

# **Tektronix Logic Analyzer Family**

## **Pattern Generator Programmatic Interface (PPI) Manual**

Copyright © Tektronix, Inc. All rights reserved. Licensed software products are owned by Tektronix or its suppliers and are protected by United States copyright laws and international treaty provisions.

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph ©(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, or subparagraphs ©(1) and (2) of the Commercial Computer Software – Restricted Rights clause at FAR 52.227-19, as applicable.

Tektronix products are covered by U.S. and foreign patents, issued and pending. Information in this publication supercedes that in all previously published material. Specifications and price change privileges reserved.

Tektronix, Inc., 14200 SW Karl Braun Drive, Beaverton, OR 97077

Tektronix and Tek are registered trademarks of Tektronix, Inc.

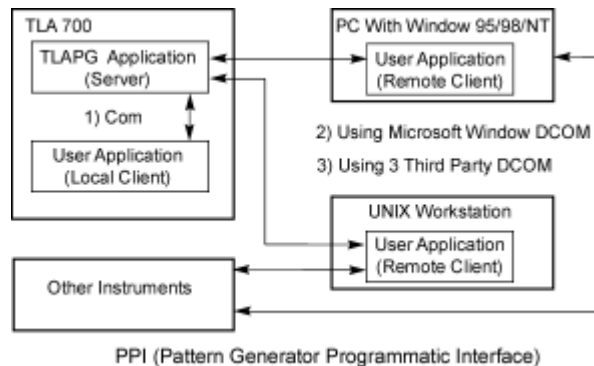
# Table of Contents

<b>Tektronix Logic Analyzer Family</b>	<b>1</b>
<b>Pattern Generator Programmatic Interface (PPI) Manual</b>	<b>1</b>
<b>Table of Contents</b>	<b>3</b>
Introduction .....	5
General Characteristics .....	5
Deliverables .....	7
Setting up PPI.....	7
Objects and Interfaces .....	7
<b>Setting up PPI</b>	<b>8</b>
Client application on the TLA 700 .....	8
Client application across the network .....	8
Share level access .....	8
User level access .....	9
Client machine using Microsoft Windows NT .....	9
Client machine using Microsoft Windows 95 .....	9
Share-level access .....	10
User-level access .....	10
Client machine using Microsoft Windows 98 .....	11
Share-level access .....	11
User-level access .....	11
Client machine using other platforms .....	12
Registering/Unregistering PPI.....	12
Server side Registration/UnRegistration .....	12
Other Issues with DCOM.....	13
Connecting to the Tektronix Pattern Generator Application (Server) .....	14
Disconnecting from the Tektronix Pattern Generator Server.....	14
Samples .....	14
Sample Visual Basic Client (Dispatch) .....	15
Sample Visual Basic Client (Vtable) .....	16
Other Samples .....	17
Errors .....	17
Messages.....	17
Slot Numbers and Expansion Mainframes .....	17
<b>Reference</b>	<b>18</b>
Quick Reference .....	18
Application Object.....	18
System Object .....	18
Module Object .....	19
Listing of Methods.....	20
IPGApplication::GetSystem .....	20
IPGSystem::GetNumModuleSlots .....	21
IPGSystem::GetFirstModuleSlot .....	22
IPGSystem::GetSWVersion .....	23
IPGSystem::GetDiagnosticsStatus.....	24
IPGSystem::GetModulePropertiesBySlot.....	25
IPGSystem::GetModuleTypeBySlot .....	27
IPGSystem::GetModuleBySlot .....	29
IPGSystem::GetModuleByName.....	31

IPGSystem::GetModuleNames .....	32
IPGSystem::LoadSystem .....	33
IPGSystem::SaveSystem .....	35
IPGSystem::Run.....	36
IPGSystem::Stop .....	37
IPGSystem::GetRunStatus.....	38
IPGModule::LoadModule.....	39
IPGModule::SaveModule .....	40
IPGModule::Import .....	42
IPGModule::Export .....	44
IPGApplication::ShowWindow.....	46
IPGModule::SetClockMode .....	47
IPGModule::SetClockPeriod.....	48
IPGModule::SetClockThreshold.....	49
IPGModule::SetClockPolarity.....	50
IPGModule::SetOutputLevel.....	51
IPGModule::GetGroupNames .....	52
IPGModule::GetGroupSize.....	53
IPGModule::GetProbeChannelNames .....	54
<b>Miscellaneous Topics</b> .....	<b>55</b>
Tektronix TLA Data Exchange Format .....	55
File Format: .....	55
Header Info Syntax:.....	56
Group Details Syntax: .....	56
Rules: .....	56

# Introduction

The Tektronix Pattern Generator Programmatic Interface (PPI) is based on Microsoft's Component Object Model (COM). It gives the Tektronix Pattern Generator the ability to be controlled from a separate user program running on the Tektronix Pattern Generator or on a remote host. PPI provides the ability to control the pattern generator modules to be controlled by third party applications. The following diagram shows the different ways this can be achieved.



The Tektronix Pattern Generator application is called the **server** and the user program is called the **client**.

- Case 1 shows the user program is running on the Tektronix Pattern Generator and communicates with the Tektronix Pattern Generator application using Microsoft COM.
- Case 2 shows the user program running on another PC and communicating with the Tektronix Pattern Generator via Microsoft DCOM (Distributed COM).
- Case 3 shows the user program is running on a UNIX workstation and communicates with the Tektronix Pattern Generator application via DCOM provided by a third party vendor. In either of these cases, the user program may talk to other instruments using whatever means required.

The user program may be written in any language or programming environment that supports COM. Some examples are Visual C++ and Visual Basic.

## General Characteristics

Some general characteristics of the programmatic interface are as follows:

- All of the exported server interfaces are *dual* interfaces (they support static and dynamic binding).
- The application must be fully initialized before a client attempts to connect to it. This includes dismissing any diagnostic errors that occur at startup time.
- If a client attempts to connect to the application before it is fully initialized, it will receive an error indicating result in an Error "access is denied".
- Local clients running on the TLA 700 will connect to an existing instance of the server, if there is one. If the server is not already running, it will be launched automatically.

Because of restrictions imposed by Microsoft Windows 98, remote clients can't launch the server automatically. The server needs to be explicitly started to initiate COM connection.

- When a client connects to the Tektronix Pattern Generator server application, the main window of the server application will be visible.

Remote clients can hide the server's main window via PPI. If the window is visible, users can directly interact with the Tektronix Pattern Generator server application. The main window status will have indicator to shown that a client is connected.

- The Tektronix Pattern Generator server application will not be terminated at the end of a client connection. The server window is always made visible when all clients have disconnected.
- PPI will operate within the main thread of the application.

