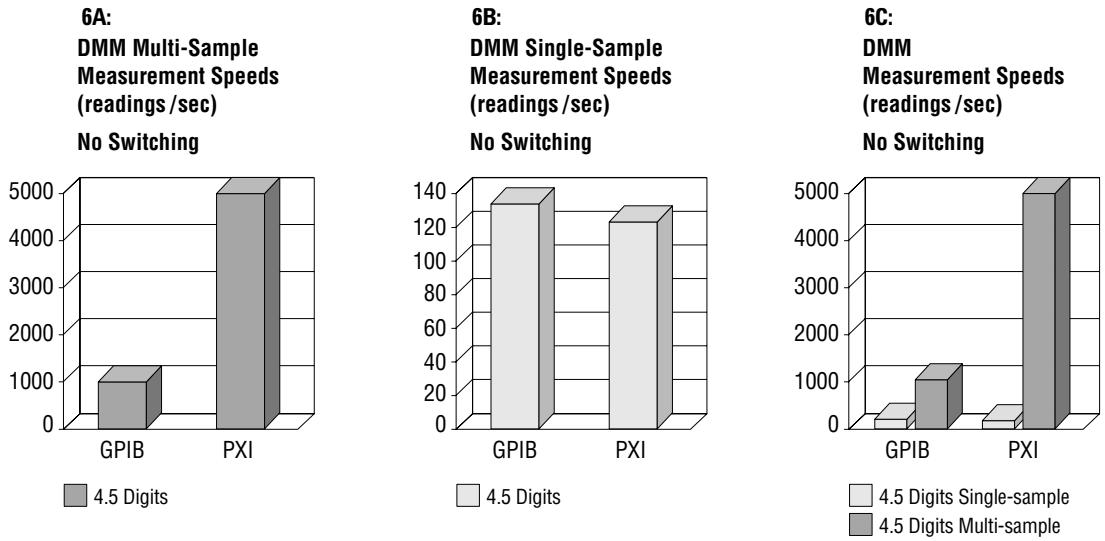


Figure 6: *4.5-digit transactional measurement speed of a GPIB and PXI DMM. "C" is a combination of A and B shown on the same scale.*

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.

It is also important to look hard at the instrument specifications. Often a measurement speed specification is related to the speed per reading when thousands of samples are taken, which is a data acquisition use model. With a functional test model, it is far more common to close some relays, take a measurement, open those relays and move on to another measurement. In such a case, the DMM's single-sample reading speed is most important and is dramatically slower than the fastest possible multi-sample reading speeds. Measurement speed without taking switching into account is shown in **Figure 6A** and **6B**. When viewed on the same scale, as shown in **Figure 6C**, it is evident just how big this difference is. At higher resolutions (5.5 and 6.5 digits), single-sample readings are faster with PXI DMMs than with GPIB DMMs. However, when relay times are taken into account, the total speed of the readings for both types of DMMs is generally less than 10/sec, so such readings tend to be done only when the extra resolution is absolutely necessary. The resulting smaller number of measurements taken at high resolution makes their contribution to overall test time less of a factor.

2. Minimize instrument state changes. Voltage, current and resistance measurements are so common that manufacturers have expended a lot of effort making the measurements as fast as possible. But even after one chooses a fast DMM and a fast interface, random range and function changes can still interfere with fast tests, as these are still relatively slow. To compensate for this, tests should be ordered such that AC and DC readings on the same instrument are not intermixed. It is also helpful to pick a range that gives the needed resolution for most measurements and then keep it there. Autozero should be set to "once" or "off", as it doubles the measurement time. (The execution time of a production test program at one manufacturing site was once improved from 90 seconds to 60 seconds merely by changing autozero from "on" to "once"!) This should only be done, however, if the temperature drift in the system is minimal. Otherwise, an autozero should be performed periodically.

