Agilent Embedded Automation with the E6601A Wireless Communications Test Set

Application Note

Agilent Technologies

Table of Contents

Introduction

Recent innovations in test equipment design have created a new class of "smart instruments" that give mobile phone manufacturers the ability to streamline their production lines. The Agilent E6601A Wireless Communications Test Set, which features an integrated PC with Windows® XP operating system, fits this category. The E6601A test set has the control and interfacing capability needed to become the centerpiece of a streamlined and automated test system. With an embedded automated test program, the E6601A can significantly improve the test speed, simplify the test system, and help reduce the cost of test.

For test engineers and programmers, this application note demonstrates how to use the built-in capability of the E6601A test set to develop a simple program for automated testing. The programming example includes setting up the programming environment and the test set, verifying connectivity, creating a programming flowchart, and writing the test program. Although several programming environments can be used in this environment, the examples provided in this application note are created in Visual Studio .NET®.

Overview of the E6601A Test Set

The Agilent E6601A is a one-box test set for mobile phone manufacturing based on a new platform. An important feature of this new platform is the embedded PC, which runs the Windows XP Professional® operating system. Test engineers and programmers now can work in a familiar computing environment using a keyboard, mouse, and monitor connected to the test set. They can run any current Windows-compatible program on the test set without having to convert the code. Furthermore, they can make use of all Windows platform features such as remote access.

A test set with a built-in PC has other advantages.

- **Saves space** – Placing a PC and monitor is always challenging on a production line. Rarely does the rack have adequate space. By combining the PC and monitor into the test set, the E6601 eliminates this headache. The front panel can be controlled through automation, with a mouse, or by standard front-panel buttons.

- **Easy to deploy and maintain** – An all-in-one solution makes moving and deployment easy. As no external PC and accessories are needed, production managers can rearrange or build new production lines more quickly. The test set is easier to maintain, and there are fewer parts and connections to worry about. Most regular maintenance, such as software upgrades and daily or monthly alignment, can even be performed remotely by LAN.

- **Helps reduce the expenses associated with buying and operating equipment** – The E6601A saves money by reducing the need for PCs, GPIB devices and accessories, racks and floor space used for computers, and computer maintenance costs. Test engineers and programmers also do not have to use proprietary test development tools and can get started testing more quickly.

Features That Simplify Automation

The E6601A has important features that simplify operating the test set and automating the test environment. Foremost is the embedded Windows XP computer, which shortens the user's learning curve; allows remote control of the test set and sharing of data between test sets; and allows online software updating, troubleshooting, and support.

Connectivity features also help make the E6601A suitable for system integration and remote control. Users can control and supervise a test procedure, communicating by means of a LAN, GPIB, USB, or virtual LAN.

- **LAN** – LAN capability provides instrument connectivity over distances and allows sequential sharing of instruments among multiple PCs. A LAN is the recommended method of connecting instruments in new test systems. For the E6601A, remote control over a LAN is an efficient way to implement remote troubleshooting and monitoring.

- **USB** – Universal Serial Bus (USB) is a quick and easy way to connect instruments to PCs on a bench top. Typically it is used to connect a single instrument to a PC.

- **GPIB** – The General Purpose Interface Bus (GPIB) is a primary instrument IO device with proven reliability. It is the most common interface for measurement instruments and can be used in the integration of test systems.

- **Virtual LAN** – In embedded programs, the E6601A uses an internal LAN interface to control the test set. In this scenario, the virtual LAN connects the embedded PC and a test component internally.

Developing a Test Program

Test programs can be developed using either the embedded PC or an external PC. For best results, Agilent recommends developing and debugging test programs on an external PC, then uploading the program into the E6601A's embedded PC for use in testing. The following instructions apply to test program development on an external PC. Loading the program into the test set is explained in a later section.

Set up the programming environment – Many programming environments can be used with the E6601A, including C++, Visual Basic®, Agilent VEE Pro and more. Today Visual Studio® is a popular tool for program development, so the examples given in this application note use the Visual Studio .NET environment and the Visual Basic programming language. You can learn more about these programming tools at the Microsoft Web site **http://msdn.microsoft.com/vstudio/**.

Install IO Libraries Suite – The IO Libraries Suite is an Agilent application for connectivity. It needs to be installed when you plan to connect instruments to a PC. The program helps you configure your system to control instruments from the PC and to transfer data. You can download the suite software from the IO Libraries Suite Web site: **www.agilent.com/find/iolib**. Under the **Software** menu, select **IO Libraries Suite** and then **IO Libraries Suite 14.1 (or above) Product Download**. After downloading, follow the installation instructions.

Connect and set up the test set – Different options are available for connecting the external PC used for programming to the E6601A test set.

- **GPIB connection**
  - Connect a GPIB cable from your computer's GPIB interface to the GPIB connector on the test set's rear panel.
  - On the test set's front panel, under **Screens**, press the **Config** key.
  - In the **Configuration Main** window, under **Instrument Connections**, type the GPIB address you want.

