

# QxBoundaryScan Analyzer

GETTING STARTED MANUAL

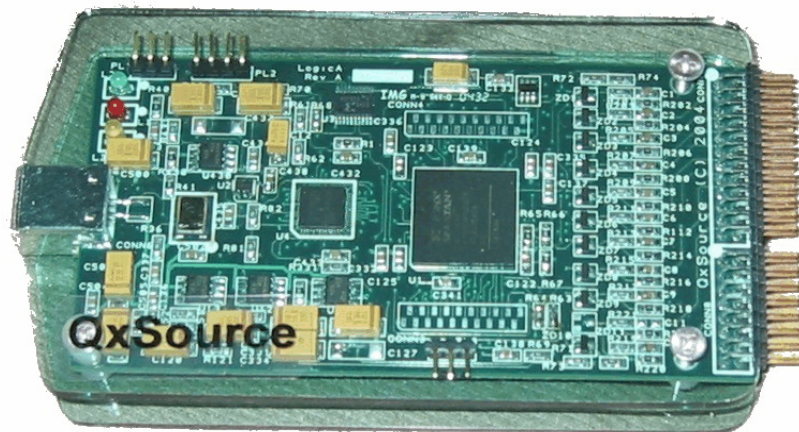
FOR

BOUNDARY SCANNER

FROM

QxSOURCE

February 2005  
Revision 3.1



## **Getting Started -Table of Contents**

**Warranty Information**

**System Requirements**

**Warnings**

---

## **Overview of This Manual**

**Overview of the JTAG Boundary Scan**

---

## **Installing the QxSource Software**

**Downloading and Installing the QxSource Software from the Internet**

**Software**

**USB Driver**

---

## **Installing the Hardware**

**Hardware**

---

## **Configuring the BSDL Scanner**

**File Selections**

**Other Setup**

**Changing the File Settings**

# **Using the Boundary Scan Software from QxSource**

**Buttons and Menus**

**Demo Mode**

---

## **Running Script Examples**

**Commands**

**Opening a BSDL Window**

**Loading Script**

**Running Script**

**Saving the Script**

## **Running a test by OLE Automation**

**What is OLE Automation**

**How to invoke an Ole Automation server**

**Commands**

**Flash Write/Read Sequence in C:**

**Code Example for Automation Client**

## QxSource Getting Started Manual

### Warranty Information

The Boundary Scan Probe is sold with a 90 day warranty starting from the date of purchase. Defective parts under warranty will either be repaired or replaced as suitable. QxSource's software is sold with no warranty, but upgrades can be downloaded from the QxSource's Website: <http://www.qxsource.com>.

QxSource makes no other warranties including but not limited to the implied warranties of products and their fitness for a particular purpose. In no event will QxSource be liable for consequential damages.

### System Requirements

Windows applications requires operating system resources. The quantity depends on your applications. If your system resources are low, close other applications until you get enough resources. Contact your system administrator or QxSource Technical Support at [support@qxsource.com](mailto:support@qxsource.com) for additional advice.

The following are minimum system requirements:

- 1GHz PC computer, Pentium III or faster.
- Running Windows 98, Me, 2000 or XP
- RAM > 256MByte.
- USB1 or USB2 Ports

### Warnings

You want to install the software **BEFORE** you plug in the USB device.

## Overview of This Manual

This getting started manual can be used to provide initial information so you can get started using your QxSource Boundary Scan instrument quickly. The text also includes detailed information for both the hardware and the software so that you can use it as a reference in the future.

### Overview of the JTAG Boundary Scan Software

The QxSource Boundary Scan software is written with the intention to give you fast access to the states of pins and flip flops in a device under test. The information for all the pins and flip flops is presented in an easy to read format on the screen of your computer. In addition to the grid format, you can view all the pins on a chip in the graphic presentation window. To present the information the software only requires a .BSDfile that outlines how the JTAG scan chain is implemented in the device under test.

### Overview of the JTAG Boundary Scan Hardware

The Boundary Scan hardware consists of a test probe with a USB2 interface and connection cables for attaching the probe to the device under test. The test probe receives power from the USB cable that is connected to your PC. For accurate functionality the test probe requires that you have appropriate software installed in your PC. The software must include the USB drivers for the Boundary Scan test program.

Note: You should not plug in the test probe to the USB cable until you have installed the software.