3. Use fast I/O. It's sometimes thought that GPIB is slow and LAN is fast. But is this really true? Tests indicate that the extra overhead of all the layers of LAN and VISA actually make LAN slower than or comparable to GPIB unless a lot of data are being transferred. For the typical transactional model of electronic functional test, LAN may not be the best choice for measurement instruments, though it can do nicely for stimulus and power supplies where frequent transfer of a lot of data is not usually required. LAN tends to run at its fastest if a direct socket connection is made. It is also useful when remote access to an instrument is required, because instruments with LAN interfaces can have built-in web servers.

Relative I/O speeds from PC to Agilent 33220A Function Generator

INTERFACE	FUNCTION CHANGE	FREQUENCY CHANGE	4K ARB	64K ARB
GPIB	99 ms	4 ms	20 ms	330 ms
USB 1.1	100 ms	5 ms	10 ms	185 ms
USB 2.0	99 ms	5 ms	8 ms	100 ms
LAN-Socket LAN-VXI-11	100 ms	5 ms 6 ms	8 ms 14 ms	110 ms

NOTE: VXI-11 (non-socket) speeds are preliminary.
Data taken on HP Kayak XU800 with an 800 MHz processor running Windows XP.

Table 1: *I/O Speed comparison for USB/LAN/GPIB capable instrument*

USB is about 3x faster than GPIB. FireWire is about 4x faster than 100Mbit LAN, so it is the best choice for a connection to VXI. MXI is faster yet, but requires a proprietary interface card in the PC. **Table 1** shows the relative speeds for various operations for a stimulus instrument having GPIB, USB and LAN interfaces [2]. The instrument's internal speed clearly dominates setup changes, making I/O choices seem moot, but download speeds get much better with LAN and USB when large amounts of data are involved.

4. Use relays wisely. 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. Armature relays (which typically switch in 10-20 ms) should be used for higher current loads, and tests that use them should be grouped so that they can stay connected as long as possible.

5. Don't use an oscilloscope to do a digitizer's job. Oscilloscopes are slow and have generally poor (8-bit) resolutions, although they have high bandwidth. Digitizers acquire data fast and have good (14-bit) resolution. Sometimes a DMM can be used as a digitizer. The point is, instrumentation should be chosen based on the required accuracy and resolution first, and the resulting list of instruments narrowed based on execution speed. For R&D and DV applications, however, an oscilloscope is a desirable addition, especially when debugging the digitizer. Since oscilloscopes do not require a trigger, they can show when a desired signal is not present. Digitizers require a trigger, so it can be a nuisance to figure out the correct settings unless one can first see that the desired waveform is actually making it to the instrument.

6. Don't program carelessly with timing delays. The "delay" statement has to be the most misused in all of test programming. A slow test program is bound to have many seconds of delays. It's natural for test programmers to insert delays when debugging test routines, but many are guilty of not removing them, or at least of not optimizing them once the real cause of the measurement problem is found. There are many ways to solve this problem. The best is to use the sophisticated tools that the system already has available. Most instruments have triggering systems, or at least can programmatically announce when they are done. Instead of randomly tacking delays into a test routine, the measurement should be analyzed for correctness and the triggering system used to take a reading as soon as the instruments are ready.

7. In a production environment, move repeatedly failing tests to the front of a test program. If there are consistent problems with a DUT, why wait until the rest of the DUT has been tested before finding them? The system should be allowed to find a DUT that is destined to fail as soon as possible. Ideally, of course, such problems should be fed back into R&D or production engineering so that they can be resolved permanently rather than letting Manufacturing Test become a sorting process for design or parts problems.

