TLAScript
Graphical Interface
Printed Help Document

www.tektronix.com
077-0034-00

Tektronix

Contacting Tektronix

Tektronix, Inc.
14200 SW Karl Braun Drive
P.O. Box 500
Beaverton, OR 97077
USA

For product information, sales, service, and technical support:

- In North America, call 1-800-833-9200.
- Worldwide, visit www.tektronix.com to find contacts in your area.

Warranty 9(b)

Tektronix warrants that the media on which this software product is furnished and the encoding of the programs on the media will be free from defects in materials and workmanship for a period of three (3) months from the date of shipment. If any such medium or encoding proves defective during the warranty period, Tektronix will provide a replacement in exchange for the defective medium. Except as to the media on which this software product is furnished, this software product is provided “as is” without warranty of any kind, either express or implied. Tektronix does not warrant that the functions contained in this software product will meet Customer’s requirements or that the operation of the programs will be uninterrupted or error-free.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period. If Tektronix is unable to provide a replacement that is free from defects in materials and workmanship within a reasonable time thereafter, Customer may terminate the license for this software product and return this software product and any associated materials for credit or refund.

THIS WARRANTY IS GIVEN BY TEKTRONIX WITH RESPECT TO THE PRODUCT IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX’ RESPONSIBILITY TO REPLACE DEFECTIVE MEDIA OR REFUND CUSTOMER’S PAYMENT IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Table of Contents

Overview

Quick Reference

Commands by Alphabetical Listing

Index

Introduction

TLAScript is a standalone application program that runs on the TLA mainframe and provides a scripting interface to the TLA application. TLAScript is an interpreter that processes commands that are read from a script file or are entered by you through the TLAScript graphical interface.

The TLAScript commands are similar to the TPI.com commands in the TPI application. Most of the TPI.com commands have a corresponding TLAScript command.

In essence, TLAScript provides a quick and easy way for you to programmatically control the TLA application using a script file rather than writing a full-blown program that directly uses the TPI application.

Although TLAScript is a good first step toward programmatically controlling the TLA system, the TLAScript application is not a powerful or full-featured TLA programming language. It has limitations when compared to other scripting/programming languages like Visual Basic and ActivePerl (a variety of Perl that has COM capabilities).

> **COM**
>
> The Component Object Model (COM) is a software architecture that allows applications to be built from binary software components. See www.microsoft.com for more information.

For example, TLAScript does not support variables and does not provide any kind of control flow capability. If you need these features you may want to investigate the two scripting/programming languages mentioned previously.

General Characteristics

TLAScript application software:

- Commands from the TLAScript are directly executed by the TLAScript application and do not have an effect on the TLA application (server) software.
- Is compatible only with the corresponding versions of the TLA Application software. For example, the TLAScript application Version 4.1 will only work with TLA Application software Version 4.1.

Starting the TLAScript Application

You have three methods of starting the TLAScript application:

- The easiest way is to start the TLAScript application using the Windows Start menu. Locate and click Start > Programs > Tektronix TLA 700 > TLAScript to display the TLAScript graphical interface.
- You can also start the TLAScript application from an MS-DOS command prompt window by entering C:\Program Files\TLA 700\System\TLAScript.exe from the command line. If switches and arguments are not used, then the TLAScript graphical interface is displayed.
- Double-click on a ".tls" file. Double-clicking has the same functionality as typing a TLAScript file and only works on text files with the TLAScript script extension ".tls".

Following are examples that use a combination of switches and arguments. See Syntax for more information about switch and argument combinations.

Examples.

```
TLAScript
```

This brings up the TLAScript graphical interface.

```
TLAScript C:\MyScript.tls
```

This executes the MyScript.tls script, then automatically connects to the default server (the local server). It does not display the TLAScript graphical interface unless an error is encountered in the script.

```
TLAScript /s MyTLA C:\MyScript.tls
```

This connects to the server named MyTLA, then executes the script named MyScript.tls. It does not display the TLAScript graphical interface unless an error is encountered in the script.

```
TLAScript /b C:\myscript.tls
```

This executes the script MyScript.tls in batch mode, the output goes to standard out, and errors go to standard error.

```
TLAScript"C:\TLA Scripts\myscript.tls"100
```

This executes the myscript.tls script within the folder TLA Scripts. The quotes are used because of the space in the folder name, and the value 100 is assigned to the script variable %1.

```
TLAScript /c Run
```

This switch executes the Run command on the default server (the local server).

Connecting to a Server

When you open the TLAScript application, it is not connected to a TLA server. You must establish a connection.

> **Server**
>
> A server is the TLA application software.

Use the Connect command without an argument to connect to the local server.

> **Local Server**
>
> A local server is usually the TLA (server) application that resides on the same machine that TLAscript is running on.

Example.

```
Connect
```

See Also. Disconnect

LA Module Commands, DSO Module Commands, TLAScript Commands, and TLA Application Commands.

Connecting to Multiple Servers

You may want to connect to multiple servers to control more than one TLA mainframe. To connect to multiple severs, use the Connect command with an argument.

Example. `Connect MyTLA`

When a new connection is established using the Connect command, that server becomes the current server.

> **Current Server**
>
> The current server is the server receiving commands when no server name is specified in the command.

When multiple servers are connected, TLAScript can only send commands to one server at a time. You can check which server is the current server by viewing the status bar in the lower right-hand corner of the TLAScript window. You can also change the current server by entering another server name in the command entry box that the TLAScript is already connected to.

The Connections command will list all the servers that TLAScript is currently connected to. The server name of a remote server is the network name of the remote TLA mainframe.

> **Remote Server**
>
> A remote server is a server (TLA application) that is running on a remote TLA mainframe on the network. To connect to a remote server you must set up the TPI application on the client and server machines. Refer to Setting Up TPI in the TPI documentation.

Multiple Servers

TLAScript allows simultaneous connections to multiple servers. The current server receives all entered commands that do not include a server specification. A server must be connected to before it can receive commands, but it does not need to be the current server for commands to be directed to it.

Commands can be directed to any connected server by preceding the command with "server**:** ". For example, **MyTLA:show** would direct the Show command to the server MyTLA even if it were not the current server.

When there are multiple servers connected, one of the servers is considered the current server. To change the current server enter the server name you want to be switched to (a connection to this server must already have been established), or by specifying a new connection with the Connect command.