## **Installing the QxSource Software**

You can download and install the QxSource software from the internet. The software is also provided on the distribution CD that is included in your shipment box.

Please put the CD in one of the CD Rom drives of your computer. The setup application of the CD Rom should auto start unless you have disabled the auto start feature of your CD Rom drive.

If the set up application does not start please browse to the CD Rom drive and double click on setup.exe..

The setup application will require that you click on the OK button to acknowledge that you want the setup application to do the install. In addition, you will be asked to checkmark that you understand and acknowledge the license agreement. After clicking OK, the software will be installed in seconds if everything is normal.

Depending on your version of Windows and if you elected to get software icons installed on your desktop you may see the QxSource Boundary Scan icons on your screen.

You can always launch the software from the start programs menu where you should have the Boundary Scan icons visible.

## **Software**

The Boundary Scan GUI lets you interact with the application by clicking on the toolbar buttons as well as selecting functionality from the menus. Please see below for the explanations of the different buttons and menus.

## **USB Driver**

The USB Driver is named QxUSBDRV.sys and should reside in the directory:  
Windows/system32/drivers

In addition to the Driver, Windows requires an .inf file that allows an enumeration. The inf file is called Qxinf.inf this file should reside in the directory:  
Windows/inf

## **Uninstalling**

1) While the USB device is connected find the driver “QxSource Instrument” in the [Control Panel | System | Hardware | Device manager].

Select Properties, Driver and click on Uninstall.

2) If you used **OLE Automation** run  
“QxBoundaryScan.exe /unregserver” from a command line.

3) To uninstall the software delete the directory “QxBoundaryScan” under “Program Files”

## Installing the Hardware

Please unpack your shipment box and visually inspect your Boundary Scan test probe and the other items that were shipped to you for defects. Plug in the USB cable to one of the USB connectors on your PC computer. Thereafter, plug in the other end of the USB cable to the Boundary Scan test probe. At this point you should see a power on indication LED at the Boundary Scan test probe.

You can also, open up the Device Manager under windows and you should see a line “QxSource Instrument.” This line in the Device Manager indicates that your Boundary Scan test probe enumerated correctly and is ready for use.

The Boundary Scan test probe has a 10 pin connector located in its upper right corner named CONN1. This connector allows you to connect the JTAG lines to your device under test. Please connect your JTAG lines to your test object. For detailed description of the connectors pin configuration please see the outline below.

### Hardware:

The 10 pin JTAG connector on the Boundary Scan test probe has pin numbers below:

The connector (Conn1) is a double row header 2 x 5 with the even number row closest to the printed circuit board. Pin1 can be found by locating the white dot on the silkscreen of the printed circuit board.

Pin (1) GND    Pin (2) TMS  
Pin (3) GND    Pin (4) TCK  
Pin (5) GND    Pin (6) NC  
Pin (7) GND    Pin (8) TDI  
Pin (9) GND    Pin (10) TDO

Pin (18) **OutProbe**    If supported by your hardware.  
Pin (20) **InProbe**     If supported by your hardware.

## Quick Start

- Install the software
- Connect the USB device
- Connect the USB device to the Target under Test
- Start the software
- Starting the software auto detects the devices in the JTAG Scan chain
- Load a .BSD file for each device
- Load a map file that maps your signals to the port names in the BSD files (If you have such map file or .ucf file)
- Click the Scan button to read in all pin fields
- Type in a signal name in the Find edit field to locate a signal
- Click the Find button
- Double click in the output field for the selected signal or signals. You may double click on the pin in the graphical grid to change its value
- Click the Set button to apply the settings if you work with the string grid.

## Configuring the BSDL Scanner

### File Selection

After you have started the Boundary Scan software it will automatically try to communicate with the device under test through USB and JTAG. By reading the ID codes from the devices under test, the software will conclude how many devices are in your JTAG Scan chain. You will be asked to identify and load a .BSD file for each device the software detected in the Boundary Scan chain.

You can cancel the file open dialog box, but you have to at some point select your .BSD files before the software lets you perform any JTAG operations. See the description for the button opening the .BSD file.

Chips that do not have “Grid/ BGA” footprint will not get the graphical display in 3.1. In the non graphical string grid that you will see, you can double click on the “Out Val” column for a given signal and the signal will change from X -> 0 -> 1. You may also have to double click on the “Out Val” column for the control cell to enable the pin.