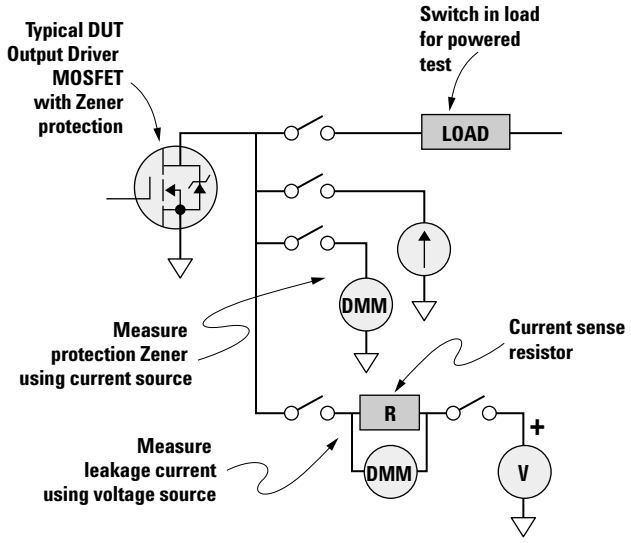


Figure 7: *Switched loads allows parametric measurements.*

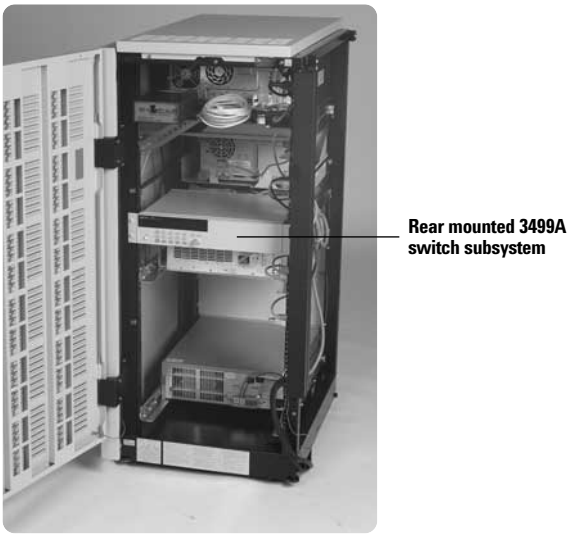


Figure 8: *Rear mounting the switching subsystem reduces rack space and minimizes cable length.*

Switching

Though a test spec may not include a list of system switches, it should. Switches are an integral part of any test system and must be chosen carefully. Reed relays and FETs are the best choice for high speed, and of the two, reeds have higher voltage and current ratings. A matrix arrangement of reed relays provides an excellent way to allow any instrument to be connected to any pin on the DUT, and it allows for easy expansion as new instruments are added to the system or more pins appear on the DUT. Armature relays are best used for lower thermal noise and higher current.

It is often necessary to provide banks of general purpose relays of varying current capability. Such relays can be used to connect DUT inputs to ground or to a supply, sometimes through resistors to simulate dirty switches. They can also be used to provide ways of disconnecting output loads in order to allow parametric tests on output transistors as shown in **Figure 7**.

While relay cards can be placed in a cardcage that is intended for high performance instruments, it is an enormous 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 relays are placed in a separate box that's 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 or newer switchbox. It also makes a clearer delineation between the instrumentation and the switching subsystems.

Placing the DUT interface panel (mass interconnect or feed-through 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 chosen box has cards in the rear, reverse-mount the switchbox using the rails on the rear of the rack as shown in **Figure 8**. 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.

DUT power

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 the application, DUTs can require anything from a few milliwatts to many kilowatts. There are many power supplies available for providing power to a DUT. The task is more complicated than simply picking the right voltage and current level. A reliable system power source can make testing the DUT a lot less frustrating by providing a stable voltage source to power the DUT and built-in measurement capability to verify DUT performance under various operating conditions. Things to consider when selecting a DUT power source are:

- Settling Time
- Output Noise
- Fast Transient Response
- Fast programming, especially down-programming response

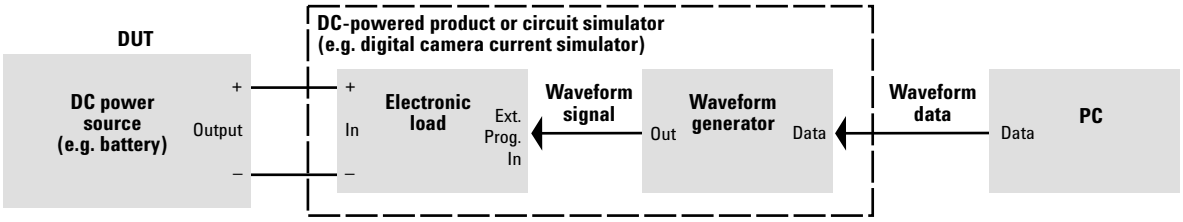


Figure 9: *Method of simulation of battery current using programmable load.*

- 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 switcher these days, freeing up much rack space
- Triggering options
- Programmable Output Impedance
- Multiple Outputs and Sequencing of Outputs

