PS Class Library for Microsoft® Visual Basic® 6.0

Programming Guide and Reference

Version 1.0.0

For Functional Test and Automate Test Equipment (ATE) Software Developer

May 2002 Edition

practical-software.com

Table of Contents

USER LICENSE AGREEMENT

THIS IS A LEGAL AGREEMENT BETWEEN YOU AND PRACTICAL SOFTWARE, ("PS") REGARDING SOFTWARE WHICH YOU MAY INSTALL ("SOFTWARE").PLEASE READ IT CAREFULLY. BY INSTALLING THE SOFTWARE AND ACCESSINGTHE SOFTWARE, YOU INDICATE YOUR AGREEMENT TO BE BOUND BY ALL OF THETERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THETERMS AND CONDITIONS, YOU SHOULD EXIT THIS PAGE AND, IF YOU RECEIVED/DOWNLOADTHE SOFTWARE, DELETE/UNINSTALL THE SOFTWARE FROM YOUR COMPUTER

1. Registered License Grant. PS grants to you on a single computer per one Unlocked code.

2. Unregistered License Grant. PS grants to you on unlimited computer and allow using SOFTWARE only 60 days after first installed.

3. Ownership. The Software is owned by Practical Software. The Software is protected by copyright laws and international treaty provisions. YOU MAY NOT MODIFY, REVERSE ENGINEER, REVERSE COMPILE OR DISASSEMBLE THE SOFTWARE. You may not remove, alter or destroy any copyright, proprietary or confidential notices placed on the Software or the documentation. You may copy the Software solely to make one backup or archival copy. You may not copy the documentation.

4. No Transfers. You may not sublicense the Software. You may not transfer the Software to a third party unless you cease all use of it, transfer all copies of it and accompanying Documentation, and the transferee agrees to be bound by the terms of this Agreement.

5. LIMITED WARRANTY

NO WARRANTIES. PS expressly disclaims any warranty for the SOFTWARE. The SOFTWARE and any related documentation is provided "AS IS" without warranty of any kind, either express or implied, including, without limitation, the implied warranties or merchantability, fitness for a particular purpose, or noninfringement. The entire risk arising out of use or performance of the SOFTWARE remains with you.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES. In no event shall PS be liable for any damages whatsoever (including, without limitation, accident, injury or death, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use this SOFTWARE, even if PS has been advised of the possibility of such damages.

6. Term. PS may immediately terminate this Agreement and the license it grants if you fail to comply with any material term or condition. Upon such termination you must immediately cease using the SOFTWARE and must delete/uninstall SOFTWARE from your computer.

ALL DISCLAIMERS HEREIN SHALL SURVIVE TERMINATION.

7. Miscellaneous. Should you have any questions concerning this document, or if you desire to contact Practical Software for any reason, please contact testdev@practical-software.com.

Hardware Requirement

- PC Pentium II 233 MHz or faster
- Memory 128 MB or more
- XGA (1024x768) Display card with 16 bits color quality or higher
- XGA (1024x768) Monitor CRT 15" (CRT 17" or LCD 15" is preferable)
- HDD 4 GB or greater
- Keyboard and PS/2 mouse
- NI IEEE-488 GPIB PCI card (now on will call NI GPIB Card). If use ISA bus type please mark sure that your PC have ISA slot available)
- Fluke 45 DVM with IEEE-488 option
- GPIB cable 1 meter length or longer

Note: if you do not have Fluke 45 DVM so you may need another brand or model instead. However, this document is referring to IEEE-488 commands over Fluke 45 so you must replace with suitable command so that to make your device works properly.

Software Requirement

- Microsoft® Windows® 98, ME, NT 4 Prof. SP6, 2000 Prof. or XP Prof. *(now on call OS)*
- Microsoft® Visual Basic® Version 6.0 SP5 *(now on call VB6)*
- National Instrument (NI) IEEE-488 GPIB card Driver Version 1.6 or later
- Practical Software (PS) Class Library *(now on call class library)*

  Note: Recommend use NT4, 2000 Prof. or XP Prof. is much better.

Setup and configure your system

Prepare you hardware, connect all keyboard, mouse, monitor to PC also if you have GPIB card so please plug it into suitable slot then connect GPIB cable between your PC via NI GPIB card to your device such Fluke 45 DVM (please do consult with your vendor's manuals for detail and safety).

For notebook machine, you may instead use NI GPIB on PCMCIA slot or USB port. Please check them from National Instruments web site for more information.

Anyway, if you cannot find any GPIB card or devices so you still play our class library by get in simulate mode that will be show and explain in later.

Assuming you has installed OS and VB6. Then you need to install NI GPIB driver and test so that make sure your NI GPIB card and device is works well (again, please do consult with NI manuals for detail and safety)

Installing PS class library, unzip file *PSTestDev.ZIP.* There are 3 files *PSTestDev.dll*, *PSVisualStyle.ocx* and *Cast.dll* that need copy into your system.

So assuming you have copied them into C:\Winnt\System32\ then you need do register them so that let OS know the new software components have been installed as below:

CD C:\Winnt\System32 ➔ Enter
C:\Winnt\System32 Regsvr32 *PSTestDev.dll* ➔ Enter
C:\Winnt\System32 Regsvr32 *PSVisualStyle.ocx* ➔ Enter
C:\Winnt\System32 Regsvr32 *Cast.dll* ➔ Enter

The rest of supporting files such VB6 Runtime, ADO/DAO Database drivers. They should include with VB6 and OS.

To uninstall PS class library, just do unregistered as below then delete them off form your system.

CD C:\Winnt\System32 ➔ Enter
C:\Winnt\System32 Regsvr32 /U *PSTestDev.dll* ➔ Enter
C:\Winnt\System32 Regsvr32 /U *PSVisualStyle.ocx* ➔ Enter
C:\Winnt\System32 Regsvr32 /U *Cast.dll* ➔ Enter

License Registration

Practical Software allows you run the class library without the registration or Unlicensed Mode. However, the feature between Unlicensed and Licensed Mode will be same but Unlicensed Mode will show the Form as Figure 1 about 5 second and show only one time after you have first created object from class library.

You can register our class library to your system by click License button then you will see form as Figure 2. To registration, you will need enter both License and Unlocked Code then click OK. For more information how to get License and Unlocked Code please see file *Order.doc* that enclosed in our software package.

Figure 1 Class library in unlicensed mode

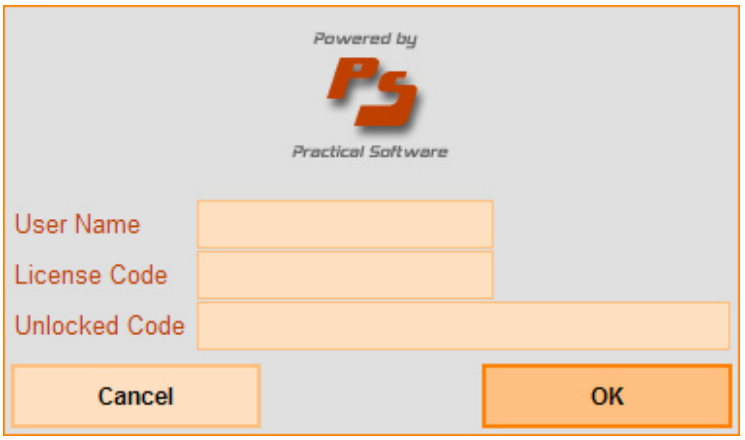

Figure 2 Class library registration form

Getting Start

This document supposes you have some experience in programming by VB6. Therefore, some action regarding that will not much be in details.

Now, let launch VB6 then open new Standard EXE project and add one command button into Form1 now your project will have one Form (name Form1) and one command button (name Command1).

Next, you need to add a new project reference by click *Project➔Reference* at menu bar (see Figure 3) then check at Practical Software X.X.X Test Development Library (see Figure 4).

Note: X.X.X is Class Library DLL Version.

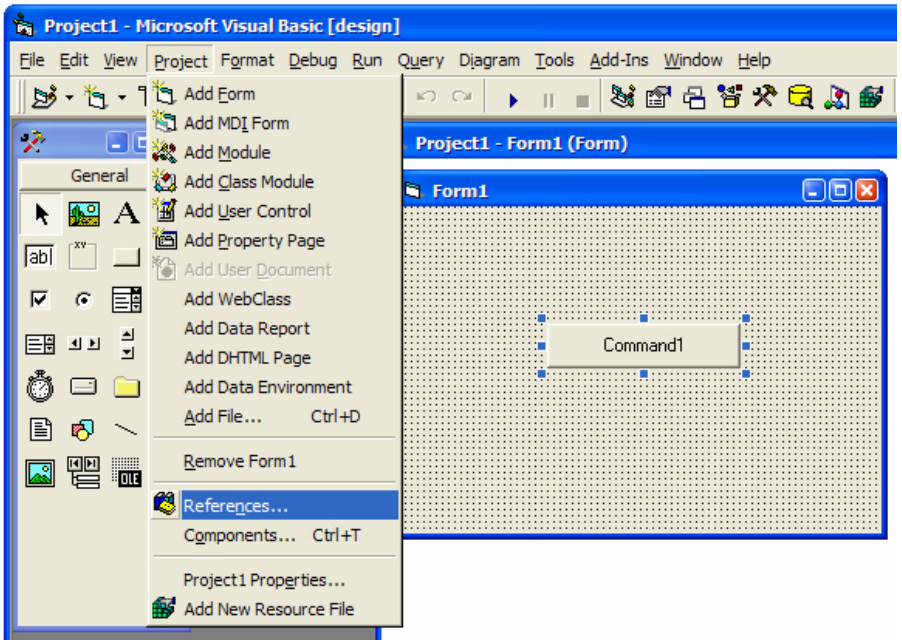

Figure 3 Microsoft Visual Basic 6 Add Reference to Project

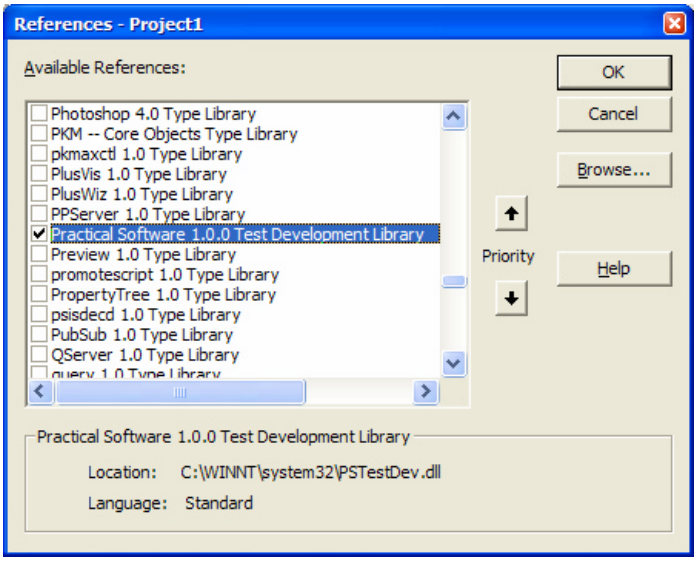

Figure 4 Microsoft Visual Basic 6 Add Reference to Project (Cont.)

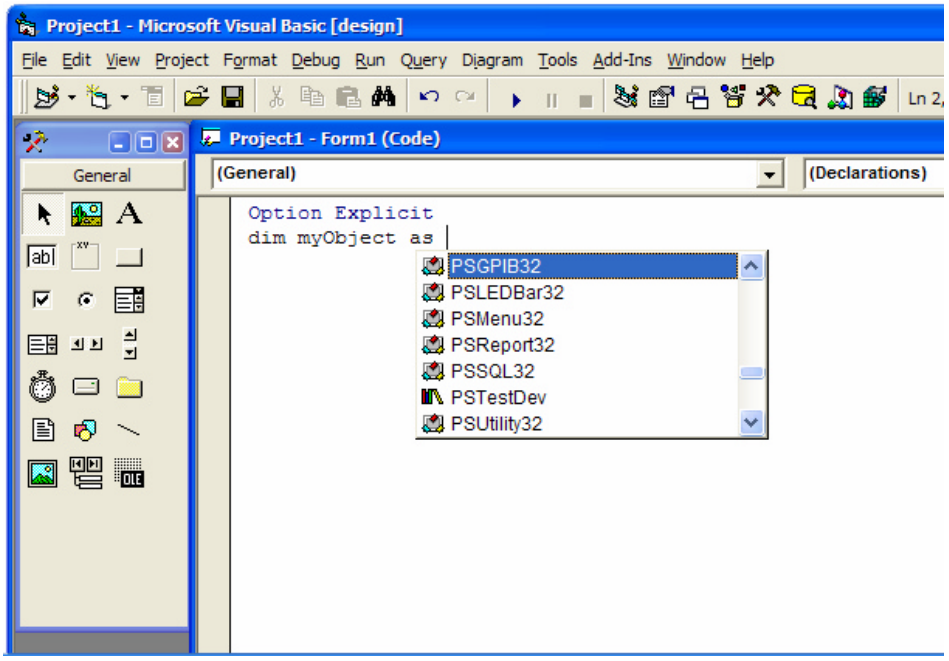

Figure 5 Microsoft Visual Basic 6 Add Reference to Project (Cont.)

If everything done correctly, you should see the available class objects (see Figure 5) in library as following:

- PSGPIB32 class
- PSLEDBar32 class
- PSMenu32 class
- PSReport32 class
- PSSQL32 class
- PSUtility32 class
- PSVirtualDevice32 class

Congratulation, we are now ready to learn how to use these class objects by step by step. Do not worry, it is very simple and make you look professional programmer. By the way, please save your new project into any location you want by use default project name (Project1)

Are you trying the object browser in VB6? Try it once! By simply press F2 in VB6 then select library PSTestDev so you will see all class objects inside even propertied, methods and Enum constants as Figure 6

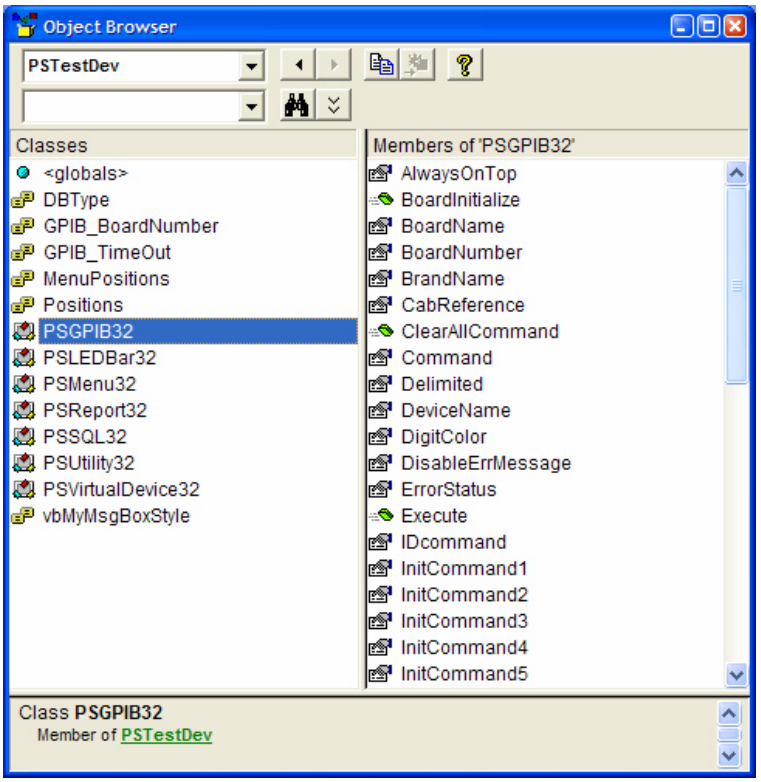

Figure 6 Object Brower

Using PSGPIB32 Class

This lesson will show how to write the code so that control your device via GPIB by use PSGPIB32 class. First, let put the below code into Form1 (Code) in Declarations section

```
Option Explicit
Dim WithEvents MyDevice1 As PSGPIB32
```

Next, put the code as below:

```
Private Sub Form_Load()
Set MyDevice1 = New PSGPIB32
End Sub
```

And:

```
Private Sub Command1_Click()
With MyDevice1
    .SimulateMode = True      'for no GPIB card or device connected
    .DeviceName = 7           'this device address is 7
    .PanelParameter = "My Test Parameter 1"
    .PanelOption = "VDC"
    .PanelStatus = "Remote"
    .BrandName = "Fluke"
    .ModelName = "45 #1"
    .PanelPosition = Center
    .StatusPosition = Centerdown
    .InitCommand1 = "*RST;CLS"
    .ReadCommand = "MEAS?"
    .ReadInterval = 250         'read every 250 ms
    .Offset = 1
```



```
    .Execute
End With
End Sub
```

Run project then click Command1 button, class library will one-time write INICommand1 into device address (name) 7 then repeated write ReadCommand into same device by every 250 ms and read data back after send each ReadCommand.