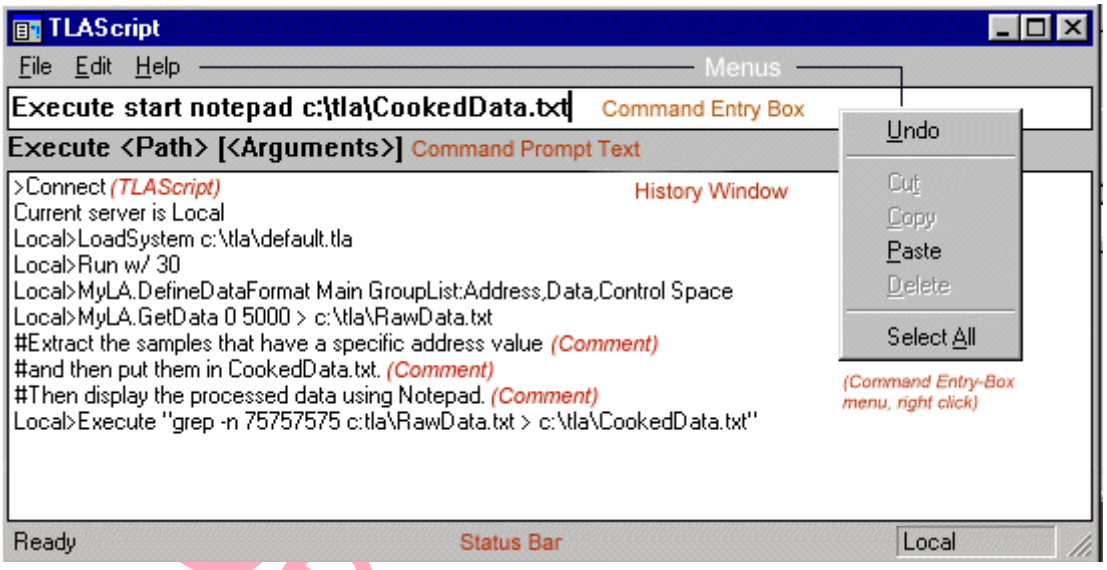
Example of a TLAScript Application

Following is an example of a TLAScript file:

```
#.....
#Connects to the local TLA server, loads the TLA setup,
#then does an acquisition which times out after 30 seconds
#.....
Connect
LoadSystem C:\TLA\Default.tla
Run/w 30
#.....
#Get the Address, Data and Control Group values for
#the first 5000 samples and save them to RawData.txt.
#.....
MyLA.DefineDataFormat Main GroupList:Address,Data,Control Space
MyLA.GetData 0 5000 > C:\TLA\RawData.txt
#.....
#Extract the samples that have a specific address value
#and put them in CookedData.txt.
#Then display #the processed data using Notepad.
#.....
Execute "grep -n 75757575 C:\TLA\RawData.txt > C"\TLA\CookedData.txt
Execute start notepad C:\TLA\CookedData.txt
```

If comments are included with the script, they will also appear in the History list. See TLAScript window.

Graphical Interface



TLAScript Graphical Interface

Command Entry Box. The Command Entry box is located at the top of the TLAScript window. In the Command Entry box, enter the next TLAScript command that you want to execute. TLAScript will attempt to match the command you entered to a list of TLAScript commands. Pressing Enter causes TLAScript to interpret your entry, execute the command, clear the window, and then be ready for the next command entry.

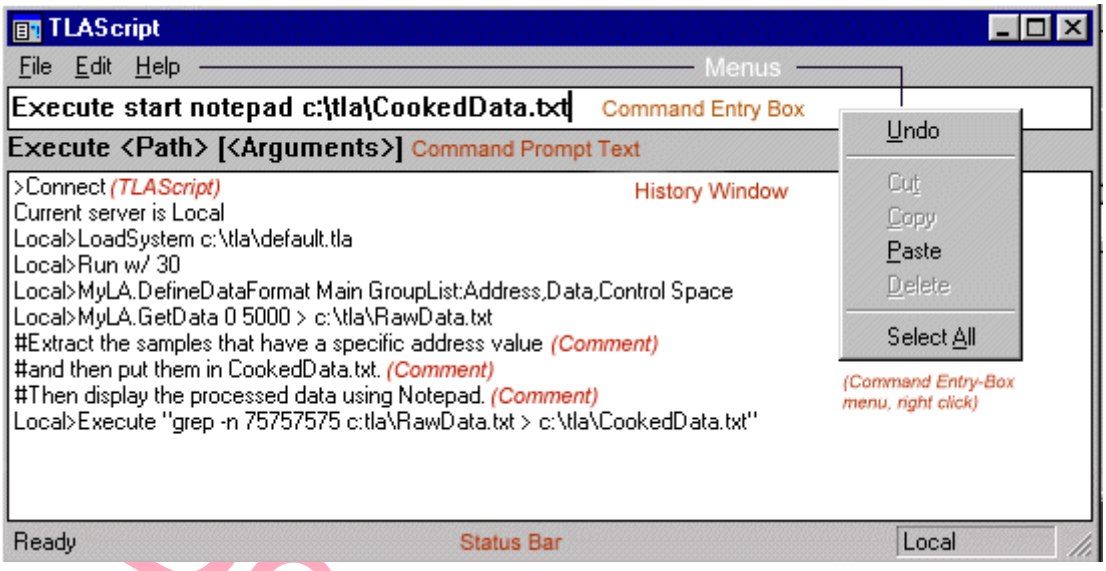
The arrow keys scroll forwards and backwards through previously typed commands.

Command Prompt Box. The Command Prompt box is located under the Command Entry box. The Command Prompt box shows the syntax of the command that you are currently entering from the Command Entry box.

History Window. The History window is located in the center of the TLAScript window. The History window stores and displays all commands previously entered in the Command Entry box, along with error messages and outputs (except binary data returned from a call to GetData).

When the maximum entries (64 K) are filled in the History window, TLAScript removes entries from the beginning of the history list as you make new entries.

Menus and Status Bar



TLAScript Graphical Interface

TLAScript window has four menus: File, Edit, Help, and command entry-box menu (right click).

File Menu

Open	Enter the name of the script you want to open. This menu selection is the same as typing in the LoadScript command.
Save	If the history list has not yet been saved, then this menu selection acts the same as the "Save As" menu item. Otherwise, "Save" simply saves the history list over the previous version that was saved (with the same option for script or history selected).
Save As...	This menu selection prompts you for a file, but with the additional option of saving the history list as a script file or as a text file.
Exit	Quits TLAScript application. This menu selection is the same as typing the Exit command.

Edit Menu

Cut, Copy, Paste	These menu selections are active only on the Command Edit Box.
Execute	This menu executes whatever command is in the Command Entry Box and is the same as pressing Enter from the Command Entry Box.
Clear History	Removes all items from the History box.

Command Entry Menu

Cut, Copy, Paste, and Delete	These behave as common Windows commands.
Select All	This selects the entire contents in the Command entry box.
Undo	This reverses only the cut, copy, paste and delete commands.

Status Bar

The Status Bar is located at the bottom of the TLAScript window. The Status Bar displays the current server, or defaults to "No Connections" if TLAScript is not connected to any server.

Syntax

```
TLAScript[/sserver] [/ccommand] [ [/b]ScriptFile[script args] ]
```

The TLAScript program executes according to the combination of switches and arguments that you use. The following is a list of switches and arguments that work in conjunction with each other.

NOTE. *Most of the TLAScript commands are case insensitive, and the arguments are delimited by spaces or tabs.*

If a module name or argument has embedded spaces, use double quotes around it.

Example.

"LA 1". Enabled False

LoadSystem "C:\my documents\my system.tla"

[/sserver]. This switch specifies the default server. When used with the `/c` switch or with the `file` argument, this switch will cause TLAScript to automatically connect to the named server before executing the command or script file. When used without the `/c` switch or the script `file` argument, this specifies the server that is connected to when no argument is given for the Connect command. The name of a remote server is the network name of the remote TLA mainframe.

To connect to a remote server you must set up the TPI application on the client and server machines. Refer to Setting Up TPI in the TPI documentation.

[/ccommand]. This switch specifies a single TLAScript command to execute. When the TLAScript application starts up, the graphical interface is not displayed, the default server is connected, and the command is executed, then exits the TLAScript application. Any output goes to standard out and any error messages go to standard error.

> **Default Server**
>
> Specify the default by using the /s switch (if you do not use this switch, the TLA server on the local server is the default). When you use the /c switch to execute a single command the default server is the server that you connect to.

[scriptfile]. This argument specifies which TLAScript file to execute. TLAScript starts up without displaying the graphical interface, executes the specified script, then exits. If an error is encountered while executing the script, the TLAScript graphical interface pops up and displays the current history and the error message, otherwise no graphical interface appears. TLAScript does not automatically connect to any server in this scenario - you must have an explicit Connect command in the scriptfile.

[/b scriptfile]. This switch specifies which TLAScript file will execute in batch mode. TLAScript starts up without displaying the graphical interface, executes the specified script, then exits. With this switch, the graphical interface is not displayed under any circumstances. An error in the script execution will

halt execution of the script and exit TLAScript with all output going to standard out and any errors going to standard error.