Loads

Many DUTs require components to be connected to their outputs in order to adequately stress the unit. 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 rack-mountable DC programmable load (the size and shape of a power supply), or perhaps a slide-out tray on which small, discrete loads can be mounted. Such loads are often connected to the DUT through relays to allow the DUT to be completely disconnected from all test system resources. **Figure 9** shows how a DC programmable load can be connected to an arbitrary waveform generator to re-create the current waveform previously captured from a battery.

Mass interconnect

It is tempting to leave off a mass interconnect when using a functional test system in a design lab, since the product design changes so much and the extra time to rewire a fixture is not productive. It's also not as likely that identical measurements on large numbers of devices will need to be made. Simple clip leads may suffice, especially for small DUTs. However, in general there are several good reasons for adding a relatively expensive interface panel to a system. Here are a few reasons, as stated by an automotive DV customer:

- "It provides a physical location for us to mount interface components such as terminal blocks, fuses, custom electronics/interfaces/conditioning, etc. between the system and the DUT. We can either mount these components to the interface frame or to a shelf attached to the frame. Without the mass interconnect, we would have to find somewhere else to place these components."
- "With the use of terminal blocks on the [interface] it allows us to very easily make wiring changes as the DUT changes, allows easy connection of multiple resources to common points, and provides easy test connections when debugging...the system."
- "[It] provides a fast and robust means of changing connections to different DUTs using the same system."

TEST SYSTEM EXAMPLE

Architectural choices

To illustrate the concepts and issues discussed in this application note, a test system will now be designed from scratch that can be used to test low frequency, low-medium pin count, low-medium power electronic modules. These are typical of the automotive and aerospace/defense industries.

Here are the architectural choices that will be made for this test system:

- Use an external PC, not an embedded PC.
- Allow extra rack space. Allow about 20-30% extra space to allow for future expansion.
- Mix modular and rack&stack (R&S) instrumentation. If there are any modular instruments, leave either 20% expansion room in the cage, or room in the rack for a bigger cage.
- Choose only industry standard interfaces.
- Place switching into a separate subsystem.
- Place the DUT interface panel (mass interconnect or feedthrough panels) in front of the switching subsystem.
- Use a matrix switching architecture for measurement instruments and low current stimulus.
- Connect high current DUT pins to GP relays that can be wired to power supplies and loads.
- Use Microsoft's Visual Studio.NET software.
- Use a rack with a top-exhaust cooling fan.

System design

The architecture will now be applied to a real world problem. The DUT is an Electronic Throttle Module for an automotive throttle body. According to the test specification, the following equipment will be required to run the tests:

- Programmable Volt/Ohm/Ammeter
- Programmable Power Supply – 0-13.5V/0-10A
- Waveform Generator capable of Pulse Width Modulation, 0-10VDC, 0-3 KHz
- Low current DC voltage source – 0-5VDC
- Waveform Analyzer
- CAN Interface
- Simulated or actual stepper motor load