In addition, class library will create 2 form panels one Device Panel and one Status Panel (see Figure 7 and Figure 8)

Figure 7 Device Panel at Center Screen

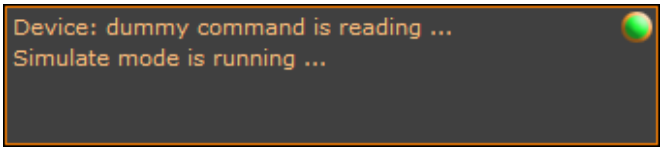

Figure 8 Status Panel at Center Down Screen

That is all for basic writing codes for access a GPIB device. For others device, they are depend on you as long as you can find the command reference manual for each device you will unlimited writing code to control them. It just put command into propertied InitCommand1, 2, 3, 4 and 5 then use methods Execute. The class library will write these commands into device respectively.

What about add one more device! It's so easy. If you have real device just connect it via GPIB cable, power on and set GPIB address. Assuming you have one more Fluke 45 and has been set address to be 5. The below code will make independently communicate to 2 devices and create 3 form panels whereas move both Device Panels to right up position on screen but Status Panel is remained (see Figure 9)

```
Option Explicit
Dim WithEvents MyDevice1 As PSGPIB32
Dim WithEvents MyDevice2 As PSGPIB32

Private Sub Form_Load()
Set MyDevice1 = New PSGPIB32
Set MyDevice2 = New PSGPIB32
End Sub

Private Sub Command1_Click()
With MyDevice1
    .SimulateMode = True        'for no GPIB card or device connected
    .DeviceName = 7             'this device address is 7
    .PanelParameter = "My Test Parameter 1"
    .PanelOption = "VDC"
    .PanelStatus = "Remote"
    .BrandName = "Fluke"
    .ModelName = "45 #1"
    .PanelPosition = RightUp
    .StatusPosition = Centerdown
    .InitCommand1 = "*RST;CLS"
```



```
    .ReadCommand = "MEAS?"
    .ReadInterval = 250         'read every 250 ms
    .Offset = 1
    .Execute
End With
With MyDevice2
    .SimulateMode = True        'for no GPIB card or device connected
    .DeviceName = 5             'this device address is 7
    .PanelParameter = "My Test Parameter 2"
    .PanelOption = "VDC"
    .PanelStatus = "Remote"
    .BrandName = "Fluke"
    .ModelName = "45 #2"
    .PanelLeft = MyDevice1.PanelLeft
    .PanelTop = MyDevice1.PanelTop + MyDevice1.PanelHeight
    .StatusPosition = Centerdown
    .InitCommand1 = "*RST;CLS"
    .ReadCommand = "MEAS?"
    .ReadInterval = 250         'read every 250 ms
    .Execute
End With
End Sub
```

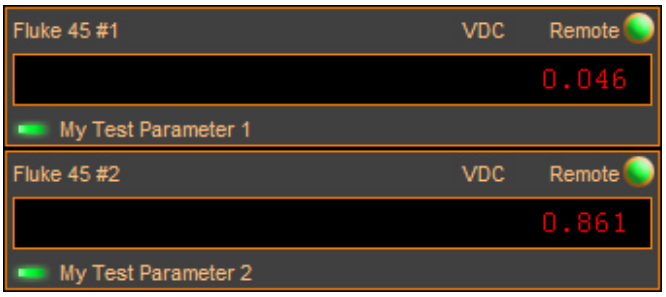

Figure 9 Stack Device Panels at Right Up screen

Using PSVirtualDevice32 Class

This class object is purposed for Non-GPIB device, for instant, you may have some device that connect via serial port, parallel port or another port and you want they work like PSGPIB32 class.
The following code is demonstrably send value that read from MyDevice2 then put it into MyVirtualDevice.

```
Option Explicit
Dim WithEvents MyDevice1 As PSGPIB32
Dim WithEvents MyDevice2 As PSGPIB32
Dim WithEvents MyVirtualDevice As PSVirtualDevice32

Private Sub Form_Load()
Set MyDevice1 = New PSGPIB32
Set MyDevice2 = New PSGPIB32
Set MyVirtualDevice = New PSVirtualDevice32
End Sub

Private Sub Command1_Click()
With MyDevice1
    .SimulateMode = True        'for no GPIB card or device connected
    .DeviceName = 7             'this device address is 7
    .PanelParameter = "My Test Parameter 1"
```



```
    .PanelOption = "VDC"
    .PanelStatus = "Remote"
    .BrandName = "Fluke"
    .ModelName = "45 #1"
    .PanelPosition = RightUp
    .StatusPosition = Centerdown
    .InitCommand1 = "*RST;CLS"
    .ReadCommand = "MEAS?"
    .ReadInterval = 250         'read every 250 ms
    .Offset = 1
    .Execute
End With
With MyDevice2
    .SimulateMode = True        'for no GPIB card or device connected
    .DeviceName = 5             'this device address is 7
    .PanelParameter = "My Test Parameter 2"
    .PanelOption = "VDC"
    .PanelStatus = "Remote"
    .BrandName = "Fluke"
    .ModelName = "45 #2"
    .PanelLeft = MyDevice1.PanelLeft
    .PanelTop = MyDevice1.PanelTop + MyDevice1.PanelHeight
    .StatusPosition = Centerdown
    .InitCommand1 = "*RST;CLS"
    .ReadCommand = "MEAS?"
    .ReadInterval = 250         'read every 250 ms
    .Execute
End With
With MyVirtualDevice
    .Caption = "Virtual Device Demonstration"
    .Suffix = "VDC"
    .Left = MyDevice2.PanelLeft
    .Top = MyDevice2.PanelTop + MyDevice2.PanelHeight
    .Show
End With
End Sub

Private Sub MyDevice2_ValueChanged(ByVal Value As Double, _
                                   ByVal RawValue As Variant)
MyVirtualDevice.SendValue Value, 2
End Sub
```

Run project then click Command1 button then you will find the forms on screen as Figure 10. The value that present on MyDevice2 (Fluke 45 #2) will present to MyVirtualDevice as well.

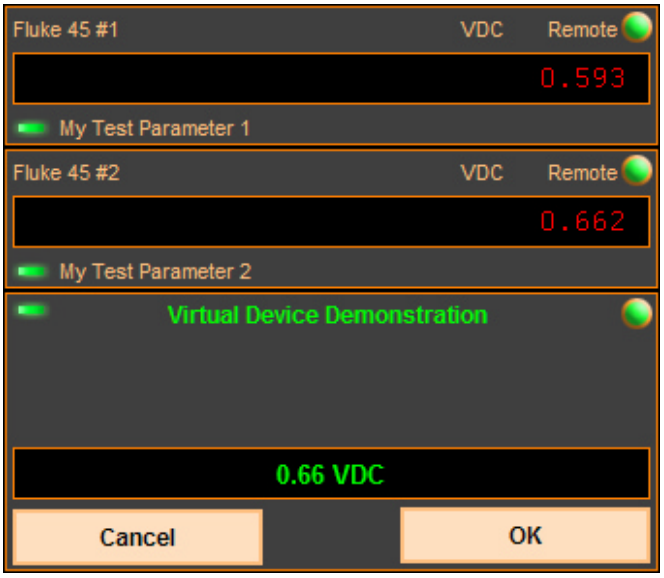

Figure 10 Virtual Device Panel

Using PSLEDBar32 Class

The horizontal LED slide bar, our intention is purpose for tuning value by operator, unlike one that provided from Microsoft. Its higher precision, support decimal point, negative number and custom define tolerantly.

The following code is demonstrably send value from MyLEDBar to MyVirtualDevice.

```
Option Explicit
Dim WithEvents MyDevice1 As PSGPIB32
Dim WithEvents MyDevice2 As PSGPIB32
Dim WithEvents MyVirtualDevice As PSVirtualDevice32
Dim WithEvents MyLEDBar As PSLEDBar32

Private Sub Form_Load()
Set MyDevice1 = New PSGPIB32
Set MyDevice2 = New PSGPIB32
Set MyVirtualDevice = New PSVirtualDevice32
Set MyLEDBar = New PSLEDBar32
End Sub

Private Sub Command1_Click()
With MyDevice1
    .SimulateMode = True        'for no GPIB card or device connected
    .DeviceName = 7             'this device address is 7
    .PanelParameter = "My Test Parameter 1"
    .PanelOption = "VDC"
    .PanelStatus = "Remote"
    .BrandName = "Fluke"
    .ModelName = "45 #1"
    .PanelPosition = RightUp
    .StatusPosition = Centerdown
    .InitCommand1 = "*RST;CLS"
    .ReadCommand = "MEAS?"
    .ReadInterval = 250         'read every 250 ms
    .Offset = 1
    .Execute
End With
With MyDevice2
    .SimulateMode = True        'for no GPIB card or device connected
```

```
    .DeviceName = 5             'this device address is 7
    .PanelParameter = "My Test Parameter 2"
    .PanelOption = "VDC"
    .PanelStatus = "Remote"
    .BrandName = "Fluke"
    .ModelName = "45 #2"
    .PanelLeft = MyDevice1.PanelLeft
    .PanelTop = MyDevice1.PanelTop + MyDevice1.PanelHeight
    .StatusPosition = Centerdown
    .InitCommand1 = "*RST;CLS"
    .ReadCommand = "MEAS?"
    .ReadInterval = 250         'read every 250 ms
    .Execute
End With
With MyVirtualDevice
    .Caption = "Virtual Device Demonstration"
    .Suffix = "VDC"
    .Left = MyDevice2.PanelLeft
    .Top = MyDevice2.PanelTop + MyDevice2.PanelHeight
    .Show
End With
With MyLEDBar
    .DecimalPoint = 2
    .Min = -80
    .Max = 80
    .ScaleMin = -100
    .ScaleMax = 100
    .PreciseStep = 0.5
    .Suffix = "VDC"
    .Left = MyVirtualDevice.Left
    .Top = MyVirtualDevice.Top + MyVirtualDevice.Height
    .Show
End With
End Sub

Private Sub MyDevice2_ValueChanged(ByVal Value As Double, _
                                   ByVal RawValue As Variant)
'MyVirtualDevice.SendValue Value, 2
End Sub

Private Sub MyLEDBar_ValueChanged(ByVal Value As Double, _
                                  ByVal IsPassed As Boolean)
MyVirtualDevice.SendValue Value, 2
```

Run project then click Command1 button. You will find the forms on screen as Figure 11. Try to scroll left/right and observe that the value that present on MyLEDBar will present to MyVirtualDevice as well. Consequently, during scrolling if value is under -80 or upper 80 the caption text color will change as defined.

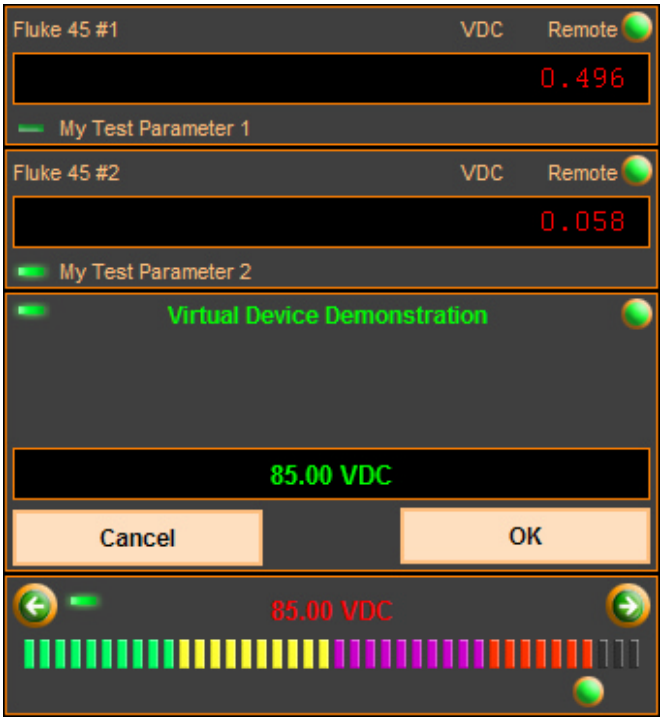

Figure 11 LED Bar

Using PSMenu32 Class

In this class library, we will give you an amazing Test Menu. You have no need to write any code so that create the Test Menu even any change in the future such delete modification, or add new test item so you still no need to modify your code.

The information is required by class library so that to create Test Menu will find in simple text files ATE.INI and Products.INI so in this lesson you will learn more how to create those files.

The intention of ATE.INI is for keep any information that relate with tester and independent with testing products whereas Products.INI relates to testing products but not concerning a tester.