### Other Setup

After you have loaded the .BSD file you will be asked to identify a .map or .ucf file for each device the software detected in the Boundary Scan chain. The map file will let you see the signal names for each pin as you have defined them.

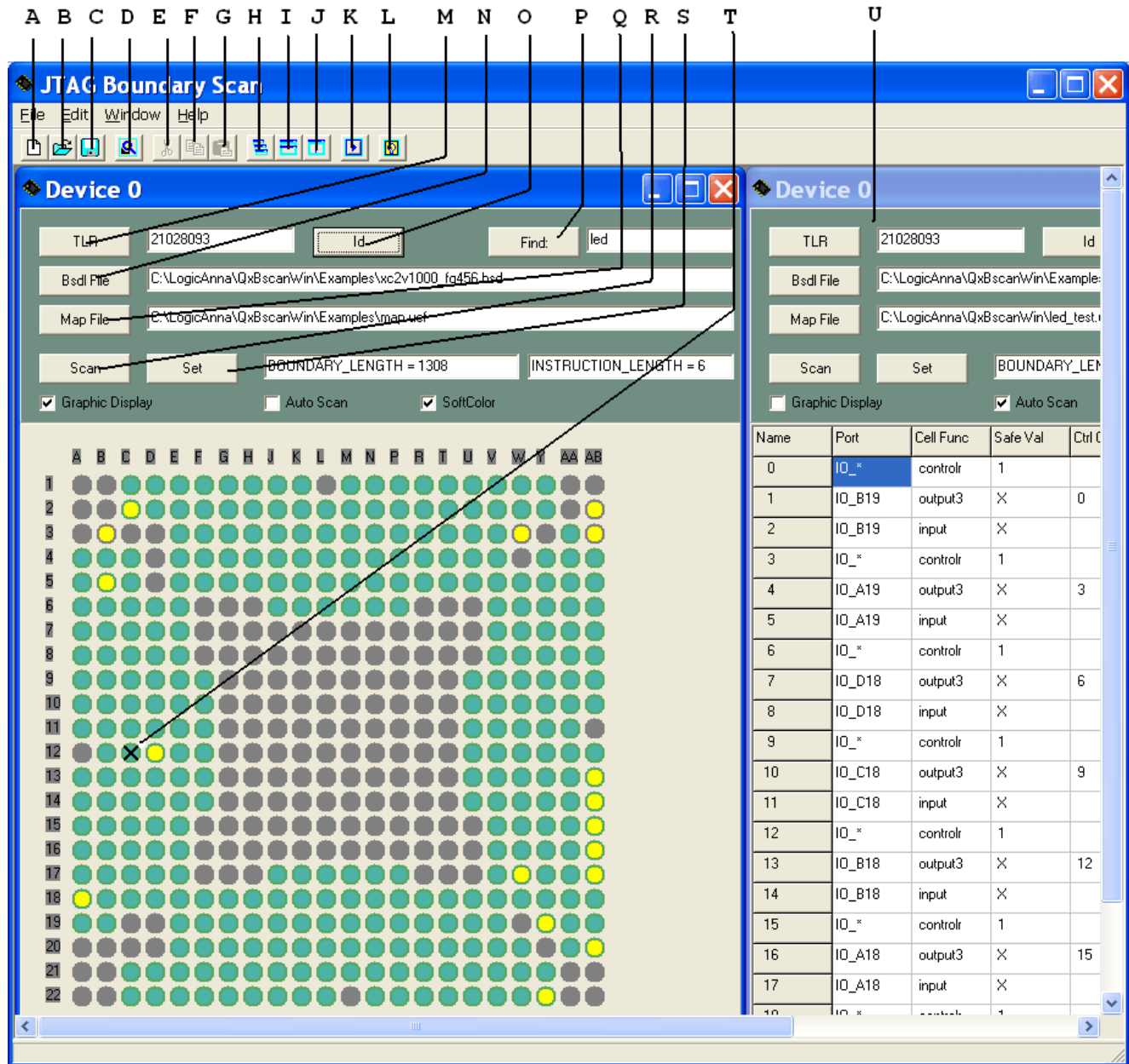
You can cancel the file open dialog box but you will not see your signal names in the signal grid. See the description for the button opening the .map file.

### Changing the File Settings

At the time you exit the Boundary Scan software your last settings for .BSD files and .map files are saved in the file lastcfg.dat

# Using the Boundary Scan Software from QxSource

## Buttons and Menus



- A. Create a new text file
- B. Open a script or text file
- C. Save a script or text file
- D. Open a new “BSDL Grid”
- E. Cut
- F. Copy
- G. Paste
- H. Cascade
- I. Horizontal tile
- J. Vertical tile
- K. Load and Run script
- L. Load and transmit SVF file to chip
- M. TLR or Test Logic Reset
- N. Load the BSDL file for the chip under test
- O. Read out the chip Id
- P. Find signal in grid (x will show in graph grid)
- Q. Load Map file
- R. Scan inputs
- S. Set outputs to specified values
- T. Shows found signal with the name as in the edit field
- U. Points to the String Grid display (alternative to graph display)

## Running Script Examples

The software can interpret script files to automate your testing. The script commands are designed such that you can test individual signals and conditionally skip the next line in the script language if the test is true. This mechanism allows you to create an error sub-routine that prints out information to your script log in case of a false test. Please see below for a list of the script language commands. The Application command line switch -p will make commands print on the status bar as they run. (A 2 Second delay per command is used so you can see what is done.)

### Script Language Commands

```
Load C:\FPGA\Virtex2\data\xc2v1000_fg456.bsd;  
Load C:\Program Files\QxBscanWin\Examples\led_test.ucf;  
Load C:\Program Files\QxBscanWin\Examples\FpgaTest.svf;
```

Start;	Script code Label, observe the “:”
TLR;	Performs Test Logic Reset
Device N;	Selects Device N in the JTAG Chain
Clear <port name   net name>;	Bring the signal <port name   net name> low
Set <port name   net name>;	Bring the signal <port name   net name> high
Disable <port name   net name>;	Disable the <port name   net name>
Sample;	Sample all Boundary chain signals
ExTest;	Force out all enabled Boundary chain signals
ProbeOut <port name   net name> <0 1>;	
Test <port name   net name> <0 1>;	Test a signal <port name   net name> to see If it is <0 1>. If test is true skip next line in script.
Goto Name;	Go to a Label in the script.
Call SubTest;	Call a subroutine that starts with Sub...
Return;	Returns from a Subroutine (to the line after the past call)
Pause	Pause 1 second
Print OK! Test was OK;	Print the string following Print
Exe Boundary Scan.exe;	Execute a program in the PC
Exit	Last line in the “main” part of the script.

### New (3.1) script Language Commands:

GroupDef <Bus,D7,D6,D5,D4,D3,D2,D1,D0>;  
GroupIn <Bus>;  
GroupOut <Bus,55>;  
FxTest; Force (**FAST**) out all enabled Boundary chain signals (No update of grid)

GetId <Device>; Show IdCode for Device on the Status Line  
GetIrLen <Device>; Show IrLen for Device on the Status Line  
Irls <Device>,<ID>; Set ID code for Device  
Scan <#Device>; Force a “Chain Scan” if #Devices=0 else SET # Devices

## **Loading a Script**

You can load a script by clicking the Open Button in the toolbar and Browse to the script of your choice. The Open button opens any text document.

## **Opening a BSDL Window**

In a script file you can instruct the software to load a .BSD file for a particular device by the code snippet below:

```
Device N;  
Load C:\Xilinx\Virtex2\data\xc2v1000_fg456.bsd;
```

## **Running Script**

You execute a script by clicking on the Run Script button. The Script Engine will automatically open a script log to which all your print statements will print in. You can save the Script Log as any text file by clicking on the Save button in the toolbar.

## **Saving a Script**

You can save a script by clicking the Save button in the toolbar and browse to the folder of your choice. The Save button will save any text document that is open in the text window.

```
// This is a test script  
Load C:\Xilinx\Virtex2\data\xc2v1000_fg456.bsd;  
Load C:\Program Files\QxBoundaryScan\Examples\map.ucf;
```

```
Start:  
TLR;  
Device 0;  
Set IO_F11;  
Sample;  
ExTest;  
Pause  
Clear led<0>; //IO_F11; //or Pin  
ExTest;  
Pause  
Set led<0>;  
ExTest;  
Pause;  
Set led<0>;
```

```
Test IO_F11 0;  
Goto End;  
Disable led<0>;  
Print OK! Test was OK;
```

```
End:  
TLR;  
Print Calling SubTest;  
Call SubTest;  
Print Back out;  
Print OK! This went Nice;  
//Exe BoundaryScan.exe;  
Exit
```

```
SubTest:  
Print OK! InSubTest...;  
Call SubTest2;  
Return;
```

```
SubTest2:  
Print OK! InSubTest2...;  
Return;
```