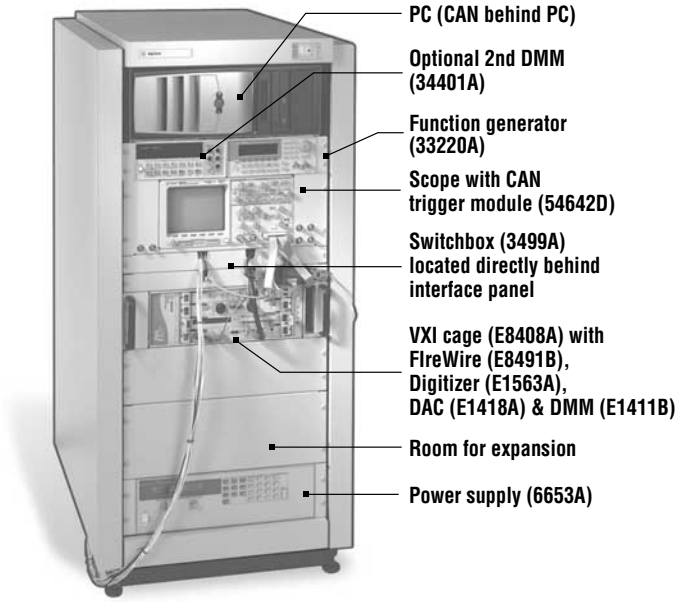


Figure 10: *Functional test system*

The DUT has 14 pins total on 3 connectors.

Looking at various catalogs, and adopting the architecture specified earlier, the instruments shown in **Figure 10** are chosen: a rack-mountable arbitrary waveform/function generator, heavy duty power supply, optional DMM, oscilloscope with CAN trigger module, dedicated switching cardcage ("switchbox") and 4-slot VXI cage. The VXI cage contains a digitizer, 16-channel DAC and high speed DMM. An RS-232C-based CAN interface is located on a shelf behind the PC.

There are four GPIB instruments – the power supply, switchbox, oscilloscope, and optional second DMM (useful for debugging since it does not require use of the PC). A USB/GPIB converter will be used for these instruments. This means a slot in the PC for a GPIB card will not be needed. It also provides access to the USB in the event the GPIB cables and instruments are eventually replaced with USB versions, thus "future-proofing" the system.

Since there are some VXI instruments, a FireWire interface will be used to control them. It's a high-speed industry standard interface.

The Arbitrary Waveform/Function Generator will be connected to the PC's LAN port using a "Crossover" cable. Should more LAN-based instruments be added in the future, a LAN hub or router can be added. The use of LAN provides an opportunity to use the instrument's built-in web server to see and edit setup information from a web browser.

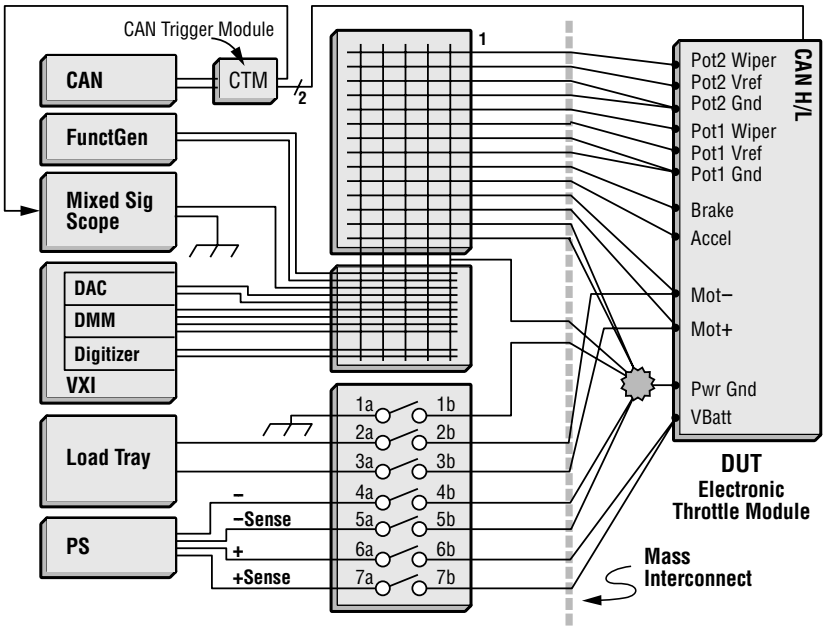


Figure 11: *Block diagram of system*

12 Many I/O interfaces will be in use – RS-232C, FireWire, USB, GPIB, LAN. By 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 USB 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 11 shows how the instruments will be connected to the switching subsystem. By using a matrix, any instrument can be connected to any DUT pin, and new instruments can be added 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. A star ground is used to avoid ground loops.