Next, please open Notepad program and type or copy follow text then save it as ATE.INI in same folder of your VB6 project.

```
;ATE.INI
[History]
Version=1.0.0

[ATE Configuration]
ID=DEMO
TestID=Production,Retest,Rework,Evaluation,Repair
TestIDDefault=Production
TestProduct=X1 Dummy Product
CustomerLogo=CompanyX.jpg
```

Next, select *File➔New from* Notepad menu bar and type or copy following text into Notepad then save it as Products.INI in same folder of your VB6 project.

```
;Products.INI
[X1 Dummy Product]
Version=1.0.0
```

```
TableName=X1_Dummy_Table
CustomerName=CompanyX Inc.
LocalPN=98726-22-309
LocalAssyPN=87289-01-309
CustomerPN=77-99-02349
Model=Eagles,Hawk
Revision=A,AA,AB
Remark=F/W Rev.,Lot No.

[X1 Dummy Product Spec]
printGroupOrder=
ScreenGroupOrder=
;Num=[DBFieldName],[MinSpec],[MaxSpec],[Target],[Unit],GroupName,ParaName,
;[CustomFlag]
1=Main_Continuity,,,,Logic,Continuity Test,Main Power I/P
2=A_Output_Continuity,,,,Logic,Continuity Test,Output Slot A
3=B_Output_Continuity,,,,Logic,Continuity Test,Output Slot B
4=C_Output_Continuity,,,,Logic,Continuity Test,Output Slot C
5=D_Output_Continuity,,,,Logic,Continuity Test,Output Slot D

6=Q1Current,100,150,,mA,Current Test,Q1 Current
7=Q2Current,100,150,,mA,Current Test,Q2 Current
8=Q3Current,100,150,,mA,Current Test,Q3 Current
9=Q4Current,100,150,,mA,Current Test,Q4 Current
10=TotalCurrent,400,500,,mA,Current Test,Total Current

11=Q1_Gain_1MHz,5,10,,dB,Gain @1MHz Test,Q1
12=Q2_Gain_1MHz,5,10,,dB,Gain @1MHz Test,Q2
13=Q3_Gain_1MHz,5,10,,dB,Gain @1MHz Test,Q3
14=Q4_Gain_1MHz,1,10,,dB,Gain @1MHz Test,Q4

15=F1_-3dB,1.5,1.8,,MHz,Frequency Response Test,Q1 @-3dB
16=F2_-3dB,1.5,1.8,,MHz,Frequency Response Test,Q2 @-3dB
17=F3_-3dB,1.5,1.8,,MHz,Frequency Response Test,Q3 @-3dB
18=F4_-3dB,1.5,1.8,,MHz,Frequency Response Test,Q4 @-3dB

19=DC_Bias_Q1,1.2,1.8,,VDC,Bias Voltage Test,Q1
20=DC_Bias_Q2,.8,1.2,,VDC,Bias Voltage Test,Q2
21=DC_Bias_Q3,1.2,1.8,,VDC,Bias Voltage Test,Q3
22=DC_Bias_Q4,1.2,1.8,,VDC,Bias Voltage Test,Q4

23=Temp_Full_Q1,,70,,C,Temp Full Load Test,Q1
24=Temp_Full_Q2,,70,,C,Temp Full Load Test,Q2
25=Temp_Full_Q3,,70,,C,Temp Full Load Test,Q3
26=Temp_Full_Q4,,70,,C,Temp Full Load Test,Q4
27=<EOT>
```

Put one more command button in your form so a new button name should be Command2 then add/modify the below code into Declarations section.

```
Dim WithEvents MyMenu As PSMenu32
Dim MyUtility As PSUtility32
```

and the rest code as following:

```
Private Sub Form_Load()
```



```
'!!!Your previous code is here!!!

Set MyMenu = New PSMenu32
Set MyUtility = New PSUtility32
End Sub

Private Sub Command2_Click()
With MyUtility
   .MyAppPath = App.Path 'let class library know where is your application path
   .INIPath = .MyAppPath 'let class library know where is your INI files
   MyMenu.ShowMenu
End With
End Sub

Private Sub Form_Unload(Cancel As Integer)
Set MyDevice1 = Nothing
Set MyDevice2 = Nothing
Set MyVirtualDevice = Nothing
Set MyLEDBar = Nothing
Set MyMenu = Nothing
Set MyUtility = Nothing
End Sub

Private Sub MyMenu_ExitClick()
Unload Me
End
End Sub
```

Run project then click Command2 button. You will find the User Entry form Test Menu form on screen as Figure 12 and Figure 13. That is all for steps to create Test Menu from class library.

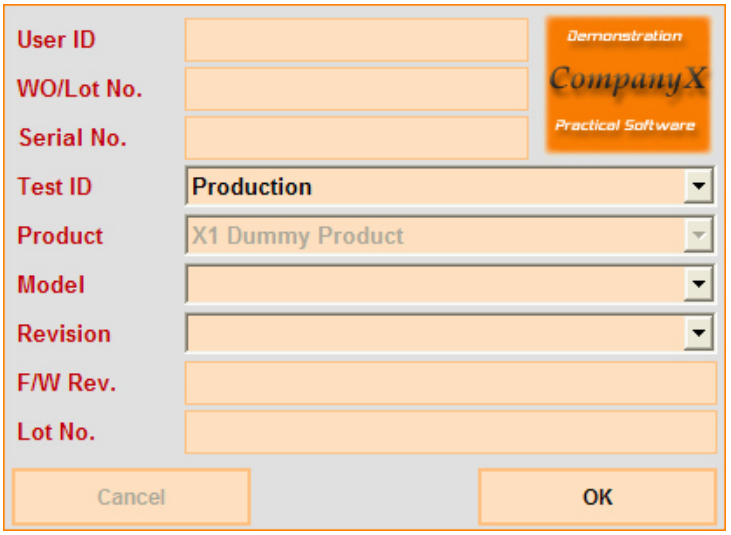

Figure 12 User Entry Form

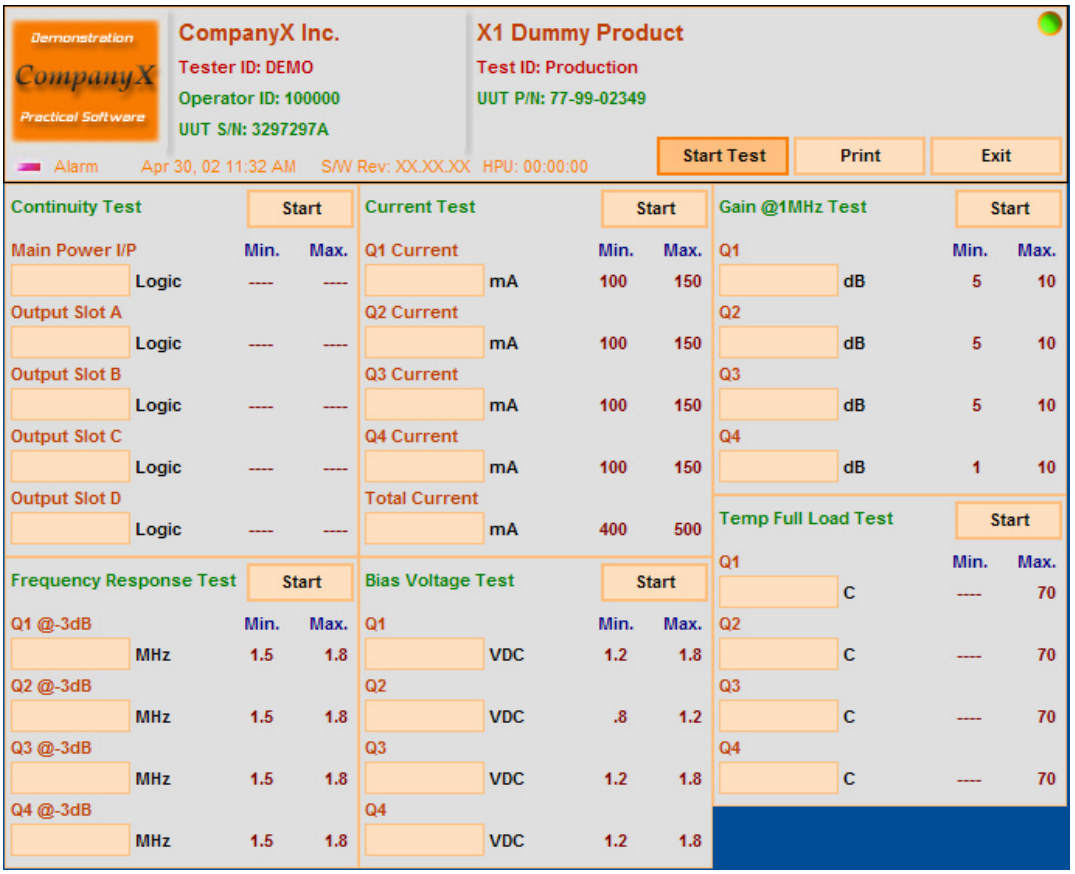

Figure 13 Test Menu

Note: After you created PSMenu32 class object in your application then you should at least one time completely use ShowMenu method otherwise VB6 may have problem due to cannot close that object.

Now, we will try to send something to Test Menu so let stop your application. The following code is demonstrably send value that read from MyDevice1 then put it into Test Menu group name Frequency Response Test and test parameter name Q1 @-3dB. Type or copy it to your project.

```
Private Sub MyDevice1_ValueChanged(ByVal Value As Double, _
                                   ByVal RawValue As Variant)
Value = Format(Value, "#,##0.00")
MyMenu.SendData "Frequency Response Test", "Q1 @-3dB", Value
End Sub
```

Run project, click Command2 then Command1 button (do not click Command1 before Command2). The Figure 14 shows the number that sent from MyDevice1. Consequently, in case of the value is out of specification. It will be red and get black color if it is within specification. Also the head menu will flash the image such Pass/Fail, this is overall result that mean if any one of test parameter is failure (out of spec.) the overall test result will be show Fail unless all test parameters are passed the overall result will be show Pass.

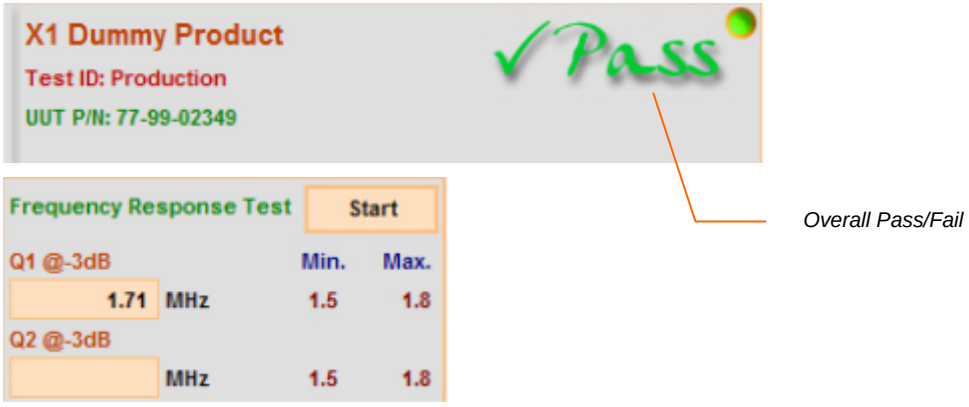

Figure 14 Send Test Data to Test Menu

Using PSUtility32 Class

In previous sample code has shown a few lines for using the PSUtility32 class. The intention for this class is provides varied functions also other class such PSMenu32 must in conjunction use with this class.

There are such provide functions as following:

- Delay Time in milli-second
- Accessing INI files
- Message box in PS style
- INI's security
- and more …

However, more details for each properties and methods will explain in Class Library Reference section.

Using PSSQL32 Class

This is one of amazing class that provide the easiness by writing test data results into database in one method:

```
Dim MyDatabase As PSSQL32

Private Sub Form_Load()
Set MyDatabase = New PSSQL32
End Sub

Private Sub Command1_Click()
MyDatabase.WriteDatabase , , , App.Path & "\TestData.MDB"
End Sub
```

There is support Microsoft® Access and Microsoft® SQL Server. More detail will explain in Class Library Reference section.

Using PSReport32 Class

Similar as above, this class also gives you the very convenience by printing test report in one method:

```
Dim MyReport As PSReport32

Private Sub Form_Load()
Set MyReport = New PSReport32
End Sub

Private Sub Command1_Click()
MyReport.PrintReport
End Sub
```

The following is a sample of print out report that generated from class library. More detail will explain in Class Library Reference section.

CompanyX Inc. X1 Dummy Product Test Report
*****PASS*****

Operator ID	8711	Test Date	05/02/2002
Project No	98726-22-309 (Eagles)	Test Time	10:09:03 AM
Local Assy No	87289-01-309	Test Duration	00:00:39
Customer P/N	77-99-02349 (AA)	Test S/W Ver.	XX.XX.XX
Serial Number	8792736A (W/O 987-03-238-02)	ATE ID	DEMO
F/W Rev.	8770A	Lot No.	09734-093-02

Test Description	Min Spec	Max Spec	Results	Units	Pass/Fail
[Continuity Test]					
Main Power I/P	----	----	PASS	Logic	Pass
Output Slot A	----	----	PASS	Logic	Pass
Output Slot B	----	----	PASS	Logic	Pass
Output Slot C	----	----	PASS	Logic	Pass
Output Slot D	----	----	PASS	Logic	Pass
[Current Test]					
Q1 Current	100.0	150.0	112	mA	Pass
Q2 Current	100.0	150.0	131	mA	Pass
Q3 Current	100.0	150.0	142	mA	Pass
Q4 Current	100.0	150.0	133	mA	Pass
Total Current	400.0	500.0	433	mA	Pass
[Gain @1MHz Test]					
Q1	5.0	10.0	7.2	dB	Pass
Q2	5.0	10.0	6.8	dB	Pass
Q3	5.0	10.0	6.1	dB	Pass
Q4	1.0	10.0	7.8	dB	Pass
[Frequency Response Test]					
Q1 @-3dB	1.5	1.8	1.77	MHz	Pass
Q2 @-3dB	1.5	1.8	1.73	MHz	Pass
Q3 @-3dB	1.5	1.8	1.68	MHz	Pass
Q4 @-3dB	1.5	1.8	1.64	MHz	Pass
[Bias Voltage Test]					
Q1	1.2	1.8	1.26	VDC	Pass
Q2	0.8	1.2	0.965	VDC	Pass
Q3	1.2	1.8	1.33	VDC	Pass
Q4	1.2	1.8	1.42	VDC	Pass
[Temp Full Load Test]					
Q1	----	70.0	69	C	Pass
Q2	----	70.0	68.5	C	Pass
Q3	----	70.0	68.6	C	Pass
Q4	----	70.0	68.9	C	Pass

End of Report

Figure 15 Sample Print out Report

First Test Application

Now, we will study the combination of all class so that to produce the real practice of test application program. In reality, you may need to gather the programming manual of each device so that to know what command that needs to write into each device in order to get the desired output back. You may ask your vendor so that to have these manuals also you may find them via Internet as well.

Define the GPIB address, as we know each device must have and set own unique address number, there are 30 (1 to 30) addresses (devices) connecting in a GPIB card also each devices will have varied method for setting own address such DIP switch, jumper or soft switch so please consult your user manual what right to do.

Grouping your test parameters, take list all each equivalent testing items and combines them in same group. For instance, Resistance/Continuity Test, Current Test, etc. This is helpful reduction the test time because you will need to initialize the device one time then test many parameters.

Preparative INI files, the class library need in conjunction works with 2 supporting files ATE.INI and Products.INI both are normal text file. You might use the existing one for modification or create new by use any editor program.

Test Groups Flow, order you test groups, which one go first and what happen if a test parameter has been failed.

Next, please find the sample VB project (Sample_My Project\MyProject.vbp) that enclosed with our software package. This VB project is supposed five GPIB devices are connected in tester as below (do not worry if you do not have those devices so we will able to run this project by simulate mode).