# Deliverables

You will be provided with the following:

- Tektronix Pattern Generator executable that exports COM interfaces
- PPI documentation in online help
- Type library
- Header files for interface and error codes
- Auxiliary scripts/programs/instructions to set up DCOM on a client machine
- Sample client applications using Visual C++, Visual Basic and perhaps other languages.

## Setting up PPI

**Tektronix Pattern Generator Server.** A separate installation program is not necessary to set up PPI on the TLA 700.

The Tektronix PG application installation program will perform the necessary setup install and configure the PPI.

**Remote Clients.** There will be a separate installation program to set up PPI on remote client machines running Microsoft Windows NT/95/98 clients.

If customers are running clients on Microsoft Windows 95 machines, they must install DCOM for Microsoft Windows 95 on their client machine. Please refer to Appendix A on instructions on setting up remote Windows 95 clients.

## Objects and Interfaces

The programmatic interface for the Tektronix Pattern Generator consists of three kinds of objects, Application, System, and Module.

**Application.** The user creates an Application object to initially connect to the application and to subsequently obtain a reference to a System object. The Application object exports a single interface called IPGApplication.

**System.** The System object provides methods for configuration, run control and save and load operations. Every client must obtain a reference to a *System* object before they can obtain references to module objects. The *System* object exports a single interface called IPGSystem.

**Module.** The Module object provides methods for module configuration, and obtaining PG statistics. Module object export a single interface called IPGModule.

Unless otherwise specified, all methods are synchronous and wait for the completion of the operation before returning.

# Setting up PPI

This document describes the steps you need to take to set up PPI.

We will use *install directory* to refer to the directory where PPI client has been installed on your client machine. This directory is C:\Program Files\ Tektronix Pattern Generator by default.

The type library to be used with PPI is Tlapg.tlb. After you have finished the following setup procedure, this file will be located in C:\Program Files\ Tektronix Pattern Generator \System\PPI on the TLA 700 and in *install directory*\System\PPI on your PPI client machine.

## Client application on the TLA 700

No special setup is required if the Tektronix Pattern Generator application has already been installed.

Do the following to demonstrate PPI:

1. Start the Tektronix Pattern Generator application on the TLA 700.
2. Run

C:\Program Files\ Tektronix Pattern Generator \Samples\PPI Samples\Vc++\test client\testclient.exe.

Click the Connect button to see if the client can connect to the Tektronix Pattern Generator.

## Client application across the network

Do the following steps:

1. Install and configure TCP/IP.
2. You may choose to have share-level or user-level access to the TLA 700 as provided by Microsoft Windows 98. This is done in the Control Panel>Network>Access Control page.

Share-level access allows a password to be assigned to each shared resource. User-level access allows a group of users to have access to each shared resource. For Microsoft Windows 98 only networks, share-level access is the only option.

Choose between share-level access and user-level access and perform the following steps:

**NOTE:** For PPI to work with share-level access, authentication is turned off and any COM client can call into a COM server running on the Tektronix Pattern Generator.

### Share level access

Do the following steps:

1. In Control Panel>Network>Access Control, choose share-level access.
2. Re-boot the machine.
3. Double-click  
C:\Program Files\Tektronix Pattern Generator\System\PPI\Share Level Access Server.reg
4. Re-boot the machine.



## User level access

Do the following steps:

1. In Control Panel>Network>Access Control, choose user-level access control and enter the name of the domain that will be used to validate user access.
2. Re-boot the machine.
3. Double-click C:\Program Files\Tektronix Pattern Generator\System\PPI\User Level Access Server.reg
4. Re-boot the machine.

Start the Tektronix Pattern Generator application on the TLA.

You can switch between user-level and share-level access later by redoing the procedure from Step 3 onwards.

## Client machine using Microsoft Windows NT

**Note:** The user requires administrative privileges to perform this setup.

1. Install and configure TCP/IP.
2. Run the Tektronix PPI Client install program supplied with the Tektronix Pattern Generator.
3. Depending on the type of access control you chose for the Tektronix Pattern Generator, double-click *install directory\System\PPI\Share Level Access Client.reg* or *User Level Access Client.reg*
4. Re-boot the client machine.
5. Run *dcomcnfg*.
6. Double-click Tlapg in the Applications page.
7. In the Location page, check the *Run application on the following computer* box. Enter the name of the TLA 700 machine in the edit field.

To verify that setup is complete:

1. Run *install directory\Samples\PPI Samples\Vc++\test client\testclient.exe* on the client machine.
2. Click the Connect button to see if the client can connect to the TLAPG. (The first time you connect it may take a few minutes.)

## Client machine using Microsoft Windows 95

Do the following steps:

1. Install and configure TCP/IP.
2. Run the Tektronix PPI Client install program supplied with the Tektronix Pattern Generator.
3. Download and install the following from Microsoft's web site. Re-boot after each installation.

<http://www.microsoft.com/com/dcom95/download-f.htm>:

- Distributed COM for Microsoft Windows 95 (DCOM95)

- Dcomcnfg (DCOM configuration utility)

The version of DCOM for Microsoft Windows 95 that was tested with PPI was 1.1

Dcomcnfg will run only if user-level access is enabled. See the following step.

You must use share-level or user-level access as chosen for the Tektronix Pattern Generator. This is done in the Control Panel>Network>Access Control page.

## Share-level access

1. In Control Panel>Network>Access Control, choose share-level access.
2. Re-boot the machine.
3. Double-click *install directory*\System\PPI\Share Level Access Client.reg
4. Re-boot the machine.

## User-level access

1. In Control Panel>Network>Access Control, choose user-level access control and enter the name of the domain that will be used to validate user access.
2. Re-boot the machine.
3. Double-click *install directory*\System\PPI\User Level Access Client.reg
4. Re-boot the machine.

Do the following steps if you have user-level access enabled:

1. Run dcomcnfg.
2. Double-click Tektronix Pattern Generator Application in the Applications page.
3. In the Location page, uncheck the Run application on this computer box and check the *Run application on the following computer* box. Enter the name of the TLA 700 machine in the edit field.

Do the following steps if you have share-level access enabled:

1. Run regedit.
2. Click on the following registry key.

HKEY\_CLASSES\_ROOT\AppID{ EF9B47D6-99AD-11d3-A413-0004ACAEB013 }

3. Using Edit>New>StringValue, add a named value *RemoteServerName*.
4. Click on the new value *RemoteServerName* and select Edit>Modify.
5. Enter the name of the TLA 700 machine as its value.

Do the following to verify that setup is complete:

1. Run *install directory*\Samples\PPI Samples\Vc++\test client\testclient.exe on the client machine.
2. Click the Connect button to see if the client can connect to the TLAPG.

(The first time you connect it may take a few minutes.)

You can switch between user-level and share-level access later by uninstalling Tektronix PPI Client and DCOM95 via the Control Panel and redoing the procedure from Step 2 onwards.

# Client machine using Microsoft Windows 98

Do the following steps if using Microsoft Windows 98:

1. Install and configure TCP/IP.
2. Run the Tektronix PPI Client install program supplied with the Tektronix Pattern Generator.
3. You must use share-level or user-level access as chosen for the Tektronix Pattern Generator. This is done in the Control Panel>Network>Access Control page.

## Share-level access

Do the following steps if you have share access enables:

1. In Control Panel>Network>Access Control, choose share-level access.
2. Re-boot the machine.
3. Double-click *install directory*\System\PPI\Share Level Access Client.reg
4. Re-boot the machine.

## User-level access

Do the following steps:

1. In Control Panel>Network>Access Control, choose user-level access control and enter the name of the domain that will be used to validate user access.
2. Re-boot the machine.
3. Double-click *install directory*\System\PPI\User Level Access Client.reg
4. Re-boot the machine.

Do the following if you have user-level access enabled:

1. Run dcomcnfg.
2. Double-click Tektronix Pattern Generator Application in the Applications page.
3. In the Location page, uncheck the Run application on this computer box and check the Run application on the following computer box. Enter the name of the TLA 700 machine in the edit field.

Do the following if you have share-level access enabled:

1. Run regedit.
2. Click on the following registry key  
HKEY\_CLASSES\_ROOT\AppID{ EF9B47D6-99AD-11d3-A413-0004ACAEB013 }
3. Using Edit>New>StringValue, add a named value *RemoteServerName*.
4. Click on the new value *RemoteServerName* and select Edit>Modify.
5. Enter the name of the TLA 700 machine as its value.

Do the following to verify that setup is complete:

1. Run *install directory*\Samples\PPI Samples\Vc++\test client\testclient.exe on the client machine.
2. Click the Connect button to see if the client can connect to the TLAPG. (The first time you connect it may take a few minutes.)

You can switch between user-level and share-level access later by redoing the procedure from Step 3 onwards.

## Client machine using other platforms

If the client application requires use of the type library, it may be generated on your platform using Tlapg.IDL in C:\Program Files\Tektronix Pattern Generator\System\PPI\src on the TLA 700.

Make sure you perform the following steps:

1. Ensure that DCOM is working on your platform.
2. Merge  
C:\Program Files\ Tektronix Pattern Generator\System\PPI\Client.reg on the TLA 700 into your registry.
3. Depending on the type of access control you chose for the Tektronix Pattern Generator, merge  
C:\Program Files\ Tektronix Pattern Generator\System\PPI\Share Level Access Client.reg or User Level Access Client.reg on the TLA 700 into your registry.
4. Add a string value named *RemoteServerName* to the key  
HKEY\_CLASSES\_ROOT\AppID\{EF9B47D6-99AD-11d3-A413-0004ACAEB013 }
5. Enter the name of the TLA 700 machine as its value.

## Registering/Unregistering PPI

### Server side Registration/UnRegistration

The server side needs both tlapg server and tlapg proxy/stub dll has to be registered from command line.

Do the following to register the server:

```
Tlapg /RegServer from command line of appropriate directory
```

Do the following to Unregister the server:

```
Tlapg /UnregServer from command line of appropriate directory
```

Do the following to register the proxy/stub dll:

```
Regsvr32 Tlapgproxy .dll
```

Do the following to Unregister the proxy/stub dll

```
Regsvr32 /u Tlapgproxy .dll
```

### Client side Registration

Client side registration also needs both of the above command lines.

When the registration is complete open the **DCOMCNFG** tool from command line and select the tlapg properties.

To set the **location** of component running from "Run application from the following computer" option. Also give the remote machine name. In **identity** page select "interactive user" option.

## Other Issues with DCOM

If the server ( Tlapg ) is going to run in Windows 95/98 DCOM software should be installed prior to run any DCOM application and also the Server part should be started first before any clients get connected to it.

Also the Windows 95/98 registry needs two entries for DCOM enabling.

EnableDCOM 'Y'

EnableRemoteClient 'Y'

Both of these entries should be done in the following:

HKEY\_LOCAL\_MACHINE\Microsoft\Ole directory in registry

## Connecting to the Tektronix Pattern Generator Application (Server)

Client applications connect to the Tektronix Pattern Generator server by creating an Application object. For example, in Visual Basic,

```
`Establish connection to TLA PG.  
Dim App As Object  
Set App = CreateObject("Tlapg.Application")
```

Once the Application object has been created, the client can call methods on it to get references to System and Module objects.

## Disconnecting from the Tektronix Pattern Generator Server




A client application that has connected to the TLAPG server may disconnect by deleting the reference to the Application object. For example, in Visual Basic,

```
`Disconnect from TLA PG.  
Set App = Nothing
```

## Samples

As mentioned previously, all of the interfaces exported by the server are dual interfaces for example, they support static and dynamic binding.

Click on one of the following buttons to show the use of dual binding. The rest of the code samples in this document use the dispatch portion of each dual interface (dynamic binding).

-  [Sample Visual Basic Client \(Dispatch\) on page 15](#)
-  [Sample Visual Basic Client \(Vtable\) on page 16](#)
-  [Other Samples on page 17](#)

## Sample Visual Basic Client (Dispatch)

```
Run
```

```
End Sub
```

```
Private Sub Run()
```

```
    'Do pattern generation. Wait for it to complete.
```

```
    System.Run
```

```
    Do
```

```
        Status = System.GetRunStatus
```

```
    Loop While (Status = 0)
```

```
End Sub
```

## Sample Visual Basic Client (Vtable)

The above client sample has been repeated below, using the vtable part of the dual interfaces.

```
Dim App As IPGApplication
Dim System As IPGSystem
Dim LA As IPGModule

Dim Status As Long

Dim S As String
Dim Data As Variant

Private Sub Form_Load()
    'Connect to server.
    Set App = CreateObject("Tlapp.Application ")
    'Get system pointer.
    Set System = App.GetSystem

    'NOTE: To load a system, fill in system path.
    System.LoadSystem("<path>")

    'Run pattern generation.
    Run
End Sub

Private Sub Run()

    'Do pattern generation. Wait for it to complete.
    System.Run
    Do
        Status = System.GetRunStatus
    Loop While (Status = 0)

End Sub
```



## Other Samples

For sample client programs, go to Start > Programs > Tektronix Pattern Generator > PPI Samples.

## Errors

All methods in all interfaces of PPI return an HRESULT (or SCODE). Refer to `tlapgerror.h` for possible error codes.

Additional error information is communicated as follows:

- Objects that use the dispatch portion of the dual interface can use the exception information argument of the *Invoke* method.
- Objects that use the Vtable portion of the dual interface can use error objects. When an HRESULT indicates an error, the client can call the standard function `GetErrorInfo()` to get more detailed information about the error.

When a method returns an error, output arguments are undefined and should not be used.

Refer to the sample programs for examples on handling errors.

## Messages

Tektronix Pattern Generator Application has instances where the user is asked to confirm a particular operation. For example, before loading a system, the user is asked whether the current system should be saved before the load operation. Since it is not possible to ask questions through the programmatic interface, the application will always proceed with the original operation as though the question were never asked. In the previous example, the load operation would proceed without saving the current system.

## Slot Numbers and Expansion Mainframes

In PPI, the slot numbers for expansion mainframes are specified by extending the slot numbers for the mainframe.

For example, consider a system configuration consisting of a TLA720 benchtop mainframe with 2 expansion frames each containing 13 slots. The slot numbers would be as follows:

Mainframe:	Slots 0-12
Expansion 1:	Slots 13-25
Expansion 2:	Slots 26-38

# Reference

This is a reference for all the objects and interfaces supported by the TLA PG Programmatic Interface.

## NOTE:

- Output arguments are not defined and should not be used if the HRESULT return code indicates an error.
- Unless otherwise specified, all methods are synchronous and wait for the completion of the operation before returning.
- All the examples in the reference section use the dispatch portion of each dual interface.

Go to page 7 for information about objects and interfaces.

## Quick Reference

This section contains a quick reference for the objects and methods in the TLA PG Programmatic Interface. These methods are described in more detail in the Reference section.

### Application Object

IPGApplication

HRESULT GetSystem( ppDispatch ) (see page 20 )

HRESULT ShowWindow( Show ) (see page 46 )

### System Object

IPGSystem

Configuration Functions:

HRESULT GetNumModuleSlots( pNumSlots ) (see page 21 )

HRESULT GetFirstModuleSlot( pSlot ) (see page 22 )

HRESULT GetSWVersion( pVersion ) (see page 23 )

HRESULT GetDiagnosticsStatus( pDiagStatus ) (see page 24 )

HRESULT GetModuleTypeBySlot( Slot, pModuleType ) (see page 27 )

HRESULT GetModulePropertiesBySlot( Slot, pModuleProperties ) (see page 25 )

HRESULT GetModuleBySlot( Slot, ppDispatch ) (see page 29 )

HRESULT GetModuleByName( ModuleName, ppDispatch ) (see page 31 )

HRESULT GetModuleNames (ModuleName) (see page 32 )

Load & Save Functions:

HRESULT LoadSystem( SystemPath ) (see page 33 )

HRESULT SaveSystem( SystemPath, UserComment, SaveData ) (see page 35 )

Run Control & Status Functions:

HRESULT Run() (see page 36 )

HRESULT Stop() (see page 37 )

HRESULT GetRunStatus( pRunStatus ) (see page 38 )

## Module Object

IPGModule

Load & Save Functions:

HRESULT LoadModule( ModulePath, ModuleName ) (see page 39 )

HRESULT SaveModule( ModulePath, UserComment, SaveData ) (see page 40 )

Export & Import Functions:

HRESULT Import( ImportFilePath, BlockNo ) (see page 42 )

HRESULT Export( ExportFilePath, BlockNo ) (see page 44 )

Module Configuration Functions:

HRESULT SetClockMode (Mode) (see page 47 )

HRESULT SetClockPeriod (Period) (see page 48 )

HRESULT SetClockThreshold (Threshold) (see page 49 )

HRESULT SetClockPolarity (Polarity) (see page 50 )

HRESULT SetOutputLevel (Level, Probe) (see page 51 )

HRESULT GetGroupNames ( GroupName ) (see page 52)

HRESULT GetGroupSize ( GroupSize ) (see page 53)

HRESULT GetProbeChannelNames ( GroupName, ProbeChName ) (see page 54 )

# Listing of Methods

## IPGApplication::GetSystem

### Description:

This method returns the interface pointer for the System object.

### IDL Syntax:

```
HRESULT GetSystem( [out, retval] IDispatch** ppDispatch )
```

### Arguments:

ppDispatch - The interface pointer for the *System* object.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object  
Dim Sys As Object  
  
Set App = CreateObject("Tlapg.Application ")  
  
'Get system.  
Set Sys = App.GetSystem
```

### Remarks:

Gets the existing system object if there is a one.

## IPGSystem::GetNumModuleSlots

### Description:

This method returns the number of slots in the TLA700 mainframe that can be occupied by instrument modules.

### IDL Syntax:

```
HRESULT GetNumModuleSlots( [out, retval] long* pNumSlots )
```

### Arguments:

pNumSlots - The number of instrument module slots in the mainframe.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim NumModuleSlots As Long

Set App = CreateObject("Tlapg.Application")

        'Get system.
        Set Sys = App.GetSystem

'Get number of module slots.
        NumModuleSlots = Sys.GetNumModuleSlots
```

### Remarks:

The value returned is 4 for the portable mainframes (TLA704 and TLA14). It is 11 for the TLA711 and 10 for the TLA720 (Slots used by the benchtop controller are not included).

## IPGSystem::GetFirstModuleSlot

### Description:

This method returns the number of the first slot in the TLA700 mainframe that can be occupied by an instrument module.

### IDL Syntax:

```
HRESULT GetFirstModuleSlot( [out, retval] long* pSlot )
```

### Arguments:

pSlot - The number of the first slot in the mainframe that can be occupied by an instrument module.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim FirstModuleSlot As Long

Set App = CreateObject("Tlapg.Application")

    `Get system.
    Set Sys = App.GetSystem

`Get first module slot.
    FirstModuleSlot = Sys.GetFirstModuleSlot
```

### Remarks:

The value returned is the slot number of the first slot that *can* be occupied by a module. This slot may or may not be currently occupied.

This slot number is 1 for the portable mainframes (TLA704 and TLA714). It is 2 for the TLA711 and 3 for the TLA720 because the first few slots are occupied by the controller module.

## IPGSystem::GetSWVersion

### Description:

This method returns the version of the application software.

### IDL Syntax:

```
HRESULT GetSWVersion( [out, retval] BSTR* pVersion )
```

### Arguments:

pVersion - The version of the application software. This is in the form "For example, "1.01.000".

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim SWVersion As String

Set App = CreateObject("Tlapg.Application")

    'Get system.
    Set Sys = App.GetSystem

'Get software version.
SWVersion = Sys.GetSWVersion
```

### Remarks:

The TLA700 server will allocate the space for the returned string. The client is responsible for freeing it when it is no longer in use.

## IPGSystem::GetDiagnosticsStatus

### Description:

This method returns the power-on diagnostics status.

### IDL Syntax:

```
HRESULT GetDiagnosticsStatus( [out] BSTR* pDiagStatus )
```

### Arguments:

pDiagStatus - The status of diagnostics.

Eg: "Pass "

The diagnostics status can take one of the following values: "Running" or "Pass" or "Fail"

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim Status As String

Set App = CreateObject("Tlapg.Application")

        `Get system.
        Set Sys = App.GetSystem

`Get diag status.
        Status = Sys.GetDiagnosticsStatus
```

### Remarks:

The TLAPG server will allocate the space for the returned string. The client is responsible for freeing it when it is no longer in use.



## IPGSystem::GetModulePropertiesBySlot

### Description:

This method returns the properties of the physical module in the specified slot.

### IDL Syntax:

```
HRESULT GetModulePropertiesBySlot( [in] long Slot, [out, retval] BSTR*
    pModuleProperties )
```

### Arguments:

Slot - The physical slot number

pModuleProperties - The properties of the physical module in the specified slot. This is of the format shown below. Fields are included as they apply. "<manufacturer>,<model>,<firmware version>,<power- on diagnostics status>,<speed>,<memory depth>"

For example:

```
PG:      "Tektronix,TLA 7PG2,2.0.1,Pass, 268 MHz,262140"
```

Refer to IPGSystem::GetDiagnosticsStatus for possible values for diagnostics status.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_EMPTY_SLOT	The specified slot is empty.
TLAPG_E_INVALID_SLOT_NUMBER	Invalid "Slot" argument.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim ModDesc As String

Set App = CreateObject("Tlapg.Application")

        `Get system.
        Set Sys = App.GetSystem

        ...
        `Get description of module in slot 3.
ModDesc = Sys.GetModulePropertiesBySlot(3)
```

**Remarks:**

For modules that occupy more than one slot, the same string will be returned for each of its slots.

The TLAPG server will allocate the space for the returned string. The client is responsible for freeing it when it is no longer in use.

## IPGSystem::GetModuleTypeBySlot

### Description:

This method returns the type of the physical module in the specified slot.

### IDL Syntax:

```
HRESULT GetModuleTypeBySlot( [in] long Slot, [out, retval] long*  
                             pModuleType )
```

### Arguments:

Slot - The slot number.

PModuleType - The type of the physical module in the specified slot.

This can be one of the following values:

Return value	Description
TLA_LA_MODULE (0)	LA module
TLA_DSO_MODULE (1)	DSO module
TLA_CONTROLLER_MODULE (2)	Controller module
TLA_UNKNOWN_MODULE (3)	Unknown module
TLA_EMPTY_SLOT (4)	Empty slot
TLA_EXPANSION_INTERFACE_MODULE (5)	Expansion interface module
TLA_PG_MODULE (6)	PG module

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_INVALID_SLOT_NUMBER	Invalid "Slot" argument.
TLAPG_E_OUT_OF_MEMORY	There is not enough memory to perform the operation.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object  
Dim Sys As Object  
Dim ModType As Long
```

```
Set App = CreateObject("Tlapg.Application")

    `Get system.
    Set Sys = App.GetSystem

`Get type of module in slot 3.
ModType = Sys.GetModuleTypeBySlot(3)
```

**Remarks:**

For instrument modules that occupy more than one slot, the same module type will be returned for each of its slots.

Refer to the section Slot Numbers and Expansion Mainframes on page 17 for information on how to specify slot numbers with expansion mainframes.

## IPGSystem::GetModuleBySlot

### Description:

This method returns the interface pointer for the logical module in the specified slot.

### IDL Syntax:

```
HRESULT GetModuleBySlot( [in] long Slot, [out, retval] IDispatch**  
                        ppDispatch )
```

### Arguments:

Slot - The slot number. This can correspond to any of the slots occupied by the logical module.

ppDispatch - The interface pointer for the module in the specified slot.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_EMPTY_SLOT	The specified slot is empty.
TLAPG_E_UNKNOWN_MODULE	The module in the specified slot is not recognized.
TLAPG_E_INVALID_SLOT_NUMBER	Invalid "Slot" argument.
TLAPG_E_OUT_OF_MEMORY	There is not enough memory to perform the operation.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object  
Dim Sys As Object  
Dim PG As Object  
  
Set App = CreateObject("Tlapg.Application")  
  
'Get system.  
Set Sys = App.GetSystem  
...  
  
'Get module in slot 3.  
Set PG = Sys.GetModuleBySlot(3)
```

**Remarks:**

Module references obtained via this method are invalidated by operations like `IPGSystem::LoadSystem()` that affect the logical modules in the system. Remember to release any module references before performing such operations.

## IPGSystem::GetModuleByName

### Description:

This method returns the interface pointer for the logical module with the specified name. The module name should be as specified in the TLAPG System window.

### IDL Syntax:

```
HRESULT GetModuleByName( [in] BSTR ModuleName, [out, retval]  
                        IDispatch** ppDispatch )
```

### Arguments:

ModuleName - The user name of the required module. This is the name that you would see in the System Window.  
ppDispatch - The interface pointer for the module with the specified name.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_INVALID_MODULE_NAME	Invalid "ModuleName" argument.
TLAPG_E_OUT_OF_MEMORY	There is not enough memory to perform this operation.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object  
Dim Sys As Object  
Dim PG As Object  
  
Set App = CreateObject("Tlapg.Application")  
  
        'Get system.  
Set Sys = App.GetSystem  
  
        ...  
        'Get module.  
Set PG = Sys.GetModuleByName("PG 1")
```

### Remarks:

Module references obtained via this method are invalidated by operations like IPGSystem::LoadSystem() that affect the logical modules in the system. Remember to release any module references before performing such operations.

# IPGSystem::GetModuleNames

## Description

This method retrieves the names of all logical modules in the system.

## IDL Syntax

```
HRESULT GetModuleNames( [out, retval] VARIANT* pModuleNames )
```

## Arguments

pModuleNames - The module names.

Module names are returned as a VARIANT. The variant is of type VT\_ARRAY and points to a SAFEARRAY. The SAFEARRAY has dimension 1 and its elements are of type VT\_BSTR. The number of modules is equal to the number of elements in the SAFEARRAY.

## HRESULT Return Codes

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_FAILED	The operation was unsuccessful.

## Example

### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim M As Variant
Dim Modules As Variant

Set App = CreateObject("Tlapg.Application")
'Get system.Set Sys = App.GetSystem
'Get module names.
Modules = Sys.GetModuleNames
'Access module names.
For Each M In Modules

'Use module name in M.

Next M
```

## Remarks:

If there are no modules, the SAFEARRAY returned will be empty.



## IPGSystem::LoadSystem

### Description:

This method loads the Pattern Generator application with the specified pattern generator system file.

### IDL Syntax:

```
HRESULT LoadSystem( [in] BSTR SystemPath )
```

### Arguments:

SystemPath - The full path to the required TLA PG system file.  
Eg: "C:\My Documents\System1.tpg"

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_LOAD_INVALID_FILE	An error occurred while opening the file for reading.
TLAPG_E_LOAD_MISMATCH	The system configuration in the file does not match the current hardware configuration.
TLAPG_E_LOAD_ERROR	An error occurred retrieving information from the file during the load operation.
TLAPG_E_SYSTEM_RUNNIG	The operation cannot be performed when the system is running.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object  
Dim Sys As Object  
  
Set App = CreateObject("Tlapg.Application")  
`Get system.  
Set Sys = App.GetSystem  
...  
`Load system.  
Sys.LoadSystem "C:\My Documents\System1.tpg"
```

### Remarks:

All file paths without machine qualifiers refer to drives mapped on the TLA 700.

Focus may be transferred to the TLA PG application window whenLoadSystem() is invoked.

Client applications need to take this into account.

## IPGSystem::SaveSystem

### Description:

This method saves the Pattern Generator system to a file.

### IDL Syntax:

```
HRESULT SaveSystem( [in] BSTR SystemPath,[in] BSTR UserComment, [in]  
    long SaveData )
```

### Arguments:

**SystemPath** - The full path to the TLA PG system file to save to.

Eg: "C:\My Documents\System1.tpg"

**UserComment** - The user comment to be saved in the file.

**SaveData** - This flag takes one of the following values:

Value	Description
TLAPG_SAVE_NO_DATA (0)	Do not save program data in file.
TLAPG_SAVE_DATA (1)	Save program data in file.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_SAVE_ERROR	An error occurred during the save operation.
TLAPG_E_SYSTEM_RUNNING	The operation cannot be performed when the system is running.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object  
Dim Sys As Object  
  
Set App = CreateObject("Tlapg.Application")  
'Get system.  
Set Sys = App.GetSystem  
  
'Save system with data.  
Sys.SaveSystem "C:\My Documents\a.tpg", "My system", 1
```

### Remarks:

All file paths without machine qualifiers refer to drives mapped on the TLA 700. If the file already exists, it will be overwritten.

## IPGSystem::Run

### Description:

This method starts the pattern generation operation.

### IDL Syntax:

```
HRESULT Run()
```

### Arguments:

None

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_SYSTEM_RUNNING	The system cannot be performed as the system is running.
TLAPG_E_NO_ENABLED_MODULES	There are no enabled modules in the current system.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim RunStatus As Long

Set App = CreateObject("Tlapg.Application")
'Get system.
Set Sys = App.GetSystem
...

'Start pattern generation and wait until it is complete.
Sys.Run
Do
    RunStatus = Sys.GetRunStatus
Loop While (RunStatus = 0)
```

### Remarks:

This method starts the pattern generation operation but does not wait for it to complete before returning. After calling this method, the method IPGSystem::GetRunStatus() can be used to find out the current run status of the system.

## IPGSystem::Stop

### Description:

This method stops the pattern generation operation.

### IDL Syntax:

```
HRESULT Stop()
```

### Arguments:

None

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_SYSTEM_NOT_RUNNING	The system is not running.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object

Set App = CreateObject("Tlapg.Application")
`Get system.
Set Sys = App.GetSystem
...

`Start Pattern Generation.
Sys.Run
...

`Stop Pattern Generation.
Sys.Stop
```

### Remarks:

This method issues a request to stop the system but does not wait for the stop operation to complete. After calling this method, the method `IPGSystem::GetRunStatus()` can be used to find out the current run status of the system.

## IPGSystem::GetRunStatus

### Description:

This method returns the current runtime status of the Pattern Generator.

### IDL Syntax:

```
HRESULT GetRunStatus( [out, retval] long* pRunStatus )
```

### Arguments:

pRunStatus - The current runtime status. This can be one of the following values:

Return Value	Description
TLAPG_RUNNING (0)	Pattern Generation has started and is currently running.
TLAPG_IDLE (1)	Pattern Generator is in Idle state. Any operations that were previously running have completed or have been stopped.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim RunStatus As Long

Set App = CreateObject("Tlapg.Application")

'Get system.
Set Sys = App.GetSystem
...
'Start pattern generation and wait until it is complete.
Sys.Run
Do
    RunStatus = Sys.GetRunStatus
Loop While (RunStatus = 0)
```

### Remarks:

Informs the status like running or idle.

## IPGModule::LoadModule

### Description:

This method loads a module from the specified Pattern Generator system/module file on to the current module.

### IDL Syntax:

```
HRESULT LoadModule( [in] BSTR ModulePath, [in] BSTR ModuleName )
```

### Arguments:

ModulePath - The full path to the required TLA PG system/module file.  
Eg: "C:\My Documents\My System.tpg"

ModuleName - The name of the module in the specified file to load.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_LOAD_INVALID_FILE	An error occurred opening the file for reading.
TLAPG_E_SYSTEM_RUNNING	The operation cannot be performed when the system is running.
TLAPG_E_FAILED	The operation was successful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim PG As Object

Set App = CreateObject("Tlapg.Application")
'Get system.
Set Sys = App.GetSystem
...

'Get module in slot 3.
Set PG = Sys.GetModuleBySlot(3)
...
'Load module.
PG.LoadModule "C:\My Documents\System1.tpg", "PG 1"
```

### Remarks:

Invoking this method will not result in a merge operation even if the destination module does not have enough channels/physical modules.

All file paths without machine qualifiers refer to drives mapped on the TLA 700.

## IPGModule::SaveModule

### Description:

This method saves the module to a file.

### IDL Syntax:

```
HRESULT SaveModule( [in] BSTR ModulePath, [in] BSTR UserComment, [in]
                    long SaveData )
```

### Arguments:

ModulePath - The full path to the TLA PG module file to save to.

Eg: "C:\My Documents\My Module.tpg"

UserComment - The user comment to be saved in the file.

SaveData - This flag takes one of the following values:

Value	Description
TLAPG_SAVE_NO_DATA (0)	Do not save program data in file.
TLAPG_SAVE_DATA (1)	Save program data in file.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_SAVE_INVALID_FILE	An error occurred while opening this file for reading.
TLAPG_SAVE_ERROR	An error occurred during the save operation.
TLAPG_E_SYSTEM_RUNNING	The operation cannot be performed when the system is running.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim PG As Object

Set App = CreateObject("Tlapg.Application")

'Get system.
Set Sys = App.GetSystem
...

'Get module in slot 3.
Set PG = Sys.GetModuleBySlot(3)
...
```



```
`Save module.  
PG.SaveModule "C:\My Documents\a.tpg","My module",1
```

**Remarks:**

All file paths without machine qualifiers refer to drives mapped on the TLA 700. If the file already exists, it will be overwritten.

## IPGModule::Import

### Description:

This method imports pattern data from an ASCII text file onto a particular Block.

### IDL Syntax:

```
HRESULT Import( [in] BSTR ImportFilePath, [in] long BlockNo )
```

### Arguments:

ImportFilePath - The file that contains the pattern data is to be imported.

BlockNo- The BlockNo onto which the contents of the buffer to be imported.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_INVALID_IMPORT_FILE	An error occurred while opening the file for reading.
TLAPG_E_UNKNOWN_IMPORT_FORMAT	Unknown import format.
TLAPG_E_IMPORT_ERROR	An error occurred during the import operation.
TLAPG_E_SYSTEM_RUNNING	The operation cannot be performed as the system is running.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim PG As Object

Set App = CreateObject("Tlapg.Application")

'Get system.
Set Sys = App.GetSystem
...

'Get module in slot 3.
Set PG = Sys.GetModuleBySlot(3)
...

'Import pattern data onto block 2.
PG.Import "C:\My Documents\Counter.txt", 2
```

**Remarks:**

All file paths without machine qualifiers refer to drives mapped on the TLA 700.

## IPGModule::Export

### Description:

This method exports pattern data of a particular block to an ASCII text file.

### IDL Syntax:

```
HRESULT Export([in] BSTR ExportFilePath, [in] long BlockNo, [in] long ExportType )
```

### Arguments:

ExportFilePath - The file onto which the pattern data is to be exported.

BlockNo - The BlockNo of which the contents of the buffer to be exported.

Export Type - This flag takes one of the following values.

Value	Description
TLAPG_E_INVALID_EXPORT_FILE	Tektronix TLA Data Exchange Format.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_INVALID_EXPORT_FILE	An error occurred while opening the file for writing.
TLAPG_E_UNKNOWN_EXPORT_FORMAT	Unknown export format.
TLAPG_E_INVALID_BLOCK_NUMBER	Invalid block number.
TLAPG_E_EXPORT_ERROR	An error occurred during the export operation.
TLAPG_E_SYSTEM_RUNNING	The operation cannot be performed as the system is running.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim PG As Object

Set App = CreateObject("Tlapg.Application")

'Get system.
Set Sys = App.GetSystem
...
```

```
'Get module in slot 3.  
Set PG = Sys.GetModuleBySlot(3)  
...
```

```
'Export pattern data of block 2 in TLA Format.  
PG.Export "C:\My Documents\Counter2.txt", 2, 0
```

**Remarks:**

All file paths without machine qualifiers refer to drives mapped on the TLA 700. If the file already exists, it will be overwritten.

## IPGApplication::ShowWindow

### Description:

This method shows/hides the TLA PG server's main window.

### IDL Syntax:

```
HRESULT ShowWindow( [in] long Show )
```

### Arguments:

Show - This flag takes one of the following values:

Value	Description
TLAPG_HIDE_WINDOW (0)	Hide the server window.
TLAPG_SHOW_WINDOW (1)	Show the server window.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object  
  
Set App = CreateObject("Tlapg.Application ")  
  
'Hide the window.  
App.ShowWindow 0
```

### Remarks:

The application window is shown by default when a client connects to the server.

## IPGModule::SetClockMode

### Description:

This method sets the Clocking mode to Internal or External for this module.

### IDL Syntax:

```
HRESULT SetClockMode([in] long Mode)
```

### Arguments:

Mode-Internal or External clock mode.

Value	Description
TLAPG_INTERNAL_MODE (0)	Internal
TLAPG_EXTERNAL_MODE (1)	External

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_SYSTEM_RUNNING	The parameter cannot be set as the system is running.
TLAPG_E_INVALID_CLOCK_NAME	Invalid clock mode.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim PG As Object

Set App = CreateObject("Tlapg.Application")

'Get system.
Set Sys = App.GetSystem
...

'Get module in slot 3.
Set PG = Sys.GetModuleBySlot (3)

'Set the Clock Mode to External
PG.SetClockMode 1
```

## IPGModule::SetClockPeriod

### Description:

This method sets the internal clock period for this module.

### IDL Syntax:

```
HRESULT SetClockPeriod([in] BSTR Period)
```

### Arguments:

Period - Clock period in string format. For example, if you want to set the clock period to 10.56 ns, then the clock period string should be "10.56ns".

Channel Mode	Min	Max	Resolution
Full	7.4626865ns (134MHz)	2.0000000s (0.5Hz)	8digit
Half	3.7313432ns (268MHz)	1.0000000s (1Hz)	8digit

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_SYSTEM_RUNNING	This parameter cannot be set as the system is running.
TLAPG_E_INVALID_CLOCK_PERIOD	Invalid clock period.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim PG As Object

Set App = CreateObject("Tlapg.Application")

'Get system.
Set Sys = App.GetSystem
...

'Get module in slot 3.
Set PG = Sys.GetModuleBySlot(3)

'Set the Clock Period to 1ms
PG.SetClockPeriod "1.0000000 ms"
```



## IPGModule::SetClockThreshold

### Description:

This method sets the external clock threshold for this module.

### IDL Syntax:

```
HRESULT SetClockThreshold([in] BSTR Threshold)
```

### Arguments:

Threshold\_Clock threshold in string format.

Value	Description
Threshold	-2.56V to 2.54V (20mV step)

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_SYSTEM_RUNNING	This parameter cannot be set as the system is running.
TLAPG_E_INVALID_CLOCK_THRESHOLD	Invalid clock threshold.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim PG As Object

Set App = CreateObject("Tlapg.Application")

'Get system.
Set Sys = App.GetSystem
...

'Get module in slot 3.
Set PG = Sys.GetModuleBySlot(3)

'Set the Clock Threshold to 1.5V
PG.SetClockThreshold to "1.5V"
```

## IPGModule::SetClockPolarity

### Description:

This method sets the external clock polarity for this module.

### IDL Syntax:

```
HRESULT SetClockPolarity([in] long Polarity)
```

### Arguments:

Polarity\_Clock polarity Normal or Invert.

Value	Description
TLAPG_POLARITY_NORMAL (0)	Normal
TLAPG_POLARITY_INVERT (1)	Invert

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_SYSTEM_RUNNING	This parameter cannot be set as the system is running.
TLAPG_E_INVALID_CLOCK_POLARITY	Invalid clock polarity.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim PG As Object

Set App = CreateObject("Tlapg.Application")

'Get system.
Set Sys = App.GetSystem
...
'Get module in slot 3.
Set PG = Sys.GetModuleBySlot(3)

'Set the Clock Polarity to Invert
PG.SetClockPolarity 1
```

## IPGModule::SetOutputLevel

### Description:

This method sets the output level for a probe for this module.

### IDL Syntax:

```
HRESULT SetOutputLevel([in] BSTR Level, [in] BSTR Probe)
```

### Arguments:

Level-Output Level in string format. (2.0V to 5.5V)  
Probe\_Probe Name in string format.

Value	Description
Level	2.0V to 5.5V (for TTL/CMOS Probes only).
Probe	A, B, ...for single module. 1A, 2B in case of merged modules.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_INVALID_OUTPUT_LEVEL	Invalid output level.
TLAPG_E_INVALID_PROBE	Invalid "Probe" argument.
TLAPG_E_FAILED	The operation was unsuccessful.