[scmptfile script args]. The argument specifies which TLAScript file to execute and passes the command line arguments to the script. The arguments to the script can be referenced in the script in the same manner as DOS batch file command line arguments. When referring to command line arguments %N refers to the Nth command line argument after the last switch. A script file counts as one of these command line arguments and would be %0. So, given the command line `TLAScript /s MyTLA C:\myscript.tls 100, %0` would be `C:\myscript.tls` and %1 would be 100.

Comments. Comments are allowed from both the command entry box and from within scripts and they are denoted by the '#' character. Any text following the '#' character, up until the end of line character, will be ignored.

See. Connecting to a server

TLAScript Commands

The following commands are handled directly by TLAScript and have no effect on any connected TLA servers:

Application

- Connect *[Server]*
- Disconnect *[Server]*
- Connections
- LoadScript *ScriptFile*
- Echo *Message*
- Execute *Executable [command line arguments]*
- Pause *[/t Time] | [Message]*
- #Comment
- <ServerName>
- Exit

See. Starting TLAScript Application.

See Also. LA Module Commands, DSO Module Commands, and TLA Application Commands.

TLA Application Commands

All the following commands result in a communication with the TLA application (server). These commands are associated with the TLA application and not to a specific module in the TLA mainframe. So you can think of these commands as being associated with the TLA system.

In addition, the following commands have similar methods in the TLA Programmatic Interface (TPI).

- DefineRangeSymbolOptions *FileFormat SymbolTypes Reserved Bound1 Bound2 OffsetType SymbolOffset*
- ExternalSignalIn *[SignalName]*
- ExternalSignalOut *[SignalName]*
- ExternalSignalOutLowTrue *[Value]*
- GetDiagCalStatus
- GetFirstModuleSlot
- GetModuleNames
- GetModuleProperties *SlotNum*
- GetModuleSlotByName *ModuleName*
- GetModuleType *SlotNum*
- GetNumModuleSlots
- GetRepetitiveStopReason
- GetRunStatus
- GetSwVersion
- Hide
- LoadSymbolFile *Path*
- LoadSystem *Path*
- Repetitive *[Value]*
- Run *[/w[WaitLength]]*
- RunCount SaveSystem
- *Path UserComments SaveData*
- Show
- Stop

See Also. LA Module Commands, DSO Module Commands, and TLAScript Commands

LA Module Commands

The following is a list of LA Module commands:

DefineDataFormat	*DataSetComponentsDataType*
DeleteChannelGroup	*UserChannelGroupName*
Enabled	*[Value]*
GetBeginTime	*DataSet*
GetBytesPerSample	
GetChannelGroup	*UserChannelGroupName*
GetChannelName	*HWChannelName*
GetData	*FirstSamples NumSamples [>[>] Path]*
GetEndTime	*DataSet*
GetGroupNames	
GetGroupSize	*GroupName*
GetNumSamples	*DataSet*
GetTimerValue	*TimerID*
GetCounterValue	*CounterID*
GetTimestampMultiplier	
GetTriggerSample	*DataSet*
GetTriggerTime	*DataSet*
GetStartTime	
LoadModule	*Path ModuleName*
LoadTrigger	*Path ModuleName*
MemoryDepth	*[Value]*
Name	*[Name]*
SaveModule	*PathUserCommentsSaveData*
SetChannelGroup	*UserChannelGroupName [ChannelNameList]*
SetChannelName	*HWChannelName[UserChannelName]*
SetEventValue	*EventID EventValue*
SetTriggerPosition LA Module	*Position*

See Also. DSO Module Commands, TLAScript Commands, and TLA Application Commands.

DSO Module Commands

The following is a list of DSO Module commands:

DefineDataFormat	*ComponentsDataType*
Enabled	*[Value]*
GetBeginTime	
GetData	*FirstSamplesNumSamples[>[>]file]*
GetDataOffset	*Channel*
GetDataRange	*Channel*
GetDataSamplePeriod	
GetEndTime	
GetNumSamples	
GetStartTime	*[/t Time] | [Message]*
GetTriggerSample	
GetTriggerTime	
LoadModule	*Path ModuleName*
Name	*[Name]*
SaveModule	*PathUserCommentsSaveData*

See Also. TLA Application Commands, LA Module Commands, and TLAScript Commands

<ServerName>

Syntax. `<ServerName>`

If there are multiple open connections to TLA servers, entering the name of one of those servers causes that server to become the current server. The current server is the server receiving commands when no server name is specified in the command.

Example. `MyTLA`

See Also. Connect, Connections, Disconnect

Comment

Syntax. `# Comment`

Comments in script files are denoted by the "#" character. All characters on the line following the "#" character are ignored.

Example.

This is a comment

Connect MyTLA # Connect to the system in the lab.

See Also. Echo, Pause

Connect

Syntax. `Connect [server]`

This command opens a connection between the TLAScript application client and a local or remote server (TLA application).

Local Server

A local server is usually the TLA (server) application that resides on the same machine that TLAScript is running on.

Remote Server

A remote server is a server (TLA application) that is running on a remote TLA mainframe on the network. To connect to a remote server you must set up the TPI application on the client and server machines. Refer to Setting Up TPI in the TPI documentation.

Once the connection is established, it remains established until you explicitly disconnect it by using the Disconnect command, or by exiting the TLAScript program. It is possible to have more than one open connection when using multiple TLA mainframes. If a server (TLA application) is not specified then the connection is made to the default server, typically the local server. If the local server (TLA application) is not the default server, then a local server connection is made by using the name of the local computer.

Default Server

Specify the default by using the `/s` switch (if you do not use this switch, the TLA server on the local server is the default). When you use the `/c` switch to execute a single command the default server is the server that you connect to.

NOTE. *Run the TLA application (server) software on the TLA mainframe before connecting to that server.*

DCOM Configuration Utility

The DCOM can be used to define where the local computer is in relationship to the "local server". This file utility is included in the Windows operating system. To access this utility, click Start > Run, and then enter the utility named comcnfg. Using this utility, the local computer can use the TLA application (server) software on a remote TLA mainframe as the "local server". If you use the DCOM configuration utility, and then use the TLAScript Connect command to connect to the local server, you will connect to that remote TLA mainframe.

NOTE. *The Distributed Component Object Model (DCOM) is a protocol that enables software components to communicate directly over a network in a reliable, secure, and efficient manner. See www.microsoft.com for more information. Due to the nature of DCOM, the first connection to a remote server can take up to one minute to establish.*

`[server]` The name of the TLA application you are connected to. Use the network name of the remote TLA mainframe for a remote TLA server. For a local server, use the connect command without an argument.

Examples.

Connect (Local server)

Connect MyTLA (Remote server)

See Also. Connections, Disconnect, <ServerName>

Connections

Syntax. `Connections`

This command returns a list of the TLA servers that the TLAScript application is currently connected to.

Example. `Connections`

See Also. Connect, Disconnect, <ServerName>

DefineDataFormat (LA Module)

Syntax. `DefineDataFormat` *`DataSet Components DataType`*

This command defines the options that are used by subsequent GetData commands to the same module. The DefineDataFormat command must be executed before the first GetData command. The options specified when you execute this command remain in effect until changed by a later DefineDataFormat command or until the connection to that TLA server is closed.

DataSet: The LA Module stores three different sets of data. This argument specifies which of the three sets of data you want to retrieve. You can use either the numeric or the symbolic value. Following are valid values:

0	Main
1	MagniVu
2	Violation

*Components:*This argument indicates which components of the data you want to retrieve. The RawWithTimestamp and AllGroupsWithTimestamp values are invalid if the *DataSet* is MagniVu. See the following valid values:

Raw	With Timestamp
Raw	WithoutTimestamp
AllGroups	With Timestamp
AllGroups	WithoutTimestamp
GroupList:	<grp>,<grp>,...,Timestamp

DataType: This argument indicates which format you want the data in when it is returned. You can use either the numeric or the symbolic value (see the following valid values). The Space, Tab, and Comma format cause data to be returned as text with the specified delimiter between groups. Binary data will not be displayed in the History Box, but it can be redirected to a file.