1.	Agilent 3499A	Switch Control	Address 9
2.	Agilent E3631A	DC Power Supply	Address 5
3.	Agilent 33120A	Signal Generator	Address 6
4.	Agilent 53131A	Frequency Counter	Address 7
5.	Fluke 45	DVM	Address 12

Preparative folders, by create the new folders as C:_New INI, C:_Test Database and C:_My Project then copy all files into folder C:_My Project.

The Figure 16 shows the test software flow diagram that equivalent with sample. It is simple and not complicates so try to step run project and learn how it works also any doubtful in class library methods, properties, events even how ATE.INI and Products.INI works, you will find the explanation in Class Library Reference section.

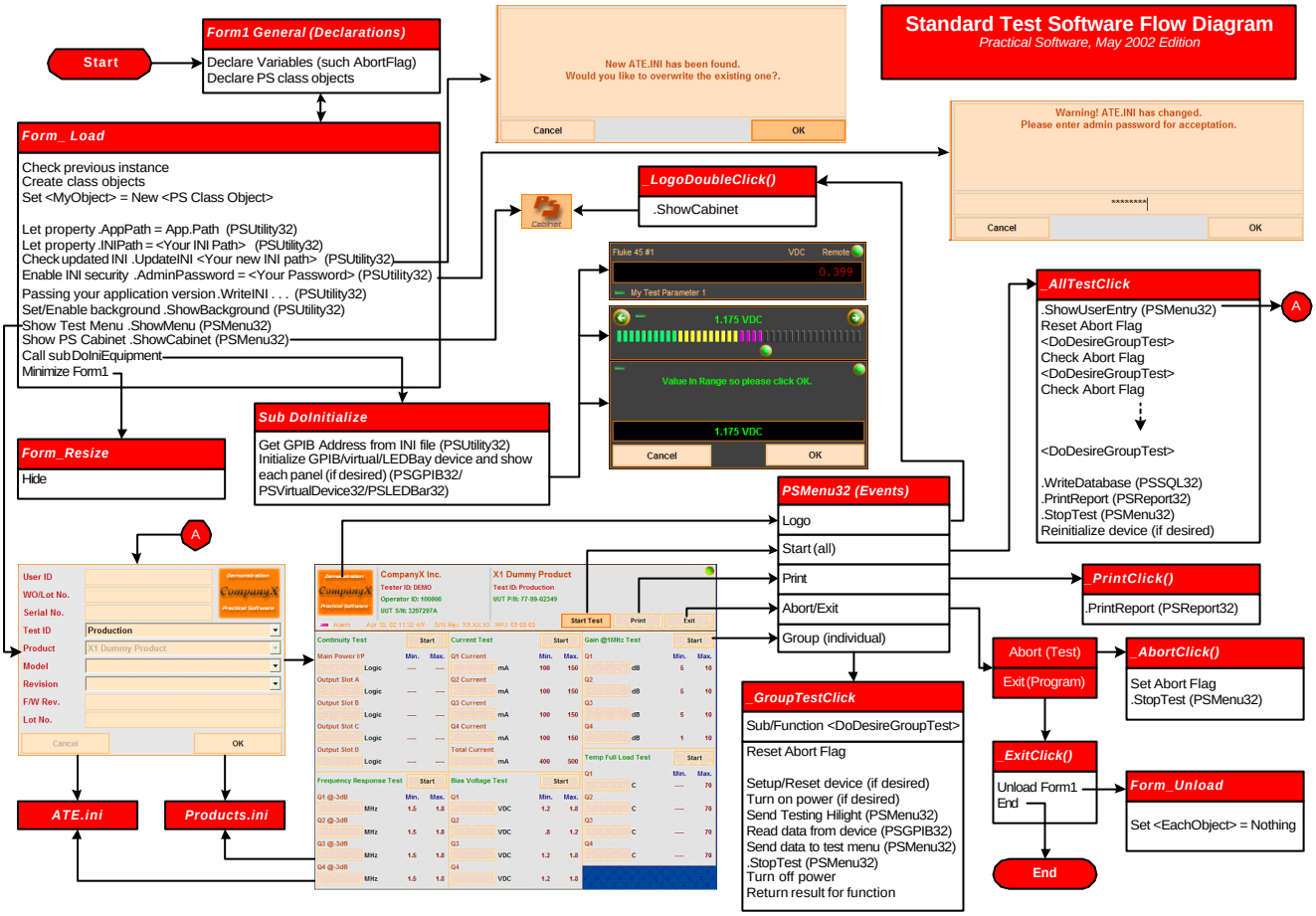

Figure 16 Standard Test Software Flow

Class Library Reference

INI Files

The INI files are an essential part that in conjunction works with the class library so this section will explain in details how to configuring these INI files. Be complete understand how to make, modify these INI files so will give you for the rest task to be easiness. For example, you will create test menus, write database even report printing by couple code lines.

The INI files are normal text file that able to read/write with any text editor program also the format of them are like traditional operating system used such Microsoft Windows 3.x that used WIN.INI and SYSTEM.INI to keep configuration values which useful by itself and other applications. There is simple structure by consist only two key words, Section Name and Key Name as below:

```
;section name must place inside a bracket also the space may include.
;key name must follow by equal sign the space may include.
[Section Name1]
Key Name1=
Key Name2=
Key Name3=

[Section Name2]
Key Name1=
Key Name2=
Key Name3=
```

There are only 2 INI files which designed for class library, ATE.INI and Products.INI which may place into any location in your hard drive. However, you need to tell the class library where they are by give the right path to .INIPath from PSUtility32 class.

ATE.INI is designed for keep the information that concerning ATE machine and Products.INI is concerned by products that to be testing. In addition, Products.INI is support for multiple products configuration that means two or more products able to place in same INI file.

Furthermore, both INI files are not only useful for class library but developer able to use them for specially keep information (see method ReadINI and WriteINI in PSUtility32 class). The following are describes how the class library in conjunction works with both INI files.

ATE.INI

[History]

Version=<Version Number>

Keep the current version of ATE.INI.

PSCheckSum=<Hex Number>

This key name and value will generate from class library. Whenever you set the password to AdminPassword property form PSUtility32 class and use method ShowMenu or ShowUserEntry from PSMenu32 class.

First, the class library will calculate the checksum value from ATE.INI the put it in above key value then every time the test menu on loading, the class library will recalculate the check sum once and compare with current number so if any changed of ATE.INI the checksum number will mismatch.

The class library will show the prompt box in order to let someone enter the password. If the password is correct as you prior set to AdminiPassword property, a new checksum value will overwrite to existing one and allow continuing testing.

[ATE Configuration]

ID=<Tester ID>

Keep the tester ID.

TestID=<Test ID1, TestID2, TestID3,…>

List of test ID that available for this machine also each item much separate by comma. These items will be showing at Test ID drop down list of user entry form and unlimited for number of items.

TestIDDefault=<Default Test ID>

The value must one of items list in TestID key.

TestProduct=<Product1,Product2,Product3,...>

List of products name that available for test also each item much separate by comma. These items will be showing at Test Product drop down list of user entry form and unlimited for number of items.

TestProductDefault=<Default TestProduct>

The value must one of items in TestProduct key.

CompanyLogo=<Picture full path and file name>

Full path and file name of picture (logo) file that will be showing in right corner of user entry form and left side of head menu form. The available picture formats are Bitmaps, GIF, JPEG, Metafiles and Icons. The recommendatory size is 100(W) x 84(H) pixels. The default picture is Practical Software logo.

CheckNetworkDrive=<Logical map network drive>

Class library will try to access this drive so that to make sure the tester is good network connection.

ReportLinePerPage=<Line Number>

Number of lines per page for report printing, the default number is 65

[Database]

UserName=<User Name>

User Name for login to SQL server. You may leave it blank if you use Access database type keep your test data instead.

Password=<Password>

The Password of UserName for login to SQL server (leave blank for no password) also you may leave it blank if you use Access database type keep your test data instead.

DatabaseName=<Database Name>

If selected Access to be your test database type,the DatabaseName value will be Access database filename (.MDB).

If selected SQL Server be your test database type,the DatabaseName value will be SQL database object.

ServerName=<Database Server Name>

SQL database server name. For Access database, please leave blank.

Products.INI

[History]

PSCheckSum=<Hex Number>

This key name and value are generated from class library. Whenever you set the password to AdminPassword property form PSUtility32 class and use method ShowMenu or ShowUserEntry from PSMenu32 class.

First, the class library will calculate the checksum value from Products.INI the put it in above key value then every time the test menu on loading, the class library will recalculate the check sum once and compare with current number so if any changed of Products.INI the checksum number will mismatch.

The class library will show the prompt box in order to let someone enter the password. If the password is corrected as you prior gave to AdminiPassword property, a new checksum value will overwrite to existing one and allow continuing testing.

[<TestProduct>]

The test product section name also must one of items in TestProduct key name in [ATE Configuration] section name in ATE.INI.

Version=<Version Number>

The version number of this product.

TableName=<Database table name>

Table name inside database (see ATE.INI in section [Database]) that class library will be written to.

LocalPN=<Local product part number>

Local product part number that will be writing into database and print out report as well.

LocalAssyPN=<Local product assy number>

Local assy. part number that will be writing into database and print out report as well.

CustomerPN=<Customer part number>

Customer product part number that will be writing into database and print out report as well.

CustomerName=<Customer Name>

This value will show on head menu also writing into database and print out report as well.

Model=<Model1,Model2,Model3,. . >

List of product model that available for selection also each item much separated by comma. These items will show at Model drop down list on user entry form and unlimited for number of items also writing into database and print out report as well.

Revision=<Rev1,Rev2,Rev3,. . .>

Product revision that available for selection also each item much separated by comma. These items will show at Revision drop down list on user entry form and unlimited for number of items also writing into database and print out report as well.

Remark=<Remark1,Remark2,Remark3,Remark4>

Remark captions, each item much separate by comma. These items will show as label caption for operator entering the special information on following text boxes on user entry form. The limited number is 4 items maximum also writing into database and print out report as well.

[<TestProduct> Spec]

This section purposes for keep the test specification by simple format. This will utilizing by various class such as report printing, database writing and test menus.

The format for specification will be:

Num=[DBFieldName],[MinSpec],[MaxSpec],[Target],[Unit],GroupName,ParaName,[Custom Flag]

Note: [is an optional]

And must finish by;

Num=<EOT>

Where:

- **'Num'** is sequence number 1, 2, 3, 4, 5 and so on (limit for 500 maximum)
- **'DBFieldName'** is a unique name that defined in table of database
- **'MinSpec'** is lowest number for acceptation
- **'MaxSpec'** is highest number for acceptation
- **'Target'** is expect number for testing
- **'Unit'** such Amps, Volts, Ohms, Hz , dB and so on
- **'GroupName'** is test group name
- **'ParaName'** is test parameter
- **'CustomFlag'** is custom information (free for use)
- **'<EOT>'** is End Of Test flag (must be in last num)

The below sample will be 2 test groups (Power 51dBm and Power 55dBm) with 3 test parameters (MyPara1, MyPara2 and MyPara3).

```
[Product1 Spec]
1=Gain_51dBm,32.5,,,dB,Power 51dBm,Gain,MyPara1
2=Current_51dBm,20,25,,Amps,Power 51dBm,Current,MyPara2
3=Harmonics_3L_51dBm,,-27.8,,dBc,Power 55dBm,Lower 3rd Harmonics,MyPara3
4=<EOT>
```

Note: In case updating either add or remove group test. They must update to conform to information in PrintGroupOrder and ScreenGroupOrder keys

PrintGroupOrder=<Group1,Group2,Group3, . . .>

Re-Order test group print out. Each item in list should present in above specification lines.

This will pick all parameters of Group1 then print them first then Group2, Group3 and so on. If not specify, the class library will scan all specification lined then perform first found (group) first print.

Example:
PrintGroupOrder=Power 55dBm,Power 51dBm

ScreenGroupOrder=<Group1,Group2,Group3, . . .>

Same as PrintGroupOrder but ScreenGroupOrder will specify that each group test menu to be show on screen by start from left to right then up to down side. In case of this property is not set, the class library will use PrintGroupOrder (if available too) instead.

PSGPIB32 class

Properties

AlwaysOnTop (Boolean)

Set/Return boolean value so that to force a device panel to be top over other forms. It must in conjunction use with either Execute or Show methods. The default value is "True".

```
Dim MyGPIBDevice As PSGPIB32

Private Sub Form_Load()
Set MyGPIBDevice = New PSGPIB32
End Sub

Private Sub Command1_Click()
With MyGPIBDevcie
    .DeviceName = 13
    .TimeOut = T10S
```



```
    .InitialCommand1 = "*CLS"
    .ReadCommand = "AR"
    .ReadInterval = 250
    .AlwaysOnTop = True
    .Execute
End With
End Sub
```

BoardNumber (Long)

Set/Return GPIB board (card) number. There is allow you install up to 10 cards (GPIB0 to GPIB9) in same machine (of course if you have enough slots on main board) also one card is support up to 30 devices. The default value is GPIB0.

```
Dim Fluke45 As PSGPIB32

Private Sub Form_Load()
Set Fluke45 = New PSGPIB32
End Sub

Private Sub Command1_Click()
With Fluke45
    .BoardNumber = GPIB0
    .DeviceName = 16
    .IntialCommand1 = "*RST;RA"
    .ReadCommand = "MEAS?"
    .Execute
End With
End Sub
```

BrandName (String)

Set/Return the device's brand name

```
Dim Keithley2000 As PSGPIB32

Private Sub Form_Load()
Set Keithley2000 = New PSGPIB32
End Sub

Private Sub Command1_Click()
With Keithley2000
    .BrandName =  "Keithley"
    .ModelName = "2000"
    .Show
End With
End Sub
```

CabReference (String)

Set/Return the string value for reference text that put into PS Cabinet during you hides the panel so that helpful re-show from PS Cabinet. The Figure 17 shows the popup menu while right click mouse on PS Cabinet then you will see the CabReference string inside menu.

```
Dim MyGPIBDevice As PSGPIB32

Private Sub Form_Load()
Set MyGPIBDevice = New PSGPIB32
End Sub

Private Sub Command1_Click()
With MyGPIBDevcie
    .DeviceName = 13
```



```
    .TimeOut = T10S
    .InitialCommand1 = "*CLS"
    .ReadCommand = "AR"
    .ReadInterval = 250
    .CabReference = "This is my device"
    .Execute
End With
End Sub
```

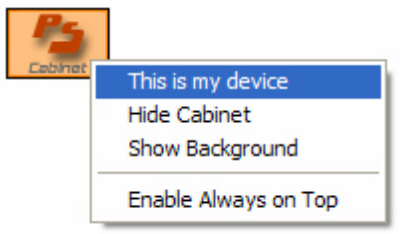

Figure 17 Cabinet Reference

Delimited (String)

Set/Return character string so that to wording identification from stream data that read out from device. The default is an ANSI code 13 (Line Feed).

```
Dim DVM As PSGPIB32

Private Sub Form_Load()
Set DVM = New PSGPIB32
End sub

Private Sub Command1_Click()
With DVM
    .DeviceName = 16
    .InitialCommand1 = "*RST"
    .InitialCommand2 = ":CONF:VOLT:DC"
    .ReadCommand = "READ?"
    .Delimited = vbLf
    .ReadInterval = 250
    .Execute
End With
End Sub
```

DeviceName (Variant)

Set/Return GPIB device name also accepts both string and number. This property need to set before use Execute method.

```
Dim DVM As PSGPIB32
Dim PowerMeter As PSGPIB32
Dim Switcher As PSGPIB32

Private Sub Form_Load()
Set DVM = New PSGPIB32
Set PowerMeter = New PSGPIB32
Set Switcher = New PSGPIB32

DVM.DeviceName = "METER"'Indirect set a GPIB address string that 'configured
via control panel
PowerMeter.DeviceName = 16    'Direct set a GPIB address number that accept
Switcher.DeviceName = "9"     'both integer and string values
```



```
End Sub
```

DigitColor (ColorConstants)

Set/Return a number of VB color constants. The setting color will show on reading window of GPIB device panel. The following code is set to yellow color as show as Figure 18.

```
Dim MyDevice As PSGPIB32

Private Sub Form_Load()
Set MyDevice = New PSGPIB32
End Sub

Private Sub Command1_Click()
With MyDevice
    .BrandName = "Fluke"
    .ModelName = "45"
    .InitCommand1 = "*RST"
    .ReadCommand = "MEAS?"
    .ReadInterval = 250
    .DigitColor = vbYellow
    .PanelPosition = Center
    .PanelOption = "Auto"
    .Unit = "mV"
    .PanelParameter = "My Parameter"
    .PanelStatus = "Remote"
    .Execute
End With
End Sub
```

Figure 18 GPIB Device Panel

DisableErrMessage (Boolean)

Set/Return a value that determines to disable error message box that generate from class library.

```
MyGPIB.DisableErrMessage = True
```

ErrorStatus (Boolean)

Set/Return error status value. 'True' is meaning error has been occurred.

```
If MyGPIB.ErrorStatus Then
    MsgBox Error, vbExclamation, "GPIB Error"
    Call My_Sub_For_Error
End If
```

IDCommand (String)

Set/Return device identification query command, due to some device is not accept the standard command (IEEE-488.2) such '*IDN?'.