A mass interconnect is an option for this system. This particular DUT only has 14 pins, so an R&D or DV environment 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 is all that may be needed. In the future, if the modules to be tested have more and more pins, or if a place is needed to put other things between the system and the DUT, a commercial mass interconnect solution may be needed. A place directly in front of the switchbox will therefore be provided for such an interface.

A 5-wire measurement bus is chosen because it allows all four leads of the DMM to be connected to different pins on the DUT, making a 4-wire Ohms measurement possible. By routing two matrix points to the same pin on the DUT (as shown in **Figure 11** on the Pot1 and Pot2 Gnd pins), the resistance measurement will be very accurate, since the remote sense location is made right at the DUT. If two wires are not used, a 4-wire Ohms measurement can still be made 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. This 5-wire bus works in part because it is rarely necessary to connect all sources and measurement devices to the DUT at once. As is true of many DUTs today, the DUT has built-in selftest capabilities that are activated via the serial interface, in this case a CAN interface. Thus, a command can be issued to, say, internally measure the voltage being fed to the Pot1 Wiper input. The DAC can generate a voltage and feed it to that pin, and the resulting internal measurement can be read back over the CAN bus. This tests not only the Pot1 Wiper pin but the ability of the internal processor to correctly measure pin voltages.

Multiple signal sources can be connected to the same pin when using a matrix arrangement. It is important not to accidentally short such sources together. Switching routines should be carefully written to either eliminate this possibility or offer warnings when such conditions occur.

		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

Figure 12:
DUT wiring spreadsheet. "Star Ground" appears in a row and a column because it sits between the DUT and the System and is therefore external to both. Connections are made from DUT pins to each of the selected System Resources using separtate wires. Other connections are made from the named resource (e.g.; "PS+Sense") to the selected System Resources.

If the DUT must be powered up and run in full functional mode, it may be necessary to modify the test system with either more instrument busses or with more devices connected directly to the DUT. The test engineer must carefully analyze the type of testing that is required and plan accordingly.

Care must be observed when using the oscilloscope. As an earth-referenced device, it requires the Star Ground to be connected to Earth Ground. A GP relay (1a/1b) was allocated to allow for this. In addition, the oscilloscope can not be used to make measurements of the stepper motor drive, which is a floating H-bridge circuit. The oscilloscope can be used to measure Mot+ or Mot– with respect to ground, but can not be used to measure Mot+ to Mot–. This is why an isolated digitizer is also required. In fact, the oscilloscope is not really required for any measurements. The benefit of including one, if cost is not critical, is in fast manual debugging.

Not shown in the drawing is the optional DMM. That DMM is used only for manual debugging. It could be added to the switch matrix though, simply by adding it to another set of 4 rows on the instrumentation matrix. Other instruments can similarly be added to the system easily.

Only one channel of DAC (High and Low leads) is shown, although sixteen are actually available. It is more typical to run all DAC lines to the mass interconnect so that individual channels can be connected as needed to various DUT pins. However, in doing so one loses the flexibility of routing a DAC to any pin.

It is helpful to make a wiring map that shows how the DUT will connect to the system. Fig. 12 shows how to make one using a spreadsheet. In the future, when it becomes necessary to test a different DUT, all that will have to be done 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 GP relay 7b and one to GP 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.

```
Public Class Form1
    Inherits System.Windows.Forms.Form

    ' This code creates and initializes an instance of the VXIplug&play wrapper class
    ' and uses hard-coded values for driver settings.
    Dim myHp34401 As _
        New Agilent.TMFramework.InstrumentDriverInterop.VxipnpWrappers.Hp34401("GPIB0::22::INSTR", True, True|
                    Namespace TMFramework
    Dim myHpe1411 As _
        New Agilent.TMFramework.InstrumentDriverInterop.VxipnpWrappers.Hpe1411("VXIO::24::INSTR", True, True|

    Dim myHpe1563 As _
        New Agilent.TMFramework.InstrumentDriverInterop.VxipnpWrappers.Hpe1563("VXIO::40::INSTR", True, True|

    Dim myHpe1418 As _
        New Agilent.TMFramework.InstrumentDriverInterop.VxipnpWrappers.Hpe1418("VXIO::9::INSTR", True, True|

    ' This code creates and initializes an instance of the IVI-COM wrapper class
    '
    Dim myAgilent66xxClass As Agilent66xxLib.IAgilent66xx
    Dim myAgilent3499Class As Agilent3499Lib.Agilent3499
    Dim myAgilent33220Class As Agilent33220Lib.IAgilent33220

    Public Sub Init_insts()

        myAgilent66xxClass = New Agilent66xxLib.Agilent66xxClass()
        myAgilent3499Class = New Agilent3499Lib.Agilent3499Class()
        myAgilent33220Class = New Agilent33220Lib.Agilent33220Class()
        myAgilent66xxClass.Initialize("GPIB0::5::INSTR", True, True, String.Empty)
        myAgilent3499Class.Initialize("GPIB0::9::INSTR", True, True, String.Empty)
        myAgilent33220Class.Initialize("TCPIP0::169.254.2.20::INSTR", True, True, String.Empty)

    End Sub
```