0	Binary
1	Space

2	Tab
3	Comma

Example.

```
LA.DefineDataFormat Main GroupList:Address,Data Space
LA.DefineDataFormat Violation RawWithTimestamp Binary
LA.DefineDataFormat MagniVu AllGroupsWithoutTimestamp Comma
```

See Also. GetData (LA Module), GetData (DSO Module)

TPI Online Help:DefineDataFormat

DefineDataFormat (DSO Module)

Syntax. `DefineDataFormat` *`Components DataType`*

This command defines the options that will be used by subsequent GetData commands to the same module. The DefineDataFormat command is executed before the first GetData command. The options specified when you execute this command remain in effect until changed by a later DefineDataFormat command or until the connection to that TLA server is closed.

Components: This argument indicates which components of the data you want to get. Possible values are listed below. A <Channel> is a number, 1 through 4.

AllChannels

ChannelList:<Channel>,<Channel>,…

DataType: The argument indicates which format you want the data in when it is returned. You can use either the numeric or the symbolic value (see below). The Space, Tab, and Comma format cause data to be returned as text with the specified delimiter between groups. Binary data will not be displayed in the History Window, but it can be redirected to a file.

0	Binary
1	Space
2	Tab
3	Comma

Example.

```
DSO.DefineDataFormat AllChannels Space

DSO.DefineDataFormat ChannelList:1,2,3 Comma

DSO.DefineDataFormat ChannelList:1 Binary
```

See Also. GetData

TPI Online Help: DefineDataFormat

DefineRangeSymbolOptions

Syntax. `DefineRangeSymbolOptions` *`FileFormat SymbolTypes Reserved Bound1Bound2 OffsetType SymbolOffset`*

This command defines the options used by subsequent TPI LoadSymbolFile commands that load a Range type of symbol file (as opposed to a Pattern type of symbol file). The DefineRangeSymbolOptions command must be executed before the first LoadSymbolFile command. The options specified when you execute DefineRangeSymbolOptions command remain in effect until changed by a later DefineRangeSymbolOptions command or until the connection to that TLA server is closed.

If the DefineRangeSymbolOptions command is not executed, these options have the following default values:

```
DefineRangeSymbolOptions AutoFormat AllSymbols 0 0 FFFFFFFF DefaultOffset 0.
```

FileFormat: This argument specifies the format of the symbol file to be loaded. You can use either a numeric or a symbolic value. Valid values are:

0	AutoFormat
1	TSF
2	IEEE695
3	OMF86
4	OMF286
5	OMF386
6	COFF
7	ELF
8	OMF51
9	OMF166

NOTE. *AutoFormat causes the TLA software to look at the symbol file and automatically determine the symbol file type. TSF (text symbol file) is the proprietary TLA application text symbol file format (a .tsf file).*

SymbolTypes: This argument specifies the types of symbols to be loaded from the symbol file. There is a numeric value for each symbol type. To specify any single type or to specify all types you can use either the numeric or the symbolic value from the list below. To specify a combination of types you should use the numeric value that is the sum of the numeric values of the types you want to specify. For example, use the value 3 to specify FunctionSymbols and VariableSymbols.

1	FunctionSymbols
2	VariableSymbols
4	SourceCodeSymbols
8	ColorSymbols
–1	AllSymbols

Reserved: Reserved for future use.

Bound1,Bound2: These arguments specify the numeric range of the symbols that will be loaded. These strings are interpreted as hexadecimal values and must be in the range of 0 to FFFFFFFF.

OffsetType: The type of offset that is applied to the symbol values. You can use either the numeric or the symbolic value (see below). DefaultOffset applies only if File Format is TSF and indicates that the offset should be read from the symbol file. CustomOffset indicates that the custom offset specified in the SymbolOffset argument is to be used as the symbol offset.

0	DefaultOffset
1	CustomOffset

SymbolOffset: This is the custom offset value you use when a symbol file is loaded. This argument is ignored if the OffsetType argument is not CustomOffset. The value is interpreted as a hexadecimal value, up to 32-bits in width. The value can be preceded by a "–" (minus sign) to indicate that the offset should be subtracted from symbol values. If there is no minus sign in front of the number, or if there is a "+" (plus sign) in front of the number, then the specified offset is added to the symbol values when they are loaded. Examples of offsets: 1F000, +C0000, –50000000.

Examples.

```
DefineRangeSymbolOptions AutoFormat AllSymbols 0
0 FFFFFFFF CustomOffset 5000
DefineRangeSymbolOptions AutoFormat 13 0
7FFFFF FFFFFF DefaultOffset 0
DefineRangeSymbolOptions IEEE695 FunctionSymbols 0
0 FFFFFFFF CustomOffset -C000
```

See Also. LoadSymbolFile

TPI Online Help: DefineRangeSymbolOptions

DeleteChannelGroup (LA Module)

Syntax. `DeleteChannelGroup` *`UserChannelGroupName`*

This command deletes the channel group with the specified name.

UserChannel Group Name: This is the name of the channel group to be deleted.

Example. `LA.DeleteChannelGroup 28BitAddr`

See Also. TPI Online Help: DeleteChannelGroup

Disconnect

Syntax. `Disconnect` *`[server]`*

This command closes a connection to either the named server or, if no argument is specified, the current server.

If the connection to the current server is closed and there are one or more other connections remaining, one of those remaining connections will become the new current server. All connections are closed when the TLAScript application is terminated.

[server]: The name of the TLA application (server) you want to disconnect from.

Example.

```
Disconnect (Local server)

Disconnect MyTLA (Remote server)
```

See Also. Connections, <ServerName>

Echo

Syntax. *Echo Message*

This command echoes the message to the History Window if the TLAScript graphical interface is present. If the TLAScript application is executing a script in batch mode, the message in an Echo command is sent to standard out.

Message: This is the message to echo.

Example. `Echo "Getting data now."`

See Also. Pause, #Comment

Enabled (LA Module)

Syntax. `Enabled` *`[Value]`*

This command retrieves the enabled status of the module.

With no argument, this command gets the enabled status of the module, and with an argument, it sets the specified value for the module.

[Value]: True or False. This argument is optional.

Example.

```
LA.Enabled True
```

See Also. TPI Online Help: Enabled

Enabled (DSO Module)

Syntax. `Enabled` *`[Value]`*

With no argument, this command gets the enabled status of the module, and with an argument, it sets the specified value.

[Value]: True or False. This argument is optional.

Example.

```
DSO.Enabled True
```

```
DSO.Enabled
```

See Also. TPI Online Help: Enabled

Execute

Syntax. `Execute` *`Executable[command line arguments]`*

This command executes an external application as if from the MS-DOS command line.

The arguments to the Execute command are treated as if they were entered at the MS-DOS command prompt from the current directory. The external program can also be a non-executable file of the type that Windows knows how to open: for example, A.avi file.

TLAScript usually waits for the external program to exit before continuing, but this is dependent on your particular operating system and executable. If you want TLAScript to start the external program and not wait for it to exit, try using the Windows start command to start the executable.

Executable: The path to the external file to execute. This can be an actual executable or a file of the type that Windows knows how to open. Include any command line arguments that you need.

You need to include both the name of the executable and the name of the file which contains arguments of the Execute command.

If an argument to the Execute command contains an embedded space, enclose that argument in double quotes.

Example.

```
Execute C:\MyProgram.exe argument1

Execute start C:\MyProgram.exe argument1

Execute mspaint.exe test.bmp

Execute C:\Program Files\Accessories\mspaint.exe""C:\My Documents\test.bmp"
```

See Also. LoadScript

ExternalSignalIn

Syntax. `ExternalSignalIn` *`[SignalName]`*

This command gets the current value of the Internal Signal that is connected to the External Signal In. With an argument, it sets the specified value.

SignalName: This argument is optional. You can use either the numeric or symbolic value.Valid values are:

0	NoSignal
1	Signal1
2	Signal2
3	Signal3
4	Signal4

Example.

```
ExternalSignalIn Signal3
```

```
ExternalSignalIn
```

See Also. TPI Online Help: ExternalSignalIn

ExternalSignalOut

Syntax. `ExternalSignalOut` *`[SignalName]`*

This command gets the current value of the Internal Signal that is connected to the External Signal Out. With an argument, it sets the specified value.

SignalName: This argument is optional. You can use either the numeric or symbolic value. Valid values are:

0	NoSignal
1	Signal1
2	Signal2
3	Signal3
4	Signal4
5	10MHzClock

Example.

```
ExternalSignalOut 10MHzClock
ExternalSignalOut
```

See Also. TPI Online Help: ExternalSignalOut

ExternalSignalOutLowTrue

Syntax. `ExternalSignalOutLowTrue` *`[value]`*

This command gets the current value of the logical polarity for the External Signal Out. With an argument, it sets the specified value.

Value: True or False. A value of True means that the External Out Signal is low when asserted. This argument is optional.

Example.

```
ExternalSignalOutLowTrue True
ExternalSignalOutLowTrue
```

See Also. TPI Online Help: ExternalSignalOutLowTrue

Exit

Syntax. `Exit`

This command exits the TLAScript application.

This is unnecessary in most script files because scripts that were invoked from the command line automatically exit the TLAScript application upon successful completion.

Example. `Exit`

GetBeginTime (LA Module)

Syntax. `GetBeginTime` *`DataSet`*

This command returns the timestamp value, in picoseconds, of the beginning of the acquisition for the specified data set. This is the time relative to the start of the acquisition that the data set was enabled to acquire data.

DataSet: The LA Module stores three different sets of data. This argument specifies which of the three data sets you are interested in. You can use either the numeric or the symbolic value (see the following valid values).

0	Main
1	MagniVu
2	Violation

Example. `LA.GetBeginTime Main`

See Also. TPI Online Help: GetBeginTime

GetBeginTime (DSO Module)

Syntax. `GetBeginTime`

This command returns the timestamp value, in picoseconds, of the beginning of the module acquisition. This is the time relative to the start of the acquisition that the module was enabled to acquire data.

Example. `DSO.GetBeginTime`

See Also. TPI Online Help: GetBeginTime

GetBytesPerSample (LA Module)

Syntax. `GetBytesPerSample`

This command returns the number of bytes required for each sample of data when a binary data type is specified.

Example. `LA.GetBytesPerSample`

See Also. TPI Online Help: GetBytesPerSample

GetChannelGroup (LA Module)

Syntax. `GetChannelGroup` *`UserChannelGroupName`*

This method is used to get the channel list assigned to a channel group.

UserChannelGroupName: This is the user defined name of the channel group whose channel list is to be retrieved.

Example. `LA.GetChannelGroup 28BitAddr`

See Also. TPI Online Help: GetChannelGroup

GetChannelName (LA Module)

Syntax. `GetChannelName` *`HWChannelName`*

This command returns the user name of the channel with specified HW Channel Name.

HWChannelName: This is the hardware name of the channel whose user name is to be retrieved. For normal acquisition channels, the syntax is the hardware pod name followed by the channel number enclosed in parentheses, (for example A0(1), A0(2), and others). The syntax for clock and qualifier channels is the type identifier "CK" or "Q" respectively, followed by a number, (for example CK0, CK1, Q0, and others).

Example. `LA.GetChannelName A2(7)`

See Also. TPI Online Help: GetChannelName

GetCounterValue (LA Module)

Syntax. `GetCounterValue` *`CounterID`*

This command returns the final value of the specified counter from the previous acquisition.

CounterID: Use a value of 1 for Counter 1 and a value of 2 for Counter 2.

Example. `LA.GetCounterValue 1`

See Also. GetTimerValue

TPI Online Help: GetCounterValue

GetData (LA Module)

Syntax. `GetData` *`FirstSample NumSamples[>[>]file]`*

This command returns the requested samples from the LA Module acquisition memory.

This data can optionally be redirected to a file. It is necessary to execute the DefineDataFormat command before the GetData command because the DefineDataFormat command specifies the characteristics of the data to be returned in subsequent GetData commands.

FirstSample: This argument is the sample number of the first sample to be returned, relative to the beginning of the data set. The first sample in a data set is sample number 0.

NumSamples: This is the number of samples to return.

Examples.

```
LA.DefineDataFormat Main RawWithTimestamp Space

LA.GetData 0 25

LA.GetData 5000 25 > C:\MyData.txt
```

See Also. TPI Online Help: GetData

Remarks. For information on important performance considerations see Tips for Improving Data Transfer Performance.

GetData (DSO Module)

Syntax. `GetData` *`FirstSample NumSamples[>[>]file]`*

This command returns the requested samples from the DSO Module acquisition memory. The data can optionally be redirected to a file. It is necessary to execute the DefineDataFormat command before the GetData command because it specifies the characteristics of the data to be returned in subsequent GetData commands.

FirstSample: This argument is the sample number of the first sample to be returned, relative to the beginning of the module acquisition memory. The first sample is sample number 0.

NumSamples: The number of samples to return.

Examples.

```
DSO.DefineDataFormat AllChannels Space

DSO.GetData 0 25

DSO.GetData 5000 25 > C:\MyData.txt
```

See Also. DefineDataFormat

TPI Online Help: GetData

GetDataOffset (DSO Module)

Syntax. `GetDataOffset` *`Channel`*

This command returns the vertical offset for a channel in the current acquisition data. This value is used to interpret binary acquisition data obtained from the module.

Channel: The channel number, 1 through 4.

Example. *`DSO.GetDataOffset 1`*

See Also. GetDataRange

TPI Online Help: GetDataOffset

GetDataRange (DSO Module)

Syntax. `GetDataRange` *`Channel`*

This command returns the vertical range of the specified channel in the module's current acquisition data.

Channel: The channel number, 1 through 4.

Example. `DSO.GetDataRange 1`

See Also. GetDataOffset

TPI Online Help: GetDataRange

GetDataSamplePeriod (DSO Module)

Syntax. `GetDataSamplePeriod`

This command returns the sample period for the module's current acquisition data.

Example. `DSO.GetDataSamplePeriod`

See Also. TPI Online Help: GetDataSamplePeriod

GetDiagCalStatus

Syntax. `GetDiagCalStatus`

This command returns the power-on diagnostics and calibration status of the TLA application. The returned results do not include external oscilloscopes.

Example. `GetDiagCalStatus`

See Also. TPI Online Help: GetDiagCalStatus

GetEndTime (LA Module)

Syntax. `GetEndTime` *`DataSet`*

The command returns the timestamp value, in picoseconds, at the end of the acquisition for the specified data set. The time is relative to the start of the acquisition when the data set stopped acquiring data.

DataSet: The LA Module stores three different sets of data. This argument specifies which of the three sets of data to use. You can use either the numeric or the symbolic value (see the following valid values).

0	Main
1	MagniVu
2	Violation

Example. `LA.GetEndTime Main`

See Also. TPI Online Help: GetEndTime

GetEndTime (DSO Module)

Syntax. `GetEndTime`

The command returns the timestamp value, in picoseconds, at the end of the acquisition for the specified data set. The time is relative to the start of the acquisition when the data set stopped acquiring data.

Example. `DSO.GetEndTime`

See Also. TPI Online Help: GetEndTime

GetFirstModuleSlot

Syntax. `GetFirstModuleSlot`

This command returns the number of the first slot in the TLA mainframe.

Example. `GetFirstModuleSlot`

See Also. TPI Online Help: GetFirstModuleSlot

GetGroupNames (LA Module)

Syntax. `GetGroupNames`

This command returns the names of all the channel groups currently defined in the module setup.

Example. `LA.GetGroupNames`

See Also. TPI Online Help: GetGroupNames

GetGroupSize (LA Module)

Syntax. `GetGroupSize` *`GroupName`*

This command returns the number of channels in a specified channel group.

GroupName: This is the name of the group.

Example. `LA.GetGroupSize Main`

See Also. TPI Online Help: GetGroupSize

GetModuleNames

Syntax. `GetModuleNames`

This command returns the names of all logical modules in the TLA mainframe.

Example. `GetModuleNames`

See Also. TPI Online Help: GetModuleNames

GetModuleProperties

Syntax. `GetModuleProperties` *`SlotNum`*

This command returns the properties of the physical module located in the specified slot. For physical modules that occupy more than one slot, the same information is returned for all slots occupied by the module. The properties can include manufacturer, model, firmware version, diagnostic status, calibration status, speed, and memory depth. Some properties do not apply to some modules. This command cannot be used with external oscilloscopes.

SlotNum This argument identifies a slot number to query.

Example. `GetModuleProperties 5`

See Also. TPI Online Help: GetModulePropertiesBySlot

GetModuleSlotByName

Syntax. `GetModuleSlotByName` *`[ModuleName]`*

This command returns the slot number of the module with the specified ModuleName. This command cannot be used with external oscilloscopes.

ModuleName: This is the name of the module. Enclose the name within double quotes if it contains embedded spaces.

The slot number returned is the lowest numbered slot occupied by the physical module. If the named module is part of a merged set of modules, the slot number returned is that corresponding to the master module.

Example. `GetModuleSlotByName "LA 1"`

See Also. TPI Online Help: GetModuleSlotByName

GetModuleType

Syntax. `GetModuleType` *`SlotNum`*

This command returns the type of the physical module located in the specified slot. For physical modules that occupy more than one slot, the same information is returned for all slots occupied by the module. This command cannot be used with external oscilloscopes. Module types include:

Controller Module

LA Module

DSO Module

Expansion Interface Module

SlotNum: This argument prompts the slot number to query.

Example. `GetModuleType 1`

See Also. TPI Online Help: GetModuleTypeByslot

GetNumModuleSlots

Syntax. `GetNumModuleSlots`

This command returns the total number of slots in the TLA mainframe, including any slots in the expansion mainframes. The returned count does not include external oscilloscopes.

Example. `GetNumModuleSlots`

See Also. TPI Online Help: GetNumModuleSlots

GetNumSamples (LA Module)

Syntax. `GetNumSamples` *`DataSet`*

This command returns the number of unsuppressed samples. These unsuppressed samples reside in the acquisition memory of the module for that specified data set.

DataSet: The LA Module stores three different sets of data. This argument specifies which of the three sets of data to use. You can use either the numeric or the symbolic value (see the following valid values).

0	Main
1	MagniVu
2	Violation

Example. `LA.GetNumSamples Main`

See Also. TPI Online Help: GetNumSample

GetNumSamples (DSO Module)

Syntax. `GetNumSamples`

This command returns the number of samples in the module acquisition memory.

Example. `DSO.GetNumSamples`

See Also. TPI Online Help: GetNumSamples

GetRepetitiveStopReason

Syntax. `GetRepetitiveStopReason`

This command returns the reason why the last repetitive acquisition stopped. The reason will be one of the following:

Example. `GetRepetitiveStopReason`

See Also. TPI Online Help: GetRepetitiveStopReason

GetRunStatus

Syntax. `GetRunStatus`

The command returns "Running" if the logic analyzer is acquiring data. Otherwise the command returns "Idle."

Example. `GetRunStatus`

See Also. Run, Stop

TPI Online Help: GetRunStatus

GetStartTime (LA Module)

Syntax. `GetStartTime`

This command returns the start date and time for the previous acquisition.

Example. `LA.GetStartTime`

See Also. TPI Online Help: GetStartTime

GetStartTime (DSO Module)

Syntax. `GetStartTime`

This command returns the start date and time for the previous acquisition.

Example. `DSO.GetStartTime`

See Also. TPI Online Help: GetStartTime

GetSwVersion

Syntax. `GetSwVersion`

This command returns the software version of the server.

This command should not be confused with getting the software version of the TLAScript application (that information is in the About TLAScript box).

Example. `GetSwVersion`

See Also. TPI Online Help: GetSwVersion

GetTimerValue (LAModule)

Syntax. `GetTimerValue` *`TimerID`*

This command returns the final value of a specified timer from the previous acquisition.

TimerID: This argument uses a value of 1 for Timer 1 and a value of 2 for Timer 2.

Example. `LA.GetTimerValue 2`

See Also. GetCounterValue

TPI Online Help: GetTimerValue

GetTimestampMultiplier (LA Module)

Syntax. `GetTimestampMultiplier`

This command returns the number of picoseconds per "tick". When the acquisition data is returned in a raw format (using the GetData command), the timestamp information in the data is returned in units of "ticks" rather than in picoseconds. To calculate the actual timestamp value in picoseconds, use the returned number of picoseconds per tick; this is a constant for any given module type.

Example. `LA.GetTimestampMultiplier`

See Also. TPI Online Help: GetTimestampMultiplier

GetTriggerSample (LA Module)

Syntax. `GetTriggerSample` *`DataSet`*

This command returns the sample number of the trigger sample for the specified data set.

DataSet: The LA Module stores three different sets of data. This argument specifies which of the three data sets to use. You can use either the numeric or the symbolic value (see the following example).

0	Main
1	MagniVu
2	Violation

Example. `LA.GetTriggerSample MagniVu`

See Also. TPI Online Help: GetTriggerSample

GetTriggerSample (DSO Module)

Syntax. GetTriggerSample

This command returns the sample number of the trigger sample.

Example. `DSO.GetTriggerSample`

See Also. TPI Online Help: GetTriggerSample

GetTriggerTime (LA Module)

Syntax. `GetTriggerTime` *`DataSet`*

The command returns the time stamp value of the trigger in picoseconds. The time is relative to the start of the acquisition when the trigger occurred.

DataSet: The LA Module stores three different sets of data. This argument specifies which of the three data sets to use. You can use either the numeric or the symbolic value (see the following valid values).

0	Main
1	MagniVu
2	Violation

Example. `LA.GetTriggerTime Main`

See Also. TPI Online Help: GetTriggerTime

GetTriggerTime (DSO Module)

Syntax. `GetTriggerTime`

The command returns the time stamp value of the trigger in picoseconds. The time is relative to the start of the acquisition when the trigger occurred.

Example. `DSO.GetTriggerTime`

See Also. TPI Online Help: GetTriggerTime

Hide

Syntax. `Hide`

The command hides the TLA application window. However, the TLA application is still running.

Example. `Hide`

See Also. TPI Online Help: ShowWindow

LoadModule (LA Module)

Syntax. `LoadModule` *`Path ModuleName`*

The command loads the specified module setup file.

Path: This argument is the full path of the file from which the module setup is loaded. This path is relative to the TLA server file system since the TLA server ultimately interprets this path. Enclose this argument in double quotes if it includes embedded spaces.

ModuleName: This argument is the LA Module name in the specified file that is loaded. This module name is needed because the file may contain more than one LA module since it could have resulted from a SaveSystem command or from a SaveModule command. Enclose this argument in double quotes if the module name includes embedded spaces.

Examples.

```
LA.LoadModule C:\MyModule.tla "LA 1"
LA.LoadModule C:\MySystem.tla MyLA
```

See Also. LoadModule (DSO Module), LoadTrigger, LoadSystem

TPI Online Help: LoadModule, SaveModule (LA Module), SaveModule (DSO Module)

LoadModule (DSO Module)

Syntax. `LoadModule` *`Path ModuleName`*

The command loads the specified module setup file.

Path: The full file path which loads the module setup. This path must be relative to the TLA server's file system since the TLA server ultimately interprets this path. Enclose this argument in double quotes if it includes embedded spaces.

ModuleName: This argument is the name of the DSO Module in the specified file that is to be loaded. This is needed because the file may contain more than one module since it could have resulted from a SaveSystem command or from a SaveModule command. Enclose this argument in double quotes if the name includes embedded spaces.

Examples.

```
DSO.LoadModule C:\MyModule.tla "DSO 1"

DSO.LoadModule C:\MySystem.tla MyDSO
```

See Also. SaveModule (DSO Module), LoadSystem

TPI Online Help: LoadModule

LoadScript

Syntax. `LoadScript` *`ScriptFile`*

This command loads a script file and executes it.

This command also allows for something similar to a subroutine in a script file (calling a script from within a script). You can only have batch script style variables in a script that is executed from the command line invocation of TLAScript application. The file name passed as an argument to this command is relative to the machine on which the TLAScript application is running.

When the LoadScript command executes a script, TLAScript reads and executes each line of the script file as if it were entered by a user into the graphical interface. It shows all commands that are executed and stops when an error is encountered. In addition, if the script file contains an Exit command it will cause TLAScript to exit.

The LoadScript command is equivalent to using the Open… item in the File menu of the TLAScript menu bar.

ScriptFile: This is the path to the TLAScript script file to load and execute. This path should be relative to the machine on which the TLAScript application is running.

Example. `LoadScript C:\TLAScript\myscript.tls`

See Also. Execute

LoadTrigger (LA Module)

Syntax. `LoadTrigger` *`Path ModuleName`*

This command causes the module to load its trigger program from the specified file.

Path: This argument is the full file path from which the module trigger is loaded. This path must be relative to the TLA server file system since the TLA server ultimately interprets this path. Enclose this argument in double quotes if it includes embedded spaces.

ModuleName: This argument is the LA Module name in the specified file that is loaded. The module name is needed because the file may contain more than one LA module since the file could have resulted from a SaveSystem command or a SaveModule command. Enclose this argument in double quotes if the name includes embedded spaces.

Examples.

```
LA.LoadTrigger C:\MyModule.tla "LA 1"
LA.LoadTrigger C:\MySystem.tla MyLAModule
```

See Also. LoadModule (LA Module), LoadModule (DSO Module), LoadSystem

TPI Online Help: LoadTrigger, SaveModule (LA Module), SaveModule (DSO Module)

LoadSymbolFile

Syntax. `LoadSymbolFile` *`Path`*

This command causes the TLA application to load a specified symbol file. If the file is a Range symbol file, the options in the most recent DefineRangeSymbolOptions command will be used when the file is loaded.

Path: This argument is the full path to the symbol file being loaded. This path must be relative to the TLA server's file system since the TLA server interprets this path. Enclose this argument in double quotes if it includes embedded spaces.

Example.

```
LoadSymbolFile "C:\My Documents\MySymbols.tsf"
```

See Also. DefineRangeSymbolOptions

TPI Online Help: LoadSymbolFile

LoadSystem

Syntax. `LoadSystem` *`Path`*

This command causes the TLA application to load the specified saved system file.

Path: This argument is the full path of the file to be loaded. This path must be relative to the TLA server file system since the TLA server ultimately interprets this path. Enclose this argument in double quotes if it includes embedded spaces.

Example. `LoadSystem "C:\My Documents\MySystem.tla"`

See Also. SaveSystem, LoadModule (LA Module), LoadModule (DSO Module)

TPI Online Help: LoadSystem

Local Server

Local Server. A local server is usually the TLA (server) application that resides on the same machine that TLAScript is running on.

MemoryDepth (LA Module)

Syntax. `MemoryDepth` *`[value]`*

This command returns a value as specified in the following table. The return values are depend on the current user specified memory depth settings.

Value: This value is a constant that specifies the acquisition memory depths to be set, according to the following table. Attempting to set acquisition depths greater than that allowed by the acquisition module in use will result in an error.

Value	Memory depth
0	128 samples
1	256 samples
2	512 samples
3	1K samples
4	2K samples
5	4K samples
6	8K samples
7	16K samples
8	32K samples
9	64K samples
10	128K samples
11	256K samples
12	512K samples
13	1M samples
14	2M samples
15	4M samples
16	8M samples
17	16M samples
18	32M samples
19	64M samples
20	128M samples
21	256M samples
22	512M samples
23	1G samples

Example.

```
LA.MemoryDepth TLA700_MEMORY_DEPTH_4
LA.MemoryDepth
```

See Also. TPI Online Help: MemoryDepth

Name (LA Module)

Syntax. `Name` *[Name]*

This command with no argument returns the current name of the Module and with an argument it sets the module name to the specified name.

Name: This argument is the new module name and is optional.

Examples.

```
LA.Name
```

```
LA.Name PCIBus
```

See Also. TPI Online Help: Name

Name (DSO Module)

Syntax. `Name` *[Name]*

The command with no argument returns the current name of the Module and with an argument it sets the module name to the specified name.

Name: The new name for the module. This argument is optional.

Examples.

```
DSO.Name
```

```
DSO.Name MyScope
```

See Also. TPI Online Help: Name

Pause

Syntax. `Pause [/t time] // [message]`

This command pauses the execution of a script file.

There are two forms of this command. If the /t switch is used, TLAScript pauses execution for the specified time (units of seconds). If a message is used, the message is displayed in a message box and TLAScript waits for you to press the OK button of that message box. These two arguments can not be used together; whichever argument is first will be the argument that is used.

[/t *time]:* Pause for the specified number of seconds.

[message]: Display the message in a message box and pause for user confirmation.

Example.

```
Pause /t 25

Pause "New acquisition data has been acquired."
```

See Also. Echo, # Comment

Repetitive

Syntax. `Repetitive [Value]`

The command return value depends on the current user-specified repetitive status of the TLA application.

Value: True or False. This argument is optional.

Example.

```
Repetitive True

Repetitive
```

See Also. TPI Online Help: Repetitive

Run

Syntax. `Run [/w [WaitLength] ]`

This command causes the TLA application to run (start an acquisition).

With no arguments, Run causes the TLA application to run and immediately returns control to the TLAScript application, (it does not wait for the application to complete the acquisition). With the /w switch and the optional WaitLength argument, TLAScript will wait for the TLA to finish running, or wait for the specified length of time (WaitLength), in seconds, and then return control.

[/w [WaitLength]]: The /w switch without a WaitLength argument causes TLAScript to wait indefinitely for the TLA run to end (no commands are processed until the run is completed). If a WaitLength argument is included, TLAScript will wait at most WaitLength seconds for the run to complete. Control will be returned after WaitLength seconds if the run has not completed.

Examples.

```
Run

Run /w

Run /w 30
```

See Also. Stop, GetRunStatus, Repetitive

TPI Online Help: Run

RunCount

Syntax. `RunCount`

This command returns the number of times that the TLA application has run since the TLA application software was started. A repetitive acquisition is considered one run, regardless of how many acquisitions were actually performed during the repetitive run.

Example. `RunCount`

See Also. Run

TPI Online Help: RunCount

SaveModule (LA Module)

Syntax. `SaveModule` *`Path UserComments SaveData`*

The command saves the Module acquisition data to the specified file.

Path: The full path of the file where the module acquisition data is saved. This path must be relative to the TLA server's file system since the TLA server ultimately interprets this path. Enclose this argument in double quotes if it includes embedded spaces. It is recommended that you use the .tla file name extension.

UserComments: A string, enclosed in double quotes, which will be saved in the file and can be used later to help identify the file contents. Pass "" for no user comment.

SaveData: Indicates whether to save acquisition data along with the module setup information. You can use either the numeric or the symbolic value (see below).

0	NoData
1	AllData
2	UnsuppressedData

Examples.

```
LA.SaveModule C:\MyModule.tla:"My comments"NoData
```

```
LA.SaveModule "C:\My Documents\MyModule.tla" "" AllData
```

See Also. LoadModule (LA Module), LoadModule (DSO Module), LoadTrigger, SaveSystem

TPI Online Help: SaveModule

SaveModule (DSO Module)

Syntax. `SaveModule` *`Path UserComments SaveData`*

This command saves the Module to the specified file and optionally saves the acquisition data.

Path: The full file path where the module is saved. This path must be relative to the TLA server's file system since the TLA server ultimately interprets this path. Enclose this argument in double quotes if it includes embedded spaces. It is recommended that you use the .tla file name extension.

UserComments: This is a string, enclosed in double quotes, which is saved in the file and used later to identify the file contents. Pass "" for no user comment.

SaveData: This indicates whether to save acquisition data along with the module setup information. You can use either the numeric or the symbolic value (see below).

0	NoData
1	AllData
2	UnsuppressedData

Examples.

```
DSO.SaveModule C:\MyModule.tla:"My comments"NoData
```

```
DSO.SaveModule "C:\My Documents\MyModule.tla" "" AllData
```

See Also. LoadModule (LA Module), LoadModule (DSO Module), SaveSystem

TPI Online Help: SaveModule

SaveSystem

Syntax. `SaveSystem` *`Path UserComments SaveData`*

This command saves the current TLA system setup to the specified file.

Path: This is the full file path where the TLA system is saved. This path must be relative to the TLA server file system since the TLA server ultimately interprets this path. Enclose this argument in double quotes if it includes embedded spaces. It is recommended that you use the .tla file name extension.

UserComments: This argument is a string, enclosed in double quotes, which is saved in the file and is used later to help identify the file contents. Pass "" for no user comment.

SaveData: This argument indicates whether to save acquisition data along with the TLA system setup information. You can use either the numeric or the symbolic value (see below).

0	NoData
1	AllData
2	UnsuppressedData

Examples.

```
SaveSystem C:\MySystem.tla:"My comments"NoData

SaveSystem "C:\My Documents\MySystem.tla" "" AllData
```

See Also. LoadSystem, SaveModule (LA Module), Save Module (DSO Module)

TPI Online Help: SaveSystem

SetChannelGroup

Syntax. `SetChannelGroup` *`UserChannelGroupName [ChannelNameList]`*

This method is used to set the channel list for a channel group, or to create a new channel group and assign channels to it.

UserChannelGroupName: This is the user defined name of the channel group whose channel list is to be set. If no channel group with this name exists, one will be created.

ChannelNameList: This is the list of channel names to be assigned to the channel group. For individual channels the syntax is the hardware pod name followed by the channel number enclosed in parentheses, (for example A0(1), A0(2), and others). Groups of contiguous channels can be specified using a similar syntax, using a range of the channels in a hardware pod are to be specified, a shorthand notation is allowed using just an empty pair of parentheses, (for example A0(), A1(), and others). The syntax for clock and qualifier channels is the type identifier "CK" or "Q" respectively, followed by a number, (for example CK0, CK1, Q0, and others). Multiple channels or channel groups can be specified using a comma separated list. Embedded spaces are not allowed.

Examples.

```
LA.SetChannelGroup 28BitAddr A3(3-0),A2(),A1(),A0()

LA.SetChannelGroup Clks CK0

LA.SetChannelGroup Clks CK0,CK1,CK2,CK3

LA.SetChannelGroup Quals Q0,Q1,Q2,Q3

LA.SetChannelGroup EmptyGrp
```

See Also. TPI Online Help: SetChannelGroup

SetChannelName (LA Module)

Syntax. `SetChannelName` *`HWChannelName UserChannelName`*

This command sets the user name for the channel with the specified HW Channel Name.

HWChannelName: This is the hardware name of the channel whose user name is to be set. For normal acquisition channels, the syntax is the hardware pod name followed by the channel number enclosed in parentheses, (for example A0(1), A0(2), and others). The syntax for clock and qualifier channels is the type identifier "CK" or "Q" respectively, followed by a number, (for example CK0, CK1, Q0, and others).

UserChannelName: This is the user name to be assigned to the qualifier channel. The user name must be unique among channels, and must not contain embedded spaces or commas.

Examples. `LA.SetChannelName A2(7) Addr23`

See Also. TPI Online Help: SetChannelName

SetEventValue (LA Module)

Syntax. `SetEventValue` *`EventID EventValue`*

This command modifies the value(s) of the specified trigger event. You can only modify the values of Group, Word, Counter and Timer events.

EventID: Identifies which event in the trigger program is to be modified.

The syntax to specify the EventID is: <State>.<Clause>.<Event>.

For example, "1.2.3" specifies the third event of the second clause of the first state. The conditional storage clause is considered a special trigger state and is identified using "S.<Event>". For example, "S.2".

EventValue: The new value for the specified event. The Group, Word, Counter, and Timer Events are specified as follows:

- Group Events: <value> or <value>,<value> depending on whether the group event is a single or double-value (range) event. Values must be in hexadecimal, with X for "don't care".
- Word Events: <value>,<value>,<value>,… One value per channel group. Values must be in hexadecimal, with X for "don't care".
- Counter Events: <value>. The value must be in decimal.
- Timer Events: <value>. The value must be in picoseconds and in increments of 4 nanoseconds. The minimum valid value is 4 ns (4000).

Examples.

```
LA.SetEventValue 1.2.3 FF (Group Event)

LA.SetEventValue 2.4.1 5A5A,0000,XX (Word Event)

LA.SetEventValue 1.1.1 256 (Counter Event)

LA.SetEventValue 1.2.3 100000 (Timer Event)
```

See Also. LoadTrigger

TPI Online Help: SetEventValue

SetTriggerPosition (LA Module)

Syntax. `SetTriggerPosition` *`Position`*

This command sets the trigger position to the specified value.

Position: This argument is the desired position of the trigger as a percent of the total memory.Valid values are integers between 0 and 100, inclusive.

Example. `LA.SetTriggerPosition 50`

See Also. TPI Online Help: SetTriggerPosition

Show

Syntax. `Show`

This command shows the TLA window of the current TLA server if it was previously hidden. This command has no effect if the TLA application window is already being shown.

Example. `Show`

See Also. Hide

TPI Online Help: ShowWindow

Standard Out and Standard Error

TLAScript is not a console application; consequently, there are some limitations that exist when TLAScript is invoked from an MS-DOS command prompt window. Specifically, if you invoke TLAScript from a Command Prompt window and pass as an argument a script file, or if you use the /c <command> switch to have TLAScript execute a command. You would expect any output generated during the execution of that script file or command to be automatically directed to the Command Prompt window. Unfortunately, this is not what happens.

Instead, the output generated by the script or command is sent to standard out or standard error. However, standard out and standard error are not visible in the Command Prompt window. If you want to capture output in this situation, you need to redirect standard out and standard error to a file.

Example. The following command lines generate no visible output:

```
TLAScript /c GetSwVersion

TLAScript MyScript.tls
```

However, these command lines will capture the output in a file:

```
TLAScript /c GetSwVersion > output

TLAScript MyScript.tls > output
```


Stop

Syntax. `Stop`

This command causes the TLA application to stop running (stop acquiring data).

Example. `Stop`

See Also. Run, GetRunStatus

TPI Online Help: Stop

Index