However, you must know the particular command so that to put it instead. If not specify, class library will use standard command as above.

```
With MyDevice
    .DeviceName = 16
    .IDCommand = "ID?"
```

```
    .Execute
    Debug.Print .BrandName, .ModelName
End With
```

InitCommand1 to 5 (String)

Set/Return initial GPIB command sequence. They are provided for long and complex command and makes the code to easier and clarifies.

```
With MyDevice
    .InitCommand1 = "VSET 1,0;ISET 1,0"
    .InitCommand2 = "VSET 2,16;ISET 2,2"
    .InitCommand3 = "VSET 3,16;ISET 3,2"
    .InitCommand4 = "VSET 4,8;ISET 4,2"
    .InitCommand5 = "OUT 1,1;OUT 2,1;OUT 3,1;OUT 4,1"
    .WaitForEachCommand = 0.5       'delay for each command was executed
    .Execute
End With
```

These commands will be write to device one by one by start with InitCommand1, 2, 3, 4 and 5 respectively also after Execute method has been done all values will be clear including WaitForEachCommand (reset to zero) and WriteFrimFile properties.

ModelName (String)

Set/Return the device's model name. (see BrandName property)

Offset (Double)

Set/Return the value offset that will add to reading value. The default value is zero.

```
Dim varOutput As Variant
With MyDevice
    .DeviceName = 16
    .InitCommand1 = "*CLS;RA"
    .ReadCommand = "AR"
    .ReadInterval = 200
    .Offset = -180
    .Execute
    varOutput = .Value
    Debug.Print varOutput
End With
```

As above code, assuming the original read value from device is 180 so the return value will be 180 + (-180) = 0

PanelHeight (Long)

Return the height value of device panel.

```
MyPanelHeight = MyMeter.PanelHeight
```

PanelHide (Boolean)

Set/Return a value that determines whether a device panel that is visible or hidden, the default value is 'False'.

```
MyMeter.PanelHide = True
```

PanelLeft (Long)

Set/Return the distance value between the left edge of device panel and left edge of its container (screen). The scale is twip*.

```
With MyDevice
    .DeviceName = 16
    .InitCommand1 = "*RST"
```



```
    .InitCommand2 = ":CONF:VOLT:DC"
    .PanelParameter = "My Parameter"
    .PanelOption = "Auto"
    .PanelStatus = "Remote"
    .PanelTop = 2000
    .PanelLeft = 3000
    .Unit = "VDC"
    .ReadCommand = "READ?"
    .ReadInterval = 250
    .Execute
End With
```

**Twip is a screen-independent unit used to ensure that placement and proportion of screen elements in your screen application are the same on all display systems. A twip is a unit of screen measurement equal to 1/20 of a printer's point. There are approximately 1440 twips to a logical inch or 567 twips to a logical centimeter (the length of a screen item measuring one inch or one centimeter when printed).*

PanelOption (String)

Set/Return the Option caption on device panel.

```
MyMeter.PanelOption = "Auto"
```

Figure 19 Customize your GPIB Device Panel

PanelParameter (String)

Set/Return the Parameter caption on device panel (see PanelOption).

```
MyMeter.PanelParameter = "My Parameter"
```

PanelPosition (Positions)

Set/Return a value specifying the position of device panel. There are 9 positions available as Center, CenterUp, CenterLeft, CenterRight, LeftUp, RightUp, CenterDown, LeftDown and RightDown. For other position, you may use the properties PanelLeft and PanelTop instead.

```
MyMeter.PanelPosition = Center
'Or
MyMeter.PanelPosition = RightUp
'Or
MyMeter.PanelPosition = CenterLeft

'Demostration how to stack the 2 device panels
Dim Keithley2000 As PSGPIB32
Dim Giga_tronics_8542C As PSGPIB32

Private Sub Form_Load()
Set Keithley2000 = New PSGPIB32
Set Giga_tronics_8542C = New PSGPIB32
End Sub

Private Sub Command1_Click()
With Giga_tronics_8542C
    .DeviceName = "13"
```



```
    .InitCommand1 = "*CLS;RA"
    .ReadCommand = "AR"
    .PanelParameter = "Test My Parameter"
    .PanelOption = "A"
    .PanelStatus = "Remote"
    .ReadInterval = 250
    .Unit = "dBm"
    .PanelPosition = Center
    .StatusPosition = CenterDown
    .Execute
End With
With Keithley2000
    .DeviceName = 16
    .InitCommand1 = "*RST"
    .InitCommand2 = ":CONF:VOLT:DC"
    .ReadCommand = "READ?"
    .PanelTop = Giga_tronics_8542C.PanelTop + _
                Giga_tronics_8542C.PanelHeight
    .PanelLeft = Giga_tronics_8542C.PanelLeft
    .PanelParameter = "Test My Parameter"
    .PanelOption = "Auto"
    .PanelStatus = "Remote"
    .ReadInterval = 250
    .Unit = "VDC"
    .Execute
End With
End Sub
```

PanelStatus (String)

Set/Return the Status caption on device panel (see PanelOption property).

```
MyMeter.PanelStatus = "Remote"
```

PanelTop (Long)

Set/Return the distance value between the top edge of device panel and top edge of its container (screen). The scale is twip* (see PanelLeft property).

```
MyMeter.PanelTop = 2000
```

PanelWidth (Long)

Return the width value of device panel.

```
MyPanelWidth = MyMeter.PanelWidth
```

RawValue (String)

Return the raw value that read from device. Normally, raw value will include various information that delimited by a special character.

```
Dim HP_8753C_GPIB As PSGPIB32
Dim strBuffer_1 As String
Dim strBuffer_2 As String

Private Sub Form_Load()
Set HP_8753C_GPIB = New PSGPIB32
End Sub

Private Sub Command1_Click()
With HP_8753C_GPIB
    .InitCommand1 = "CHAN1;FORM4"
    .ReadCommand = "OUTPFORM"
```



```
    .Execute
    strBuffer_1 = .RawValue
    .InitCommand1 = "CHAN2;FORM4"
    .Execute
    strBuffer_2 = .RawValue
End With
End Sub
```

ReadCommand (String)

Set/Return read command that to be access to device. This property is in conjunction works with method Execute.

```
Dim Keithley2000 As PSGPIB32

Private Sub Form_Load()
Keithley2000 = New PSGPIB32
End Sub

Private Sub Command1_Click()
With Keithley2000
    .DeviceName = 16
    .InitCommand1 = "*RST"
    .InitCommand2 = ":CONF:VOLT:DC"
    .ReadCommand = "READ?"
    .PanelPosition = Center
    .PanelParameter = "Test My Parameter"
    .PanelOption = "Auto"
    .PanelStatus = "Remote"
    .ReadInterval = 250
    .Unit = "VDC"
    .Execute
End With
End Sub
```

ReadInterval (Long)

Set/Return period of repeat time (in millisecond) to read data from a device. The default is zero mean one-time read. It must use in conjunction with ReadCommand property and Execute method.

```
MyDevice.ReadInterval = 250          'read every 250 millisecond
```

ScaleFactor (Double)

Set/Return scale factor value, The default is 1.

```
Dim varOutput As Variant
With MyDevice
    .DeviceName = 16
    .InitCommand1 = "*CLS;RA"
    .ReadCommand = "AR"
    .ReadInterval = 200
    .Offset = 100
    .ScaleFactor = 1000
    .Execute
    varOutput = .Value
    Debug.Print varOutput
End With
```

As above code, assuming the reading value from device is 0.01 so the return value will be [(0.01 x 1000) + 100] = 110.

SimulateMode (Boolean)

Set/Return a value that determines whether GPIB device accessing is in simulate mode or not. The default value is False

In case of this property is set to True so the methods Execute or ScanDevice will not real access to GPIB device. This helpful you in order to develop the test program at backend station without GPIB board and device installed. The Figure 20 is shown the information while get in simulate mode. Also the return reading value from class library will be random number.

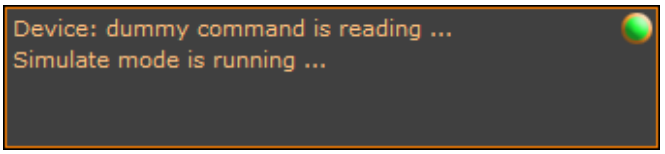

Figure 20 Simulate Mode Message in Status Panel

```
With Fluke45
    .DeviceName = 10
    .SimulateMode = True
    .InitCommand1 = "*RST;RA"
    .InitCommand2 = "RA"
    .InitCommand2 = "AUTO;FORMAT 1;OHMS"
    .InitCommand3 = "AUTO;FORMAT 1;VDC"
    .ReadCommand = "MEAS?"
    .Execute
End With
```

StatusHeight (Long)

Return the height value of status panel.

```
MyPanelHeight = MyMeter.StatusHeight
```

StatusHide (Boolean)

Set/Return a value that determines whether a status panel that is visible or hidden, the default value is False.

The status panel is common use for all GPIB devices so you will able to set these properties such StatusPosition, StatusTop, StatusLeft or StatusHide from any PSGPIB32 class that you have created.

```
MyMeter.StatusHide = True
```

StatusLeft (Long)

Set/Return the distance value between the left edge of status panel and left edge of its container (screen). The scale is twip (see PanelLeft property).

StatusPosition (Positions)

Set/Return a value specifying the position of commom status panel (see PanelPosition).

```
MyMeter.StatusPosition = CenterUp
'Or
MyMeter.StatusPosition = RightUp
'Or
MyMeter.StatusPosition = LeftDown
```

StatusTop (Long)

Set/Return the distance value between the top edge of device panel and top edge of its container (screen). The scale is twip* (see PanelTop property).

StatusWidth (Long)

Return the width value of status panel.

```
MyPanelWidth = MyMeter.StatusWidth
```

TimeOut (GPIB_TimeOut)

Set/Return the timeout period for accessing a GPIB devices. The default value is 13 or 10 second (see below table for available timeout period).

```
MyDevice.TimeOut = 14          'set time out for 30 second
'or
MyDevice.TimeOut = T30S
```

Name	Number	Timeout Period
TNONE	0	Infinite timeout
T10us	1	Timeout of 10 us
T30us	2	Timeout of 30 us
T100us	3	Timeout of 100 us
T300us	4	Timeout of 300 us
T1ms	5	Timeout of 1 ms
T3ms	6	Timeout of 3 ms
T10ms	7	Timeout of 10 ms
T30ms	8	Timeout of 30 ms
T100ms	9	Timeout of 100 ms
T300ms	10	Timeout of 300 ms
T1s	11	Timeout of 1 s
T3s	12	Timeout of 2 s
T10s	13	Timeout of 10 s (default)
T30s	14	Timeout of 30 s
T100s	15	Timeout of 100 s
T300s	16	Timeout of 300 s
T1000s	17	Timeout of 1000 ms (maximum)

Table 1 GPIB device timeout definition

Unit (String)

Set/Return the Unit caption on device panel (see PanelOption property).

```
MyMeter.Unit = "mV"
```

Value (Variant)

Return value from a device after Execute method has been done.

```
Dim varOutput As Variant
With MyDevice
    .DeviceName = 16
    .InitCommand1 = "*CLS;RA"
    .ReadCommand = "AR"
    .ReadInterval = 250
    .Execute
    Do
        DoEvents
        varOutput = .Value
```



```
        Debug.Print varOutput
    Loop
End With
```

WaitForEachCommand (String)

Set/Return the delay time in second.

```
With MyDevice
    .InitCommand1 = "VSET 1,0;ISET 1,0"
    .InitCommand2 = "VSET 2,16;ISET 2,2"
    .InitCommand3 = "VSET 3,16;ISET 3,2"
    .InitCommand4 = "VSET 4,8;ISET 4,2"
    .InitCommand5 = "OUT 1,1;OUT 2,1;OUT 3,1;OUT 4,1"
    .WaitForEachCommand = 0.5       'delay for each command was executed
    .Execute
End With
```

The class library will wait for 500 milisecond after write each line of command set also this property will reset to zero after Execute method has been done.

WriteFromFile (String)

Set/Return path and filename of data that to be writing into GPIB device.

```
With MyDevice
    .DeviceName = 10
    .InitCommand1 = "*RST"
    .InitCommand2 = ":FREQ 2330MHZ"
    .InitCommand3 = "RADIO:ARB:STATE 0"          'Set ARB Off
    .Execute
    .InitCommand1 = "MMEMORY:DELETE:ARB"         'Delete Arb files
    .Execute
    .WriteFromFile = "c:\product\sI_File.wfm"    'Up load I file
    .Execute
    .WriteFromFile = "c:\product\sQ_File.wfm"    'Up load Q file
    .Execute
End With
```

The above code shows how to transfer data from file drive into GPIB device.

Methods

ClearAllCommand

Reset GPIB command properties. After this method is done the Properties as following will be empty:

- InitCommand1, 2, 3, 4 and 5
- WriteFromFile
- ReadCommand
- WaitForEachCommand (will be zero)

```
MyGPIB.ClearAllCommand
```

Done

Unload a GPIB device panel. You must ensure this device (class) is no longer use.

```
GPIBDevice.Done
```

Execute

Starting send the GPIB command lines that set in IniCommand1 to 5 and ReadCommand properties to device.

```
Dim varOutput As Variant
With MyDevice
    .DeviceName = 16
    .InitCommand1 = "*CLS;RA"
    .ReadCommand = "AR"
    .Execute
    varOutput = .Value
End With
```

Also after done this method, all IniCommand 1 to 5 and WriteFromFile properties will be clear as well as WaitForEachCommand will be zero but not effect to ReadCommand.

Hide

Hide the GPIB device panel.

```
MyDevice.Hide
```

ScanDevices

Scanning the GPIB devices that connected to computer and power is on. The class library will return the array variable for all devices found.

```
Dim MyDevicesList As PSGPIB32

Private Sub Form_Load()
Set MyDevicesList = New PSGPIB32
End Sub

Private Sub Command1_Click()
Dim varDevicesList() As Variant
Dim i As Long
With MyDevicesList
    .DisableErrMessage = True
    .StatusPosition = Center
    .ScanDevices varDevicesList()
     For i = 0 To UBound(varDevicesList)
        Debug.Print varDevicesList(i)
     Next i
End With
End Sub
```

The above code will return the output as below (depend on which devices connected on your system).

```
GIGA-TRONICS,8542C,1832601,4.09,ADDRESS 13
KEITHLEY INSTRUMENTS INC.,MODEL 2000,0593036,A02,ADDRESS 16
```

Show

Show the GPIB device panel.

```
MyDevice.Show
```

Events

ValueChanged

Occurrence if reading value form device is changed.

```
Dim WithEvents mFluke45 As PSGPIB32

Private Sub Form_Load()
Set mFluke45 = New PSGPIB32
End Sub
```



```
Private Sub mFluke45_ValueChanged(ByVal Value As Double, ByVal RawValue As _
                                  Variant)
Debug.Print Value
End Sub
```

PSLEDBar32 class

Properties

AlwaysOnTop (Boolean)

Set/Return boolean value so that to force a LED Bar panel to be top over other forms. It must in conjunction use with Show method. The default value is "True".

```
Dim WithEvents myLEDBar As PSLEDBar32

Private Sub Form_Load()
Set myLEDBar = New PSLEDBar32
With myLEDBar
    .AlwaysOnTop = False
    .Position = Center
    .Suffix = "mVDC"
    .ScaleMin = -120
    .ScaleMax = 120
    .Min = -100
    .Max = 100
    .PreciseStep = 0.125
    .DecimalPoint = 3
    .DigitColorPass = vbGreen
    .DigitColorFail = vbRed
    .Show
End With
End Sub
```

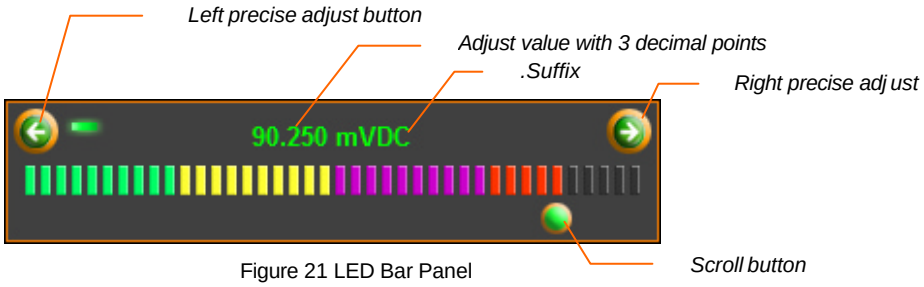

Figure 21 LED Bar Panel

CabReference (String)

Set/Return the string value for reference text that put into PS Cabinet during you hides the panel so that helpful re-show from PS Cabinet (see CabReference property in PSGPIB32)

```
myLEDBar.CabReference = "My LED Bar"
```

DecimalPoint (Long)

Set/Return the number of decimal points of alignment value (see above AlwaysOnTop property).

DigitColorFail (ColorConstants)

Set/Return the failed color of adjust value that out of limit. The Figure 22 is shown the scale setting by .ScaleMin < .Min < .Max < .ScaleMax (see Min and Max properties).

Figure 22 Scale Setting

DigitColorPass (ColorConstants)

Set/Return the passed color of adjust value that within limit (see Min and Max properties).

Height (Long)

Return the height value of LED bar panel.

```
MyLEDBarHeight = MyLEDBar.Height
```

IsPassed (Boolean)

Return 'True' if the Min <= Adjust Value <= Max otherwise will return 'False'.

```
If MyTestMenu.IsPassed then DoSomething
```

Left (Long)

Set/Return the distance value between the left edge of LED bar panel and left edge of its container (screen). The scale is twip (see PanelLeft property).

Max (Double)

Set/Return the maximum limit number. If adjust value go higher than this number then also adjust value will change the color as set in DigitColorFail property (see above AlwaysOnTop property).

In opposite the adjust value will change the color as set in DigitColorPass property.

Min (Double)

Set/Return the minimum limit number. If adjust value go lower than this number then also adjust value will change the color as set in DigitColorFail property (see above AlwaysOnTop property).

In opposite the adjust value will change the color as set in DigitColorPass property.

Position (Positions)

Set/Return a value specifying the position of LED bar panel. There are 9 positions available as Center, CenterUp, CenterLeft, CenterRight, LeftUp, RightUp, CenterDown, LeftDown and RightDown. For other position, you may use the properties Left and Top instead.

```
MyLEDBar.Position = RightUp
```

PreciseStep (Double)

Set/Return the number of precise adjust step. The default value is (ScaleMax – ScaleMin) / 40.

```
MyLEDVar.PreciseStep = 0.125
```

ScaleMax (Double)

Set/Return the maximum number that adjustable (see AlwaysOnTop property).

ScaleMin (Double)

Set/Return the minimum number that adjustable (see AlwaysOnTop property).

Suffix (String)

Set/Return suffix string of adjust value.

```
MyLEDBar.Suffix = "mVDC"
```

Top (Long)

Set/Return the distance value between the top edge of LED bar panel and top edge of its container (screen). The scale is twips (see PanelLeft property).

Value (Double)

Return an adjust value.

```
MyValue = MyLEDBar.Value
```

Width (Long)

Return the width value of LED bar panel.

```
MyLEDBarWidth = MyLEDBar.Width
```

Methods

Done

Unload a LED bar panel. You must ensure this panel (class) is no longer use.

```
MyLEDBar.Done
```

Hide

Hide the LED bar panel.

```
MyLEDBar.Hide
```

SendValue

You also able to send the adjust value from outside so that to overwitre the present one.

```
MyLEDBar.SendValue myValue, myDecimalPoint
```

Show

Show the LED bar panel.

```
MyLEDBar.Show
```

Events

ValueChanged

Occurrence if adjust value is changed.

```
Dim WithEvents mBiasVoltageTune As PSLEDBar32

Private Sub Form_Load()
Set mBiasVoltageTune = New PSLEDBar32
With mBiasVoltageTune
    .ScaleMin = 0
    .ScaleMax = 100
    .Min = .ScaleMin
    .Max = .ScaleMax
    .Show
End With
End Sub
```



```
Private Sub mBiasVoltageTune_ValueChanged(ByVal Value As Double, _
                                          ByVal IsPassed As Boolean)
Debug.Print Value, IsPassed
End Sub
```

PSMenu32 Class

As the earlier, you have learned about how to create test menu. Now on, we will go in details one by one.

The Figure 23 and Figure 24 are samples of user entry form and head menu that class library generate then by use the below information.

;Sample ATE.INI for create a header menu
[History]
Version=1.0.0

[ATE Configuration]
ID=DEMO
TestID=Production,Retest,Rework,Evaluation,Repair
TestIDDefault=Production
TestProduct=Template Product
CheckNetworkDrive=Z:\
CustomerLogo=CompanyX.jpg

;Sample Products.INI
[Template Product]
Version=1.0.0
TableName=Template_Table_Name
CustomerName=CompanyX Inc.
LocalPN=XXXXXXXXXX
LocalAssyPN=XXXXXXXXXX
CustomerPN=XXXXXXXXX
Model=Model A, Model B
Revision=Rev 1,Rev 2,Rev 3
Remark=Remark1,Remark2,Remark3,Remark4

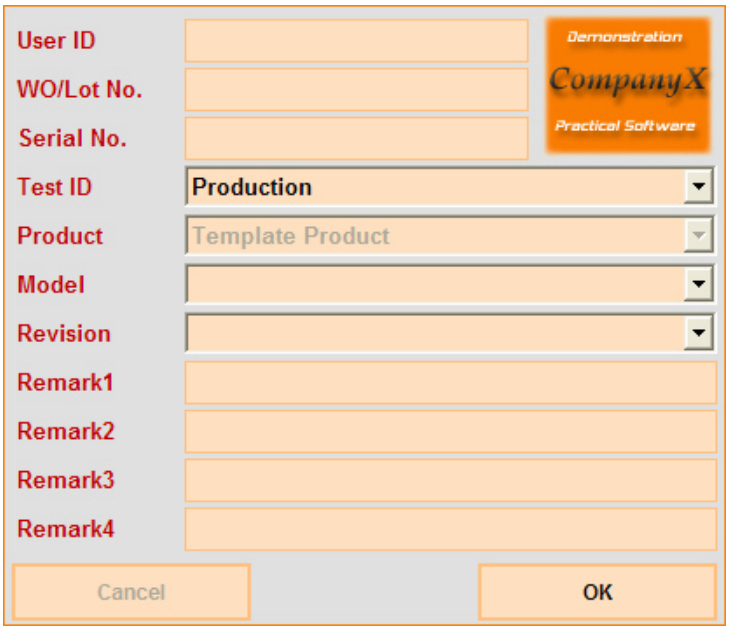

Figure 23 User Entry Form

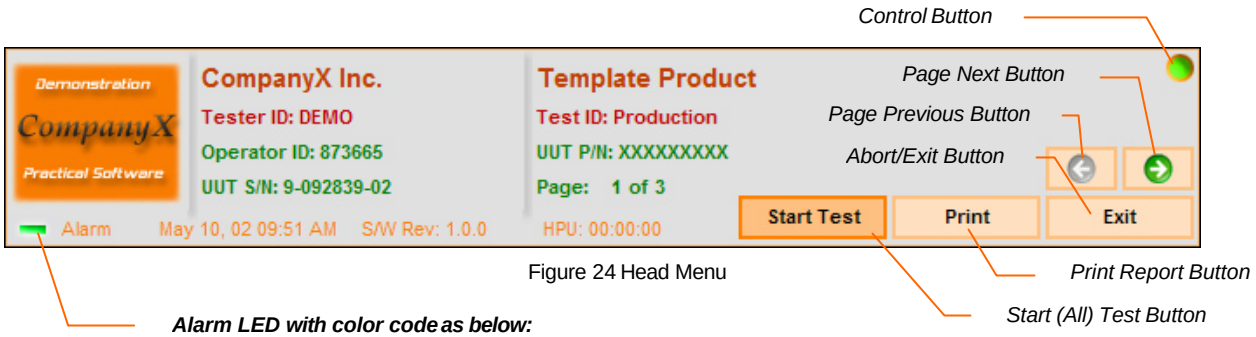

Figure 24 Head Menu

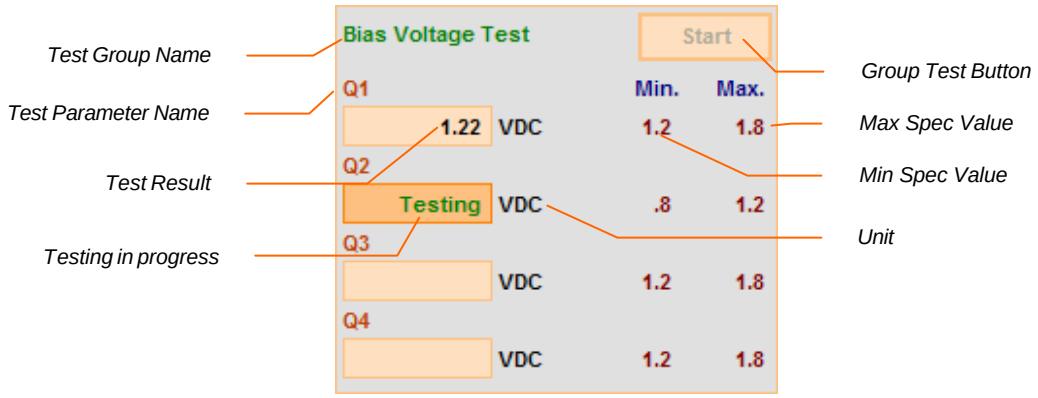

Figure 25 Test Group Menu

In addition, the Figure 25 is shown the test group menu that class library pick information form Products.INI as below.

```
;Sample Products.INI
[X1 Dummy Product Spec]
;Num=[DBFieldName],[MinSpec],[MaxSpec],[Target],[Unit],GroupName,ParaName,
[CustomFlag]
1=DC_Bias_Q1,1.2,1.8,,VDC,Bias Voltage Test,Q1
2=DC_Bias_Q2,.8,1.2,,VDC,Bias Voltage Test,Q2
3=DC_Bias_Q3,1.2,1.8,,VDC,Bias Voltage Test,Q3
4=DC_Bias_Q4,1.2,1.8,,VDC,Bias Voltage Test,Q4
5=<EOT>
```

Properties

ATEID (String)

Return ATE ID in string that keep in ATE.INI

```
myATEID = MyTestMenu.ATEID
```

DisableGroupButton (Boolean)

Set/Return a value whether enable or disable the group test button on test group menu (see Figure 25). The default value is 'False'.

```
MyTestMenu.DisableGroupButton = True
MyTestMenu.ShowMenu
```

EnableTestPrompt (Boolean)

Set/Return a values whether enable or disable the prompt message while click all start test button on group test button. The default value is 'True'.

```
MyTestMenu.EnableTestPrompt = False
MyTestMenu.ShowMenu
```

MenuPosition (MenuPositions)

Set/Return the position of test menu. There are 3 positions available such MenuCenter, MenuLeft and MenuRight. The default position is 'MenuLeft'.

```
MyTestMenu.MenuPosition = MenuRight
MyTestMenu.ShowMenu
```

Model (String)

Return model in string that selected form user entry form. However, it is one of list from key name 'Model' from Products.INI

```
MyTestMenu.ShowUserEntry
MyProductModel = MyTestMenu.Model
```

OpreratorID (String)

Return operator ID in string that entered form user entry form.

```
MyTestMenu.ShowUserEntry
OperatorID = MyTestMenu.OperatorID
```

OverallPassFail (Boolean)

Return a current overall test result.

```
TestResult = MyTestMenu.OverallPassFail
```

Revision (String)

Return revision in string that selected form user entry form. However, it is one of list from key name 'Revision' from Products.INI

```
MyTestMenu.ShowUserEntry
MyProductRevision = MyTestMenu.Revision
```

SerialNumber (String)

Return serial number in string that entered form user entry form.

```
MyTestMenu.ShowUserEntry
SerialNumber = MyTestMenu.SerialNumber
```

StartTime (Date)

Return a test start time that set after operator completely key all data and click OK on user entry form.

```
TestStartTime = MyTestMenu.StartTime
```

TestID (String)

Return test ID in string that selected form user entry form. However, it is one of list from key name 'TestID' from ATE.INI

```
MyTestMenu.ShowUserEntry
MyTestID = MyTestMenu.TestID
```

TestProduct (String)

Return test product in string that selected form user entry form. However, it is one of list from key name 'TestProduct' from A TE.INI

```
MyTestMenu.ShowUserEntry
MyTestProduct = MyTestMenu.TestProduct
```

WorkOrderLotNumber (String)

Return work order or lot number in string that entered form user entry form.

```
MyTestMenu.ShowUserEntry
WOLotNumber = MyTestMenu.WorkOrderLotNumber
```

Methods

ClearResults

Reset all existing test results that shown in test menu.

```
MyTestMenu.ClearAllData
```

GetCustomFlag, GetDBFieldName, GetMaxSpec, GetMinSpec, GetTargetSpec, GetUnit

Return the information and specification in string that defined in Products.INI in [<*Product Name*> Spec] section. Suppose you have the following string in Products.INI.

;Sample Products.INI

```
[MyProductName Spec]
1=DataFieldTestNo1,-20,10,0,dB,Group1,TestNo1,Yes
2=DataFieldTestNo2,1,3, ,Amps,Group2,TestNo2,No
3=<EOT>
```

```
MyTestMenu.ShowUserEntry
Debug.Print MyTestMenu.GetCustomFlag("Group1", "TestNo1")
```

Above will return "Yes"

```
Debug.Print MyTestMenu.GetDBFieldName("Group1", "TestNo1")
```

Above will return "DataFieldTestNo1"

```
Debug.Print MyTestMenu.GetMaxSpec("Group2", "TestNo2")
```

Above will return "3"

```
Debug.Print MyTestMenu.GetMinSpec("Group2", "TestNo2")
```

Above will return "1"

```
Debug.Print MyTestMenu.GetTargetSpec("Group1", "TestNo1")
```

Above will return "0"

```
Debug.Print MyTestMenu.GetUnit("Group1", "TestNo1")
```

Above will return "dB"

HideCabinet

Hide the cabinet.

```
MyTestMenu.HideCabinet
```

HideMenu

Hide the test menu also this will clear all test data results.

```
MyTestMenu.HideCabinet
```

HidePassFail

Hide overall Pass/Fail on head menu.

```
MyTestMenu.HidePassFail
```

SendData

Send the test result to test menu (see Figure 25).

```
With MyTestMenu
    'Make the Testing is in progress (flash word "Testing")
    .SendData "Bias Voltage Test", "Q1", "Testing"
    'If test data is out of spec. then DoSomething
If Not .SendData ("Bias Voltage Test", "Q1", MyDevice.Value, NoMovePage) _ then
DoSomething
End With
```

ShowCabinet

Show the cabinet.

```
MyTestMenu.ShowCabinet
```

ShowMenu

Show the test menu also if first time use this method or you never use ShowUserEntry method. The class library will automatically call ShowUserEntry first then ShowMenu.

```
MyTestMenu.ShowMenu
```

ShowPassFail

Show overall Pass/Fail on head menu.

```
MyTestMenu.ShowPassFail
```

ShowUserEntry

Show user entry form.

```
MyTestMenu.ShowUserEntry
```

StopTest

Let class library know you need to stop the current testing so that class library will enable the button on head menu and test group menu.

```
MyTesrMenu.StopTest
```

Events

AbortClick

Occurrence when abort/exit button on head menu is clicked (see Figure 24) also change the button abort/exit caption to "Exit".

```
Private Sub mTestMenu_AbortClick()
mIsAbortTest = True
mTestMenu.StopTest
End Sub
```

AllTestClick

Occurrence when start (all) test button on head menu is clicked also change the button abort/exit caption to "Abort".

```
Private Sub mTestMenu_AllTestClick(ByVal ProductName As String)
If Not mTestMenu.ShowUserEntry Then
```

```
    mTestMenu.StopTest
    Exit Sub
Else
    'opearator may changed product during do enter test information
    'if single product test in INI file the below line may ignore
    ProductName = mTestMenu.TestProduct
End If
mIsAbortTest = False
DoStartAllTest
mDatabase.WriteDatabase , , msAccess, App.Path & "\TestData.MDB"
mReport.PrintReport
End Sub
```

ExitClick

Occurrence when abort/exit button on head menu is clicked.

```
Private Sub mTestMenu_ExitClick()
Unload Me
End
End Sub
```

GroupTestClick

Occurrence when group test button on test group menu is clicked (see Figure 25) also change the button abort/exit caption to "Abort".

```
Private Sub mTestMenu_GroupTestClick(ByVal ProductName As String, ByVal _
                                     GroupName As String)
mIsAbortTest = False
Select Case ProductName
    Case "Template Product"
        Select Case GroupName
            Case "Group A Test"
                DoGroupATest GroupName, "Set_Ohm"
            Case "Group B Test"
                DoGroupBTest GroupName, "Set_Curr"
            Case "Group C Test"
                DoGroupCTest GroupName
            Case "Group D Test"
                DoGroupDTest GroupName
            Case "Group E Test"
                DoGroupETest GroupName, "Set_Volt"
            Case "Group F Test"
                DoGroupFTest GroupName, "Set_Volt"
            Case "Group G Test"
                DoGroupGTest GroupName
        End Select
    Case "X1 Dummy Product"
        Select Case GroupName
            Case "Continuity Test"
                DoContinuityTest GroupName
            Case "Current Test"
                DoCurrentTest GroupName
            Case "Gain @1MHz Test"
                DoGain1MHzTest GroupName
            Case "Frequency Response Test"
                DoFreqResponseTest GroupName
            Case "Bias Voltage Test"
                DoBiasVoltageTest GroupName
            Case "Temp Full Load Test"
```



```
                DoTempFullLoadTest GroupName
        End Select
End Select
mTestMenu.StopTest
End Sub
```

Hide

Occurrence when test menu go hide.

LogoDoubleClick

Occurrence when logo picture on head menu is double clicked.

PageNextClick

Occurrence when page next button on head menu is clicked.

PagePreviousClick

Occurrence when page previous on head menu is clicked.

PrintClick

Occurrence when print button on head menu is clicked.

PSReport32 Class

Methods

PrintReport

This class provides the easiness by print out the test report in one method.

myObject.*PrintReport [SoftwareVersion]*

Where:

Optional [SoftwareVersion]: A test software version number. Also you may use the method WriteINI in PSUtility32 so that to pass this number to class library via INI file by following sample:

```
MyUtility.WriteINI True, "ATE Configuration", "TestSoftwareVersion", _
                          App.Major & "." & App.Minor & "." & App.Revision
```

The ATE.INI in section [ATE Configuration] is concerned on this method as following:

[ATE Configuration]
ReportLinePerPage=75

```
;sample code for print out report
Dim MyReport As PSReport32

Private Sub Form_Load()
Set MyReport = New PSReport32
End Sub

Private Sub Command1_Click()
MyReport.PrintReport
End Sub
```

Normally, print report method should apply after full test is done otherwise your test information on report may not contain the complete information.

PSSQL32 Class

Methods

WriteDatabase

This class provides the easiness by writing test data results into database in one method.

myObject.WriteDatabase *[ProductName], [SoftwareVersion], [DatabaseType], [DatabaseName]*

Where:

Optional [ProductName]: A test product name, if not specify, class library will use product name that operator is selected from user entry form.

Optional [SoftwareVersion]: A test software version number. Also you may use the method WriteINI in PSUtility32 so that to pass this number to class library via INI file by following sample:

```
MyUtility.WriteINI True, "ATE Configuration", "TestSoftwareVersion", _
                          App.Major & "." & App.Minor & "." & App.Revision
```

Optional [DatabaseType]: Either "Access" or "MSSQL" (Microsoft® SQL Server), default is "Access"

Optional [DatabaseName]: A database name, for Access it must be full filename and path. For MSSQL, it will be database object name. If not specify, the class library will try to get it from ATE.INI

The ATE.INI in section [Database] is concerned on this method as following:

;For ATE.INI that use Access database type.
[Database]
UserName=
Password=
ServerName=
DatabaseName=c:_Test Database\TestData.Mdb

;And table name will keep in Products.INI (use for both Access and MSSQL)
[MyProductName Product]
TableName=My_Table_Name

```
;sample code for write Access database
Dim MyDatabase As PSSQL32

Private Sub Form_Load()
Set MyDatabase = New PSSQL32
End Sub

Private Sub Command1_Click()
MyDatabase.WriteDatabase
End Sub
```

;For ATE.INI that use MSSQL database type.
;Server name may also use IP address instead
ServerName=mySQLServerName
DatabaseName=myTestData
UserName=myUserName
Password=myPassword

```
Dim MyDatabase As PSSQL32

Private Sub Form_Load()
Set MyDatabase = New PSSQL32
End Sub

Private Sub Command1_Click()
myDatabase.WriteDatabase ,  ,  msSQLServer
```



```
End Sub
```

Normally, write database method should apply after full test is done otherwise your test record data may not contain the complete information. If error occurs, class library will keep the data into a local drive in text format (SQL statement) at location \<SystemRoot>\PSDATABAK\<YourTestProductName>\ and also class library will try to transfer (write) it to database once if the next time writing is succeeded.

For using Access database, you will not to create database and table. The class library will do both for you if it not found in specified location but class library will use append data instead if both database and table are found in that location. Thus, if you have modified the testing or INI file such change field name, add a test parameter or remove a test parameter. You must consider that will affect the current database so you may need to modify the database manually or backup your database or table then remove it from specified location.

However, due to the security of Microsoft® SQL Server, class library will not support automate create database and table so you need to create it manually or first use Access type then transfer it into SQL Server anyway this may not complete duplication of database and table structure so you may need to correct such primary key, index and security. Please consult with you relevant person who is in charge for Information Technology (IT) in your organization.

PSUtility32 Class

This class library provides for miscellaneous functions such INI security, delay time, read/write the INI files, message is PS style and so on. This helpful development due to support the common functions that not included in other class.

Properties

AdminPassword (String)

Set/Return the password that to enable the INI security check so if you set the AdminPassword property. The class library will automate validate the INI files while test menu loading. It found the changed of either ATE.INI or Products.INI the class library will show the prompt box (see Figure 33) so that to let enter the correct password and do not allow to continue testing.

Leave this property bank if you do not want enable this feature.

```
Dim WithEvents MyTestMenu As PSMenu32
Dim MyUtility As PSUtility32

Private Sub Form_Load()
Set MyTestMenu = New PSMenu32
Set MyUtility = New PSUtility32
With MyUtility
    .AppPath = App.Path
    .INIPath = .AppPath
    .UpdateINI "C:\_New INI"
    .AdminPassword = "password"
    .WriteINI True, "ATE Configuration", "TestSoftwareVersion", _
                    App.Major & "." & App.Minor & "." & App.Revision
    .BackgroundColor = &H404040
    .ShowBackground
End With
With MyTestMenu
    .ShowMenu
    .ShowCabinet
End With
End Sub
```


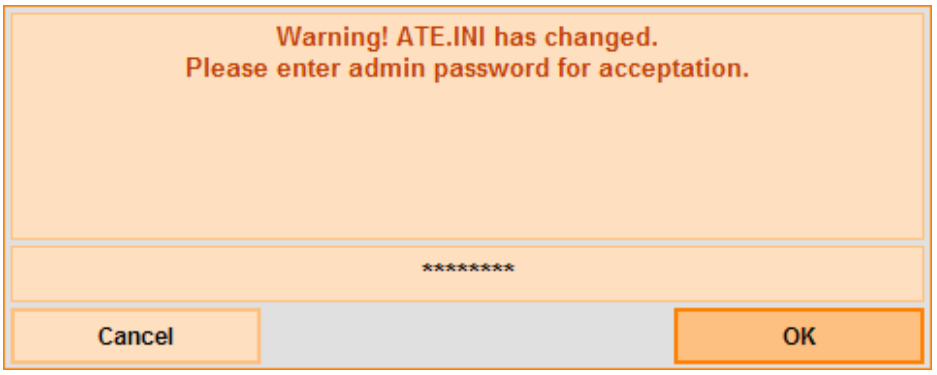

Figure 26 INI 's security check prompt box

Once, the correct password is entered the new check sum value will re-calculate and put it into ATE.INI or Products.INI file in key name "PSChecksum".

AppPath (String)

Set/Return the location that you put a test executive (.EXE) program.

```
MyUtility.AppPath = App.Path
```

BackgroundColor (ColorConstants)

Set/Return a background color. The default is vbBlack.

```
MyUtility.BackgroundColor = &H404040
MyUtility.ShowBackground
```

INIPath (String)

Set/Return class library know the location that keep your desire INI files. Also this property should set before apply UpdateINI, ShowMenu , ShowUserEntry or others method that need to read information from INI files.

```
MyUtility.INIPath = App.Path
```

Methods

Delay

Delay time in second.

```
MyUtility.Delay 1       'delay 1 second
```

GetDLLVersion

Return class library *(PSTestDev.DLL)* version string.

```
Debug.Print MyUtility.GetDLLVersion
```

GetINIVersion

Return either INI version of ATE.INI or Products.INI that keep in section name "[History]" and key name "Version". However, for Products.INI, the product name must be providing due to more than one products may contain in one Products.INI file.

```
Debug.Print MyUtility.GetINIVersion("myProductName")  'from Products.INI
Debug.Print MyUtility.GetINIVersion                   'from ATE.INI
```

HideBackground

Hide background.

```
MyUtility.HideBackground
```


InfoBox

Show/Receive the message/input box of PS style.
MyUtility.InfoBox *([PicturePathOrMessage], [Buttons], [ShowInputBox], [MsgReturn], [LockMessage], _ [Position], [NoModal], [PasswordChar]) As VbMsgBoxResult*
Where:

Optional [PicturePathOrMessage]: Either picture filename or message to be show. If class library detected that is not valid picture filename, class library will assume this is a message.

Optional [Buttons]: A message box button variables, default is vbOKOnly

Optional [ShowInputBox]: A boolean whether show input box or not, default is "False"

Optional [MsgReturn]: If a ShowInputBox is enabled so you need to provide a variant variable so that to receive in return data.

Optional [LockInputBox]: Lock the input box so that not allow editing, default is "True"

Optional [Position]: Position of infobox. There are 9 positions available as Center, CenterUp, CenterLeft, CenterRight, LeftUp, RightUp, CenterDown, LeftDown and RightDown. The default is "Center".

Optioanl [NoModal]: A boolean whether show infobox in modal or non modal, default is "False".