Figure 13: *Code to initialize instruments in Visual Basic.NET using VXIPlug&Play and IVI-COM drivers. This code is automatically generated by "toolkit" software.*

Write the software

In design and DV environments, engineers use both graphical and text-based languages to develop tests. In manufacturing, test executives are the norm, in which test engineers often create sequences of tests that have been developed either by themselves or by others, in languages that can again be a mixture of graphical and textual. In this example, we will use Visual Basic and Visual C++ in Microsoft's new Visual Studio.NET development environment. This lets us tap into the solutions provided by thousands of engineers around the world who are creating not only .NET itself, but also all the many tools that work with .NET to make the test engineer's job easier.

This test system and application software to test the throttle module was developed in about one week without ever consulting a manual (except for the CAN box that did not have VISA/VXIPNP/IVI-COM drivers), and all instruments were actually communicating in the VB program within a few minutes of loading the software. The RS-232C-based CAN box had to be manually programmed by writing a C++ program that made the various function calls to the box and then declaring those routines in VB so that they could be called from within VB. However, this was not as painful as it was in the past because the .NET environment lets the programmer mix C++, C# and VB programs all in the same development environment. That single improvement saved many hours of time. **Figure 13** shows an example of the code required to initialize all the instruments and perform resets on them. This code was generated automatically by Agilent W1140A-TK1 T&M Toolkit software.

It is usually necessary to create a Graphical User Interface (GUI) to make the task of executing a test sequence easier. This is easily accomplished in Visual Studio. **Figure 14,** (pg. 15), shows the GUI that was created in Visual Basic.NET. Although there are some changes in VB from Visual Studio 6.0, they did not interfere with the quick creation of this code. Programming in VB is straightforward. One draws the text boxes and other objects on the screen by grabbing them from a list and dropping them on the form, then sizing them and editing their properties in the Properties window. Double clicking on the object creates a stubbed subroutine that will be executed when the user clicks or selects that object.