## Running a test by OLE Automation

### What is OLE Automation

OLE Automation is a method by which one application can control another application. In case of the QxSource Boundary Scanner the Boundary Scanner software acts as the server and the applications that you write will be the clients. Your client applications can be written in C, C++, Basic or Pascal. The client software can easily launch the server application as described below. You must register the OLE Automation server with windows see the “Note” below.

The client application can thereafter invoke any of the commands available in the scripting language by using the corresponding OLE Automation call. Please see below for an example.

The complete source code for an OLE Automation client is available from QxSource. The example code was written in Pascal to target the Delphi environment. The source code could serve as an example for how to create a client in other languages.

The Application command line switch “-p” will make commands print on the status bar as they run. (A 2 Second delay per command is used so you can see what is done.)

The Application command line switch “-n” will make sure that no dialogues open during software startup. The OLE script will have to load .BSD files.

**Note:**OLE commands can run in “TestMode” or “RunMode”.In “TestMode” OLE commands will only be checked for correct syntax and signal names. Set/use “QxRunMode” to have the commands interact with the target.

**Note:** You register an automation server by running  
“QxBoundaryScan.exe /regserver” from a command line.

**Note:** You unregister an automation server by running  
“QxBoundaryScan.exe /unregserver” from a command line.

## How to invoke an OLE Automation server

```
////////////////////////////////////
//
// C++
//
#include "QxBoundaryScan.h"
IBoundaryScript BoundaryScript;
//BOOL Ok = BoundaryScript.CreateDispatch( "QxBoundaryScan.BoundaryScript" );
//if ( (BoundaryScript.m_lpDispatch == NULL) ||(Ok == 0))
// error();

////////////////////////////////////
//
// Delphi
//var QxBoundary:Variant;
//QxBoundary:=CreateOleObject('QxBoundaryScan.BoundaryScript');

////////////////////////////////////
//
// Basic
//Dim QxBoundary As New QxBoundaryScan.BoundaryScript
```

## Commands

function	QxClear	(const S: WideString):	Integer; safecall;
function	QxDevice	(const S: WideString):	Integer; safecall;
function	QxDisable	(const S: WideString):	Integer; safecall;
function	QxExe	(const S: WideString):	Integer; safecall;
function	QxExTest:		Integer; safecall;
function	QxLoad	(const S: WideString):	Integer; safecall;
function	QxPrint	(const S: WideString):	Integer; safecall;
function	QxSample:		Integer; safecall;
function	QxSave	(const S: WideString):	Integer; safecall;
function	QxSet	(const S: WideString):	Integer; safecall;
function	QxSetRunMode:		Integer; safecall;
function	QxSetTestMode:		Integer; safecall;
function	QxTest	(const S: WideString):	Integer; safecall;
function	QxTLR: Integer;		safecall; safecall;
procedure	QxOpenNewText;		safecall; safecall;
function	QxProbeOut	(const S: WideString):	Integer; safecall;
function	QxGroupDef	(const S: WideString):	Integer; safecall;
function	QxGroupIn	(const S: WideString):	Integer; safecall;
function	QxGroupOut	(const S: WideString; value: Integer):	Integer;
function	QxFxTest:		Integer; safecall;
function	QxGetId(const Device: WideString):		Integer; safecall;
function	QxGetIrLen(const Device: WideString):		Integer; safecall;
function	QxIris	(const S: WideString; ID: Integer)	Integer; safecall;
function	QxScan	(const S: WideString):	Integer; safecall;

## Flash Write/Read Sequence in C:

```
QxGroupDef ("DataBus,D7,D6,D5,D4,D3,D2,D1,D0"); //Define Data Bus
QxGroupDef ("AddrBusL,A7,A6,A5,A4,A3,A2,A1,A0"); //Define Address Bus Bus
QxGroupDef ("AddrBusH,A15,A14,A13,A12,A11,A10,A9,A8"); //Define Address Bus Bus
```

```
QxGroupOut ("DataBus",55);//Setup Data value
QxGroupOut ("AddrBusL",33);//Setup Address value
QxGroupOut ("AddrBusH",07);//Setup Address value
```

```
QxClear ("Wr"); //Set write line low
QxFxTest; //Show the lines to the Flash (Note FxTest to be FAST!)
QxSet ("Wr"); //Set write line high
QxFxTest; //Show the lines to the Flash (Note FxTest to be FAST!)
```

```
QxClear ("Rd"); //Set read line low
QxFxTest; //Show the lines to the Flash
QxSet ("Rd"); //Set read line high
QxExTest; //Show the lines to the Flash (Note ExTest to Update Data)
QxGroupIn ("DataBus",val); //Get the Data
```

Please note that the QxGroupDef command defines a bus of 8 bits and that any of the upper bits can be undefined. The least significant bit of the bus is to the left in the definition. See examples below.

```
QxGroupDef ("bus,,D5,D4,D3,D2,D1,D0"); //Define a bus of 6 signals
```

If you do QxGroupOut ("bus",0X03); you will get bit D0 and D1 high after sending the QxFxTest or QxExTest where QxExTest will update the graphical grid. QxFxTest will not update the graphical grid but will be faster.

The signal definition can be built up by your names as "Chip\_Enable" or chip pin names such as: "IO\_C4".

Chips that do not have "Grid/ BGA" footprint will not get the graphical display in 3.1. In the non graphical string grid that you will see, you can double click on the "Out Val" column for a given signal and the signal will change from X -> 0 -> 1. You may also have to double click on the "Out Val" column for the control cell to enable the pin.

From the script or OLE you use "QxSet" and "QxDisable" to do the same thing.

## Code Example for Automation Client

```
var v:Variant;
procedure TOleTest.StartServerBClick(Sender: TObject);
begin
  v := CreateOleObject('QxBoundaryScan.BoundaryScript');
end;

procedure TOleTest.TestBClick(Sender: TObject);
begin
  v.QxDevice('0');
  v.QxSample;
  v.QxSetRunMode;
  v.QxSample;
  TestResE.Text:= IntToStr(v.QxTest(TestSignalE.Text + ' 1'));
end;

procedure TOleTest.ExTestBClick(Sender: TObject);
begin
  v.QxExTest;
end;

procedure TOleTest.SetSignalBClick(Sender: TObject);
begin
  v.QxSet(SetSignalE.Text);
end;

procedure TOleTest.ClrSignalBClick(Sender: TObject);
begin
  v.QxClear(SetSignalE.Text);
end;

procedure TOleTest.SampleBClick(Sender: TObject);
begin
  v.QxSample;
end;
```

## //Code Example for Automation Client Page2

```
procedure TOleTest.RTLBClick(Sender: TObject);
begin
  v.QxTLR;
end;

procedure TOleTest.RunModeBClick(Sender: TObject);
begin
  v.QxSetRunMode;
end;

procedure TOleTest.TestModeBClick(Sender: TObject);
begin
  v.QxSetTestMode;
end;

procedure TOleTest.LoadBClick(Sender: TObject);
begin
  v.QxLoad(LoadE.Text);
end;

procedure TOleTest.DeviceBClick(Sender: TObject);
begin
  v.QxDevice(DeviceE.Text);
end;

procedure TOleTest.DisableBClick(Sender: TObject);
begin
  v.QxDisable(DisSignaE.Text);
end;

procedure TOleTest.PrintBClick(Sender: TObject);
begin
  v.QxPrint(PrintE.Text);
end;
```

### **//Code Example for Automation Client Page3**

```
procedure TOleTest.SaveBClick(Sender: TObject);  
begin  
  v.QxSave(SaveE.Text);  
end;
```

```
procedure TOleTest.ExeClick(Sender: TObject);  
begin  
  v.QxExe(ExeE.Text);  
end;
```

```
procedure TOleTest.FormCreate(Sender: TObject);  
begin  
  LoadE.Text := ExtractFilePath(Application.ExeName) + 'Examples\xc2v1000_fg456.bsd';  
  SaveE.Text := ExtractFilePath(Application.ExeName) + 'Examples\SaveFile.txt';  
end;
```

```
procedure TOleTest.NewBClick(Sender: TObject);  
begin  
  v.QxOpenNewText;  
end;
```

```
procedure TOleTest.HelpBClick(Sender: TObject);  
begin  
  Form2.ShowModal;  
end;
```

## **Index**

To be defined.