Optional [PasswordChar]: Set the password character on input box, default is blank (non password character)

```
Dim strMessage As String
Debug.Print Tools.InfoBox("Test Information", vbAbortRetryIgnore, , , Centerup)
Debug.Print Tools.InfoBox("Enter any message", vbOKCancel, True, strMessage, _
                          False)
Debug.Print strMessage
Debug.Print Tools.InfoBox(App.Path & "\MyPicture.jpg", vbOKOnly, False, , _
                          Centerup)
Debug.Print Tools.InfoBox(App.Path & "\MyPicture.jpg ", vbYesNo, True, _
                          "Is this my picture?")
```

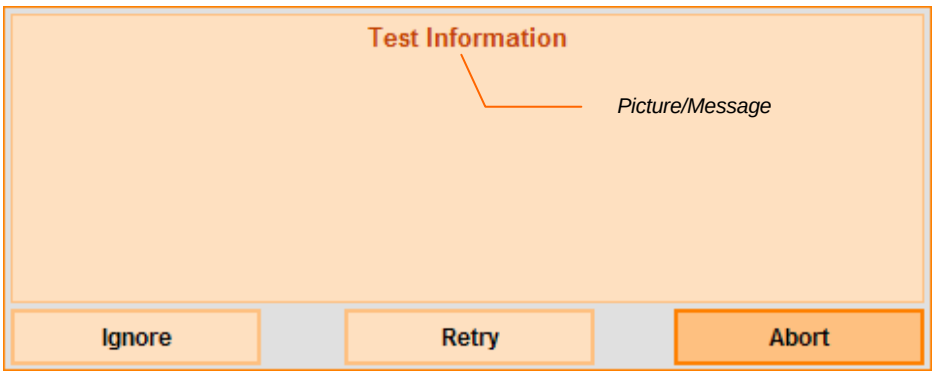

Figure 27 InfoBox with message but without input box

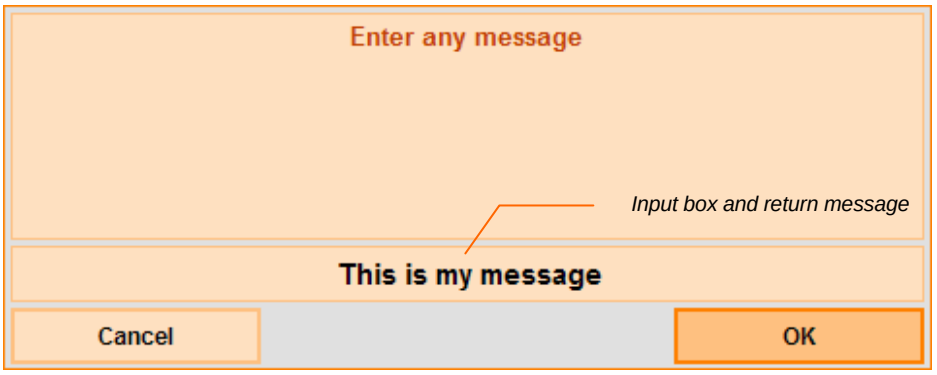

Figure 28 InfoBox with message and input box

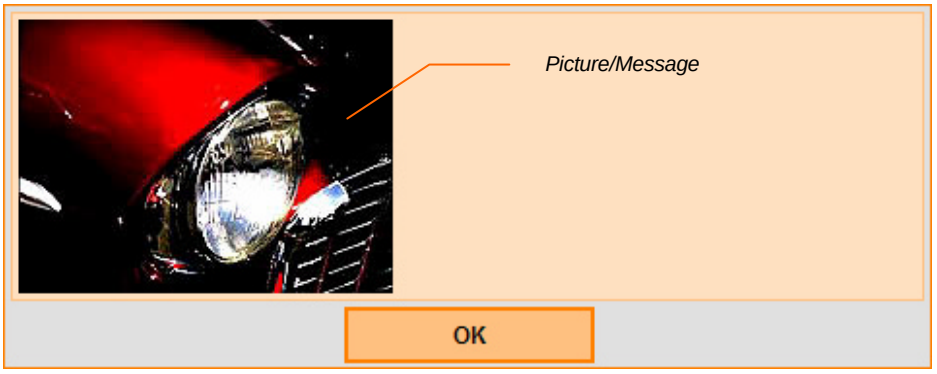

Figure 29 InfoBox with picture but without input box

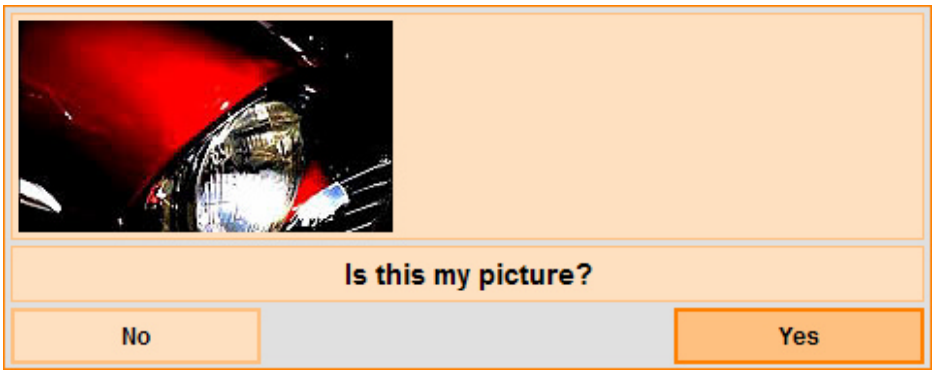

Figure 30 InfoBox with picture and input box

OpenINI

Open an INI file by program Notepad.exe.

```
'This for open ATE.INI
MyUyility.OpenINI True

'This for open products.ini
MyUyility.OpenINI False
```

ReadINI

Read information from either ATE.INI or Products.INI. Assuming, you have the information of INI as below:

```
;sample ate.ini
[GPIB Address]
PSUPP1=4
```



```
PSUPP2=5
DVM1=10
DVM2=11
DVM3=12
DVM4=13

;sample products.ini
[Product XYZ]
Frequency1_MHz=1845
Frequency2_MHz=1845.1
TestPower1_dBm=51
TestPower2_dBm=45
```

To get data form ate.ini at section name is [GPIB Address] and key name is PSUPP1, the following code will return number 4.

```
Debug.Print MyUtility.ReadINI(True, "GPIB Address", "PSUPP1")
```

To get data form product.ini at section name is [Product XYZ] and key name is Frequency1_MHz, the following code will return number 1845.

```
Debug.Print MyUtility.ReadINI(False, "Product XYZ", "Frequency1_MHz")
```

SendStatusMessage

Sending the message to status panel.

MyObject.SendStatusMessage *Message, [LineNumber], [ClearAllMessage]*

Where:

Message: the message string that to be send.

Optional [LineNumber]: a line number that to place the message. There are 4 lines (0 – 3) available. The default is zero.

Optional [ClearAllMessage]: clear all message lines before place the new one. The default is False.

The following code will send the 4 message lines to status panel as Figure 31

```
mUtility.SendStatusMessage "This is Line 0", 0
mUtility.SendStatusMessage "This is Line 1", 1
mUtility.SendStatusMessage "This is Line 2", 2
mUtility.SendStatusMessage "This is Line 3", 3
```

Figure 31 Sent messages to status panel

However, the status panel also use by PSGPIB32 class so the message may be overwritten by that class.

ShowAbout

Show PS about form as Figure 32

```
mutility.ShowAbout
```


Figure 32 About form

In addition, if you wish to register class library into your system you can do so by click on License button.

ShowBackground

Show background.

```
MyUtility.ShowBackground
```

UpdateINI

Let class library know where is your new INI files then if class library found newer one (check file date/time not version) it will be automate updating by change the existing file name. Normally new update files should keep in share network drive so that you will perform update INI file from center.

The prompt box as Figure 33 will show after class library has detected the newer one form specified location. If confirm update so click OK button, the existing one will rename to another one such ATE XXX.INI or ProdXXX.INI where XXX is number between 000 to 999 then the new one will overwrite the existing one.

```
myUtility.UpdateINI "z:\_New INI"
```

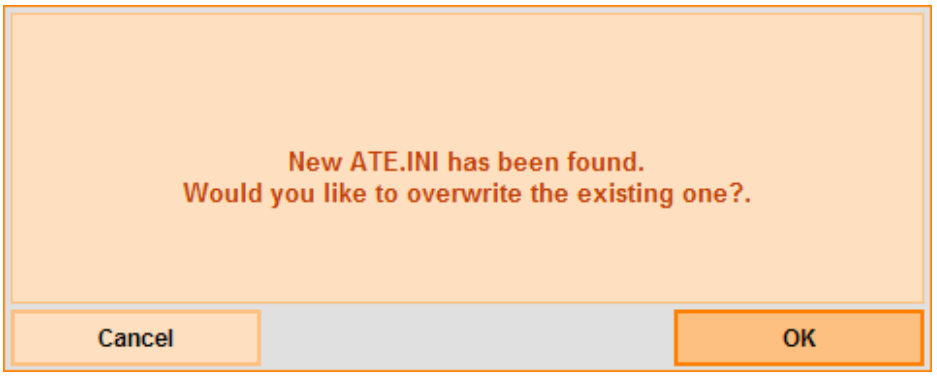

Figure 33 INI Update Prompt Box

However, if you have enabled the INI security check by set the password for AdminPassword property in your application. You must ensure that the newer INI is already updating the checksum (see how to update checksum from AdminPassword property) by class library, before you put it into specified location otherwise the target test station will unable to run your application unless the correct password is entered.

WriteINI

Write information to either ATE.INI or Products.INI. If there are missing either or both of section and key name, class library will create new one automatically.

```
'write data to ate.ini
MyUtility.WriteINI True, "Cal Data", "XParameter", "999.99"

'write data to products.ini
MyUtility.WriteINI False, "Test Product XYZ", "ResultX", "PASS"
```

The results of above will be following:

```
;ATE.INI
[Cal Data]
XParameter=999.99

;Products.INI
[Test Product XYZ]
ResultX=PASS
```

Caution: if you enable INI security check after you use this method for write anything then the class library will be prompt to enter the password.

PSVirtualDevice32 Class

This class object is purposed for Non-GPIB device, for instant, you may have some device that connect via serial port, parallel port or another interface port and you want they work like PSGPIB32 class.

However, due to it not real device so the read value must send from outside device or your sub procedure. The following code shows sending the random value to virtual device also if that sent value is in spec (Min <= Value <= Max) the virtual device panel will be Figure 34 and Figure 35 if it out of spec.

```
Dim MyVirtualdevice As PSVirtualDevice32
Dim myUtility As PSUtility32

Private Sub Command1_Click()
Set MyVirtualdevice = New PSVirtualDevice32
Set myUtility = New PSUtility32
With MyVirtualdevice
    .AlwaysOnTop = False
    .CabReference = "My Virtual Device"
    .Caption = "This is my caption!"
    .DigitColorFail = vbMagenta
    .DigitColorPass = vbWhite
    .Max = 70
    .Min = 30
    .Position = Center
    .Suffix = "VDC"
    .Show
    Do
        DoEvents
        .SendValue Rnd * 100, 3
        myUtility.Delay 0.5
        Debug.Print .Value & .IsPassed
    Loop
End With
End Sub
```


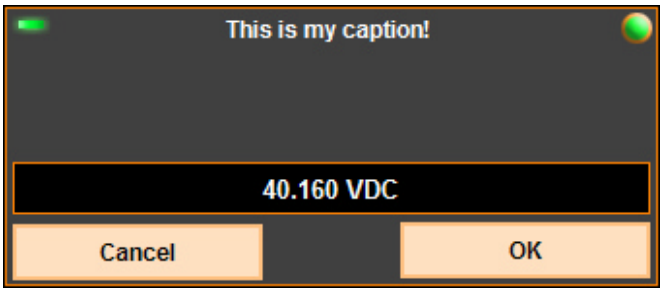

Figure 34 Virtual device panel with sent data is in spec (white color).

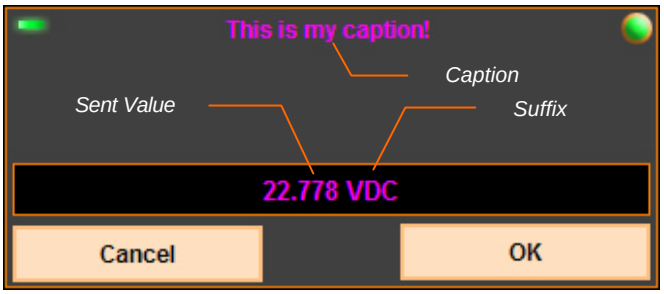

Figure 35 Virtual device panel with sent data is out of spec (magenta color).

Properties

AlwaysOnTop (Boolean)

Set/Return boolean value so that to force a virtual device panel to be top over others form. The default value is "True".

```
MyVirtualDevice.AlwaysOnTop = False
MyVirtualDevice.Show
```

CabReference (String)

Set/Return the string value for reference text that put into PS Cabinet during you hides the panel so that helpful re-show from PS Cabinet (see CabReference property in PSGPIB32)

```
MyVirtualDevice.CabReference = "My Virtual Device"
```

Caption (String)

Set/Return the string value so that sending to virtual device panel (see Figure 34).

```
MyVirtualDevice.Caption = "This is my caption!"
MyVirtualDevice.Show
```

DigitColorFail (ColorConstants)

Set/Return the failed color of caption, sent value and suffix that out of spec (see Figure 34).

DigitColorPass (ColorConstants)

Set/Return the passed color of caption, sent value and suffix that out of spec (see Figure 35).

Height (Long)

Return the height value of virtual device panel.

```
MyVirtualDeviceHeight = MyVirtualDevice.Height
```

IsPassed (Boolean)

Return 'True' if the Min <= Sent Value <= Max otherwise will return 'False'.

```
If MyVirtualDevice.IsPassed then DoSomething
```

Left (Long)

Set/Return the distance value between the left edge of virtual device panel and left edge of its container (screen). The scale is twip.

Max (Double)

Set/Return the maximum limit number. If sent value goes higher than this number then also suffix and caption will change the color as set in DigitColorFail property.

In opposite the will change the color as set in DigitColorPass property.

Min (Double)

Set/Return the minimum limit number. If sent value goes lower than this number then also suffix and caption will change the color as set in DigitColorFail property.

In opposite the will change the color as set in DigitColorPass property.

Position (Positions)

Set/Return a value specifying the position of virtual device panel. There are 9 positions available as Center, CenterUp, CenterLeft, CenterRight, LeftUp, RightUp, CenterDown, LeftDown and RightDown. For other position, you may use the properties Left and Top instead.

```
MyVirtualDevice.Position = RightUp
```

Suffix (String)

Set/Return suffix string of sent value.

```
MyVirtualDevice.Suffix = "VDC"
```

Top (Long)

Set/Return the distance value between the top edge of virtual device panel and top edge of its container (screen). The scale is twip.

Value (Double)

Return a sent value.

```
MyValue = MyVirtualDevice.Value
```

Width (Long)

Return the width value of virtual device panel.

```
MyVirtualDeviceWidth = MyVirtualDevice.Width
```

Methods

Done

Unload a virtual device panel. You must ensure this panel (class) is no longer use.

```
MyVirtualDevice.Done
```

Hide

Hide the virtual device panel.

```
MyVirtualDevice.Hide
```

SendValue

You also able to send the adjust value from outside so that to overwrite the present one.

```
MyVirtualDevice.SendValue myValue, myDecimalPoint
```

Show

Show the LED bar panel.

```
MyVirtualDevice.Show
```

Events

CancelClick

Occurrence if cancel button is clicked.

```
Dim WithEvents MyVirtualdevice As PSVirtualDevice32

Private Sub MyVirtualdevice_CancelClick(ByVal SentValue As Double, ByVal _
                                        IsPassed As Boolean)
Debug.Print SentValue, IsPassed
End Sub
```

OKClick

Occurrence if OK button is clicked.

```
Private Sub MyVirtualdevice_OKClick(ByVal SentValue As Double, ByVal IsPassed _
                                    As Boolean)
Debug.Print SentValue, IsPassed
End Sub
```

Table of Figures