### Examples:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim PG As Object

Set App = CreateObject("Tlapg.Application")

'Get system.
Set Sys = App.GetSystem
...

'Get module in slot 3.
Set PG = Sys.GetModuleBySlot(3)

'Set the Output Level to 4,75 Volts for Probe A.
PG.SetOutputLevel "4.75", "A"
```

## IPGModule::GetGroupNames

### Description:

This method retrieves the names of all groups defined in the module setup.

### IDL Syntax:

```
HRESULT GetGroupNames( [out, retval] VARIANT* pGroupNames )
```

### Arguments:

pGroupNames - The group names.

Group names are returned as a VARIANT. The variant is of type VT\_ARRAY and points to a SAFEARRAY. The SAFEARRAY has dimension 1 and its elements are of type VT\_BSTR. The number of groups is equal to the number of elements in the SAFEARRAY. The groups are returned in the same order as they are specified in the LA Setup Window.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_FAILED	The operation was unsuccessful.

### Example:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim PG As Object
Dim G As Variant
Dim Groups As Variant

Set App = CreateObject("Tlapg.Application")
`Get system.Set Sys = App.GetSystem
`Get module in slot 3.
Set PG = Sys.GetModuleBySlot(3)
`Get group names.
Groups = PG.GetGroupNames
`Access group names.
For Each G In Groups
`Use group name in G.
Next G
```

### Remarks:

If there are no groups defined, the SAFEARRAY returned will be empty.

## IPGModule::GetGroupSize

### Description:

This method retrieves the number of channels in a specified group defined in the module setup.

### IDL Syntax:

```
HRESULT GetGroupSize( BSTR GroupName, [out, retval] long* pGroupSize )
```

### Arguments:

pGroupSize - The number of channels in the specified group.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_INVALID_GROUP_NAME	Invalid "GroupName" argument.
TLAPG_E_FAILED	The operation was unsuccessful.

### Example:

#### Visual Basic

```
Dim App As Object
Dim Sys As Object
Dim PG As Object
Dim G As Variant
Dim Groups As Variant
Dim GroupSize As Long

Set App = CreateObject("Tlapg.Application")
`Get system.Set Sys = App.GetSystem
`Get module in slot 3.
Set PG = Sys.GetModuleBySlot(3)
`Get group names.
Groups = PG.GetGroupNames
`Access group sizes.
For Each G In Groups

GroupSize = PG.GetGroupSize( G )

Next G
```

### Remarks:

Please note that this method returns the actual number of channels in the specified group.

## IPGModule::GetProbeChannelNames

### Description:

This method retrieves the names of the channels defined in the group.

### IDL Syntax:

```
HRESULT GetProbeChannelNames( [in] BSTR GroupName, [out, retval] BSTR*  
pProbeChlNames )
```

### Arguments:

pProbeChlNames - The channel names corresponding to the group.

### HRESULT Return Codes:

Return Code	Description
S_OK	The operation succeeded.
TLAPG_E_INVALID_GROUP_NAME	Invalid "GroupName" argument.
TLAPG_E_FAILED	The operation was unsuccessful.

### Example:

#### Visual Basic

```
Dim App As Object  
Dim Sys As Object  
Dim PG As Object  
Dim G As Variant  
Dim Groups As Variant  
Dim ProbeChlNames As String  
  
Set App = CreateObject("Tlapg.Application")  
`Get system.Set Sys = App.GetSystem  
`Get module in slot 3.  
Set PG = Sys.GetModuleBySlot(3)  
`Get group names.  
Groups = PG.GetGroupNames  
`Access group names.  
For Each G In Groups  
ProbeChlNames = PG.GetProbeChlNames( G )  
Next G
```

### Remarks:

If there are no groups defined, the SAFEARRAY returned will be empty.

# Miscellaneous Topics

## Tektronix TLA Data Exchange Format

File Format:

[vectors]

Sample[]		Addr[15:0](Hex)	Data[15:0](Hex)	Timestamp[]
0	0000	0000	0	
1	FFFF	0001	10.0000000	ns
2	0000	0002	10.0000000	ns
3	FFFF	0003	10.0000000	ns
4	0000	0004	10.0000000	ns
5	FFFF	0005	10.0000000	ns
6	0000	0006	10.0000000	ns
7	FFFF	0007	10.0000000	ns
8	0000	0008	10.0000000	ns
9	FFFF	0009	10.0000000	ns
10	0000	000A	10.0000000	ns
11	FFFF	000B	10.0000000	ns
12	0000	000C	10.0000000	ns
13	FFFF	000D	10.0000000	ns
14	0000	000E	10.0000000	ns
15	FFFF	000F	10.0000000	ns
16	0000	0010	10.0000000	ns
17	FFFF	0011	10.0000000	ns
18	0000	0012	10.0000000	ns
19	FFFF	0013	10.0000000	ns
20	0000	0014	10.0000000	ns
21	FFFF	0015	10.0000000	ns
22	0000	0016	10.0000000	ns
23	FFFF	0017	10.0000000	ns
24	0000	0018	10.0000000	ns
25	FFFF	0019	10.0000000	ns
26	0000	001A	10.0000000	ns
27	FFFF	001B	10.0000000	ns
28	0000	001C	10.0000000	ns
29	FFFF	001D	10.0000000	ns
30	0000	001E	10.0000000	ns
31	FFFF	001F	10.0000000	ns
32	0000	0020	10.0000000	ns

33	FFFF	0021	10.0000000 ns
34	0000	0022	10.0000000 ns
35	FFFF	0023	10.0000000 ns
36	0000	0024	10.0000000 ns
37	FFFF	0025	10.0000000 ns
38	0000	0026	10.0000000 ns
39	FFFF	0027	10.0000000 ns

## Header Info Syntax:

[vectors]

Sample[]            Address[15:0](Hex)        Data[7:0](Hex)    Timestamp[]

1. The first line should start with "[vectors]"
2. The second line should have the group names as shown above.
3. "Sample[]" should be the first column and "Timestamp[]" should be the last column as shown above.
4. Tab is used as a separator between each item in a row.

## Group Details Syntax:

Syntax : <GroupName>[MSB:LSB](Radix)

Example : Data[31:0](Hex)

## Rules:

1. If Radix is not specified, it will be assumed as HEX.
2. If no channel and radix information are present, it will not be treated as a group. (Ex: Sample[], Timestamp[])
3. If no channel information is present but Radix is present, the number of channels for that group will be assumed as '1'.
4. Radix formats supported are Binary, Octal, Decimal and Hex. Radix can be specified in the column header using the first three characters. For example, BIN, OCT, DEC or HEX.
5. Number of channels = MSB - LSB + 1

In this example,

GroupName        : Data

No. of channels : 32 ( 31 - 0 + 1 )

Radix             : HEX