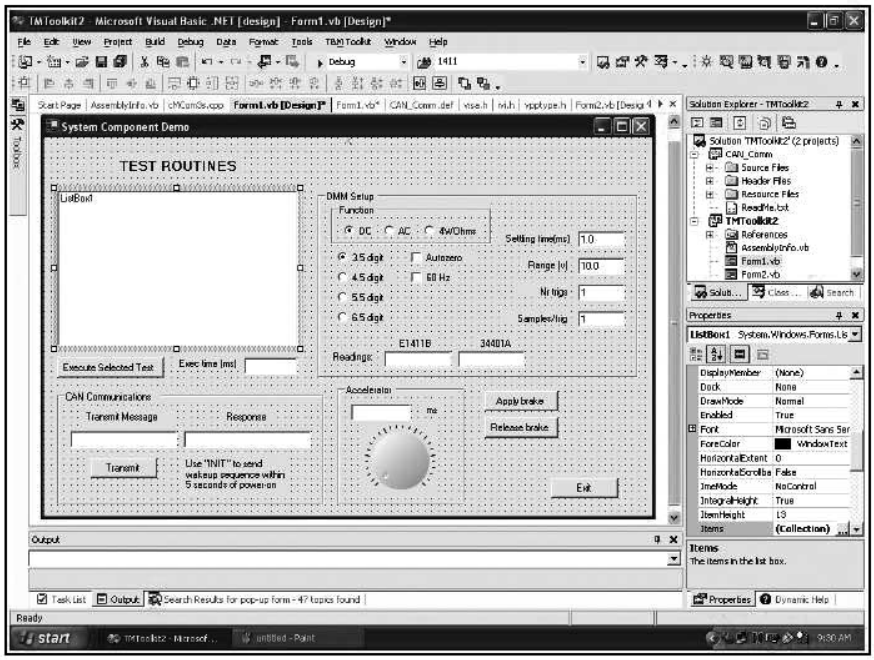


Figure 14: *Graphical User Interface developed in Visual Basic.NET*

CONCLUSION

Designing a functional test system takes up-front planning to minimize development time, maximize throughput, and plan for future expansion. Minimizing development time means picking software that has good support and good Test & Measurement tools and that is fast and easy to use. Maximizing throughput is more complicated than simply choosing fast instruments; the entire measurement cycle must be considered. Designing a system that can accommodate more instruments, more switches and bigger, more power-hungry DUTs in the future without a complete re-design maximizes the re-usability of a functional test system.

REFERENCES

[1] *"Interview with Alan Cooper: The Future of Software Development and the .NET Platform",*

Fawcette Technical Publications,
www.ftpconferences.com/cooperu/future_interview.asp

[2] *"Instrumentation I/O Enters a New Age: An Introduction to GPIB, USB and LAN Interfaces",*

B. Kolts,
Agilent Technologies, 2003

DATA SHEETS

3499A/B/C Switch Family
http://www.agilent.com/find/3499

34401A Digital Multimeter
http://www.agilent.com/find/34401a

33220A Function /Arbitrary Waveform Generator, 20 MHz
http://www.agilent.com/find/33220a

6653A 500W System Power Supply, 35V, 15A
http://www.agilent.com/find/6653a

54642D 2+16 Channel, 500 MHz Mixed-Signal Oscilloscope
http://www.agilent.com/find/54642D

VXIbus products
http://www.agilent.com/find/vxi

www.agilent.com



www.agilent.com/find/emailupdates
Get the latest information on the products and applications you select.

Pathways to Exceptional Test
Let Agilent help you streamline system development and lower the true cost of test. Our world-class measurement science and support can get you accurate results in a hurry. Agilent's instruments are optimized for use in systems, and our open industry software and I/O standards take the hassle out of creating test code. To see how you can get maximum leverage from Agilent's resources, go to www.agilent.com/find/buildyourown

Was This Useful?
If you found the information in this application note useful, feel free to visit these other Agilent websites dedicated to helping the engineer who solves difficult test problems: Visit
www.Agilent.com/find/appcentral
www.Agilent.com/find/connectivity
www.Agilent.com/find/adn

By internet, phone, or fax, get assistance with all your test & measurement needs
Online assistance:
www.agilent.com/find/assist

Phone or Fax

United States:
(tel) 1 800 452 4844

Canada:
(tel) 1 877 894 4414
(fax) (905) 282 6495

China:
(tel) 800 810 0189
(fax) 800 820 2816

Europe:
(tel) (31 20) 547 2323
(fax) (31 20) 547 2390

Japan:
(tel) (81) 426 56 7832
(fax) (81) 426 56 7840

Korea:
(tel) (82 2) 2004 5004
(fax) (82 2) 2004 5115

Latin America:
(tel) (305) 269 7500
(fax) (305) 269 7599

Taiwan:
(tel) 0800 047 866
(fax) 0800 286 331

Other Asia Pacific Countries:
(tel) (65) 6375 8100
(fax) (65) 6836 0252
(e-mail) tm_asia@agilent.com

Product specifications and descriptions in this document subject to change without notice.

© Agilent Technologies, Inc. 2004
Printed in the USA, January 27, 2004
5989-0154EN



Agilent Technologies

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP