Agilent
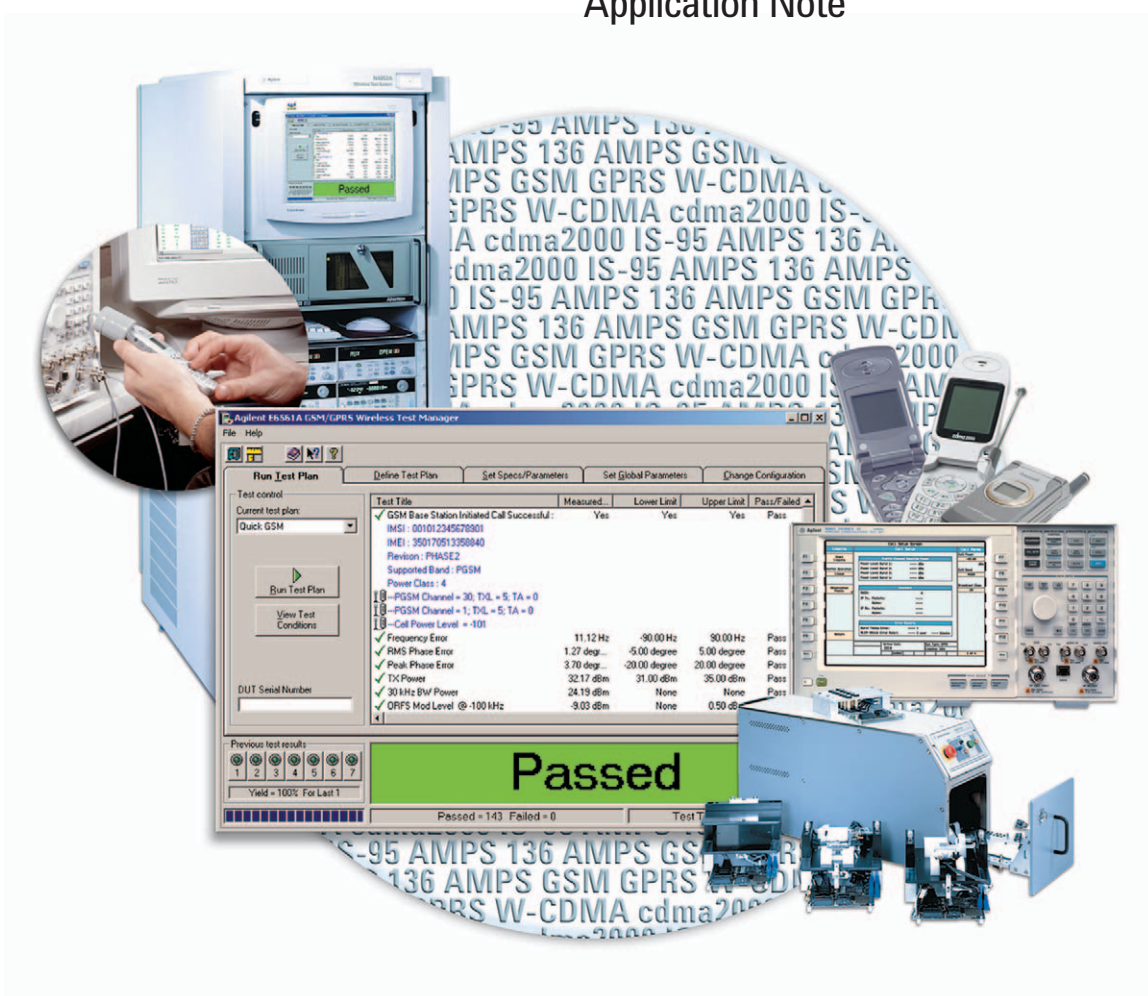# Customizing the E656XC/N588XA Wireless Test Manager

Application Note

Agilent Technologies

Table of Contents

1 Introduction

The Wireless Test Manager (WTM) is used in conjunction with the Agilent 8960 Wireless Communications Test Set to provide versatile and customized testing options for a wide variety of wireless device technologies.

The WTM is a Visual Basic .NET® program, which has the ability to completely dictate testing and measurements run on the 8960 test set through the use of a computer. The 8960 acts as a serving cell by sending messages to a wireless device and receive information in return to make a data or voice connection.

This document explains how to manipulate the WTM program in order to perform extra test procedures and implement some of the program's advanced capabilities. To perform the tasks in the following sections, the WTM Development and Runtime modes must be installed on the computer.

This application note follows the procedures for using the WTM program in compliance with the GSM/GPRS mobile test on the 8960 test set, but it should be mentioned that the procedures in this application note generally work with all test applications (TAs) and lab applications (LAs) on the 8960 test set.

If there are any problems with the program, or questions that are not answered in this application note, please refer to the help files in the WTM program. These help files can be accessed either from within the WTM program, or you can click the **Start** button, and then point to **All Programs, Agilent Wireless Test Manager .NET, Help,** and then click the name of the help file you want to open.

The following information applies to each of the sections in this application note. Please review this important information before proceeding further.

2 General Comments

2.1 WTM architecture

The WTM is composed of several interactive functions. Each of these functions is shown in the following graphic.
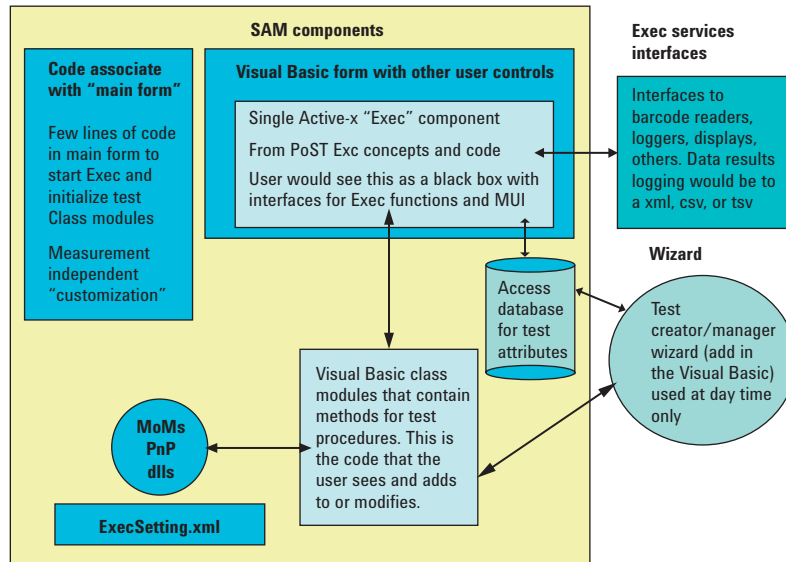
- **SAM components** – Software application for manufacturing (SAM), and the components that are a part of the SAM, allow RF testing of mobile devices.

- **"Main Form" code** – Initializes all components in the SAM to include the Executive code.

- **Visual Basic class modules** – Organized to contain collections of routines to accomplish different tasks. For example, the GSMGPRSEGPRSTransmitter.vb class module contains test steps that are associated with transmitter testing such as the output RF spectrum test.

- **MoMs, PnP, DLLs** – These are interfaces that allow the test steps to accomplish their testing. MoMs are Agilent-proprietary methods of measurement (MoM), PnP stands for Plug-n-Play drivers, and DLLs are dynamic link libraries that allow interfacing with items such as the Agilent fading system.

- **ExecSetting.xml** – This file contains settings used by the executive interface to run the software. When many of the global settings are changed in the program form, they are changed in this file.

- **Access database** – This contains all of the settings for individual test plans such as the specification and parameter settings.

- **Wizards** – These are programs that allow fast and easy modification of the WTM. It is highly recommended that you use these wizards when modifying the WTM. If you make modifications manually, you can easily overlook some of the settings that must be changed.

- **Executive services interface** – Interfaces to barcode readers, loggers, display, and other items. Results can be stored in xml, csv, or tsv, or reduced/graphical csv.

2.2 Setting up the wizard

There are two wizards included in the WTM program that help to modify the test plan code in a more efficient manner. The wizards lead you through several steps to gather information on what needs to be added to the program and then the wizards automatically insert these items into the test code. These wizards can help with adding items such as test steps, instruments, and parameters.

The names of the wizards are the 'Add Wireless Test Wizard' and the 'Add Instrument Control Wizard.' The easiest way to use these tools is to have the wizard buttons available on the WTM interface. Set up these options now if they do not already appear on the toolbar in the upper left of the Visual Basic .NET project window.
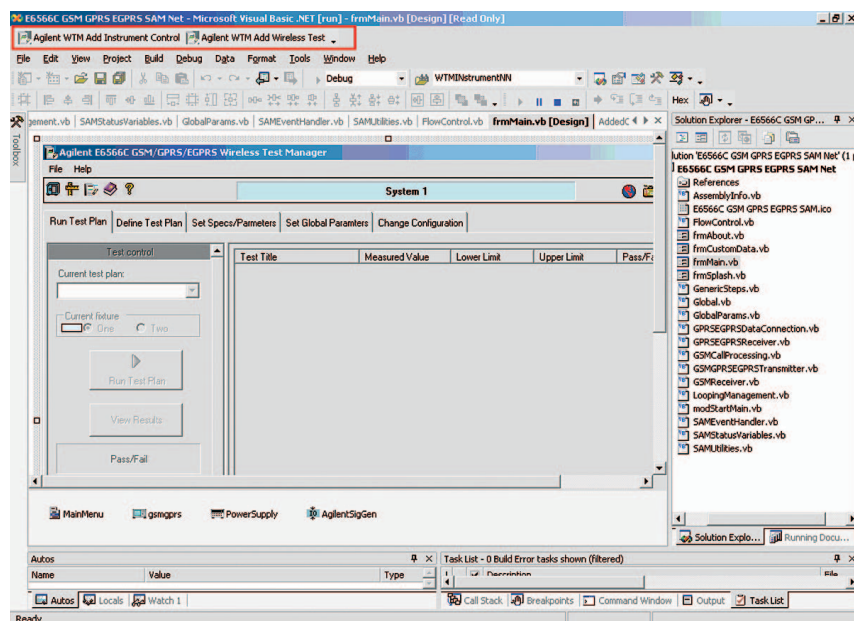
Activate the buttons by going to the **Tools** menu and selecting **Add-in Manager**. This opens a window with a list of the available wizards. Select the **Agilent WTM Add Instrument Control** and **Agilent WTM Add Wireless Test** items. Click **OK** to add the wizards to the toolbar.

Many procedures in the following sections can be completed manually, but it is not recommended unless directed otherwise. If a command, variable name, or other information is typed incorrectly, it may trigger complications with the rest of the program that could potentially cause the program to generate an error or operate incorrectly.

2.3 Selecting a database in the wizard

When using the **Add Wireless Test Wizard**, you are prompted to select a database to make the wizard changes in. If the database is installed in the default location, it can be accessed at C:\Program Files\Agilent\WirelessTestManager\E656XX\TestData and the database name is xxxx_DB_English.mdb. For example, the location could be C:\Program Files\Agilent\WirelessTestManager\E6566C\TestData and the database name would be GSMGPRS_SPTA_DB_English.mdb. Any changes made to the test plan with the wizard are recorded to that particular database.

2.4 Saving the project

Always make sure to save a copy of the project before making any changes. This precaution ensures that if something were to go wrong, it will be possible to return to previous work rather than having to reinstall the program.

The easiest way to save project files is to copy the WTM folder into a completely different folder where the files will not interfere with the project. Save the new copies under a different name than the original files. For example, a good class name would be 'GlobalParameters_BACKUP.vb.' This name would make it easy to catalog the files and use them again later.

Changes made in the Visual Basic .NET environment have a lasting effect and it is important to be able to revert back to the original copy of the project in the event that something goes wrong, or if the changes you have made do not yield the results you want.

2.5 Aligning the code

In the WTM program, the database used by the program holds a large volume of information such as values for specifications, parameters, and many other items which the program needs to run correctly. When modifying parts of the test code, it is important to align the database to the test code. This way, the values between the code and database match and the program can run correctly.

To align the database, click the **Agilent Add Wireless Test** button in the upper-left corner of the Visual Basic .NET window. Select the database you want (if using the default setting, the location is C:\Program Files\Agilent\WirelessTestManager\E656XX\ TestData and the database name is xxxx_DB_English.mdb) and then click **Next**. On the following screen, choose the **Align database to test code** option, and then click **Next**.
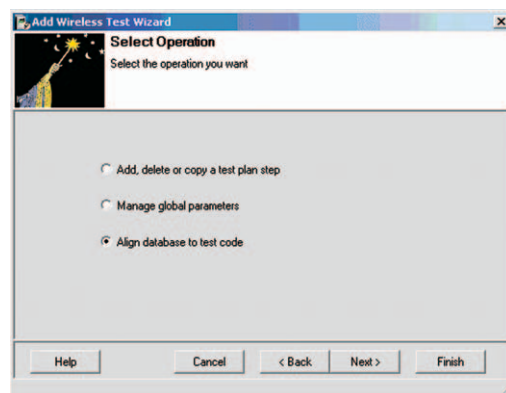
The wizard then goes through all of the classes to align the database. After this process is complete, click **Finish**.

The final page of the wizard provides the option to view the log file for the database alignment. If you would like to see the log, select this option and click **Finish**. A new window is opened displaying the alignment log.
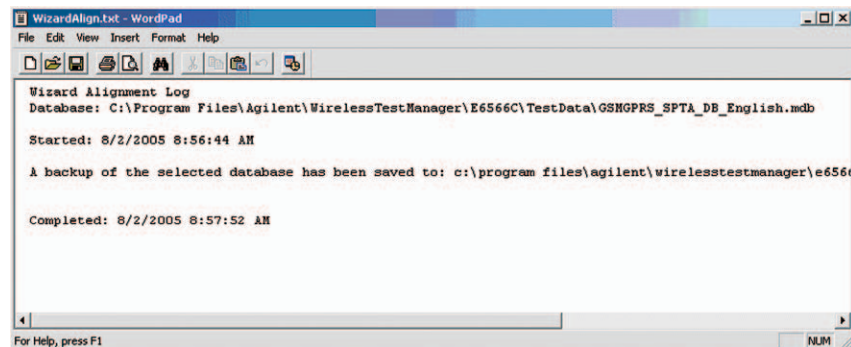
2.6 Updating the database revision

The database revision option allows you to save copies of the database at different points as you modify the WTM program. For example, if you add a new class module to the project, you could make a new database revision. This creates another revision of the project.

Typically, when the Add Wireless Test Wizard is run, the last page gives you the option to create a new database revision.
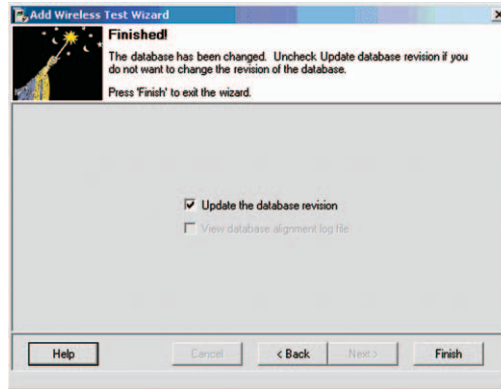
Be careful when making and using revisions. It is important to remember what is in each revision and to align it with the code before running the program. This prevents the program from operating incorrectly.

3 Managing Specifications and Parameters for a Test Step

3.1 Introduction

There are times when the code in the WTM program does not include the specifications and parameters to perform certain tests. This section describes how to add these items to the code. Note: to perform this step, Visual Basic .NET, and WTM Run-time and Development mode must be installed on your computer.

3.2 Overview

Specifications and parameters are added by the wizard but there a few things to be aware of about the process in which the wizard introduces these items.

The wizard adds a parameter to the test step, but not in a manner that you may expect. It completely comments out the original test step code and adds a new skeleton test step after which it includes the new parameter. This is done so that the original test step is essentially left untouched in case it needs to be returned to later.

3.3 Adding a specification or parameter

As previously mentioned you must use the wizard in order to add a specification or parameter. You can choose a parameter from the existing list or create a new one. Then, it must be manually applied to the desired test step.

3.3.1 Adding a specification or parameter using the wizard

Click on the **WTM Add Wireless Test** button in the upper left corner of the Visual Basic .NET environment to open the wizard. Select the desired database (see Section 2.2), and then click **Next**. When prompted, choose the **Add, delete or copy test plan step** option. On the next page, select **Manage specifications and parameters for a step**. Select the step you want to modify, and then click **Next**.

On the following screen, select any extra specifications that need to be added. To do this, highlight the specification in the **Available specifications** list box on the left and click the right arrow button (indicated by the arrow in the following graphic). By clicking the button, the selected specification is added to the **Current specifications** box on the right.
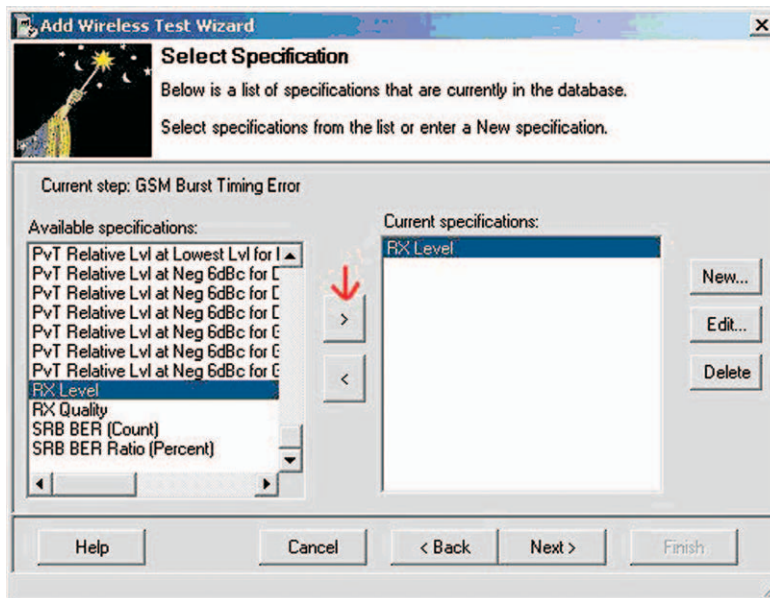
3.3.1.1 Creating a new specification or parameter

It is also possible to create a new specification or parameter. To create a new specification, click the **New** button on the right of the **Select Specification** window in the wizard. When clicked, this button opens a dialog box prompting you for details about the new specification you want to create (for example, name, limits, units, or data type). This dialog box is shown in the following graphic.
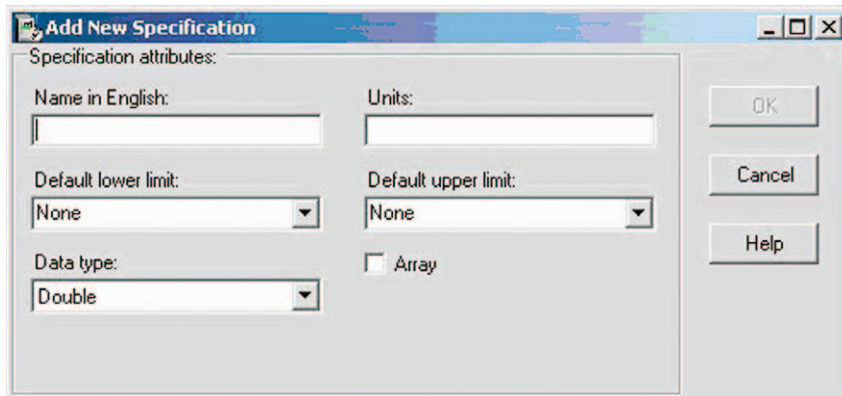
Back on the parameters page, add the parameter(s) you want in the same way used to add specifications. Alternatively, you can create a new parameter to include in the test step. When you have finished adding specifications and parameters, complete the wizard. If there is a large number of specifications and parameters, as is the case for the EGPRS Fast General Test, it may take a few minutes for the wizard to finish.

3.3.2 Integrating specifications or parameters into the test step

When inspecting the test step that the specification or parameter has been added to, it is easy to notice that the original code for the step has been entirely commented out. A new skeleton step code, which includes the new parameter, has been placed below the old code. The easiest way to add the new specification or parameter to the test step is to insert the new lines into the original test code.

As a quick overview on integrating specifications or parameters: the first step is to uncomment the original step code. Then, copy and paste the lines that include the new specification or parameter into the appropriate place in the old code. Do this according to where the lines had appeared in the new code. Make sure to comment out the block of wizard-added code so that it does not affect the test. Next, apply the specification or parameter throughout the original code as required.

The following graphic shows how the original step code was completely commented out by the wizard.
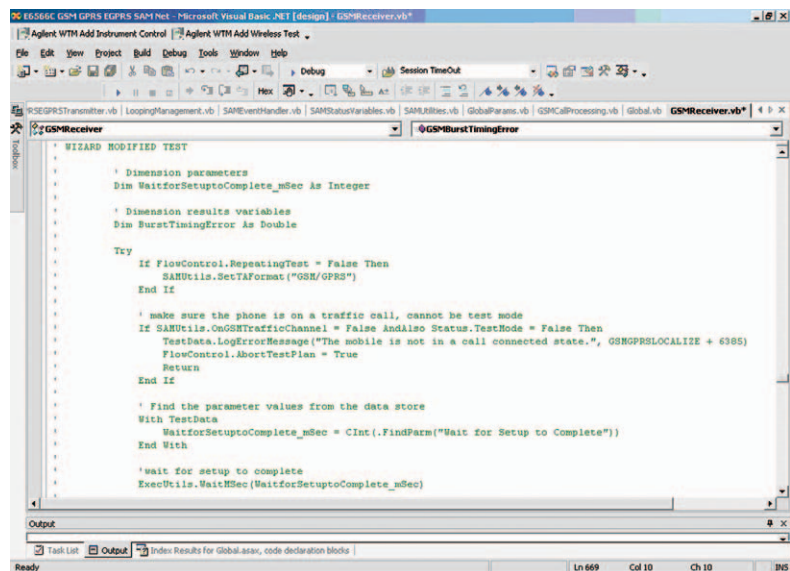
Go to the code which the wizard added (as shown in the following graphic) and review the new components, making note of where they are located. In this example, the parameter added is BandIndicator.
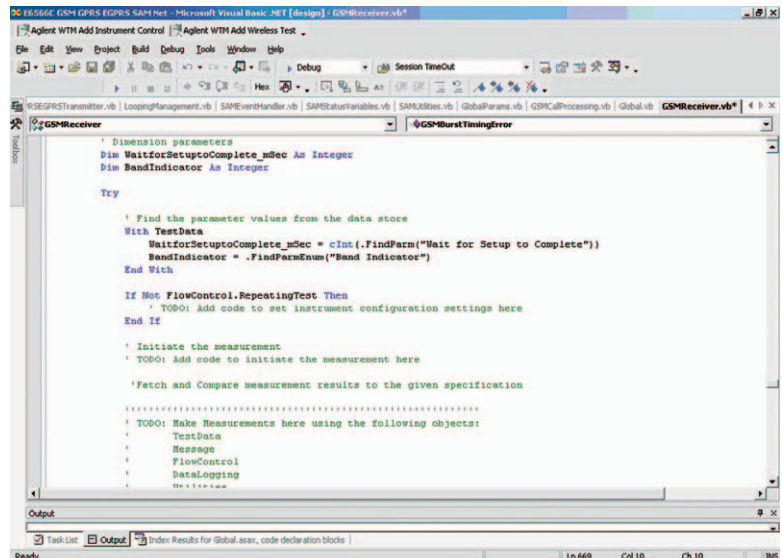
The next step is to copy and paste the lines pertaining to the new specification or parameter into the appropriate places in the old code. Place these lines in accordance with where they appeared in the new code, as shown in the following graphic. The red boxes indicate where the new specification or parameter information has been added into the old step code. After adding the code, be sure to uncomment all of the code for the old step and comment out all of the new step code that the wizard had added. Be careful not to comment out the 'End Sub' statement.
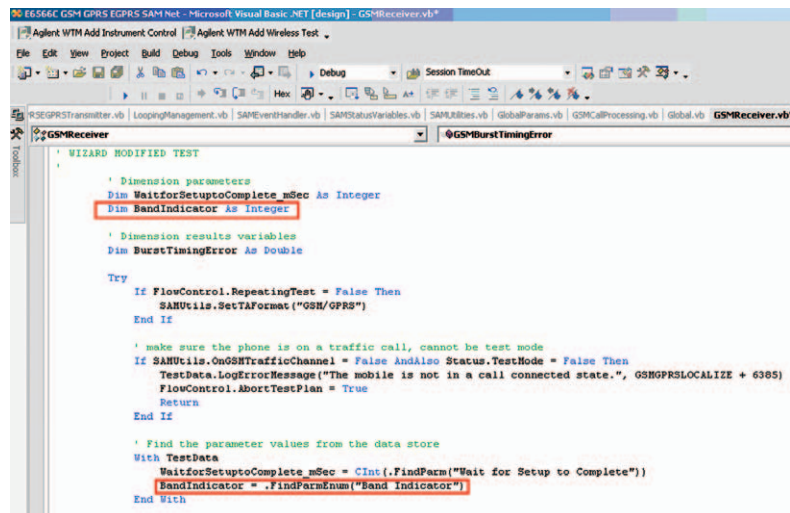
After the specification or parameter has been successfully added to the test code, it is time to give it functionality. A simple statement could be written as follows:

```
If BandIndicator = 1 Then
    gsmgprs.BaseStationEmulation.CallParameters.TCH.
    TrafficChannelARFCN(20, dmmtCellBandEnum.dmmtPGSM)
End If
```

This code says that if BandIndicator equals 1, the cell band will be set to PGSM and the broadcast channel will equal 20.

When adding code to the WTM, placement is important. There are parameters and specifications at the beginning of a test step and following this section is the code that controls the initial setup for a test step. Do not add any code to these sections. It would be safe to add code before the start of the measurements, but after the parameters received their values from the test set.

Save the project to make sure the changes are permanent.

3.4 Editing a specification or parameter

You can change characteristics of a specification or parameter such as its name, limits, units, or default values. This section describes how to modify existing specifications or parameters.

3.4.1 Editing specifications or parameters using the wizard

Click on the **WTM Add Wireless Test** button to open the wizard, and follow the steps as described in Section 3.3.1 until you navigate to the **Select Specification** or **Select Parameter** window. Select the specification or parameter you want to edit and click the **Edit** button. This opens a dialog box allowing you to make changes to the characteristics of the specification or parameter.

You must be extremely careful when changing a standard specification or parameter of the WTM program. Modifying a specification or parameter that had originally been a part of the program affects the operation of the test steps that include those particular specifications and parameters.

3.5 Deleting a specification or parameter

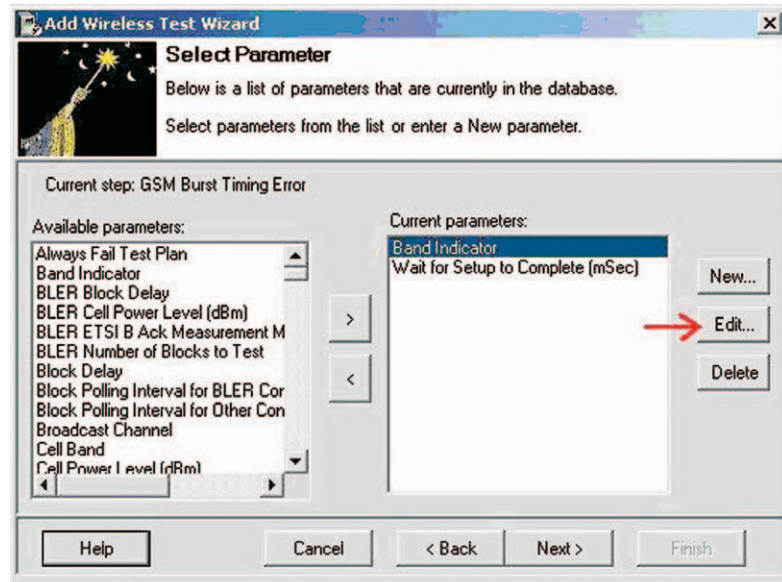When deleting a specification or parameter, it is important to use the wizard to perform this action. The wizard completely disables the specification or parameter from affecting the rest of the code in any way. It comments out the item from the code and deletes its information in the comment code at the beginning of a test step.

3.5.1 Deleting specifications or parameters with the wizard

Click on the **WTM Add Wireless Test** button to open the wizard, choose the appropriate database, select **Add, delete or copy a test step**, and then select **Manage specifications and parameters for a step**. Choose the correct test step and click **Next**. In the Current specifications box on the right, select the parameter you want to delete, and then click **Delete**. Finally, click **Next** and **Finish** to apply the changes and exit the wizard.

3.5.2 Applying deletion changes to a test step

This process is very similar to adding a parameter to a test step, so it is recommended that you review Section 3.3.2.

The original test step code has been commented out by the wizard. New code has been added below that omits the specification or parameter that was deleted through the wizard. The best way to implement these changes is to comment out the new code and uncomment the original code. At this point, since the wizard has deleted the specification or parameter, it is safe to delete the lines of code containing that specification or parameter. Delete these lines according to where they had been deleted in the code added by the wizard. Make sure that no other steps are affected by deleting this specification or parameter.

3.6 Summary

The wizard provides a powerful way to add, delete, or edit specifications and parameters. This ensures that items are instantiated correctly and helps to cut down on possible typing errors.

In general, adding, editing, or deleting a specification or parameter includes the following procedures.
- Running the wizard
- Manually changing the code to integrate the specification or parameter
- Saving the project or class modules being modified

4 Changing a Test Step for the Wireless Test Manager

4.1 Introduction

This section describes how to change an existing test step in the Visual Basic .NET code. This procedure is appropriate for any step requiring modification. In order to perform this task, Visual Basic .NET, and the WTM Run-time and Development mode must be installed on your computer.

4.2 Overview

Test steps consist of three main parts
- Specifications
- Parameters
- Code for test procedures
  - Test set up
  - Measurements
  - Results

Before changing a test step, make sure that you have saved a copy of the project prior to making any modifications (see Section 2.3).

It is also recommended that you carefully review the changes being made to the test step. Make sure to thoroughly investigate what test components are affected by this test step change. Also check to see if there are steps in different modules which may be affected as well. Changing the specifications and parameters are covered in a later section of this application note.

4.3 Changing a test step

There are a number of ways in which a test step can be changed. This section provides one example to illustrate some points to remember while making modifications.

4.3.1 Step change examples

To modify a test step, navigate to the class that includes the test step that you want to change. Change the step as needed. Be careful about what code you modify because it could potentially produce errors or cause the program to operate incorrectly.

For instance, refer to the following section of code:

```
Dim  MeasurementTimeout  As Double
If   FlowControl.RepeatingTest = False Then
     .
     .
    MeasurementTimeout = SAMUtils.
    CalculatedMeasurementTimeout((NumberofBitstoTest /
    20000) * 1.5)
     .
     .
End If
```

Let the code be modified such that the multiplication value is changed to 2. This is a simple procedure that only requires one character to be altered in the code. When modifying code within the WTM project, it is very important to comment any change being made. This makes it much easier to review your work and track any problems that arise after code modification.

```
Public Sub GSMMeasureBitError()
        .
        .
    Dim  MeasurementTimeout As Double
        .
        .
    If   FlowControl.RepeatingTest = False Then

        MeasurementTimeout = SAMUtils.
        CalculatedMeasurementTimeout((NumberofBitstoTest
        / 20000) * 2) 'Changed 1.5 to 2,
        Date: 7/20/05, Author: John Smith
        .
        .
    End If
End Sub
```

This example relates to code from the class GSMReceiver and the test step GSMInitiateBitError. The code was changed by increasing a number in the formula for MeasurementTimeout. By changing 1.5 to 2 the MeasurementTimeout value becomes larger. This action extends the period of time before the test set ends the test due to no activity during the measurement. Make sure to save the changes.

4.4 Summary
Being able to change a test step is a basic advantage of the Visual Basic .NET-supported WTM development mode. It is very important to remember to save a copy of the project before and after changing anything in the program. In this way, it is always possible to return to the starting point without having to reinstall the program. In general, changing a test step includes the following procedures:
- Making appropriate changes
- If a variable value was changed, resetting it to the original value before the end of the sub
- Saving the project or class modules being modified

5 Managing Global Parameters for the WTM

5.1 Introduction
This section describes how to control the global parameters of the WTM. The difference between normal parameters and global parameters is that global parameters can affect all the steps of a test plan, while normal parameters only affect one step of a test plan. It is also possible to set different global parameters for each test plan. In order to perform this task, Visual Basic .NET and WTM Run-time and Development mode must be installed on your computer.

5.2 Overview

Global parameters are very useful components of the test program. Due to their sharing properties, the amount of code needed in the program is reduced and they provide a guaranteed uniformity throughout the test plan for certain parameters.

Since global parameters are meant to be uniform throughout the entire test plan, it is strongly recommended that you use the wizard in order to change global parameters.

When adding a global parameter, it is simply placed in the commented code of the global parameter class for use by all of the test plans. It is left to the developer to decide how to implement the new global parameter.

5.3 Adding global parameters

When the wizard is run, it automatically adds information to the global parameters comment code and adds the global parameter into the program menu interface that controls the testing of the WTM.

5.3.1 Adding global parameters using the wizard

To add a global parameter, click on the **WTM Add Wireless Test** button to open the wizard, choose the appropriate database, and then click **Next**. Select the **Manage global parameters** option, and then select to **Add, delete or modify the global parameters**. The next page shows a list of the existing global parameters. Click the **New** button on the right of the page, enter details for the new global parameter in the boxes under Parameter attributes, and then click **OK**. To add the new global parameter to the code, click **Next**.
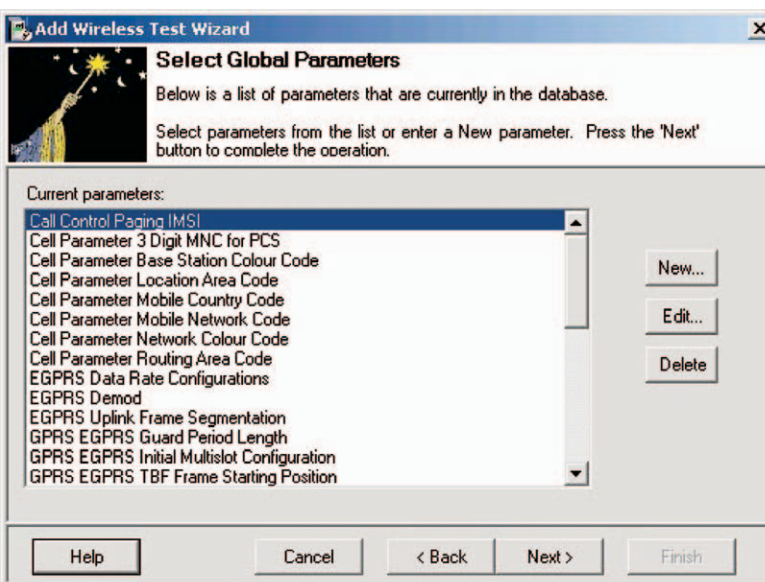
5.4 Editing global parameters
This is a simple process when implemented with the wizard. Changes take place in the global parameters comment code and in the WTM program interface.

5.4.1 Editing global parameters using the wizard
Click on the **Agilent WTM Wireless Test** button. Select the database you want and then choose **Manage global parameters**. On the following window, there are two choices. The first is to 'Add, delete or modify the global parameters.' By choosing this option, any modifications that are mad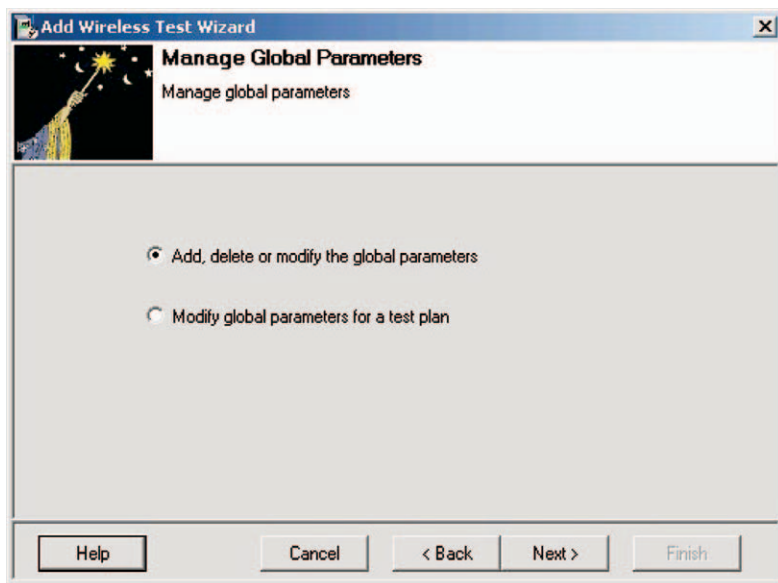e affect every test plan. The second choice, 'Modify global parameters for a test plan' is self-explanatory in that the modifications only affect one test plan.

When choosing the first option, the next screen shows a list of all of the current parameters. Highlight the parameter you want to modify and click the **Edit** button on the right of the window. This opens a window with the parameter characteristics. Edit them as necessary. Click **OK** and then **Next** to apply the changes.

If the second option on the Manage Global Parameters screen is chosen, the proceeding window shows a list of available test plans. Select the test plan which the changes are applied to and click **Next**. Edit them as necessary. Click **OK** and then **Next** to apply the changes..

5.5 Deleting global parameters
Before starting this section, please be aware that deleting a global parameter is not recommended. Global parameters affect entire test plans and deleting a parameter could potentially stop the program from operating correctly.

Using the wizard to delete a global parameter does several things. First, it deletes the parameter from the global parameter comment code and comments out any code in the test plan that includes the parameter. Secondly, it removes the parameter from the list in the WTM program interface. This illustrates how important it is to use the wizard to handle modifications so that nothing is overlooked.

5.5.1 Deleting global parameters using the wizard

Return to the Add Wireless Test Wizard, select the database, choose **Manage global parameters**, and then **Add, delete or modify the global parameters**. When the list of available parameters is displayed, select the parameter you want to remove. Click the **Delete** button on the right of the page. Finally, click **Next** and **Finish** to apply the changes and exit the wizard.

5.6 Summary

Global parameters are shared throughout each step of a test plan. By using the wizards, you can change the global parameters for every test plan, or only one test plan. Using the wizard increases coding efficiency and reduces the potential for typing errors.

When setting a global parameter in the WTM run-time user interface (not the Visual Basic .NET code) on the Global Parameters tab, the new value for the global parameter is saved automatically. It is changed only for that specific test plan, but it remains the same, even if the program is closed and reopened. Likewise, any changes made on the Set Specs/Parameters tab are saved automatically as well.

6 Creating Output Messages in the Wireless Test Manager

6.1 Introduction

In some cases, when running a test plan, you may want to have a customized output message. For example, this message could be used to pause the test procedure, or to show results in the middle of a test step. Note that Visual Basic .NET and the WTM Run-time and Development modes must be installed on the computer.

6.2 Overview

There are a few different ways to send messages through the WTM during the testing process. One way is by using a message box. This box could be displayed halfway through a test or after a test has been completed. Another method would be to use the 'Results' area of the WTM user interface to prompt messages during the testing. Finally, the Pass/Fail section on the bottom of the WTM window could also be used to show messages.

All of these different methods for showing information during the testing process can be used by simply writing a few lines of code in the WTM Development mode.

6.3 Adding a message box

A message box is a useful and versatile tool. If it is written into the code so that it is called during the middle of a test, the box could stall the testing process of the WTM until a command button has been pressed to continue. Being able to customize the message box gives you a wide range of options for how to implement the output message.

6.3.1 Coding for message boxes

There are many possible functions a message box can provide. Whichever purpose the message box is used for determines what class the code for the box is written in.

This section provides an example of how to add a message box to the code in order to pause the testing process of the WTM. First, decide what purpose the box is going to serve and then choose the most appropriate test step to implement the message box in. For this example, the message box simply suspends the test process. It is placed in the GSMMeasureBitError step in the GSMReceiver class.

A good location to add the message box would be immediately before the 'initiate measurement' section of the code. You would not want to add code too far before this location because the first section consists of wizard comment code and modifying that section could potentially cause the program to fail. The next section of code declares and allocates memory space of variables used for specifications and parameters. Again, this code should not be modified unless necessary. Finally, the code after the specification and parameter variable declarations is used for the initial test set up of the step. The area to add the code is shown in the following screen shot.

There are three different ways to write code for this message box. The first way is to use a message box that simply displays text and no icons.

```
If FlowControl.EmbeddedSystem.MsgBox("Would you like to
    continue testing?", exEmbeddedMsgSytleEnum.exMsgBoxYesNo,
    "Continue Testing") = exEmbeddedMsgAnswerEnum.exMsgBoxNo
    Then
    FlowControl.AbortTestPlan = True
    Exit Sub
End If
```

The section of code above opens a simple message box that looks like the following box. When the **No** button is clicked, the test plan aborts, effectively ending the test at the location you want while retaining the information found prior to the event. If you click **Yes** in this situation, the box closes and the test continues running as normal.
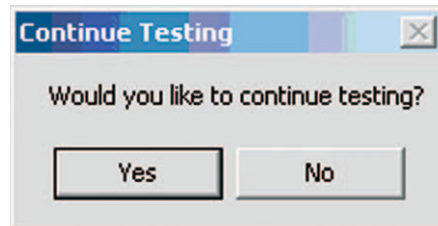
Another way to implement this message box is as follows:

```
If  MessageBox.Show("Would You Like to Continue Testing?",
    "Continue?", MessageBoxButtons.YesNo, MessageBoxIcon.
    Question) = DialogResult.No Then
    FlowControl.AbortTestPlan = True
    Exit Sub
End If
```

The message box that follows has a picture of a question mark in it. It executes the same actions when the buttons are clicked. For instance, if the **No** button is pressed the test aborts and if the **Yes** button is pressed, the test continues.

This last example illustrates changing the buttons of a message box. The following code calls a box that can hold an icon in it, but instead of having Yes and No buttons, it simply has an OK button.

```
If  FlowControl.EmbeddedSystem.MsgBox("The test is
    working correctly.", exEmbeddedMsgSytleEnum.exMsgBox
    Information, "Test Status") = exEmbeddedMsgAnswerEnum.
    exMsgBoxNo Then
    FlowControl.AbortTestPlan = True
End If
```
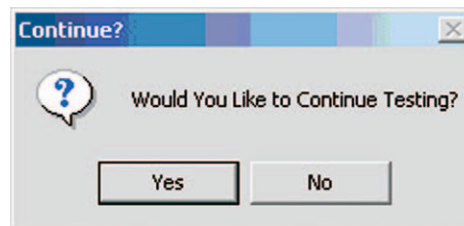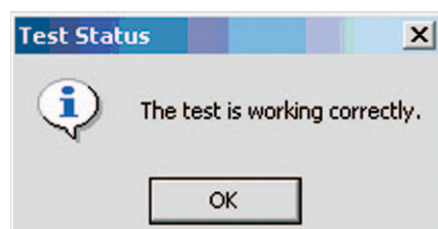

Each of these code examples above can be modified to show different buttons and icons in the boxes by using the Visual Basic Intellisense that appears as the code is being written.

6.4 Adding a message to the results area

The results area is characterized as the main portion of the WTM user interface form. This is the section where all of the test outcomes are displayed. It is possible to add alerts and messages into this region as the test runs.

6.4.1 Coding for the results area

Putting messages in the results section of the WTM user interface is convenient for presenting information to the user so that they are aware of what is happening during the middle of the test. For instance, comments could be displayed throughout the test to explain what is happening or highlight an important measurement. This is another procedure that requires only a few lines of code.

The easiest way to put comments or other items in the results area of the WTM user interface is by using the command TestData.PrintMessage ("Put your comment here"). TestData is the main command that sends information to cells of the result area. Intellisense shows a number of different ways the TestData command could be implemented. This example simply adds a comment as shown below.

`TestData.PrintMessage` ("This is a user defined comment during testing")
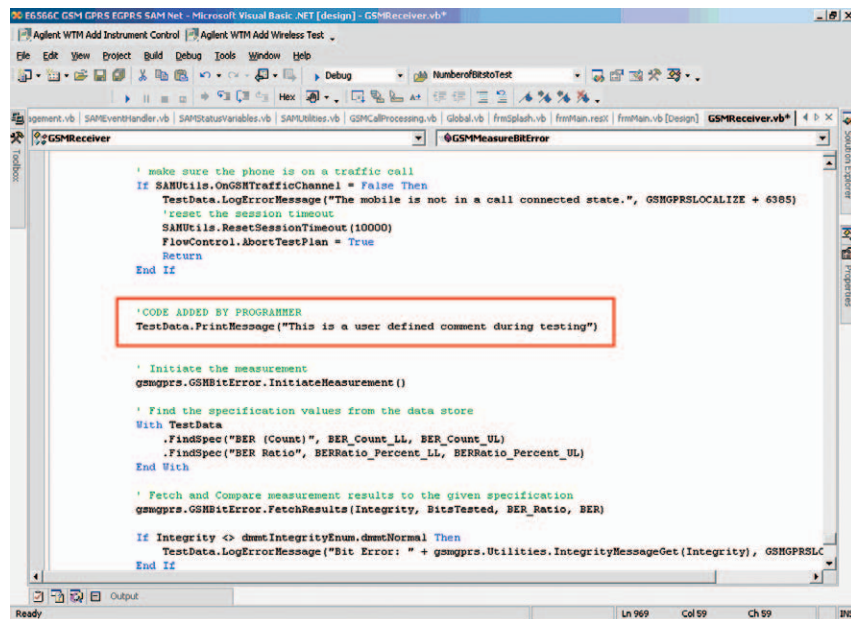
During the test, this line appears in the results area according to where it was placed in the code. The WTM user interface with the added comment is shown in the following screen shot.
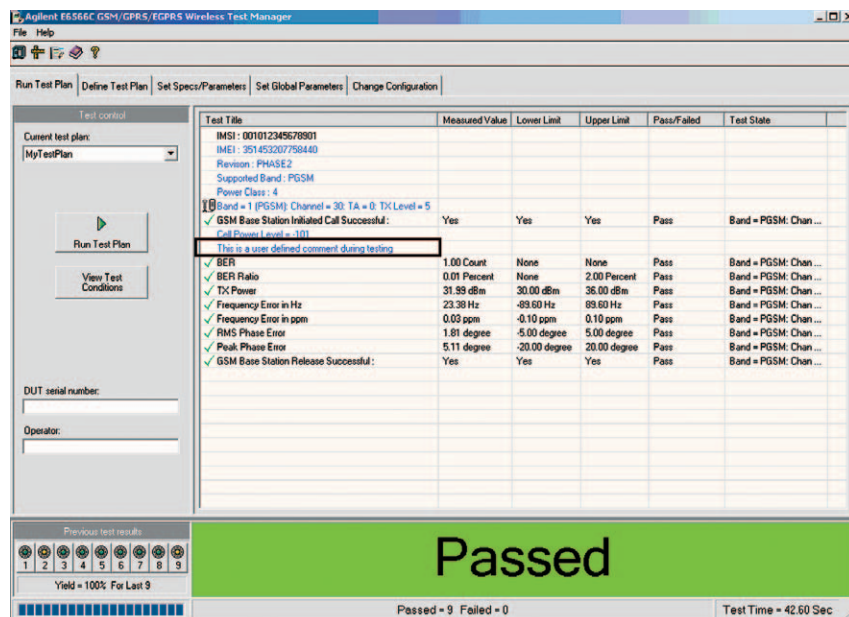
6.5 Adding a message to the Pass/Fail box

The Pass/Fail box is used to show what stages the test plan is in at a certain point in time. When trying to establish a call, the Pass/Fail box tells you what stages the program is going through to accomplish this. Most importantly, it displays whether the device under test (DUT) has passed or failed the chosen test plan.

6.5.1 Coding for the Pass/Fail box

You can place output messages in the Pass/Fail box. The only consideration when placing messages in this box, is that they do not stay there permanently. As soon as the test plan determines that a new message needs to be placed in the Pass/Fail box, it replaces the previous message.

The following block of code specifies the font size and alignment as well as the message to be added to the box.

```
With  Message.Memo
      .FontSize = 14
      .Alignment = HorizontalAlignment.Center
      .NewText("Here is user added text in the pass/fail box")
End With
```
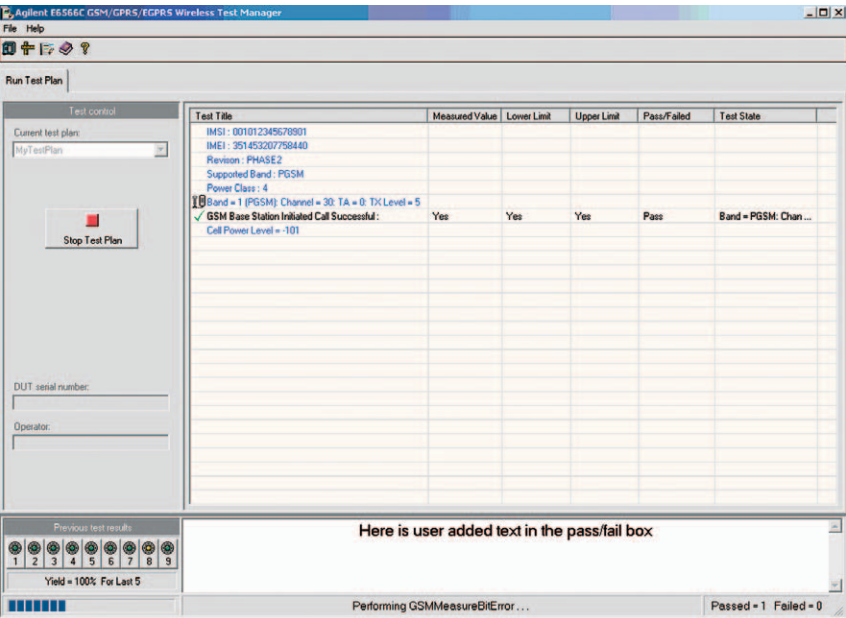

It is also possible to add a comment by simply using the command:

`Message.Memo.NewText` ("Put your comment here")

This puts the comment in the box at a default font size and alignment. The Intellisense associated with the Message.Memo command shows the programmer other options to change, such as the font type and background color.

Output messages can also be displayed on the center portion of the status bar at the bottom of the WTM window, below the Pass/Fail box. If you would like to display text on the status bar, use the command:

`Message.TitleDisplay.NewText` ("Put your comment here")

The following screen shot shows a message displayed on the status bar (highlighted by a black outline).
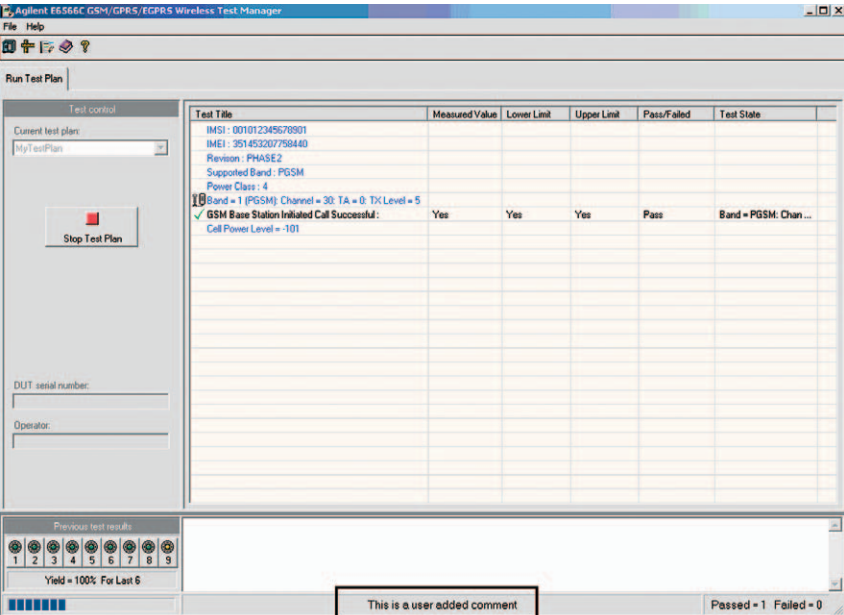
6.6 Summary

There are many ways that an output message can be displayed while using the WTM. The most appropriate method depends on the purpose of the output message. For example, it could be used to pause the analysis, insert comments, or perhaps display results halfway through the test plan to report progress.

You can add an output message to the test project using any of the following techniques:
- Message boxes
- The test results area on the Run Test Plan tab
- The Pass/Fail box
- The center portion of the Status Bar

7 Managing Test Steps for the Wireless Test Manager

7.1 Introduction
Manipulating a test step is a basic function used for customizing the WTM test program code. This section provides the procedures for successfully adding, deleting, and copying a test step. Note that Visual Basic .NET and the WTM Development and Run-time modes must be installed on your computer.

7.2 Overview
Test steps are the basic building blocks of the test plan code as a whole. This is where measurements are initiated and results are derived. Being able to add, delete, and copy a test step provides great flexibility to manipulate the WTM program for specialized tests and purposes.

There are many ways in which a test plan could potentially be affected by a test step. This makes it important to use the Add Wireless Test Wizard when adding, deleting, or copying test steps. Using the wizard ensures that all instances of the test step are affected by the modifications made.

7.3 Adding a test step
This process adds a completely new test step to the WTM program. Using the wizard, a class module is chosen to place the step in, a name is assigned, and specifications and parameters are added. The minimum amount of test code is placed in the class module to begin the test step.

7.3.1 Using the wizard to add a test step
To add a test step, click on the **Agilent WTM Add Wireless Test** button in the upper left corner to open the **Add Wireless Test Wizard**. Select the appropriate database (see Section 2.2) and click **Next**. Select the **Add, delete or copy a test step** option and then choose **Add a new step**.

The next window has a list of class modules available. Choose the appropriate module to add the test step to and click **Next**. Alternatively, a new class module can be created to add the test step to by clicking the **New** button on the right. Be aware that this action creates a new file.
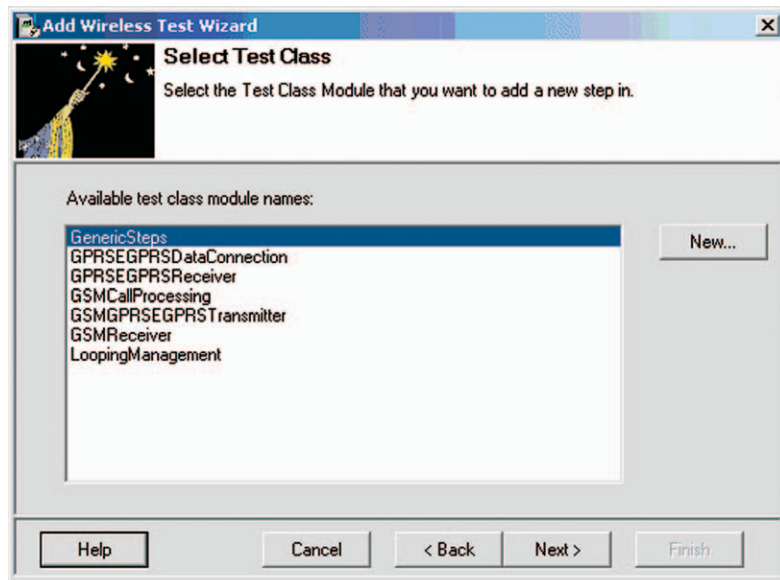
7.3.2 Creating the test step

After choosing a class module, it is time to create the actual test step. The following screen shot provides an example of how to fill out the Add New Step page.

When typing the step name, make sure that you do not use any of the Visual Basic keywords. Some examples of these words include New, Me, Public, Dim, and Try.

The next box allows you to type a new test group name or select an existing one. The test group name is used by the available steps filter on the WTM user interface to reduce the number of steps which are displayed at any one time. This makes it easier to locate the step you are trying to find.

7.3.3 Specifications and parameters

The next two windows in the wizard allow you to add specifications and parameters to the test step. They can be added by highlighting the specifications and parameters in the box on the left and clicking the right arrow button (located in the center of the wizard page).

If a parameter is accidentally added, it can be quickly removed without affecting the test step. On the same wizard window, highlight the parameter in the box on the right and click the left arrow (located in the center of the wizard page). After you have finished adding specifications and parameters, click **Next**.
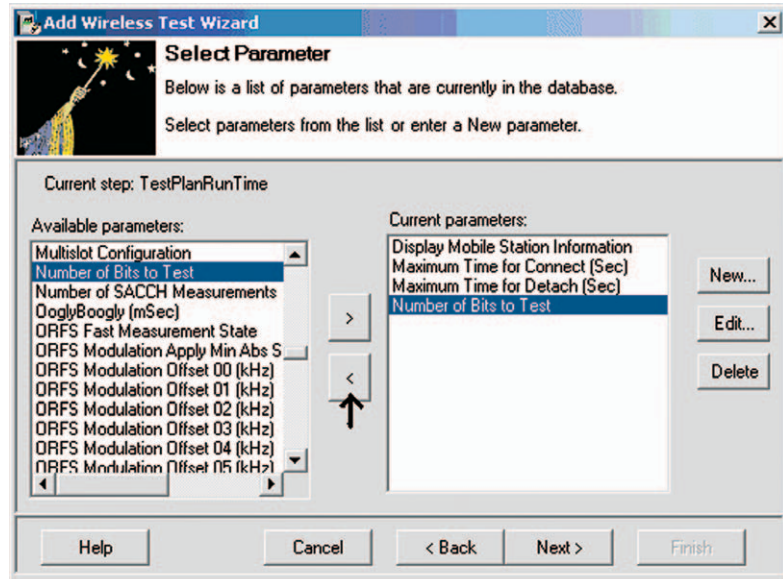
The next page (as shown in the following screen shot) displays all of the specifications and parameters that you have selected for the test step. Confirm your selections and click **Next**.
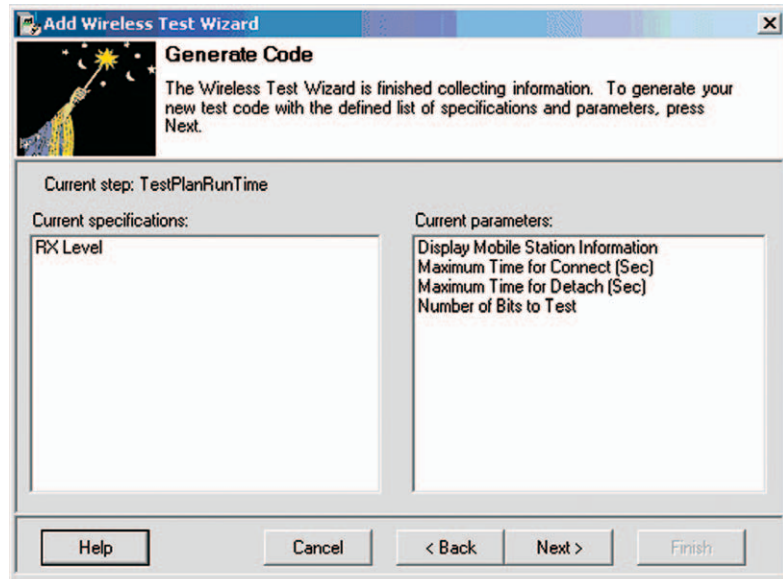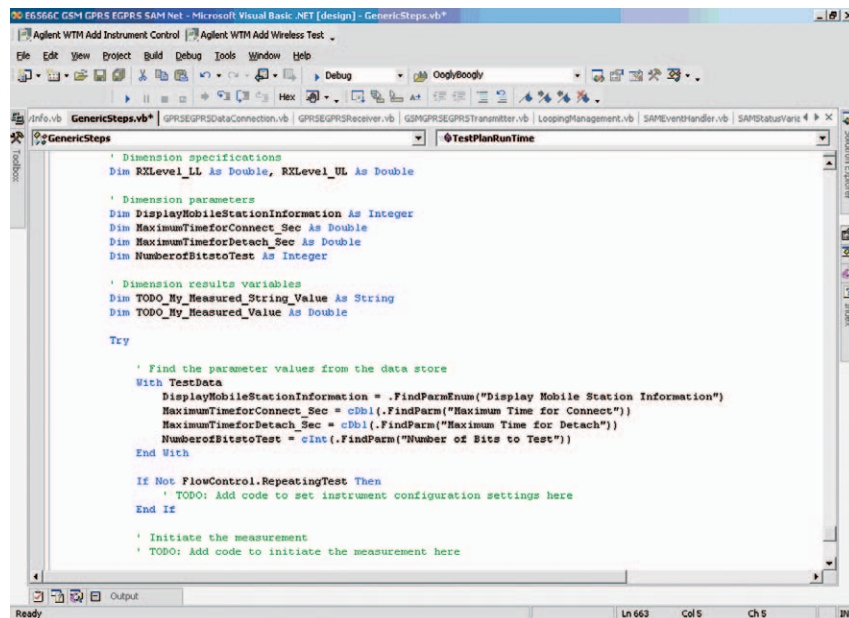
The test step is then added to the code. Click **Finish** to close the wizard. A final page asks if you want to update the database revision. If you select this option, a new revision of the database is created so that you can return to the original revision if necessary.

Note that if you modify a standard parameter or specification in the future, the change could possibly affect other test steps that use those same specifications or parameters. This may cause the program to fail if the test steps cannot operate correctly.

7.3.4 Coding the new test step
After the wizard has finished, the code for the new test step is placed at the bottom of the selected class module. Some code is already present in the step, along with the selected parameters and specifications.

The following screen shot shows the code that was automatically added by the wizard:

The first section of code consists of specifications and parameters for the test step. Do not add or delete anything in this section when editing the code manually.

After the specifications and parameters is a section of code that relates to the initial set up of the test step. This block of code brings in information from the DUT required for the test step to begin, and to provide results for test inquiries such as the 'Maximum connection time.'

27
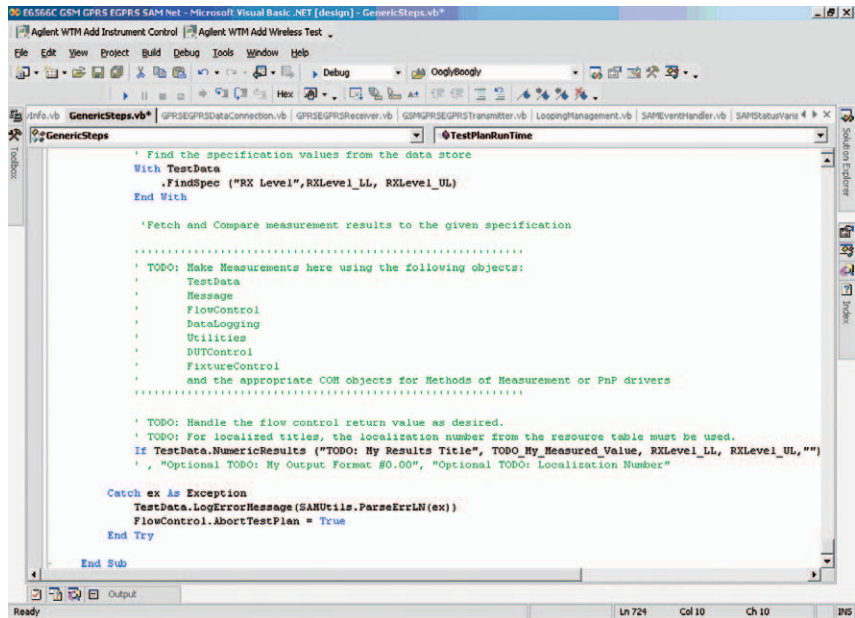
In the wizard-generated code, there is a line that checks if the test step is going to repeat the step process more than once. The line is as follows:

```
If Not  FlowControl.RepeatingTest    Then
        ' TODO: Add code to set instrument configuration
        settings here
End If
```

Throughout the wizard-generated test step, there are comment lines labeled 'TODO.' These lines let the programmer know where it is necessary to write custom code for the test step in order for it to work as intended.

Within the 'If-Not...Then' statement above, code must be added to respond to the length of the test step. If the step is going to be repeated several times, as is the case when measuring bit error, then FlowControl.RepeatingTest should be set to True. For other measurements that do not need to be run more than once, FlowControl.RepeatingTest should be set to False. Add code to this statement to initiate a timeout for the measurement so that it does not continue to loop more than needed.

The rest of the test step code relates to making measurements and performing comparisons as needed. This consists of finding the specification values and testing them to see if they are within an acceptable range.

The wizard-generated code has placed 'TODO' messages in the test step with comments which guide the programmer through the task of writing the step code correctly. In particular, there is a comment with a list of objects that are available for making measurements.
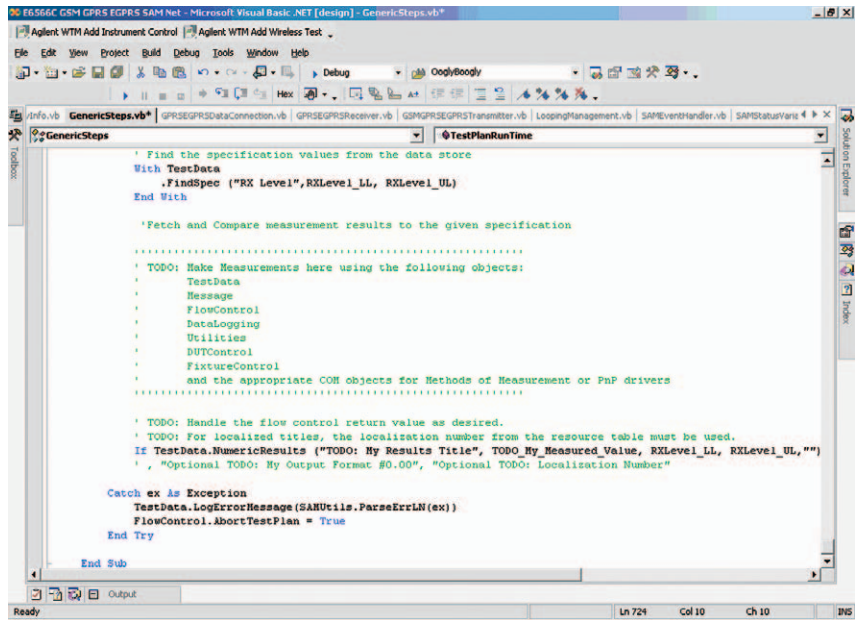
After reviewing the test step and the 'TODO' comments, write code as appropriate for the operations you require the test step to do. Remember to add comments whenever you modify the code to document what you have changed.

7.4 Copying a test step

The wizard greatly simplifies copying a test step. It copies the original test step code and replaces the original name with the new name you specify for the copied test step. This reduces the amount of time and effort required to write another test step.

7.4.1 Using the wizard to copy a step

Run the wizard, select **Add, copy or delete a test plan** and then **Copy test plan**. The next screen lists the available test steps. Highlight the test you want to copy and click **Next** to open the **Copy Step** dialog box (as shown in the following screen shot). Type a name for the test plan copy, ensuring that the name is different from the original test plan name and click **OK**.
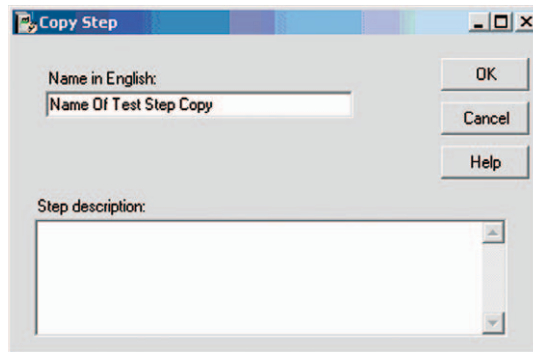
The next screen shows all of the specifications and parameters included in the copy of the test step. These specifications and parameters cannot be modified at this time. If a change is required, return to the wizard after the step has been successfully copied and choose the **Manage specifications and parameters for a test step** option.

To finish copying the test step, click **Next** to apply the changes and then click **Finish**.

If the step has been copied successfully, the wizard places the copied test step at the bottom of the class that contains the original test step. Unlike some other wizard functions, the entire test step is copied. This includes the parameters, specifications, and all of the code that was in the original step.

7.5 Deleting a test step

When using the wizard to delete a test step, the code is not actually deleted, but instead the code for that step is commented out. This is intended to prevent the loss of the source code in case the step is needed for use or reference later on.

7.5.1 Using the wizard to delete a test step

Run the wizard, choose **Add, delete or copy a test step** and then select **Delete a step**. The next page shows a list of available steps to delete. Choose the step you want and click **Next**. The following page confirms the step that is to be deleted. Click **Next** to finish the procedure.

All of the test step code, and any instances of the step, are commented out by the wizard. This is done so that it is possible to reference the test step again if necessary.

7.6 Summary

Test steps are the basic building blocks of the WTM project. The ability to manipulate these steps is important in ensuring a successful test project. The wizard greatly simplifies the process of creating, removing, and managing these steps. As always, remember to save a copy of the project before making any changes to it.

Adding, copying or deleting test steps includes the following procedures:
- Running the wizard
- Writing customized code for added steps and copied steps
- Saving the project or class modules being modified

8 Managing an Instrument in the Wireless Test Manager Using the General Purpose Interface Bus (GPIB)

8.1 Introduction

There are some tests that require the use of an additional instrument in addition to the 8960 test set. This section explains how to add such an instrument by using the General Purpose Interface Bus (GPIB). Instruments can also be added through an IVI.COM driver. In order to perform this task with a GPIB, it is necessary to have Visual Basic .NET and WTM Development and Run-time modes installed on the computer.

8.2 Overview

The GPIB is a standard device used to connect computers and laboratory instruments. It standardizes the signal allowing the instruments and computers to pass data and control information between each other.

The gpioControl item can be used to incorporate a generic instrument interface into the WTM program. This is a versatile control interface that can be used for almost all general instrumentation.

When adding an instrument to the WTM, extra code is required in several class modules. Because of this, the process of adding an instrument is greatly simplified if you use the Add Instrument Control Wizard that automatically adds the necessary code in the appropriate locations.

Another way to add an instrument to the WTM program is through an IVI.COM driver. See Section 9.

8.3 Adding an instrument

There are several steps involved in adding an instrument. You must add the gpioControl .NET component to the main design form and then run the wizard. The wizard places code for the instrument in all of the required areas. The programmer then adds code to control the instrument as required.

8.3.1 Adding the toolbox

Open the design view of frmMain by right clicking on the **frmMain class** in the Solution Explorer window and then clicking **View Designer** (as shown in the following screen shot on the left). On the left of the Visual Basic development environment there is a toolbox that appears when the mouse pointer is paused over it. If the Toolbox option is not visible, open the **View** menu, and then click **Toolbox** (as shown in the following screen shot on the right).
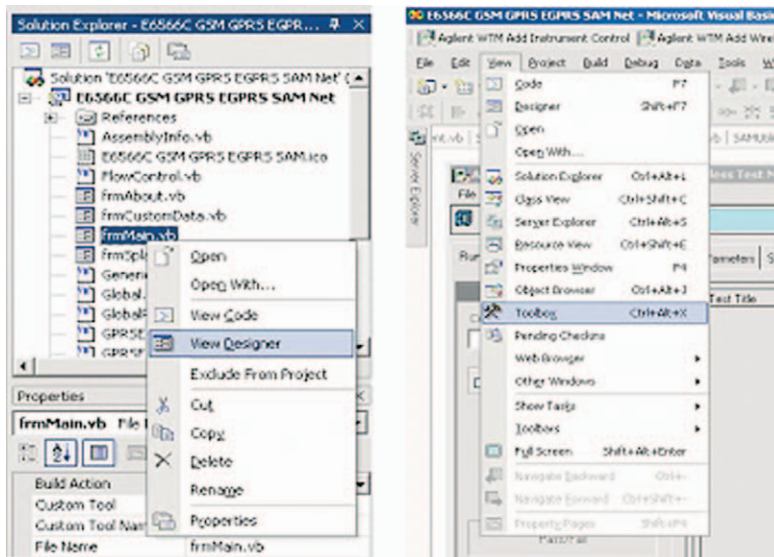
This positions the Toolbox on the left of the Visual Basic development environment. In order to hide the Toolbox when it is not needed, click the **Auto Hide** button in the upper right corner of the window so that it is pointing to the side instead of downwards.

8.3.2 Adding the GPIO component

A GPIO (General Purpose Input/Output) component is useful for most of the general instruments that connect to the system. This component allows an instrument that uses GPIB to integrate into the WTM on your computer system.

The frmMain design form must be the active window for this step. In the Toolbox on the left of the Visual Basic development environment, choose the **gpioControl** component. If the component is not included in the list, then it needs to be added. To do this, right click the **Toolbox**, and then select **Add/Remove Items** to open the **Customize Toolbox** dialog box. On the .NET Framework Components tab, select **gpioControl** and then click **OK**. The component now appears in the Toolbox on the General tab.
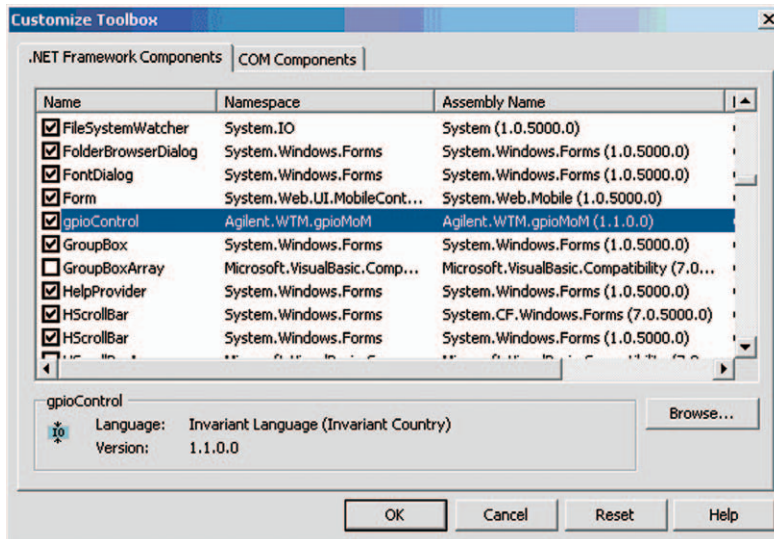
8.3.3 Adding an instrument using the wizard

After the gpioControl component has been added to the Toolbox, double-click its icon to add it to frmMain. You can also do this by dragging the icon onto the design window. The icon is now displayed in the lower part of the frmMain window. Click on the icon and set the properties as required (for example, change its name property to SigGen). The red outlines in the following screen shot indicate the areas of the screen being used in the above procedure. After setting the properties, save the project.
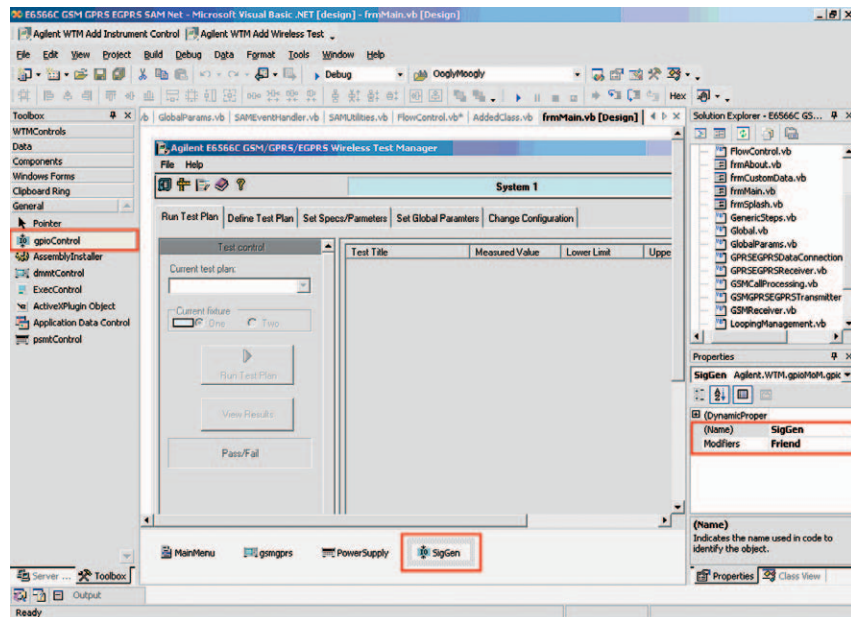
Next, open the Add Instrument Control Wizard by clicking on the **WTM Add Instrument Control** button in the upper left corner of the Visual Basic .NET development environment. Select the new instrument in the Available instrument components box and click **Next**.
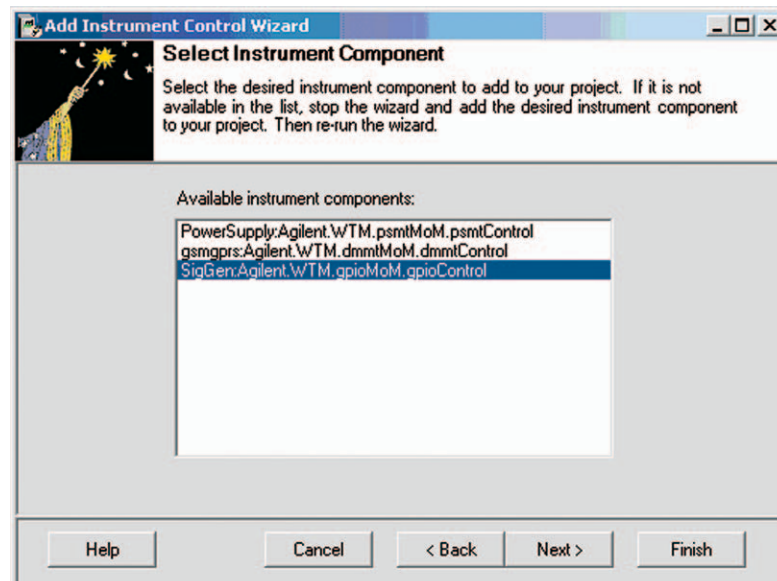
Type the user interface (UI) name in English and select the class modules that will use the instrument. You can choose the classes by selecting the class name in the Available test class module names box on the left, and then clicking the right arrow button in the center of the wizard page to copy the class module into the Current class module names box. Click **Next** to add the required code to the selected classes. Click **Finish** to exit the wizard.
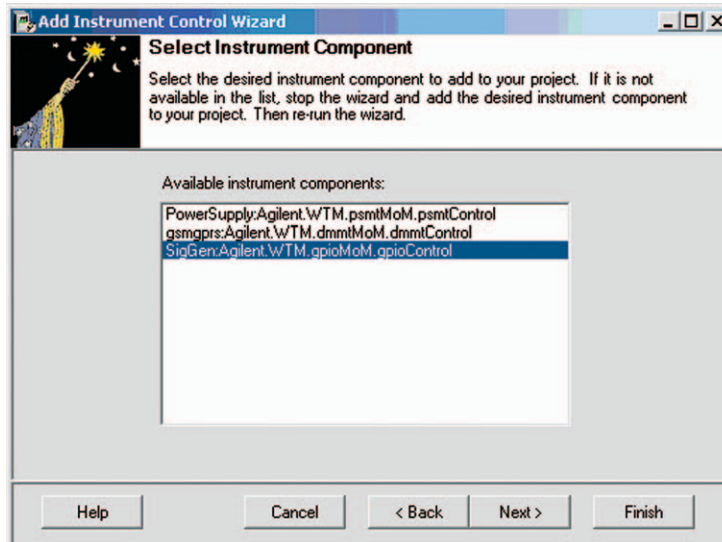
Make sure to go through the project and review all of the code that has been added. This code appears in frmMain, FlowControl and whichever class module(s) you selected in the wizard.

After the wizard has added an instrument, you must be very careful about any changes made to the properties regarding that instrument. For instance, if you were to change the name of the instrument, those changes would not be recognized in the code that had been previously written for the instrument.

8.3.4 Adding an instrument in the WTM interface
After adding the instrument to the program code, it is also necessary to add the instrument using the WTM UI.
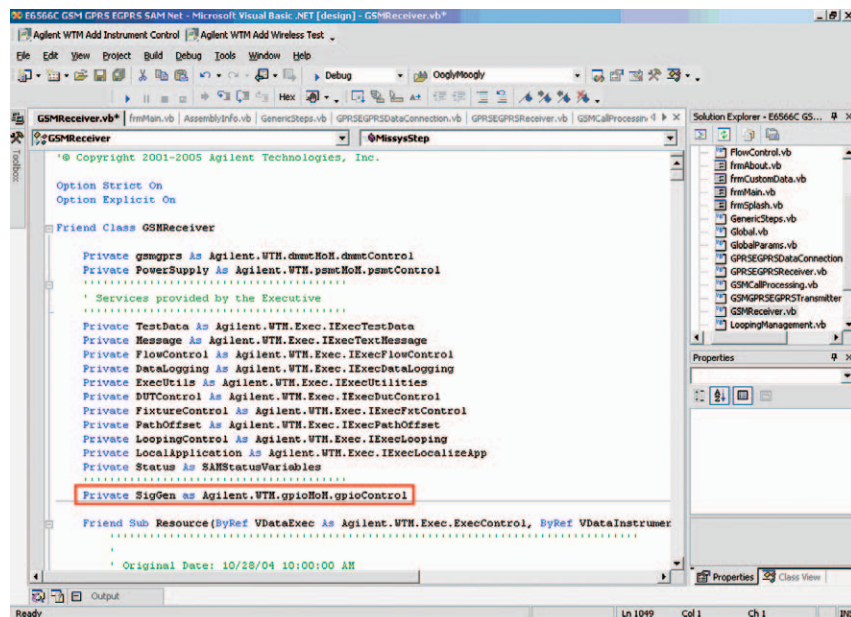
Go to the top of the Visual Basic window and click the **Run** button to open the WTM user interface. After the program loads, click the **Change Configuration** tab at the top of the window. In the Select a configuration type to change box on the left of the window, select the **Instrument Control** option. Choose the correct address for the instrument. If you do not know it, click ***IDN?** and the results of the address search are displayed in the lower part of the window.
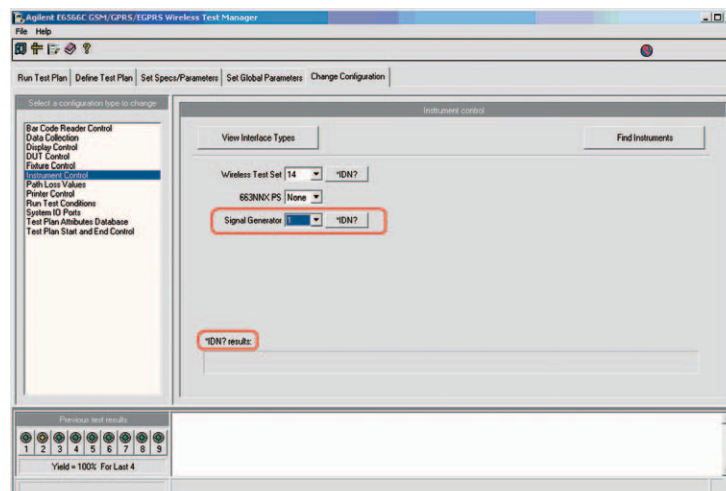
8.3.5 Coding for the new instrument

After the instrument has been included in the WTM UI, you can close the program. Return to the Visual Basic .NET test code.

At this point, go to the class module(s) in which the instrument has been added using the wizard. In the upper portion of the code, there is an added line that instantiates the test instrument.

If you scroll down and look at the 'Resource' step of the class module, there is another section where code has been added.

Now, within the class module, add code to handle the direct input and output to the instrument. All of the commands consist either of the form *instrumentname*.DirectIO. Output *"GPIB Command"* or *instrumentname*.DirectIO.Enter *"GPIB Command?"* depending on whether the command is a query or a setting.

It is important to be careful about where the new code is being added. As the comments say, do not modify any of the wizard-added commentary at the top of a test step. Below this are the specifications and parameters followed by the first section of code that may have a comment of this general form:

```
'Find the parameter values from the data store.'
```

Do not add any code to this section unless it is to reset the instrument to a known state at the beginning of a procedure. The best place to add instrument code is directly before the section that initiates measurements (as shown in the following screen shot). This code tells the program how it will be using the instrument and what information to gather.
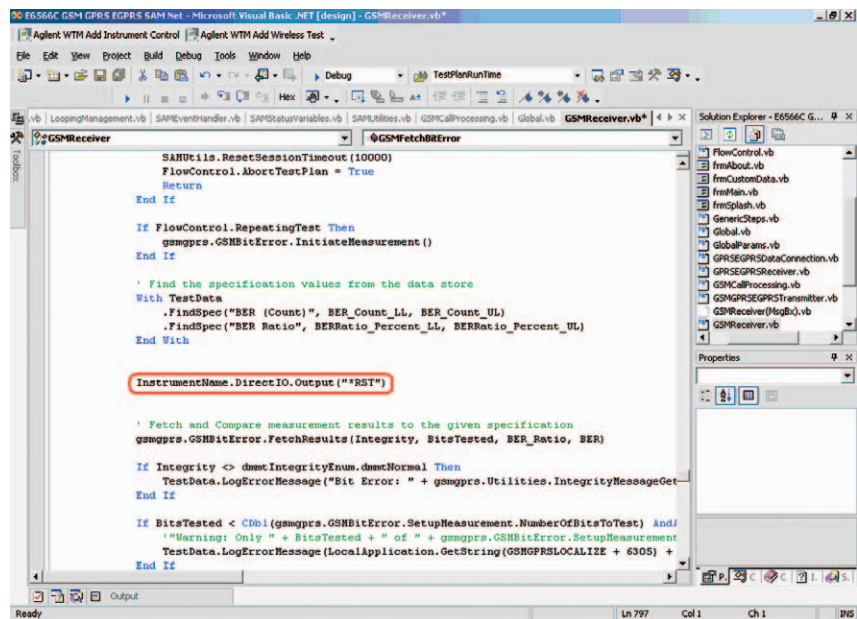
8.3.6 Turning off the instrument
At the end of the test, remember to turn off the radio frequency (RF) of the instrument so that the instrument does not cause any conflicts with other tests. The following line turns off the RF.

```
SigGen.DirectIO.Output("OUT:POW OFF")
```

Place this code both after and inside of the Exception block. Putting the code inside the Exception Block ensures that even if an error occurs, the instrument still turns off.

8.4 Removing an instrument
It is highly recommended to use the wizard to remove instruments. Since coding for instruments is spread through many class modules, the wizard is efficient in making sure every instance of the instrument code is processed for the deletion.

8.4.1 Removing an instrument using the wizard

When removing an instrument, the code is not deleted, it is only commented out. Also, the instrument icon remains at the bottom of the frmMain window even after deletion. If you want, the icon can be deleted by right clicking on it and choosing **Delete** from the pop-up menu.

To delete an instrument, click the **Agilent WTM Add Instrument Control** wizard in the upper left of the Visual Basic .NET development environment. Choose the **Remove Instrument** option, and then click **Next**. The next window shows a list of available instruments to remove. Choose the instrument you want and click **Next**. The wizard goes through all of the class modules and comments out any code relating to the instrument. The following window is shown when the process is complete.
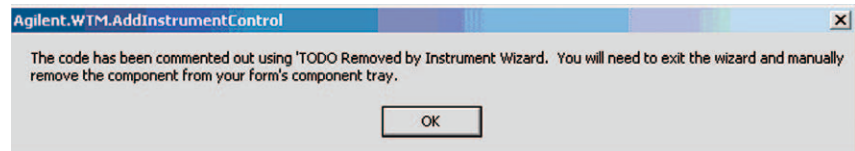
8.5 Summary

There are a number of different instruments that can be added to the WTM to help with testing. By using the gpioControl component and setting the properties as required, the wizard can be used to easily and efficiently add an instrument to the test project. Removing an instrument is just as easy with the help of the wizard. Remember to save a copy of the project code before you make any changes. Always look through the class modules to see where code has been added or removed.

Adding or deleting instruments includes the following procedures:
* Adding a gpioControl component to the Main form
* Running the wizard
* Writing customized code to use the instrument in the test project
* Saving the project or class modules being modified

9 Adding an IVI.COM driver to the Wireless Test Manager

9.1 Introduction

There are times when additional hardware is required to perform tests with the WTM. To provide efficiency in employing different instruments with the WTM, an IVI.COM driver is required. This section explains how to add such a driver. Note that Visual Basic .NET and the WTM Development and Run-time modes must be installed on your computer.

9.2 Overview

The interchangeable virtual instrument (IVI) driver is a very efficient technology designed for using hardware from different companies with relative ease. This means that you can write a program for one instrument, a signal generator for example, and then change it with a signal generator from another company, and the hardware will still work for general purpose tests without have to re-write the code.

When including an IVI.COM driver in the project, it is recommended that you add it manually. This is because the wizard does not reasonably reduce the work required to complete the task. Code needs to be added to the classes frmMain, FlowControl, and any other test classes where you want to use the IVI driver.

9.3 Adding an IVI.COM driver

Before adding an IVI.COM driver to the WTM project, you must first install the driver on your computer. Then, only a few steps are involved in adding the driver. The driver must be added to the references for the project and then code needs to be written in the appropriate classes.

9.3.1 Downloading an IVI.COM driver

An IVI.COM driver can be downloaded from Agilent's Web site at **www.agilent.com/find/adn**. Save the IVI driver you require to your computer and install it. After installation, the IVI driver is automatically available within the Visual Basic .NET project.

9.3.2 Adding the IVI.COM reference

This step can be performed in the Visual Basic .NET environment by using either the Project menu or the Solution Explorer window. The following procedure uses the Project menu method.

Click the Project menu located at the top of the Visual Basic .NET window. In the menu that is displayed, click the **Add Reference** option.
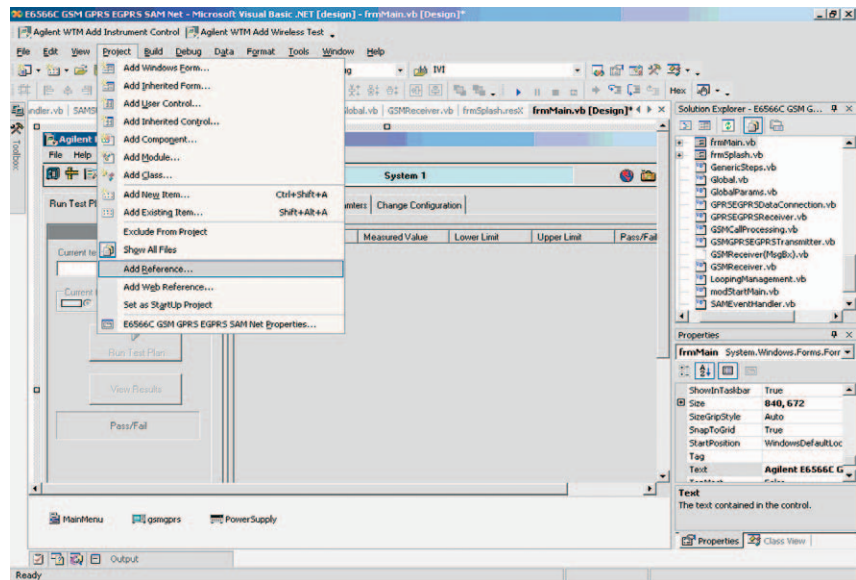
The Add Reference dialog box is displayed showing all the available references for the program (as shown in the following screen shot). Click the **COM** tab and scroll down through the choices to find **IVIAgilentRFSigGen**. Highlight this component and click **Select** on the right so that the component name appears in the box below. Then click **OK**.
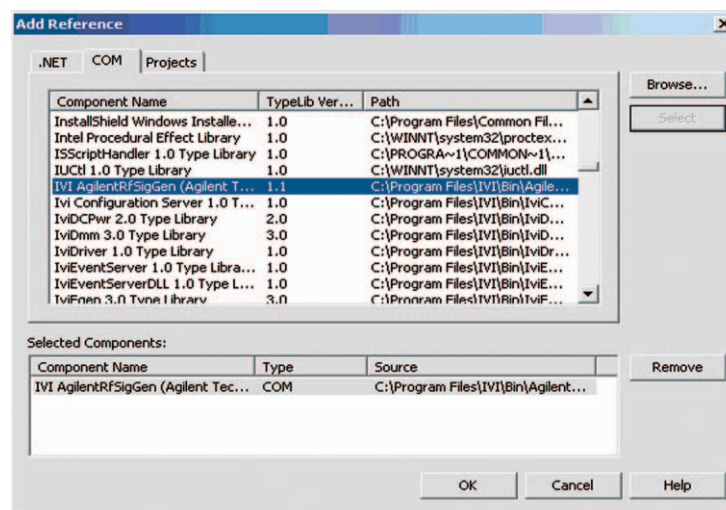
To check that the component has been added successfully, expand the References tree structure in the Solution Explorer window. This shows a list of all of the available references. AgilentRFSigGen is now a part of that list.

If you would like to see information about the added component, select the **View** menu, and then click **Object Browser**. Click **Ivi.RFSigGen.Interop** in the box on the left, and a list of members for this reference is displayed in the box on the right.

9.3.3 Adding code for IVI.COM driver

To correctly install the IVI driver, there are several places in which code needs to be added to the project. It is possible to add the instrument associated with the IVI driver by running the wizard, however, due to the amount of modification needed to correct the wizard code, it is not a recommended procedure.

9.3.3.1 Code in frmMain

The first class to add code to is frmMain. The lines of red code below are those that need to be added to the class.

```
In FormMain=========

Friend Class frmMain

'Create an instance of the driver
    Dim RFGenIVI As New
        Agilent.AgilentRfSigGen.Interop.AgilentRfSigGen

Private Sub frmMain_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Load

    Instruments.Add("RFGenIVI", RFGenIVI)   'RF Generator

Private Sub frmMain_Closing(ByVal sender As Object, ByVal
e As System.ComponentModel.CancelEventArgs) Handles MyBase.
Closing

    RFGenIVI.Close()
```

9.3.3.2 Setting up the UI instrument name

Please note that in the class FlowControl, the name of the instrument used in StartOfInitialTestPlan, "Agilent SigGen,", must match the name used in the properties for the exec WTMINstrumentNN. This line of code is shown as follows:

```
...Initialize(Cstr(InstResourceDesc("Agilent SigGen")),...
```

To set the name for the exec WTMINstrumentNN, go to the design view of frmMain. If the Properties window is not visible on the WTM project screen, it can be opened by pressing **F4** or by clicking the **View** menu and then **Properties window**.

In the Properties window, click the drop down box at the top of the window and choose the option 'Exec Agilent.WTM.Exec.ExecControl.'
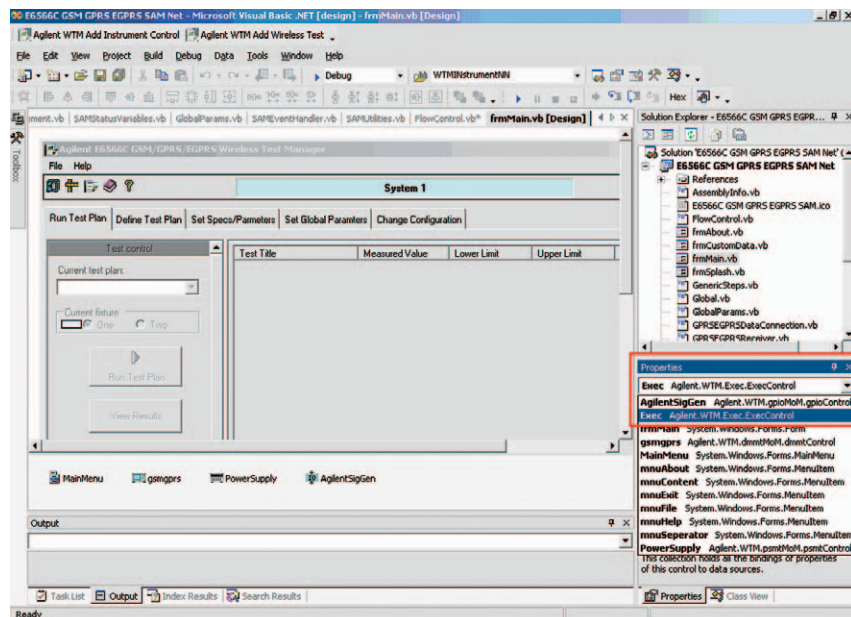
Scroll down through the Properties window to WTMInstrument03Name and type the name of the instrument.
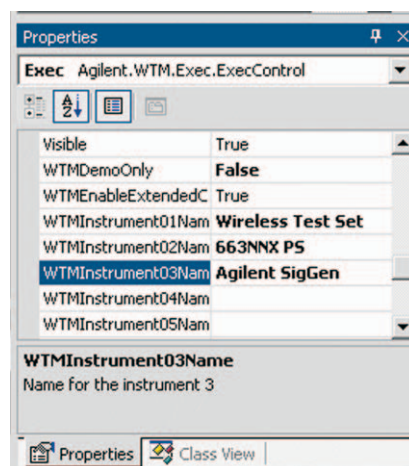
Be careful when changing the instrument name property. If you need to actually change the name in the properties window, be aware that the old name of the instrument will NOT be replaced in the wizard-generated code for the added instrument. So, if the name in the Properties window is changed, you must go back through the code and replace the old UI name for the instrument with the UI new name. This can be done by pressing **CTRL + F** to open a Find window. Type the old instrument name in the box and click **Find Next**. Replace each of these instances with the new instrument name.

9.3.3.3 Code in FlowControl

The following code is needed in the FlowControl class.

```
In Class FlowControl======

Friend Class FlowControl

   Private RFGenIVI As
       Agilent.AgilentRFSigGen.Interop.AgilentRfSigGen

Friend Sub Resource(ByRef VDataExec As
   Agilent.WTM.Exec.ExecControl, ByRef VDataInstruments As
   SortedList)

   RFGenIVI = DirectCast(VDataInstruments.Item("RFGenIVI"),
   Agilent.AgilentRfSigGen.Interop.AgilentRfSigGen)

Public Function StartOfInitialTestPlan(ByVal InstResourceDesc
   As SortedList) As Boolean

   RFGenIVI.Initialize(Cstr(InstResourceDesc("My Instrument
   Name")), False, False)

   If RFGenIVI.Initialized = False Then
       'TODO: add error code here
       Exit Function
   End If
```

9.3.3.4 Code in other classes

After completing the code for frmMain and FlowControl, you only need to add code to the classes in which you want to use the instrument corresponding to the IVI driver.

```
In Class My Test Class (as many as required)==========

Friend Class MyTestClass

   Private RFGenIVI As Agilent.AgilentRfSigGen.Interop.
       AgilentRfSigGen

Friend Sub Resource(ByRef VDataExec As
   Agilent.WTM.Exec.ExecControl, ByRef VDataInstruments As
   SortedList)

   RFGenIVI = DirectCast(VDataInstruments.Item("RFGenIVI"),
       Agilent.AgilentRfSigGen.Interop.AgilentRfSigGen)
```

After all of the above code has been written into the project, you can modify the classes to include any testing procedures that make use of the instrument.

Remember to turn off the instrument at the end of the test step. See Section 8.3.6.

9.4 Summary

The IVI.COM driver is a valuable tool that helps to make testing easier and more efficient when using extra hardware. The most efficient way to add the driver to the WTM program is to manually edit the code rather than use the wizard. As always, make sure to save copies of the classes that are being changed before you make modifications to the code.

Adding an IVI.COM driver includes the following procedures:
  • Adding IVIAgilentRFSigGen to the references
  • Adding code to the necessary classes
  • Writing customized code to utilize the instrument in the test project
  • Saving the project or class modules being modified

10 Adding and Deleting Class Modules in the Wireless Test Manager

10.1 Introduction
This section describes how to add or delete a class module in the WTM. Classes are used to create new objects and methods that help to develop new testing routines. To complete the following procedure, Visual Basic .NET and WTM Development and Run-time modes must be installed on your computer.

10.2 Overview
The WTM program comes with a number of standard classes required for the WTM to run tests properly. It is also possible to create your own classes for other tests not included in the project. Please note that you must not delete any of the standard Agilent classes from the project. This can prevent the program from operating correctly.

The Add Wireless Test Wizard can help you to add a class to the WTM project. When you use the wizard to add a class, you are also required to add a step during the creation process.

10.3 Adding a class module
This is a straightforward procedure that you can perform using the wizard. When beginning the wizard, choose the option to **Add, delete or copy a test step**. The next window allows you to choose a class where you want to add the test step. Instead of selecting from the list of available class modules, we will construct a new class altogether.

10.3.1 Creating a new class
Click on the **Agilent WTM Add Wireless Test** button in the upper left corner to open the wizard. Select the appropriate database (see Section 2.2) and click **Next**. Select the **Add, delete or copy a test step** option and then choose **Add a new step**.

The next page, Select Test Class, shows a list of available class modules to add the new test step. At this point, you have the option of creating an entirely new class module by clicking on **New....**
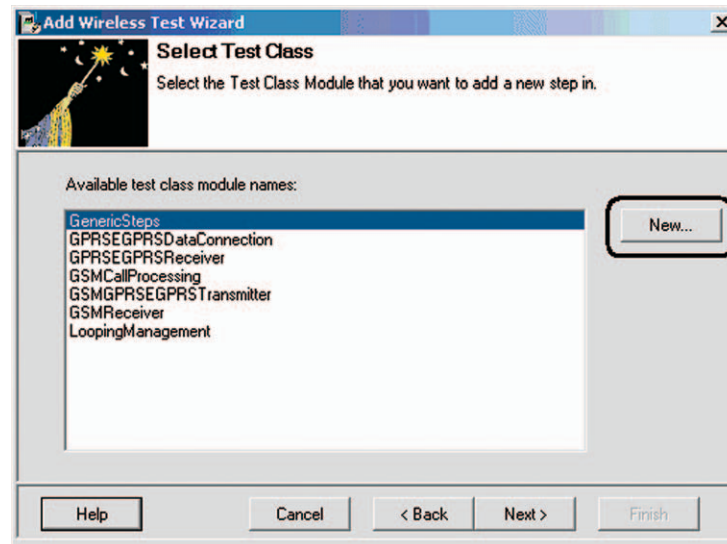
A dialog box opens prompting you to type the name of the new class. When choosing an appropriate name for the class, be sure that you do not name it after any of the Visual Basic .NET keywords such as New, Me, Public, Dim, or Try.
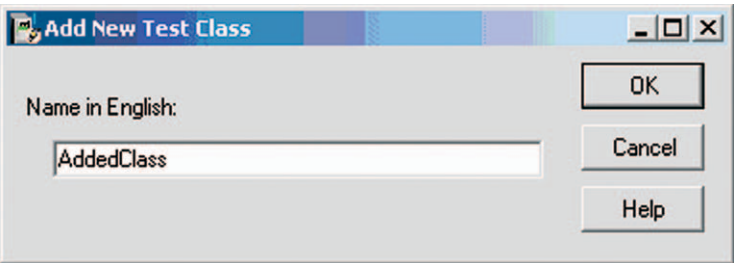
After typing a name and clicking **OK**, the new class is added to the list of available modules.
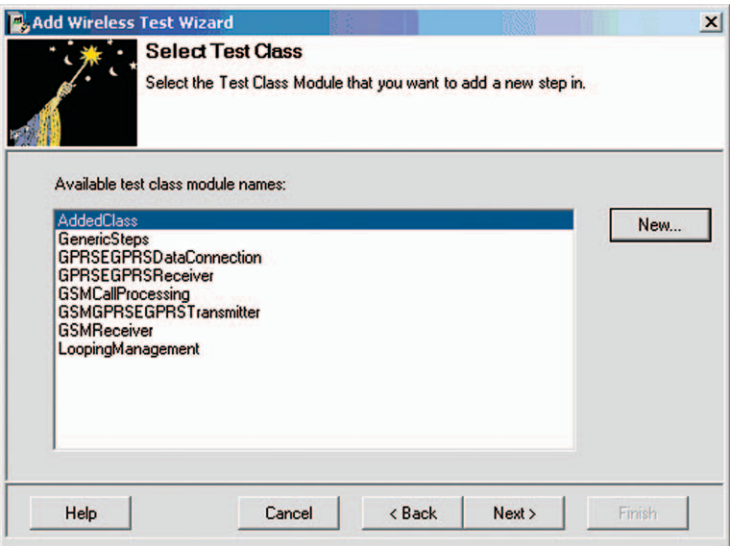
Highlight the new class module and click **Next**. The next page helps to create the test step that is to be placed into the new class module. Enter the details as required and click **Next** (if this step causes any problems, please refer to Section 7.3 of this application note).
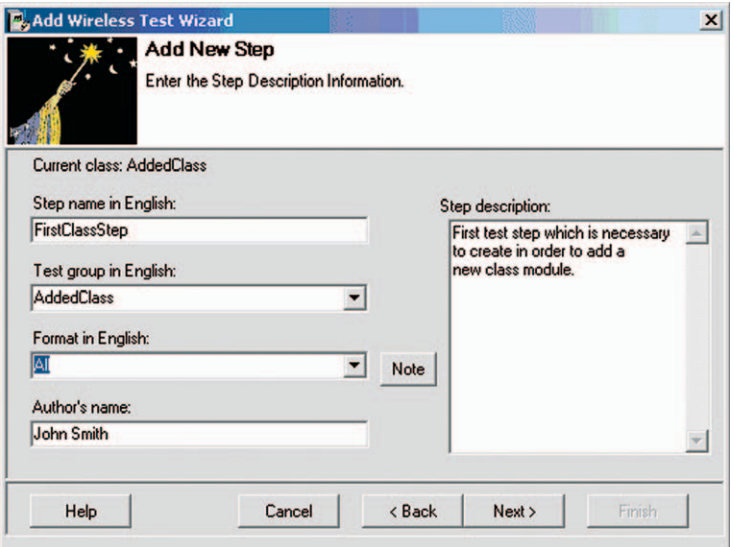
The wizard pages that follow allow you to add specifications and parameters to the new test step. For a better understanding of this process, refer to Section 3.3.1. After this process is completed, click **Next** and all of the necessary changes are made. Click **Finish** to complete the wizard procedures.

The new class can now be accessed in the Solution Explorer window. In this window, classes are listed in alphabetical order. If the Solution Explorer window is not already visible, click **View** in the **Project** menu and then choose **Solution Explorer**. Alternatively, you can use the keyboard shortcut **CTRL + ALT + L**.

When viewing the code inside the new class, you can see that a basic class code structure and general variables have already been added. These variable declarations make various Agilent tools and controls automatically available to you for coding your new class.

10.4 Removing a class module

This step can be applied when a class is no longer required for your testing needs. Please note that it is NOT recommended to remove any of the class modules developed by Agilent. Many of these classes are fundamental to the operation of the program. Removal of these classes may result in the program not functioning correctly, or cause test steps to operate erroneously.

10.4.1 Removing a class

For this part of the procedure, the removal of a class module is accomplished without using the wizard. Before proceeding with this section, create a back up of the project files.

Go to the Solution Explorer window on the right of the WTM project screen and select the class you want to delete. Right-click to open the **Context** menu and select the **Exclude from Project** option.
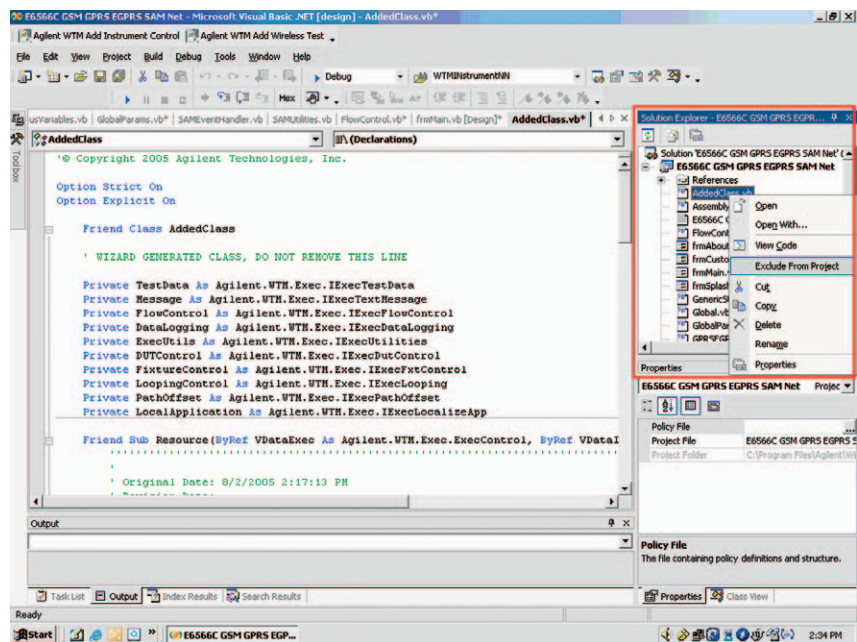
After you have excluded the class from the project, you need to right-click the class again and select **Delete**. Think carefully before performing this action, as it removes the class and its code permanently from the project as well as the folders associated with it.

Next, go to the code for FrmMain and read through it carefully in order to find the lines indicated by arrows in the sample below and comment them out.

```
Private Sub frmMain_Load(ByVal sender As
Object, ByVal e As System.EventArgs) Handles MyBase.Load

    ⟶ Dim AddedClass as New AddedClass
        .
        .
        .
    ⟶ AddedClass.Resource (Exec, Instruments)
        .
        .
        .
    ⟶ TestClasses.Add ("AddedClass", AddedClass)
```

These statements can all be found in Sub frmMain_Load. It is recommended that you comment out these lines rather than delete them, just in case you are accidentally deleting code that may potentially be needed again. One way to make sure that all instances of 'AddedClass' have been commented out is to go to the **Edit** menu, select **Find and Replace** and then **Find**. By searching through each area where this word appears, the code can be commented out so that it will not affect the project.
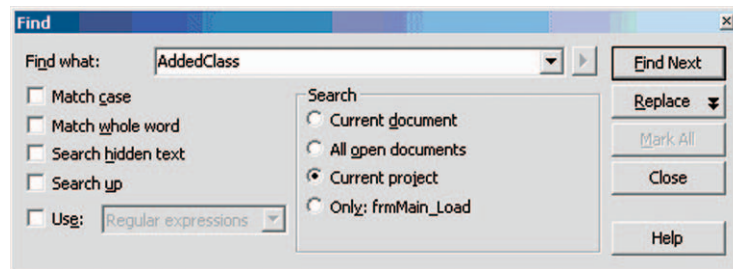
As always, make sure to save the project before making any changes to it. Before deleting a class module it is recommended that you save a copy of the class in another folder which would not interfere with the WTM project. Creating a backup makes it possible to retrieve the class again if it is ever needed.

10.5 Summary
There are a number of different reasons why you may want to add a new class module. For example, to self-contain your custom test plan steps from the standard WTM test plan steps. In some instances, you may want to remove a class from the WTM project. However, you must exercise caution when performing this task to ensure that you do not delete anything which you may need at a later date.

Adding a class module includes the following procedures:
  • Running the wizard
  • Writing customized code to utilize the instrument in the test project
  • Saving the project or class modules being modified

11 Adding LAN Control to the Wireless Test Manager

11.1 Introduction

The WTM local area network (LAN) control allows you to control the WTM remotely over a LAN connection. This procedure requires Visual Basic .NET and the WTM Development and Run-time modes be installed on your computer.

11.2 Overview

LAN control is a powerful feature of the WTM. If a user is using one computer, they can control a WTM program on an entirely different computer using the LAN control feature. An example of how to implement the LAN connection is to send a message from one computer to another that would use the testing power of the WTM.

The WTM LAN interface example can be found on the installation CD for the WTM. There is a folder on the CD titled 'WTM LAN Interface Examples' and within that folder is a program named 'WTM LAN Interface Example.sln.' This program sets up the LAN connection.

The following diagram shows how the remote computer connects to the WTM server computer (the computer running the original WTM program) and then how the WTM server is connected to the 8960 test set through a GPIB connection.
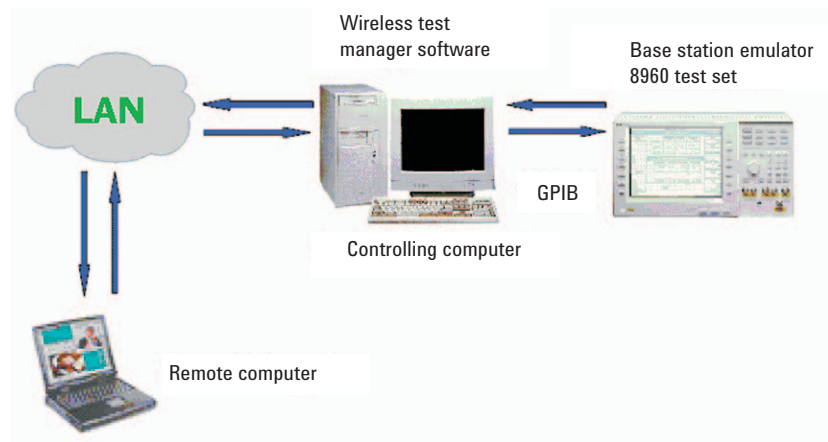
11.3 Setting up the LAN connection

Before proceeding with the following steps, there must be a working LAN connection or network available. When using the LAN control, the two computers must be on the same network and behind the company firewall.

11.3.1 Configuring the WTM to accept a LAN control

Run the WTM program and click on the **Change Configuration** tab. In the Select a configuration type to change box, click **System IO Ports**. The System IO ports area is displayed on the right (as shown in the following screen shot).
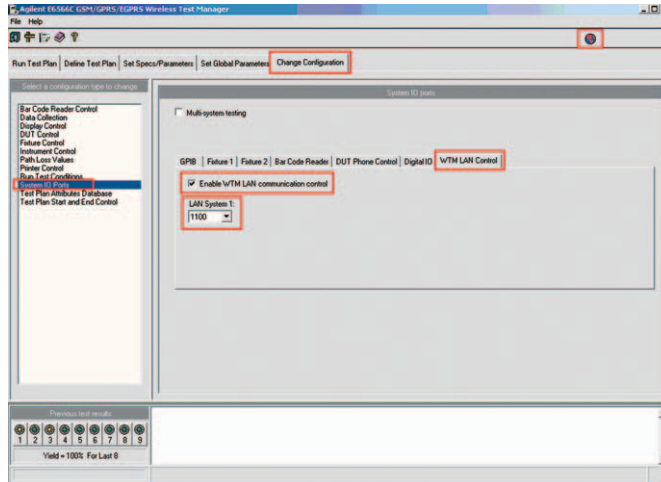
In this area, click the **WTM LAN Control** tab. Select the **Enable WTM LAN communication control** check box. When selected, a small globe appears in the upper right area of the WTM program. When the globe is red, it means that LAN control is not active. When the globe is green, it means that a LAN control program is connected to the WTM and is actively communicating.

Use the LAN System 1 list box to select the socket number of the TCP port for the computer. This completes the set up for LAN control in the WTM program. Click the **Run Test Plan** tab to return to testing.

11.3.2 Starting the LAN control program

The LAN control program is provided on the installation CD for the WTM. To find this file, open My Computer and navigate to your computer's CD drive. Right-click the CD drive's icon and click either **Open** or **Explore**. Copy the WTM LAN Interface Examples folder to your computer's hard disk drive.
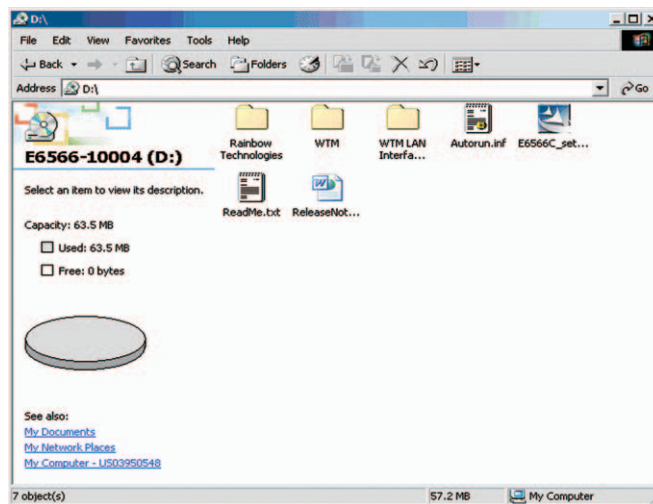
Using the copy of the folder you have just saved to your computer's hard disk drive; navigate to the folder "C:\ ...\WTM LAN Interface Examples\WTMLanClientExample\". Double-click the "**WTMLanClientExample.sln**" file to open the WTM LAN project in Visual Basic .NET.
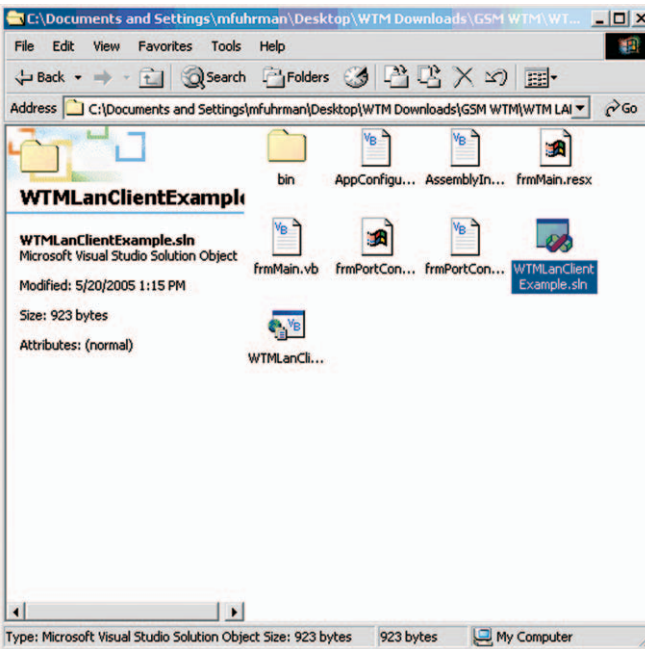
After the project has opened, click the **Run** button at the top of the Visual Basic .NET window. The WTM LAN client example program displays the Port Configuration dialog box (as shown in the following screen shot).
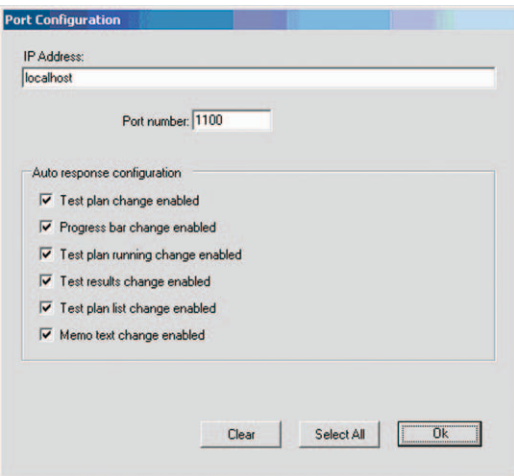
By default, all of the check boxes under Auto response configuration are selected automatically. You can clear any of these check boxes as required. Make sure that you type the appropriate port number and IP address in order to connect with the correct WTM server, and then click **OK**.

After clicking the **OK** button, the WTM LAN Client Demo window opens. The globe in the upper right corner of the original WTM program (server) turns green to indicate that it is connected to a WTM LAN interface.

Through the WTM LAN program, it is possible to control other WTM programs on the same network. The LAN control can only have power over WTM programs that are already running on another computer. The WTM LAN control interface is shown in the following screen shot.
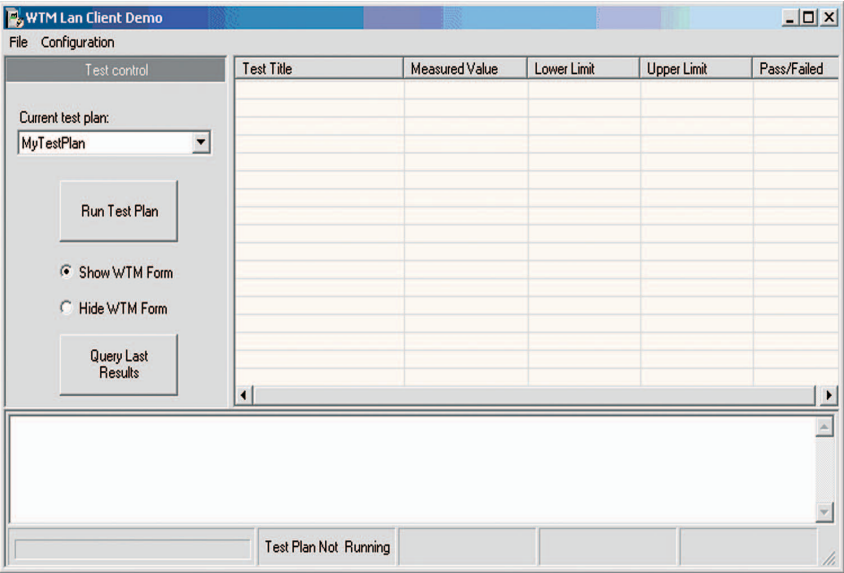
If the WTM LAN program does not successfully connect to the WTM server program with the IP address and port number specified, then the program closes and the globe on the WTM server program remains red. You must run the program again and attempt a connection with another IP address or port setting. (Note: the port number must be set to the same value on the client and server.) You may need to talk to your information technology (IT) department to determine a port value that is available, since some firewall setups on individual computers may affect available port numbers.

11.4 Running the program
The WTM LAN interface can only control other WTM programs that are already open. From the LAN control interface it is possible to control virtually anything in the WTM server program. The following section describes this process.

11.4.1 Running tests with WTM LAN control
After the WTM LAN interface window has opened on the screen, it is a simple process to run a test on another computer within the network. In the Current test plan drop-down list on the left of the LAN interface window, there is a list of test plans. These test plans on the LAN interface match the available test plans on the WTM server being controlled. Choose a test plan and click **Run Test Plan**. This starts running the test plan on the computer connected to the 8960. As the test runs, the results appear on the WTM LAN Interface screen.

If you click the **Stop Test Plan** button in the LAN Interface window, the test does not end immediately. The program continues to send a call to the mobile before it aborts the test plan. The sequence of events before the test finally stops is shown in the following screen shot.
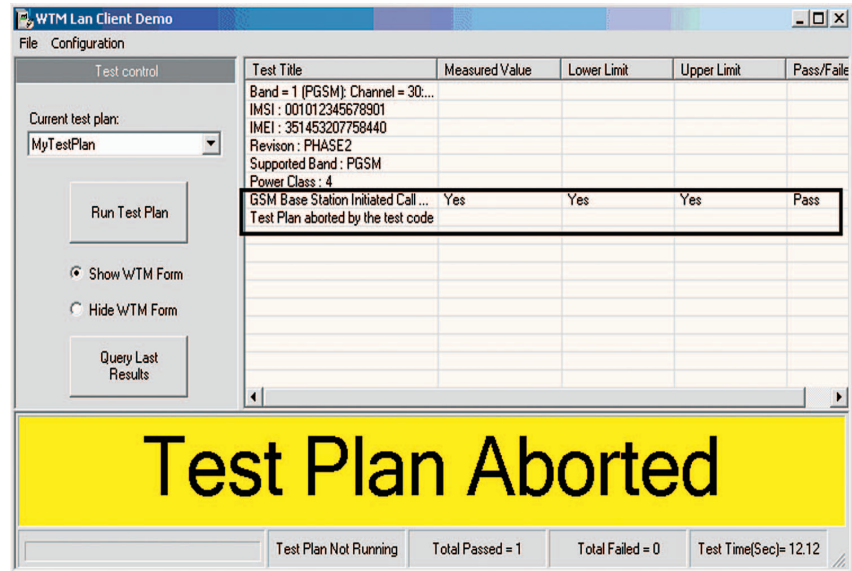
If the **Stop Test Plan** button is clicked in the middle of a test, then the test ends as soon as the test step it was executing is finished.
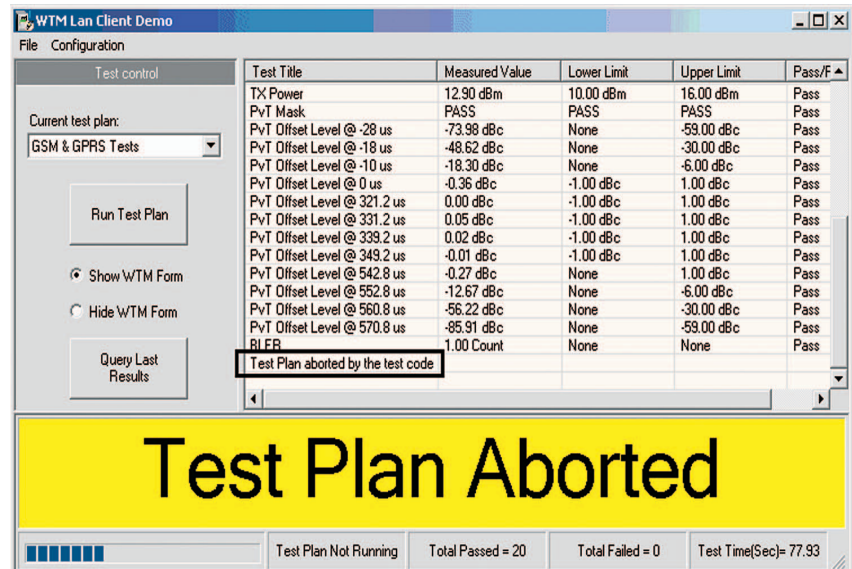
If the original WTM server program runs a test, the LAN Interface automatically shows the same results and output messages. In this way, the LAN interface monitors exactly what is happening on another remote WTM program.

11.5 Coding example

Through the WTM LAN Interface, it is possible to control almost anything in the original WTM program. By using the LAN control project in Visual Basic .NET, code can be added to change settings for the 8960, send messages, and perform other functions. These properties make the WTM LAN control a very powerful tool. The following section provides an example of how to control the WTM through the LAN interface.

11.5.1 Sending a message to the WTM server

In this example, the LAN control sends a message to the WTM server. The message is sent by clicking a button that has been added to the LAN control interface. The following screen shot shows the modified WTM LAN interface with the additional button.
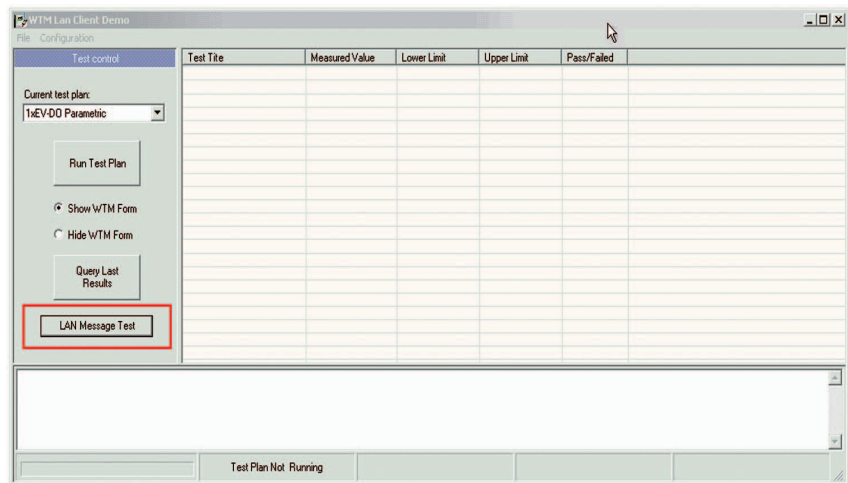
The name of this new button is **Button1** and the code it uses for sending messages is under a subroutine labeled Button1_Click. The code added to this subroutine creates an IO.StreamWriter() object, which has the ability to send strings. The messages that the LAN control is sending to the WTM server are 'XYZ.DoSomething' and 'XYZ.DoSomethingElse.' The user is allowed to use any format necessary except for a string beginning with 'WTM.xxxx.'
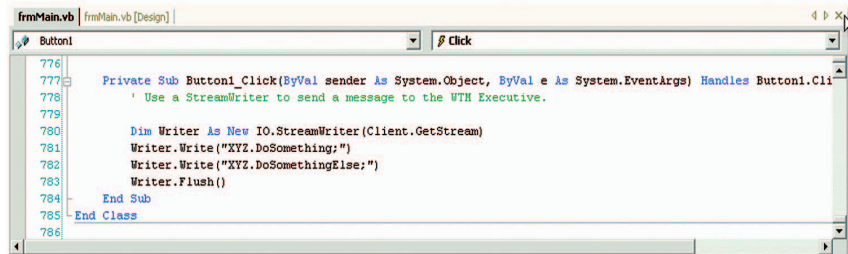
In the Exec Control there is an event that lets the user see what is being sent from the LAN control interface to the WTM server. This event is named Exec_LanMessageReceived() and it allows the user to see traffic consisting of strings. In this example, the strings being sent are 'XYZ.DoSomething' and 'XYZ.DoSomethingElse.'

The following screen shot is an example of how to write code that controls the contents of a message that appears in the memo area of the WTM server. At line 289 of the code is a statement that separates string messages into array items depending on if the program detects a ';' symbol in the string or not. Next, there is a block of code containing a case statement. It has been written so that if the program receives the message 'XYZ.Do Something' it prints 'Do Something' in the memo box of the WTM server and if it receives 'XYZ.DoSomethingElse' then it prints 'Do Something Else' in the memo box as well.

Since a custom formatting is being used instead of the format, WTM.xxxx, the Cancel-Action parameter must be set to **True** in order to stop the Exec Control from trying to process the message. If the Exec Control did try to process the custom message, an error would appear since the message is not of the form WTM.xxxx.

Now that the code has all been written, the **LAN Message Test** button on the WTM LAN Control Interface can be pressed. If everything is working correctly, the messages 'Do Something' and 'Do Something Else' appear in the memo box on the WTM Server Interface.
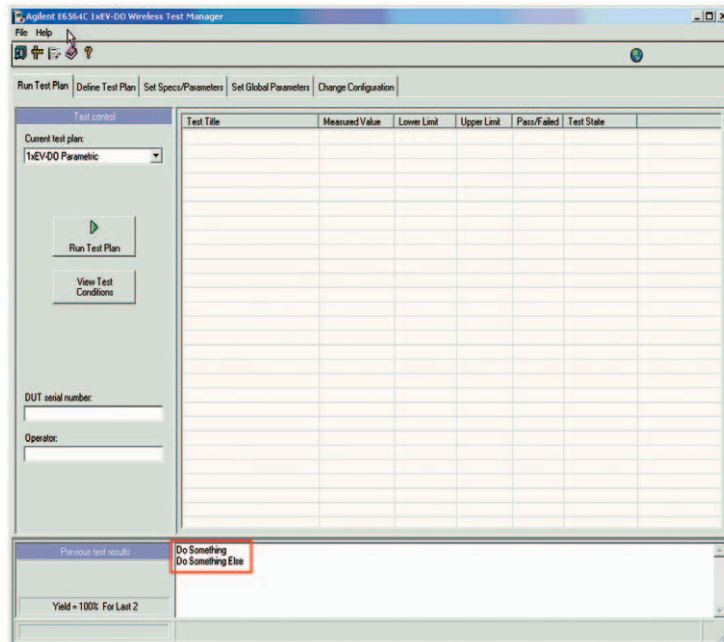
As a final note, any of the public interfaces can be accessed when using the Exec Control and if information needs to be passed to other classes within the project, this can be achieved by setting up a variable in the Global.vb class.

11.6 Other LAN examples

In the "C:\ ...\WTM LAN Interface Examples\" folder, there is another folder named "WTM LAN Client Monitor Example\". Within this folder, there is another Visual Basic project "WTMLANClientMonitorExample.sln". This is another LAN program that can remotely connect to a separate computer to monitor the progress of tests being run by the WTM program.

The monitor example is not quite as functional as the WTM LAN client because it simply lets you watch what is happening on the controlling computer. One cannot start or stop test(s) remotely through the monitor example or control any other aspect of the testing. This program is simply used for monitoring test progress.

11.7 Summary

The WTM LAN control interface can be a very effective tool. It gives you the ability to remotely control a WTM program on a completely different computer. The LAN control can monitor the test progress, change settings, send messages, and other WTM functions.

Adding LAN control includes the following procedures:
 • Changing settings in the WTM server
 • Running the WTM LAN interface example
 • Writing customized code to control the WTM server through the WTM LAN control

12 Sending and Receiving Serial DUT Commands

12.1 Introduction

An additional test step, "Send Receive DUT Command" is now included in current versions of the WTM products to allow direct reading and writing with a DUT, for example using GPIB or serial commands. In addition, a number of existing test plan steps have been modified to add parameters that support direct DUT communication directly within these steps. This procedure uses the C.02.00 version of the E6568C WTM to illustrate using serial commands with a DUT. This procedure requires Visual Basic .NET and the WTM Development and Run-time modes be installed on your computer.

12.2 Overview

It is recommended that you refer to the section "Send Receive DUT Command Test Step and Additional DUT Test Parameters Description" in the appendix of this application note. This section provides details of the test plan step, and additional parameters that have been added for direct DUT communication.

12.3 Setup the COM port

Ensure the device drivers are installed and that the device is connected to the computer on a COM port, for example USB or serial port. Before setting up the COM port within WTM it is recommended that you check the connection in Windows®. To do this, open **Network Connections**.

Right-click the **Dial-up device** and select **Properties**. Note the COM port as shown in the following screen shot.
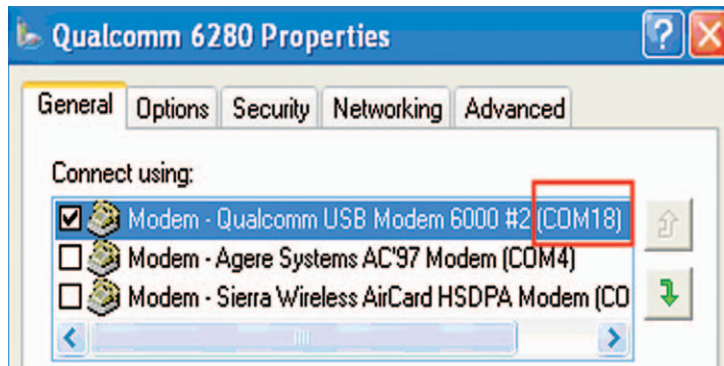
In WTM, click the **Change Configuration** tab, and then select **System IO Ports** from the Select a configuration type to change box. Click the **DUT Phone Control** tab and select the appropriate COM port in DUT System 1, as shown in the following screen shot.
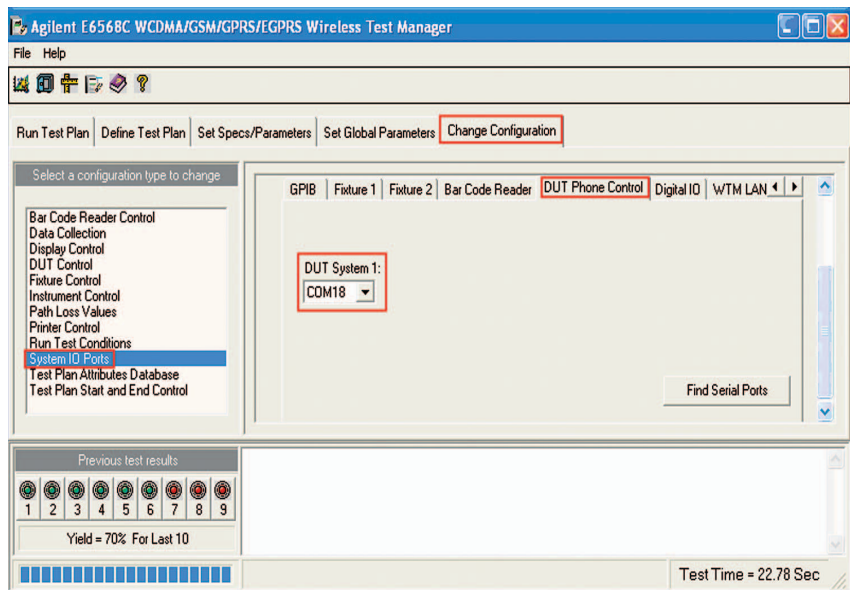
12.4 Perform a quick COM port test

On the Change Configuration tab, select **DUT Control** from the Select a configuration type to change box. Check that the baud rate and other parameters are set correctly. Change the Transmit termination to **LFCR** (Line Feed Carriage Return). Query the device with a simple ATI command, I = identity. Type **ATI**, click **Send**, and then click **Receive** to read the return value (see the following screen shot). This simple test will prove basic communication has been established.
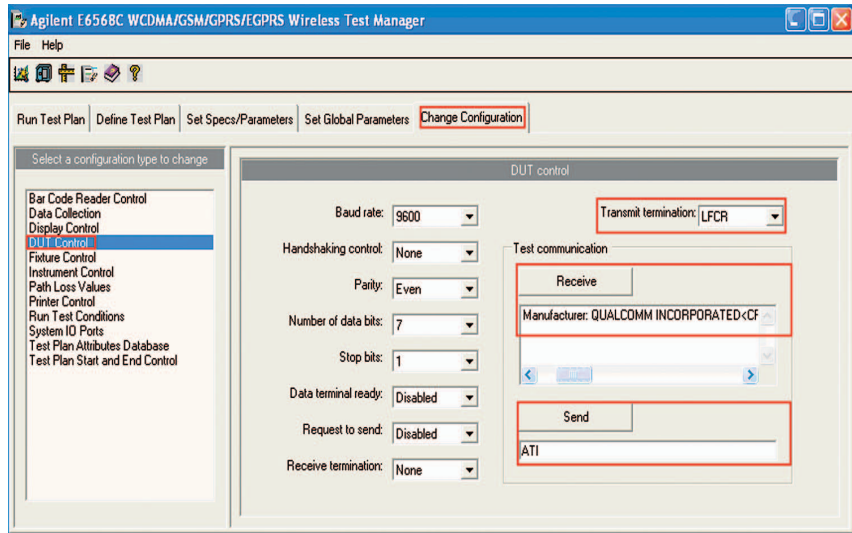
12.5 Configuring the test step

Create a new test plan. Ensure you select the format or multi-format. For example, select the **Define Test Plan** tab, and click **Add New Plan**. Decide if the plan is single or multi-format, this is dependant on the LA being used. In this example, name the test step "Test Device".
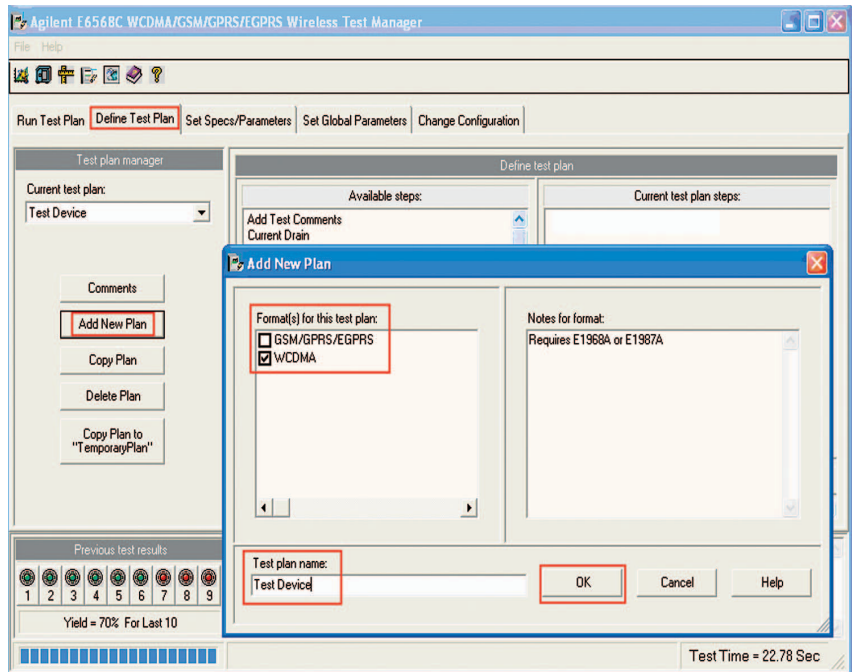
With the blank new test step in place, use the Available steps filter to select **Generic Steps**, and then drag the **Send Receive DUT Command** test step across to the Current test plan steps box.
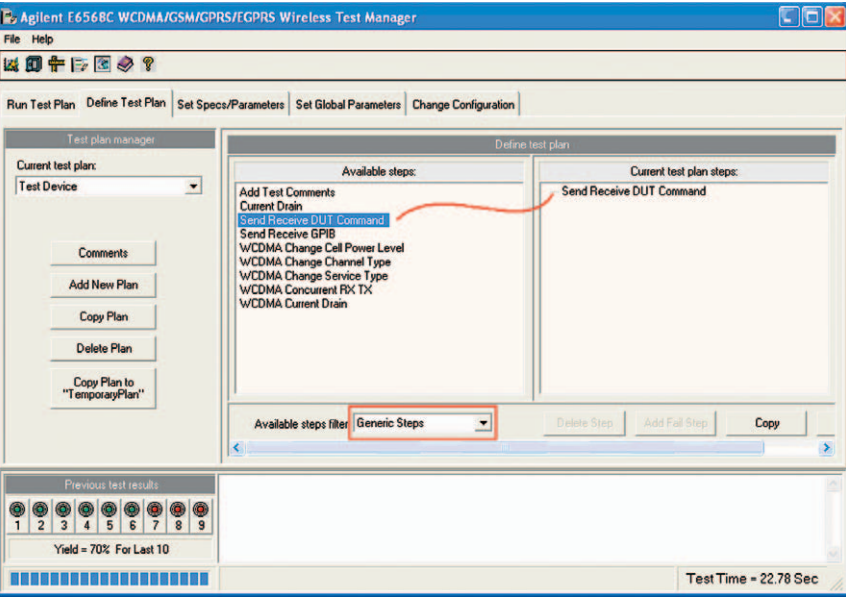
Next, you need to configure the test step to send and receive an AT command with the DUT, for example ATI. Select the **Set Specs/Parameters** tab, and then click the **>>** button adjacent to the Send/Receive DUT Commands with Specs table.
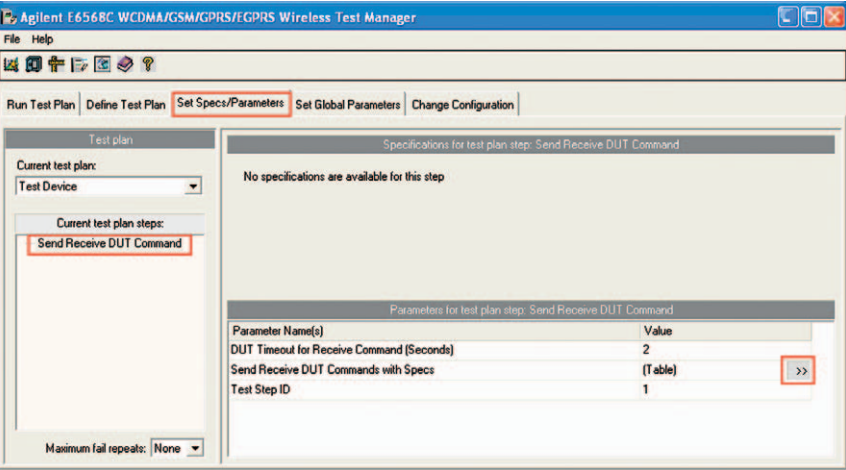
An example of sending and receiving is provided in the following screen shot. Note that the Test Step ID in the table corresponds to the higher level Test Step ID parameter. If 1 is selected, then only commands with Test Step ID 1 will be executed. This allows you to have multiple groups within one test step and to debug by removing an individual command line. What would happen if you gave Wait mSec an ID of 2 and pressed run? Answer: It would ignore this step as the higher level is set to run only Test Step ID = 1.

Parameter Table for: Send Receive DUT Commands with Specs

If the Action is to 'Receive from DUT', then the 'Command or Information' column should contain the exact expected data. If 'Display and Compare' is selected, the compare format of LL, UL [,Location] is used for numerical data were Location identifies the data location when multiple items are returned.

Test Step ID	Action	Test Title	Command or Information	Data Action	Compare Data
1	Send to DUT	Send Command	ATI	Display	LL, UL [, Location]
1	Wait mSec	Wait	200	Display	LL, UL [, Location]
1	Receive from DUT	Receive Command	Add command or information	Display and Compare	QUALCOMM

Add New Row   Delete Row   Reload Defaults   Save as Defaults   OK   Cancel

Run the test step as normal and check that it passes.

Now build a more complex group of commands within the same test step. These should have the ID 2. The group should setup an AMR BS originated voice call, have the DUT answer and end the call.

Run the test to ensure it works. The appendix shows a solution. The appendix also provides information and references to other test steps which have an AT structure implemented within them. For example, steps that perform call setup.

Appendix

Introduction

This appendix provides some additional useful information about the WTM
- Overview of the Exec and MoM programming interfaces
- Description of the WTM online help facilities
- Procedure for resetting the Administrator mode password
- Modifying the MoM XML file to add a TA or LA reference
- List of WTM LAN interface commands
- Send Receive DUT Command Test Step and additional DUT test parameters descriptions

Overview of the Exec and MoM programming interfaces

This section provides an overview of the Exec and MoM programming interfaces. These interfaces provide a variety of methods, properties, and event handlers that can be used in the Visual Basic .NET WTM development environment. The WTM interfaces you have available in the development environment is dependant on the WTM products you have purchased. The following list summarizes each of these APIs:

- **Agilent.WTM.Exec:** This API is provided with all WTM products and contains non-application specific methods, properties, and event handlers that manipulate the WTM framework or test executive. For example, there are interfaces to control data logging, DUT control, flow control, and fixture control. Event handlers are provided to act on fixture events, LAN messages, WTM GUI button presses, as well as many other non-application specific events.
- **Agilent.WTM.cmltMoM:** This API is provided with the Agilent N5880A cdma2000/IS-95/AMPS Enhanced Wireless Test Manager.
- **Agilent.WTM.wmltMoM:** This API is provided with the Agilent N5882A W-CDMA Enhanced Wireless Test Manager.
- **Agilent.WTM.xmltMoM:** This API is provided with the Agilent N5884A 1xEV-DO Enhanced Wireless Test Manager.
- **Agilent.WTM.cmmtMoM:** This API is provided with the Agilent E6560C cdma2000/IS-95/AMPS Wireless Test Manager.
- **Agilent.WTM.wmmtMoM:** This API is provided with the Agilent E6562C WCDMA Wireless Test Manager.
- **Agilent.WTM.xmmtMoM:** This API is provided with the Agilent E6564C 1xEV-DO Wireless Test Manager.
- **Agilent.WTM.dmmtMoM:** This API is provided with the Agilent E6566C GSM/GPRS/EGPRS Wireless Test Manager.
- **Agilent.WTM.cmctaMoM and Agilent.WTM.xmctaMoM:** These APIs are provided with the Agilent E6567C cdma2000/IS-95/AMPS/1xEV-DO Wireless Test Manager.
- **Agilent.WTM.dmctaMoM and Agilent.WTM.wmctaMoM:** These APIs are provided with the Agilent E6568C W-CDMA/GSM/GPRS/EGPRS Wireless Test Manager.
- **Agilent.WTM.ag34970MoM:** This API is provided with all WTM products and contains methods and properties for controlling the Agilent 34970A Data Acquisition/Switch Unit.
- **Agilent.WTM.gpioMoM:** This API is provided with all WTM products and contains generic methods and properties for control of other measurement-related devices.
- **Agilent.WTM.psmtMoM:** This API is provided with all WTM products and contains methods and properties for controlling power supplies.

Description of the WTM online help facilities

The WTM products also include extensive help facilities. Here is a summary of what is available:

- **Graphical user interface (GUI) help:** Also, know as software application for manufacturing (SAM) or software application for lab (SAL) help. There is one of these help files for each WTM product (for example, E6568C SAM Help). This help contains information about the WTM framework (test executive) and information on the application-specific test steps, parameters, and specifications. You can access this help from within the WTM GUI using the Help icon in the toolbar or by right-clicking anywhere on the user interface and selecting **View Help** from the popup menu. If you highlight one of the test steps, right-click, and select to view the Help, the Help opens at the page specific to the test step you highlighted.

- **Developer help:** There are numerous help files that document the WTM Visual Basic .NET development environment:
  - **MoM and Executive API help files:** The developer help can be viewed for each of the WTMs APIs directly in the Visual Basic .NET development environment. Either view the interfaces, methods, and properties in the Visual Basic .NET object browser where the help is displayed in the lower pane for the currently highlighted part of the API, or alternatively, in the code window press **F1** when you have the cursor placed within the part of the WTM API you want help on. This "dynamic help" works in the same way as the standard Visual Basic .NET API.
  - **Wizard help:** The two WTM development wizards, Add Wireless Test Wizard and Add Instrument Control Wizard, have their own help files that you can access by clicking the **Help** button within the wizard screens.

All of the WTM help files can also be accessed from the Windows Start menu on the taskbar: **Start > All Programs > Agilent Wireless Test Manager .NET > Help**. Portable (.chm) versions of the API help are also contained in here.

Procedure for resetting the administrator mode password

When you install a WTM product and run the WTM GUI (run-time) for the first time, you are asked to type and confirm a password. This password is used to access the Administrator mode of operation which provides additional configuration features (for example, selecting which parts of the user interface are available in Operator mode). In some instances, you may want to reset this password (for example, if you have forgotten the password you originally set). You can use the following procedure to do this:

- Using My Computer or Windows Explorer, navigate to the appropriate TestData folder for the installed WTM that you want to reset the password for. For example, if you want to reset the password for the E6566C GSM/GPRS/EGPRS Wireless Test Manager, and the product has been installed in the default directory location, you would navigate to the folder:
  C:\Program Files\Agilent\WirelessTestManager\E6560C\TestData\
- Open the **ExecSettings.xml** file in this folder with a text editor (for example, Notepad).
- Use the text editor's **Find** tool to locate the text "password".
- Change **"<Password>}geivd</Password> to <Password></Password>** then save and close the file.
- The next time you start the WTM GUI, you will be asked to enter and confirm a pass word for access to Administrator mode.

Modifying the MoM XML file to add a TA or LA reference

The following examples using the E6568C WTM illustrates how to modify a MoM XML file to add a reference to an 8960 TA or LA.

1. Using My Computer or Windows Explorer, navigate to the location C:
   \Program Files\Agilent\WirelessTestManagerNet\E6568C\TestData\Agilent.WTM.
   dmctaMoMConfig.xml.
2. Right-click on the file and select Properties.
3. Clear (uncheck) the file's read-only attribute, and then click OK.
4. Right-click the Agilent.WTM.dmmtMoMConfig.xml file again and select Edit to open the file in a text editor such as Notepad.
5. Within all **"<format format_name="XXXX">"** sections in the file, copy and paste the following code:

```
<ta_info>
<ta_name>GSM/GPRS Lab App E</ta_name>
<ta_revision>E.00.12</ta_revision>
<supported>yes</supported>
</ta_info>
```

6. Save and close the file.
7. Reapply the file's read-only attribute. To do this, right-click the file, select **Properties** and select **Read-only**.
8. Repeat steps 2 to 7 above for the Agilent.WTM.wmctaMoMConfig.xml file in the same location.

Please note that single format WTM products like the E6564C or E6566C only require the TA or LA reference to be added into one XML file (for example, Agilent.WTM.dmmtMoMConfig.xml for the E6566C).

List of WTM LAN interface commands

The WTMNet LAN interface can be used to control the operation of the WTM or obtain testing information from the WTM. The WTM must be on the Run Test Plan tab before LAN control can occur.

LAN functionality is turned on via the Change Configuration tab and selecting **LAN** on the System IO Tab. When LAN is enabled a globe icon will appear in the top right. If a client is not attached to the WTM the globe icon will be red. If a client is attached the globe icon will be green.

When in the Developer mode, the following are the interface commands to the WTM LAN connection:

- **WTM.TestData.SetCurrentTestPlan** *Desired test plan name*
  Sets the WTM current test plan to the desired test plan name.

  If the desired test plan name is not found, an error is returned via a LAN message. The test plan cannot be changed if a test plan is currently running.

- **WTM.TestData.QueryAllTestResults**
  Returns all the test information currently in the WTM tests results information list view control. Each line of information will be returned in the following format:

  "WTMQ" & ControlChars.Tab & "TestResults" & ControlChars.Tab & TestResults

  The test results will be a tab-separated string. Information in each tab location will correspond to information in the columns of the list view control. This item cannot be queried if a test plan is currently running.

- **WTM.TestData.TotalTestTimeSec**
  Returns the total test time for the last run in seconds. The information will be returned in the following format:

  "WTMQ" & ControlChars.Tab & "TotalTestTimeSec" & TestTime

  This item cannot be queried if a test plan is currently running.

- **WTM.TestData.TotalPassed**
  Returns the number of results that passed for the previous test run. The information will be returned in the following format:

  "WTMQ" & ControlChars.Tab & "TotalPassed" & NumberPassed

  This item cannot be queried if a test plan is currently running.

- **WTM.TestData.TotalFailed**
  Returns the number of results that failed for the previous test run. The information will be returned in the following format:

  "WTMQ" & ControlChars.Tab & "TotalFailed" & NumberFailed

  This item cannot be queried if a test plan is currently running.

- **WTM.TestData.AvailableTestPlans**
  Returns a tab-separated list of the currently available test plans. The information will be returned in the following format:

  "WTMQ" & ControlChars.Tab & " AvailableTestPlans" & TabSeperatedListOfCurrent AvailableTestPlans

  This item cannot be queried if a test plan is currently running.

- **WTM.TestData.CurrentTestPlan**
  Returns the currently-selected test plan. The information will be returned in the following format:

  "WTMQ" & ControlChars.Tab & "CurrentTestPlan" & CurrentTestPlan

  This item cannot be queried if a test plan is currently running.

- **WTM.TestData.SummaryResults**
  Returns the summary results (passed, failed, aborted) for the last run. The information will be returned in the following format:

  "WTMQ" & ControlChars.Tab & " SummaryResults" & SummaryResults

  This item cannot be queried if a test plan is currently running.

- **WTM.TestData.MemoText**
  Returns the current information in the memo data text. The information will be returned in the following format:

  "WTMQ" & ControlChars.Tab & " MemoText" & TextCurrentlyInMemoArea

- **WTM.FlowControl.TestPlanRunning**
  Returns a true or false indication for the test plan running condition. The information will be returned in the following format:

  "WTMQ" & ControlChars.Tab & " TestPlanRunning" & RunningCondition

- **WTM.FlowControl.ProgressBarMaximum**
  Returns the maximum value of the progress bar. The information will be returned in the following format:

  "WTMQ" & ControlChars.Tab & "ProgressBarMaximum" & MaxValue

- **WTM.FlowControl.ProgressBarValue**
  Returns the current value of the progress bar. The information will be returned in the following format:

  "WTMQ" & ControlChars.Tab & " ProgressBarValue " & CurrentValue

- **WTM.FlowControl.StartTestPlan**
  This command will start the currently selected test plan. This item will be ignored if a test plan is currently running.

- **WTM.FlowControl.AbortTestPlan**
  This command will abort the currently selected test plan.

- **WTM.Display.ShowForm**
  This command will make the WTM form visible.

- **WTM.Display.HideForm**
  This command will make the WTM form invisible.

The above commands/queries are all of the form "client send command and receive response from the WTM if one is available." It may be desired to have the WTM tell the client when a given item has changed. This can be accomplished by using the following auto response configuration commands. If an auto response item is set to **True**, then the WTM will automatically send a response to the client (without the client sending a request) when the desired item is changed. The available auto response items are as follows:

- **WTM.AutoResponse.MemoTextChangeEnabled True or False**
- **WTM.AutoResponse.TestPlanChangeEnabled True or False**
- **WTM.AutoResponse.ProgressBarChangeEnabled True or False**
- **WTM.AutoResponse.TestPlanRunningChangeEnabled True or False**
- **WTM.AutoResponse.TestResultsEnabled True or False**
- **WTM.AutoResponse.TestPlanListEnabled True or False**

The following is an event that the Exec exposes for the WTM LAN connection:

Exec_LanMessageReceived(ByVal Message As String, ByRef CancelAction As Boolean) Handles Exec.LanMessageReceived

- This event is raised any time the WTM receives information from the WTMNet LAN interface. This event is raised before the Exec processes the command
- Message contains the information received from the WTMNet LAN interface
- If the application sets CancelAction= True, then the Exec will not process the command

Send Receive DUT command test step and additional DUT test parameters description
Send Receive DUT command test step

Description
The Send Receive DUT Command test plan step allows direct reading and writing with a DUT. All command strings you want to include are built into the table, each with its own test step ID in the table. When this test plan step is executed, a command or series of commands from the table is executed, the one with the test step ID that matches the Test Step ID parameter.

Specifications
Upper and lower specification limits are included in the table, Send Receive DUT Commands with Specs, described in the following section.

Parameters
- DUT Timeout for Receive Command (Seconds)

  Description
  The DUT Timeout for Receive Command (Seconds) parameter specifies the amount of time to wait during a 'Receive from DUT' action for the information to be returned on the serial connection.

  Accessing parameter name(s)
  To access the parameters for a test plan step in the WTM, click on the **Set Specs/ Parameters tab**.

  Default value
  The default value for this parameter is: 2

  To change the default value, click on the value and a drop-down arrow appears. Select the value you want from the list.

Data range
The value you enter must be in the following range: 0 to 60.

- **Send receive DUT commands with Specs**

Description
The Send Receive DUT Commands with Specs parameter is a table that contains both parameters and specifications. The first five columns are parameters, and the last one is a specification limit. The following parameters are included:
- **Test Step ID:** This parameter is used to identify each available DUT command. When this test step is executed, the command or information string will be sent to the DUT or test set. The Test Step ID parameter (not the column in the table) allows you to select which of the available commands are sent during execution of this step. If multiple rows have the same test step ID, all of the associated GPIB commands will be sent to the target device in top to bottom sequence.
- **Action:** If **Send to DUT** is selected, the command or information string is sent to the DUT. The contents of the Receive Data Action and Compare Data fields are ignored. If **Receive from DUT** is selected, the command or information string should contain the exact expected return. If **Wait mSec** is selected, the command or information string should contain the amount of time, in milliseconds, to wait. If **Send GPIB to Test Set** is selected, the command string is sent to the current test set in use.
- **Test Title:** Each command is referenced by a title; this title appears on the test summary.
- **Command or Information:** This field is used to identify the information to be used with the action selected. For the **Send to DUT** and **Send GPIB to Test Set** actions, this field contains the command to send to the selected device. For the **Receeive from DUT** action, this field should contain the expected data to be returned from the DUT. For the **Wait mSec** action, this field should contain a number identifying the amount of time to pause the program. If characters are needing to be sent or are contained in the expected data, they must be in the following format: ~(2,10,13). The numbers are parsed and added to the command or information string as characters.
- **Data Action:** This field is only used if the Action field is set to **Receive from DUT** or **Send GPIB to Test Set**, and if the GPIB command is a query. If **None** is selected, no information is compared or displayed. If **Display** is selected, the data received from the DUT or test set will appear on the test summary. If **Display and Compare** is selected, the compare data field is used to obtain the specifications and the results will appear on the test summary.
- **Compare Data:** This field contains the specifications to compare against if the Data Action field is set to **Display and Compare**. The format of this field for numeric data is LL, UL [, Location] where LL is the lower limit, UL is the upper limit, and the optional location identifies the data location when multiple items are returned from the DUT or test Sset. That is, 0, 10 , 2 would result in a lower limit of 0 and an upper limit of 10 with a data location of 2. So if the returned data was "+1,8.345" then the value of 8.345 would be compared to the 0, 10 limits.

Accessing parameter name(s)

To access the parameters or specifications for this test plan step in the WTM, click on the **Set Specs/Parameters** tab. Click on the **>>** button in the (Table) field to open the table. To edit the values, click on the appropriate cell to select it. Enter the value you want, or click on the drop-down arrow to select the value or **OFF**.

Default value

The default values are listed below:

Test Step ID	Action	Test Title	Command or Information	Data Action	Compare Data
1	Send to DUT	Send Command	Add command or information here	Display	LL, UL [, Location]

Data range

Allowable ranges are shown below:

Test Step ID	Action	Test Title	Command or Information	Data Action	Compare Data
Any numeric value	Send to DUT Receive from DUT Wait mSec Send GPIB to Test Set	Any String up to 50 characters long	Any String up to 50 characters long	None Display Display and Compare	Any String up to 50 characters long

- **Test step ID**

    Description

    The Test step ID parameter specifies which of the available commands in the GPIB Commands with Specs or Send Receive DUT Commands with Specs table are sent during execution of this step.

    All the commands are built in the table, and each copy of the test plan step will send all the commands with Test step ID field that match this parameter.

    Accessing parameter name(s)

    To access the parameters for test plan step in the WTM, click on the **Set Specs/Parameters** tab.

    Default value

    The default value for this parameter is: 1

    To change the default value, click on the value and a drop-down arrow appears. Select the value you want from the list.

Other parameters added to support DUT commands within existing test steps

Some WTM products feature additional parameters that have been added to support DUT communication in a number of test plan steps. For example, additional parameters have been added to the following test steps in the E6568C WTM. Refer to the E6568C Help file for further information on each of these parameters:

Test plan step	Additional DUT command parameter(s)
EGPRS start data connection	• DUT Timeout for Receive Command (Seconds) • EGPRS Start Data Conn DUT Commands with Specs • Operator interaction required
GPRS start data connection	• DUT Timeout for Receive Command (Seconds) • GPRS Start Data Conn DUT Commands with Specs • Operator interaction required
GSM base station-initiated call	• DUT Timeout for Receive Command (Seconds) • GSM BS Initiated Call DUT Commands with Specs • Operator interaction required
GSM mobile-initiated call	• DUT Timeout for Receive Command (Seconds) • GSM MS Initiated Call DUT Commands with Specs • Operator interaction required
W-CDMA origination	• DUT Timeout for Receive Command (Seconds) • WCDMA Origination DUT Commands with Specs • Operator interaction required
W-CDMA page	• DUT Timeout for Receive Command (Seconds) • WCDMA Page DUT Commands with Specs • Operator interaction required
W-CDMA registration	• DUT Timeout for Receive Command (Seconds) • WCDMA Registration DUT Commands with Specs • Operator interaction required

Send receive DUT command solution

Test Step ID	Action	Test Title	Command or Information	Data Action	Compare Data
1	Send to DUT	Send Command	ATI	Display	LL, UL [, Location]
1	Wait mSec	Wait	200	Display	LL, UL [, Location]
1	Receive from DUT	Receive Command	Add command or information	Display and Compare	QUALCOMM
2	Send GPIB to Test Set	RLC Reestablish = Off	CALL:RLC:REES OFF	Display	LL, UL [, Location]
2	Send GPIB to Test Set	Set Paging Service = AMR	CALL:PAG:SERV AMR	Display	LL, UL [, Location]
2	Send GPIB to Test Set	BS Originate Voice Call	CALL:ORIG	Display	LL, UL [, Location]
2	Wait mSec	Wait	10000	Display	LL, UL [, Location]
2	Send to DUT	Answer Call	ATA	Display	LL, UL [, Location]
2	Wait mSec	Speak (ECHO VOICE)	10000	Display	LL, UL [, Location]
2	Send GPIB to Test Set	End Call	CALL:END	Display	LL, UL [, Location]

Add New Row   Delete Row   Reload Defaults   Save as Defaults   OK   Cancel

---

Agilent E6568C WCDMA/GSM/GPRS/EGPRS Wireless Test Manager

File   Help

Run Test Plan | Define Test Plan | Set Specs/Parameters | Set Global Parameters | Change Configuration

Test control

Current test plan:

Test Device

Run Test Plan

View Test Conditions

Test Title	Measured Value	Lower Limit	Upper Li
DownLink Channel = 10700			
UpLink Channel = 9750			
RLC Reestablish = Off = CALL:RLC:REES OFF			
Set Paging Service = AMR = CALL:PAG:SERV AMR			
BS Originate Voice Call = CALL:ORIG			
Wait = 10000			
Answer Call = ATA			
Speak (ECHO VOICE) = 10000			
End Call = CALL:END			

Previous test results

1 2 3 4 5 6 7 8 9

Yield = 100% For Last 10

Passed

Passed = 0 Failed = 0   Test Time = 22.50 Sec

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

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

Agilent Technologies