- **LAN connection**
  There are two ways to connect the PC and the E6601A over a LAN. One method is to connect the PC to the E6601A directly using a cross-over cable—in effect, setting up a small, PC-to-test-set-only LAN. The other method is to connect the PC to the E6601A over an office or other existing LAN or hub. In this case a regular LAN cable is used. Whichever approach you choose, the following steps apply.

  - **Get the IP address of the E6601A**
    On the E6601A desktop, click **Start**, then select **Run**. A Run window pops up. Input **CMD** in the command bar, then click **OK**. A black command window is displayed. Type **ipconfig** after the flashing cursor and press the **Enter** key. The Windows IP configuration is displayed. The IP address, which is located under **Ethernet Adapter Local Area Connection**, is the address for connection.

  - On your computer, click on the **Agilent IO Control** icon in the taskbar and select **Agilent Connection Expert** from the pop-up menu. When the Agilent Connection Expert is launched, select **E6601A** in the LAN folder. Click the **Change Properties** button, which is on the right side of the window. A **Change Configurable Properties of This LAN Device** window is displayed. Type in the IP address of the E6601A, which you located in the previous step, and then click **OK**.

Verify the test connection – On your computer, click on the **Agilent IO Control** icon in the taskbar and select **Agilent Connection Expert** from the pop-up menu.

When the Agilent Connection Expert is launched, instruments connected to the external PC's GPIB interfaces should appear automatically. Click on **Refresh All** to update the list of instruments on these interfaces.

Now you can verify the test connection by sending some commands to the test set. See Figure 1.

- Right-click on the test set you want to connect. In this example a LAN is used to connect to an E6601A. In the pop-up menu, click **Send Commands To This Instrument**. The Agilent Interactive IO program opens.
- In **Agilent Interactive IO** window, type a command—for example, "*IDN?"—in the Command box. Then, click **Send Command** and the command will be sent to the E6601A. Click **Read Response** and the IDN of the E6601A is displayed in the **Instrument Session History** window.
- To close Agilent Interactive IO, click the **Connect** menu, and then click **Exit**.

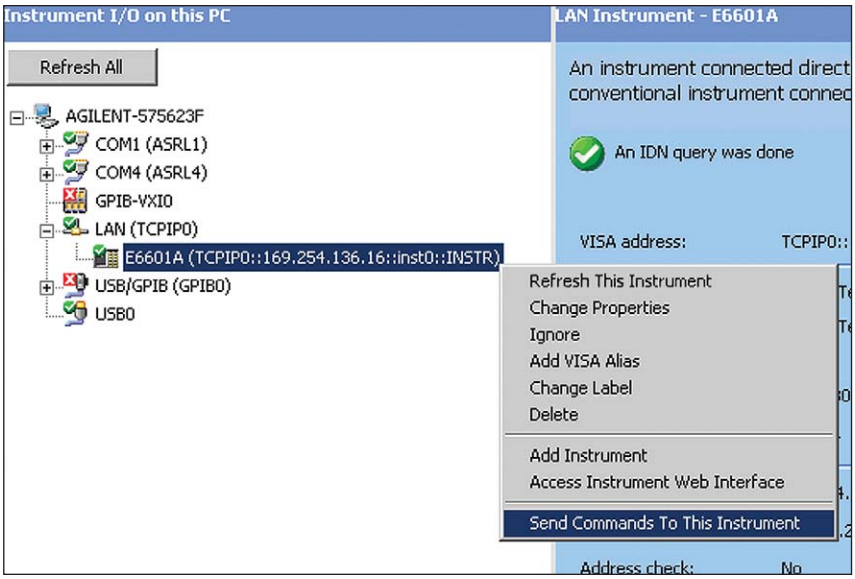

Figure 1. Agilent Connection Expert helps you set up and manage instrument connections.

Access the SCPI command documentation – Unique calibration applications provide the E6601A test set with technology-specific transmitter and receiver measurements for calibrating mobile phones. The programming examples in this application note are based on a configuration for GSM/GPRS mobile phone testing.

The SCPI command documentation for the E6831A GSM/GPRS Calibration Application is contained in the *GPIB Commands (GSM/GPRS CA)* topic which is part of the E6601A's online help. This book contains syntax, descriptions, ranges, *RST settings, test set requirements, and single-line examples of each command provided in Visual Basic .NET using the VISA COM library. Another tool for programming is the Agilent E6601A SCPI Discovery tool, which is used in programming environments such as Visual Studio .NET. This tool makes programming easy by providing fast access to the entire command structure and by allowing you to drag the commands you need to the code. See Figure 2. For more information on the SCPI Discovery tool, please refer to the E6601A Web page at **www.agilent.com/find/e6601a**. More detailed use is available in the built-in help system supplied with the tool.

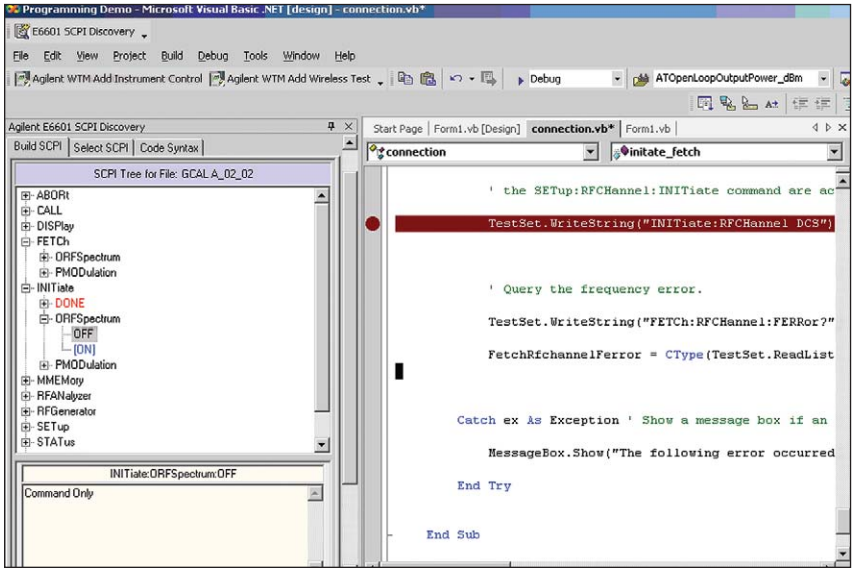

Figure 2. Agilent E6601A SCPI Discovery is a useful programming tool used in popular environments such as Visual Studio .NET.

Programming Details

A simple GSM test program is developed according to the programming flowchart shown below in Figure 3.

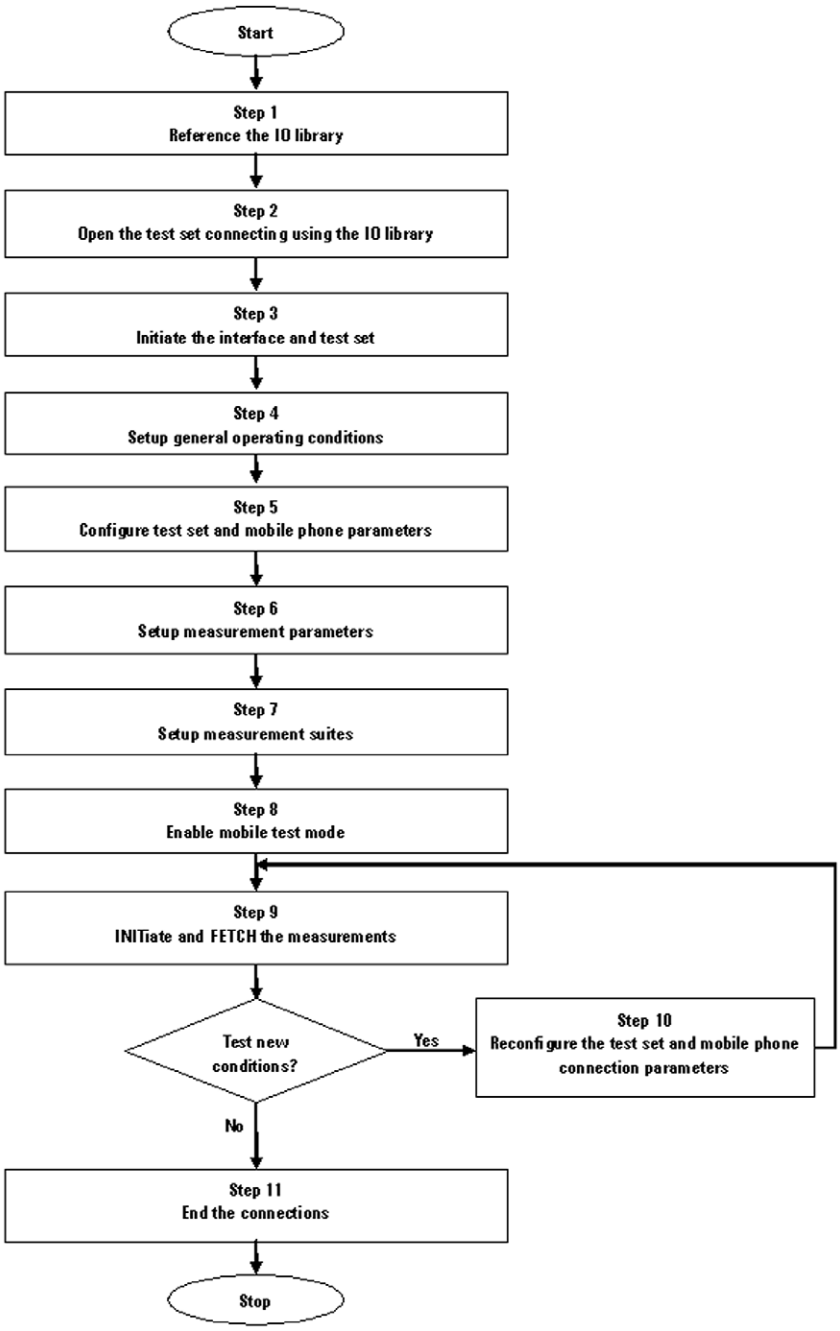

Figure 3. Flowchart forE6601A test program development.

Step 1. Reference the IO library – Start Visual Studio .NET and create a new Visual Basic .NET application. In the **Solution Explorer** window, right-click **References** and click **Add Reference**. See Figure 4.

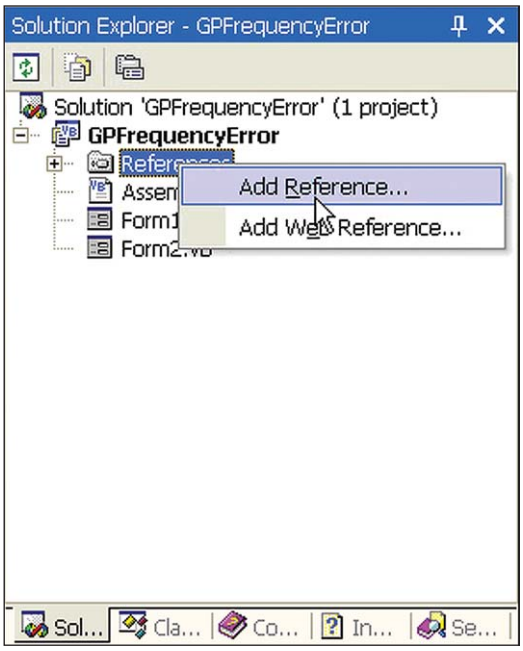

Figure 4. Solution Explorer window in Visual Studio .NET.

In the **Add Reference** dialog box, click the **COM** tab and scroll down until you see the **VISA COM 3.0 Type Library** component. See Figure 5.

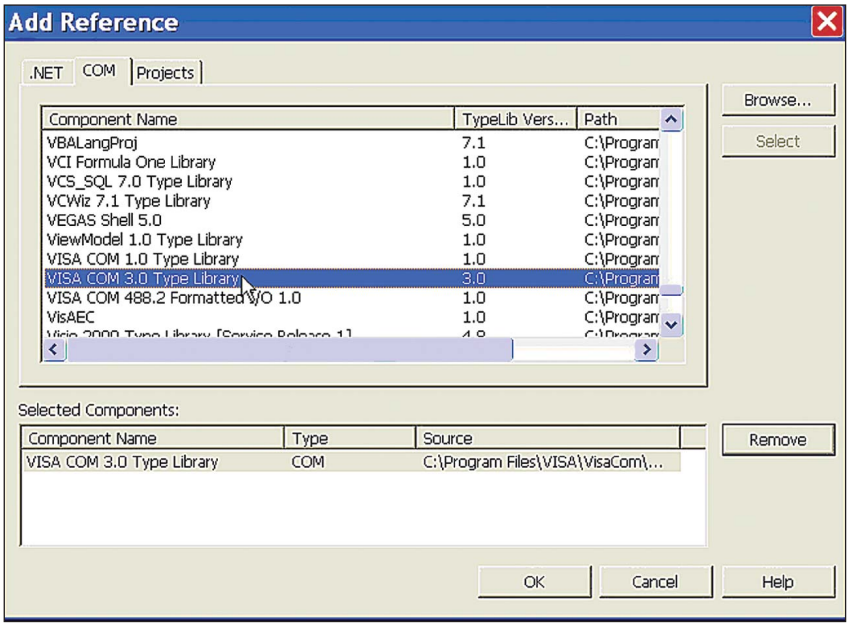

Figure 5. Add Reference window in Visual Studio .NET.

Double-click on the **VISA COM 3.0 Type Library** entry to add the component, and then click **OK** to close the Add Reference dialog box.

Step 2. Open the test set connection – Computers communicate with the test set by sending and receiving messages over an interface. When you use the Agilent VISA COM library in Visual Basic .NET, the FormattedIO488 object's WriteString, WriteNumber, WriteList, or WriteIEEEBlock methods are used for sending commands and queries. To use the VISA COM library, you must first open a connection to the test set using the ResourceManager object's Open method. After a query is sent, the response is read using the ReadString, ReadNumber, ReadList, or ReadIEEEBlock methods.

The following Visual Basic .NET statements open the connection to the test set and send a command to change the brightness of the test set's display.

```
Option Explicit On
Option Strict On
Imports Ivi.Visa.Interop

Dim ResManager As New ResourceManager
Dim TestSet As New FormattedIO488

' Open the connection to the test set. Get the "VISA  Address" from the
' Agilent Connection Expert (installed with Agilent IO Libraries Suite).
' In this sample, the "VISA Address" used for GPIB interface is
' "GPIB0::14::INSTR"
TestSet.IO = CType(ResManager.Open("GPIB0::14::INSTR"), IMessage)
'If it is a LAN interface, the “VISA Address” is
TestSet.IO = CType(ResManager.Open("TCPIP0::localhost::inst0::INSTR"),
IMessage)

 ' Send a command.
TestSet.WriteString("DISPlay:BRIGhtness MEDium")
```

Step 3. Initiate the interface and the test set – To ensure consistent, repeatable performance, you need to start the program, your computer, and the test set in a known state. Without correct initialization, your program may run correctly in one instance but not in another. This variability could be due to changes in the configuration of the test set made during previous program runs or from the front panel.

If you are using the Agilent VISA COM library, you can use the resource session object's Clear method to clear the interface buffer. If you are using GPIB, the resource session object's Clear method also resets the test set's parser. The parser is the program that reads the instructions sent to the test set.

- **Fully presetting the test set** – It is important to get the test set to a known state before each production session. Sending the *RST command fully presets the test set, which ends all current measurement processes and restores all values to defaults.

- **Clearing the test set's error queue** – Before each production session, it is useful to clear the error queue of any old messages. Clearing the error queue assures that any messages logged are relevant to the current production session.

The code in the example below fully presets and clears the error queue.

```
' Clear the interface buffer and reset the test set's parser.
TestSet.IO.Clear()

' Fully preset the test set.
TestSet.WriteString("*RST")

' Clear the test set's error queue.
TestSet.WriteString("*CLS")
```

Step 4. Set up general operating conditions – General operating conditions include amplitude offsets and the display mode.

- **Set the amplitude offsets** – Amplitude offsets compensate for losses or gains in the cables and fixtures between the test set and the mobile phone being tested. Up to 100 frequencies can be assigned by using the command SYSTem:CORRection.

- **Set the display mode** – For faster test execution, you can disable the front panel display on the test set using the command DISPlay:MODE:TRACKing.

The following example shows how to set the amplitude offsets and display mode.

```
' Set offset frequencies and set the state to On for the frequencies,
' offsets and RF IN/OUT Amplitude Offset State.
TestSet.WriteString("SYSTem:CORRection:SFRequency 1710.2
MHZ,1805.2 MHZ,1784.8 MHZ,1879.8 MHZ")

' Set offset values correcting for losses in dB.
TestSet.WriteString("SYSTem:CORRection -2.55,-3.12,-3.68,-4.23")
```

The following example shows how to set the test set's display mode.

```
' Turn the tracking mode to Off.
TestSet.WriteString("DISPlay:MODE:TRACking OFF")
```

Step 5. Configure the test set and wireless device parameters – Different parameters are set for different measurements. However, most measurements require settings for channel, power, and band. You will need to do the following to configure the test set and mobile phone test parameters.

- **Set up channels** – Specify the channel associated with every band and then set the active band.
- **Set RF generator power** – Set the output power of the test set.
- **Set RF analyzer level** – Set the expected input power level to the RF analyzer and the uplink burst for single-slot measurements on dual-uplink PDTCHs.
- **Set expected burst type** – This parameter is used for measurement synchronization.

The following example executes these actions:

- Setting the ARFCN for a number of different bands
- Setting the active band
- Setting the RF generator power
- Setting the RF analyzer level

```
***Configure Band and Channel parameters***
'

' Set the RF analyzer frequency control to auto.
TestSet.WriteString("RFANalyzer:CONTrol:FREQuency:AUTO:
GCALibration ON")

' Set ARFCN to 512 for the DCS band.
TestSet.WriteString("TRANsceiver:CHANnel:DCS 512")

' Set ARFCN to 5 for the EGSM band.
TestSet.WriteString("TRANsceiver:CHANnel:EGSM 5")

' Set ARFCN to 259 for the GSM450 band.
TestSet.WriteString("TRANsceiver:CHANnel:GSM450 259")

' Set active GSM/GPRS band.
TestSet.WriteString("TRANsceiver:BAND:GCALibration DCS")

' Set the RF Generator power state to on and the level to -70 dBm.
TestSet.WriteString("RFGenerator:POWer:GCALibration -70")

' Set the RF analyzer expected input power control mode to manual.
TestSet.WriteString("RFANalyzer:CONTrol:POWer:AUTO:
GCALibration ON")

' Set uplink burst 2 to be measured for single slot measurements.
TestSet.WriteString("RFANalyzer:BURSt 2")

' Set the Mobile Station TX Level for burst 1 in the DCS band to 10.
TestSet.WriteString("RFANalyzer:MS:TXLevel:DCS:BURSt1 10")

' Set the Mobile Station TX Level for burst 2 in the DCS band to 12.
TestSet.WriteString("RFANalyzer:MS:TXLevel:DCS:BURSt2 12")

' Set the test set to expect Training Sequence Code (TSC) 2 in the
' midamble burst.
TestSet.WriteString("RFANalyzer:BURSt:TYPE TSC2")
```

Step 6. Set up the measurement parameters – Two types of measurement parameters must be set:

- **Generic measurement parameters** – Count, timeout, trigger
- **Measurement-specific parameters** – The setup configurations for a specific measurement

The example below illustrates how to configure measurement or measurement suite parameters.

```
' Configure Power and Modulation Quality Measurement Suite parameters.
' Set Power and Modulation Quality suite trigger arm to single.
TestSet.WriteString("SETup:PMODulation:CONTinuous OFF")

' Set Power and Modulation Quality suite multi-measurement count to 100.
TestSet.WriteString("SETup:PMODulation:COUNt 100")

' Set VISA timeout to 10 seconds. This timeout should be longer than the
' following measurement timeout value.
TestSet.IO.Timeout = 10000

' Set Power and Modulation Quality suite timeout to 5 seconds.
TestSet.WriteString("SETup:PMODulation:TIMeout 5S")

' Set the time offsets for the Power versus Time power measurement.
TestSet.WriteString("SETup:PMODulation:PVTime:TIME -28uS,-
10uS,321.2uS,552.8uS,570.8uS")

' Set the burst synchronization mode to Midamble.
TestSet.WriteString("SETup:PMODulation:SYNC MIDamble")

' Configure RF Channel Measurement Suite parameters.

' Set RF Channel suite trigger arm to single.
TestSet.WriteString("SETup:RFCHannel:CONTinuous OFF")

' Set the state to on and the multi-measurement count value to 150.
TestSet.WriteString("SETup:RFCHannel:COUNt 150")

' Set RF Channel suite timeout to 5 seconds.
TestSet.WriteString("SETup:RFCHannel:TIMeout 5S")

' Set RF Channel suite filter to 1 kHz.
TestSet.WriteString("SETup:RFCHannel:FILTer BWKHZ1")

' Set trigger delay time to 1.5 ms for the 1 kHz filter.
TestSet.WriteString("SETup:RFCHannel:TRIGger:DELay:BWKHZ1 1.5mS")
```

Step 7. Set up the measurement suites – A measurement suite contains a collection of measurements that are all made using the same set of signal samples.

There are two ways to perform a measurement within a measurement suite:

```
SETup:<Measurement Suite>
INITiate:<Measurement Suite>, ...
FETCh results
```

and

```
INITiate:<Measurement Suite> <Measurement>, ...
FETCh results
```

Below are examples of how to configure measurement suites using these two methods.

SETup:<Measurement Suite> method example

```
' Setup Initiate
TestSet.WriteString(“SETup:PMODulation:INITiate PFERror”)

'Initiate a phase and frequency error measurement.
TestSet.WriteString("INITiate:PMODulation")

' Fetch phase and frequency error results.
TestSet.WriteString("FETCh:PMODulation:PFERror?")
FetchPmodulationPferror = CType(TestSet.ReadList(), Array)
```

INITiate:<Measurement Suite> <Measurement>, ... method example

```
' Initiate a phase and frequency error measurement.
TestSet.WriteString("INITiate:PMODulation PFERror")

' Fetch phase and frequency error results.
TestSet.WriteString("FETCh:PMODulation:PFERror?")
FetchPmodulationPferror = CType(TestSet.ReadList(), Array)
```

Step 8. Make a connection – This step requires making a connection between the test set and the mobile phone. Typically, you need to command the phone to synchronize to the test set's signal and begin transmitting back an appropriate signal. However, for the measurements in which signal decoding is not performed—for example, Tx or Rx calibration—this synchronization may not be necessary.

Step 9. Initiate and fetch measurement – This step is a continuation of the measurement procedures described in Step 7.

Step 10. Re-configure the test set and wireless device connection parameters – After performing a set of measurements on a mobile phone using the configuration established in Step 5, you may want to change this configuration and test the wireless device again. This step involves changing testing conditions such as channel, wireless device transmit power level, and cell power.

The following example changes the settings for a new band (using the ARFCN that was set previously in Step 5) and sets a new uplink power control level for the new band.

```
' Set active GSM/GPRS band to GSM450. This will use ARFCN 259 defined
' previously.
TestSet.WriteString("TRANsceiver:BAND:GCALibration GSM450")

' Set the traffic channel wireless device uplink power control
' level for the GSM450 band to 21.
TestSet.WriteString("RFANalyzer:MS:TXLevel:GSM450:BURSt1 21")
```

Step 11. End the connection – This step ends the transmission by the mobile device. It usually includes three procedures:

- **End the wireless device transmission** – You must send the necessary test mode commands to the mobile phone to end its transmission. These commands are specific to the phone.

- **Partially preset the test set** – It is good practice to partially preset the test set and stop all measurement processes without resetting all parameter values to their defaults.

```
' Send this preset command to partially preset the test set without
' resetting the parameter values to defaults.
TestSet.WriteString("SYSTem:PRESet3")
```

- **Close the connection to the test set** – When your program no longer needs a connection to the test set, it is important to close the connection using the FormattedIO488 object's Close method.

```
' Close IO interface to test set
TestSet.IO.Close()
```

Moving the Test Program into the E6601A's PC Controller

Once a test program has been developed and debugged on the external PC, it is loaded into the E6601A and the external PC can be disconnected. The transition is simple. In the programming shown so far, all that really needs to be changed is the resource descriptor for the test set.

In the example that uses the GPIB interface between the PC and test set, the code reads:

```
TestSet.IO = CType(ResManager.Open("GPIB0::14::INSTR"), IMessage)
```

To run the same program internally, the Virtual LAN (as described earlier) should be used. It can be done in two ways.

1. **Hard coded,** using the instrument name (for example, the instrument name is E6601A0429):

   ```
   TestSet.IO = CType(ResManager.Open("TCPIP0::E6601A0429::inst0::INSTR"),
   IMessage)
   ```

2. **Generic,** which allows easier transport to other test sets:

   ```
   TestSet.IO = CType(ResManager.Open("TCPIP0::localhost::inst0::INSTR"),
   IMessage)
   ```

After changing the resource descriptor in the program, rebuild the .EXE and move it to the E6601A (a USB flash drive is useful for this step) and run it as before. Communication now takes place using the virtual LAN instead of the GPIB connection used in the earlier examples.

Conclusion

As this example shows, writing a program to automate the E6601A test set is straightforward using Windows-based programming tools. Setting up the programming environment and the test set, verifying connectivity, creating a programming flowchart, and writing the actual test program can all be accomplished with relative ease. The program is then uploaded and run on the E6601A's internal PC. This powerful capability, when added to the E6601A's other time- and space-saving features, makes the test set a valuable addition to a fast-paced, high throughput production line.

For more information about the E6601A, visit our Web site: **www.agilent.com/find/E6601A**.

Appendix: Sample Test Program for GSM

The following programming example executes a GSM test. It includes a suite of power and modulation quality measurements, and a suite of RF channel measurements.

```
Option Explicit On
Option Strict On
Imports Ivi.Visa.Interop

Public class Demo
Private ResManager As New ResourceManager
Private TestSet As New FormattedIO488
Friend Sub Measurement()
Dim FetchPmodulationPferror as Array
' Open the connection to the test set. Get the "VISA Address" from the
' Agilent Connection Expert (installed with Agilent IO Libraries Suite).
' In this sample, the "VISA Address" used for GPIB interface is
' "GPIB0::14::INSTR"
TestSet.IO = CType(ResManager.Open("GPIB0::14::INSTR"), IMessage)
 ' Clear the interface buffer and reset the test set's parser.
TestSet.IO.Clear()

' Fully preset the test set.
TestSet.WriteString("*RST")

' Clear the test set's error queue.
TestSet.WriteString("*CLS")

' Set offset frequencies and set the state to On for the frequencies, offsets
' and RF IN/OUT Amplitude Offset State.
 TestSet.WriteString("SYSTem:CORRection:SFRequency 1710.2 MHZ,1805.2
MHZ,1784.8 MHZ,1879.8 MHZ")

' Set offset values correcting for losses in dB.
TestSet.WriteString("SYSTem:CORRection -2.55,-3.12,-3.68,-4.23")

' Turn the tracking mode to Off.
TestSet.WriteString("DISPlay:MODE:TRACking OFF")
' ***Configure Band and Channel parameters***

' Set the RF analyzer frequency control to auto.
TestSet.WriteString("RFANalyzer:CONTrol:FREQuency:AUTO:GCALibration ON")

' Set ARFCN to 512 for the DCS band.
TestSet.WriteString("TRANsceiver:CHANnel:DCS 512")

' Set ARFCN to 5 for the EGSM band.
TestSet.WriteString("TRANsceiver:CHANnel:EGSM 5")

' Set ARFCN to 259 for the GSM450 band.
TestSet.WriteString("TRANsceiver:CHANnel:GSM450 259")

' Set active GSM/GPRS band.
TestSet.WriteString("TRANsceiver:BAND:GCALibration DCS")

' Set the RF Generator power state to on and the level to to -70 dBm.
TestSet.WriteString("RFGenerator:POWer:GCALibration -70")

' Set the RF analyzer expected input power control mode to manual.
TestSet.WriteString("RFANalyzer:CONTrol:POWer:AUTO:GCALibration ON")

' Set uplink burst 2 to be measured for single slot measurements.
 TestSet.WriteString("RFANalyzer:BURSt 2")

' Set the Mobile Station TX Level for burst 1 in the DCS band to 10.
TestSet.WriteString("RFANalyzer:MS:TXLevel:DCS:BURSt1 10")

' Set the Mobile Station TX Level for burst 2 in the DCS band to 12.
TestSet.WriteString("RFANalyzer:MS:TXLevel:DCS:BURSt2 12")
```

```
' Set the test set to expect Training Sequence Code (TSC) 2 in the
' midamble burst.
TestSet.WriteString("RFANalyzer:BURSt:TYPE TSC2")

' Configure Power and Modulation Quality Measurement Suite parameters.

' Set Power and Modulation Quality suite trigger arm to single.
TestSet.WriteString("SETup:PMODulation:CONTinuous OFF")

' Set Power and Modulation Quality suite multi-measurement count to 100.
TestSet.WriteString("SETup:PMODulation:COUNt 100")

' Set VISA timeout to 10 seconds. This timeout should be longer than the
' following measurement timeout value.
TestSet.IO.Timeout = 10000

' Set Power and Modulation Quality suite timeout to 5 seconds.
TestSet.WriteString("SETup:PMODulation:TIMeout 5S")

' Set the time offsets for the Power versus Time power measurement.
TestSet.WriteString("SETup:PMODulation:PVTime:TIME -28uS,-
10uS,321.2uS,552.8uS,570.8uS")

' Set the burst synchronization mode to Midamble.
TestSet.WriteString("SETup:PMODulation:SYNC MIDamble")

' Configure RF Channel Measurement Suite parameters.

' Set RF Channel suite trigger arm to single.
TestSet.WriteString("SETup:RFCHannel:CONTinuous OFF")

' Set the state to on and the multi-measurement count value to 150.
TestSet.WriteString("SETup:RFCHannel:COUNt 150")

' Set RF Channel suite timeout to 5 seconds.
TestSet.WriteString("SETup:RFCHannel:TIMeout 5S")

' Set RF Channel suite filter to 1 kHz.
TestSet.WriteString("SETup:RFCHannel:FILTer BWKHZ1")

' Set trigger delay time to 1.5 ms for the 1 kHz filter.
TestSet.WriteString("SETup:RFCHannel:TRIGger:DELay:BWKHZ1 1.5mS")

' Setup Initiate
TestSet.WriteString(“SETup:PMODulation:INITiate PFERror”)

' Initiate a phase and frequency error measurement.
TestSet.WriteString("INITiate:PMODulation PFERror")

' Fetch phase and frequency error results.
TestSet.WriteString("FETCh:PMODulation:PFERror?")
FetchPmodulationPferror = CType(TestSet.ReadList(), Array)

' End connection
TestSet.WriteString("SYSTem:PRESet3")

' Close IO interface to test set
TestSet.IO.Close()


End sub
End Class
```

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.

www.agilent.com

Agilent Technologies' Test and Measurement Support, Services, and Assistance
Agilent Technologies aims to maximize the value you receive, while minimizing your risk and problems. We strive to ensure that you get the test and measurement capabilities you paid for and obtain the support you need. Our extensive support resources and services can help you choose the right Agilent products for your applications and apply them successfully. Every instrument and system we sell has a global warranty. Two concepts underlie Agilent's overall support policy: "Our Promise" and "Your Advantage."

Our Promise
Our Promise means your Agilent test and measurement equipment will meet its advertised performance and functionality. When you are choosing new equipment, we will help you with product information, including realistic performance specifications and practical recommendations from experienced test engineers. When you receive your new Agilent equipment, we can help verify that it works properly and help with initial product operation.

Your Advantage
Your Advantage means that Agilent offers a wide range of additional expert test and measurement services, which you can purchase according to your unique technical and business needs. Solve problems efficiently and gain a competitive edge by contracting with us for calibration, extra-cost upgrades, out-of-warranty repairs, and onsite education and training, as well as design, system integration, project management, and other professional engineering services. Experienced Agilent engineers and technicians worldwide can help you maximize your productivity, optimize the return on investment of your Agilent instruments and systems, and obtain dependable measurement accuracy for the life of those products.

United States:
(tel) 800 829 4444
(fax) 800 829 4433
Canada:
(tel) 877 894 4414
(fax) 800 746 4866
China:
(tel) 800 810 0189
(fax) 800 820 2816
Europe:
(tel) 31 20 547 2111
Japan:
(tel) (81) 426 56 7832
(fax) (81) 426 56 7840
Korea:
(tel) (080) 769 0800
(fax) (080) 769 0900
Latin America:
(tel) (305) 269 7500
Taiwan:
(tel) 0800 047 866
(fax) 0800 286 331
Other Asia Pacific Countries:
(tel) (65) 6375 8100
(fax) (65) 6755 0042
Email: tm_ap@agilent.com
Contacts revised: 09/26/05

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

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

Printed in USA, August 27, 2006
5989-5528EN

Agilent Technologies