Xgig Generator and Target Emulator

Version 8.1

User’s Guide

Xgig Generator and Target Emulator

Version 8.1

User's Guide

Viavi Solutions
1-844-GO-VIAVI
www.viavisolutions.com

Notice

Every effort was made to ensure that the information in this manual was accurate at the time of printing. However, information is subject to change without notice, and Viavi reserves the right to provide an addendum to this manual with information not available at the time that this manual was created.

Copyright/Trademarks

Copyright release

Reproduction and distribution of this guide is authorized for US Government purposes only.

Terms and conditions

Specifications, terms, and conditions are subject to change without notice. The provision of hardware, services, and/or software are subject to Viavi's standard terms and conditions, available at www.viavisolutions.com/en/terms-and-conditions.

Federal Communications Commission (FCC) Notice

This product was tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This product generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this product in a residential area is likely to cause harmful interference, in which case you will be required to correct the interference at your own expense.

The authority to operate this product is conditioned by the requirements that no modifications be made to the equipment unless the changes or modifications are expressly approved by Viavi.

Contents

About this Guide

Congratulations on your purchase of the Xgig Maestro software which, depending on the licenses you obtain, includes Xgig BERT, Xgig Jammer, Xgig Generator, Xgig Target Emulator, Delay Emulator, and Xgig Load Tester.

Who Should Read this Guide

This guide is intended for networking professionals in research and development who need to monitor and test network performance. It is assumed that users of this guide have an engineering background.

What this Guide Contains

This guide is organized into two parts:

- PART ONE: Using Xgig Generator
- PART TWO: Using Xgig Target Emulator

The chapters contain the following information:

Chapter 1, “Introducing Xgig Generator" describes the Generator features and capabilities.

Chapter 2, “Using the Generator Tab" describes the Generator tab interface and provides procedures to use it.

Chapter 3, “Using Generator with Ports" describes the elements of the Xgig Generator Device window and how to use them.

Chapter 4, “Using Generator in Edit Only Mode" describes the elements of the Xgig Generator Macro File and Functions File windows and how to use them.

Chapter 5, “Trace to Script Feature" describes the Trace to Script feature that generates a test script/program automatically from a captured trace.

Chapter 6, “Introducing Xgig Target Emulator" describes the Target Emulator features and capabilities.

Chapter 7, “Using the Target Emulator Tab" describes the Target Emulator tab interface and provides procedures to use it.

Chapter 8, “Using Target Emulator" describes the elements of the Xgig Target Emulator Device window and how to use them.

Appendix A, “Macro Variables and Functions" describes macro variables and provides a list of the various macro functions.

Conventions

The following conventions are used in this guide.

Message Formats

This guide uses the following format to highlight special messages:

Note: This format is used to highlight information of importance or special interest.

Caution: This format is used to highlight information that will help you prevent equipment failure or loss of data.

Typographical Conventions

This guide uses the following typographical conventions:

bold sans serif	Commands
italics	Directory names, book titles, named keys, for example the *Enter* key
`courier font`	Screen text, user-typed command-line entries

Technical Assistance

If you require technical assistance, call 1-844-GO-VIAVI (1-844-468-4284) or e-mail Techsupport-snt@viavisolutions.com.

For the latest TAC information, go to http://www.viavisolutions.com/en/services-and-support/support/technical-assistance.

PART ONE: Using Xgig Generator

Chapter 1

Introducing Xgig Generator

In this chapter:

Xgig Generator Overview

This chapter provides an overview of the Xgig Generator and lists its features. The Xgig Generator includes hardware and software that tests a device over a SAS or SATA domain by sending legal or illegal frames and traffic and comparing incoming payloads to expected data patterns. Xgig Generator combines powerful and reliable hardware with an easy-to-use software interface capable of testing all aspects of SAS and SATA protocol. The software development kit (SDK) includes an extensive function library for customizing the test environment.

Xgig Generator, used with the Xgig Analyzer software, provides a complete test solution including the ability to analyze the data generated and to capture and display SAS and SATA data in a customized trace view.

Generator ports only work in single chassis or master chassis in sync group. Locking generator ports in slave chassis is not supported.

New Features

The following new features have been released in this version of Xgig Generator.

- Xgig Generator provides 64-bit Dynamic and Static API libraries for the Generator Software Development Kit (SDK) library for use in 64-bit environments.
- Xgig Generator 32-bit client software is supported on both 32-bit and 64-bit versions of the Windows operating systems. Refer to the *Xgig Maestro Software Installation Guide* for the complete list of supported operating systems.
- Xgig Generator supports control from a computer (host) to either an Xgig 1000 chassis or an Xgig 5000 chassis (device) directly via a USB cable. Refer to the *Xgig Maestro Introduction Guide* for information.

Xgig Generator Features

Xgig Generator is part of the Xgig family of instruments. Generator has the following features:

- Ability to send Vendor unique SATA I/O Commands
- Ability to set SLUMBER/PARTIAL capable bit in SAS Identify Frame
- Calculate current clock tick counts while processing SCSI I/O Commands
- Send Customized Identify frame after OOB and Speed Negotiation
- Remove any previous error injection settings while reloading tester port
- SubPageCode parameter support for log select and log sense APIs
- Support for converting Bus Doctor Traces using the Trace To Script Tool and API
- Execute all SAS SMP commands based upon SAS 2.0 Specification
- Ability to send SAS Zone Configuration commands
- Ability to modify rejection type of received OPEN_ADDRESS Frames
- Send SAS/SATA command without SOF Frame
- Discover all devices and expanders located within Level 0 up to Level 3
- Supports SAS at 12G speed using 12G wide blade
- SAS Zoning has been added to allow users to configure the SAS zoning topology that connects to the Generator
- Supports SAS at 12G speed using Xgig 1000 Chassis
- Generator supports Tcl script editing.
- Generator supports 12G SAS tuning option with preset.
- The option for Tuning and Equalization for 6G SAS/SATA has been relocated to the context menu of the Parameters Status Table.

SAS/SATA Generator Application Programming Interface (API)

The Generator software allows you to write and compile tests using C/C++. This functionality requires that you be familiar with the API function library and the process of writing a compiled test. To this end, the Maestro software includes a Generator API Help system, available in the Help menu of the Maestro main window. Using this API Help system along with this manual will help you understand and control the functionality of the Generator software.

Chapter 2

Using the Generator Tab

In this chapter:

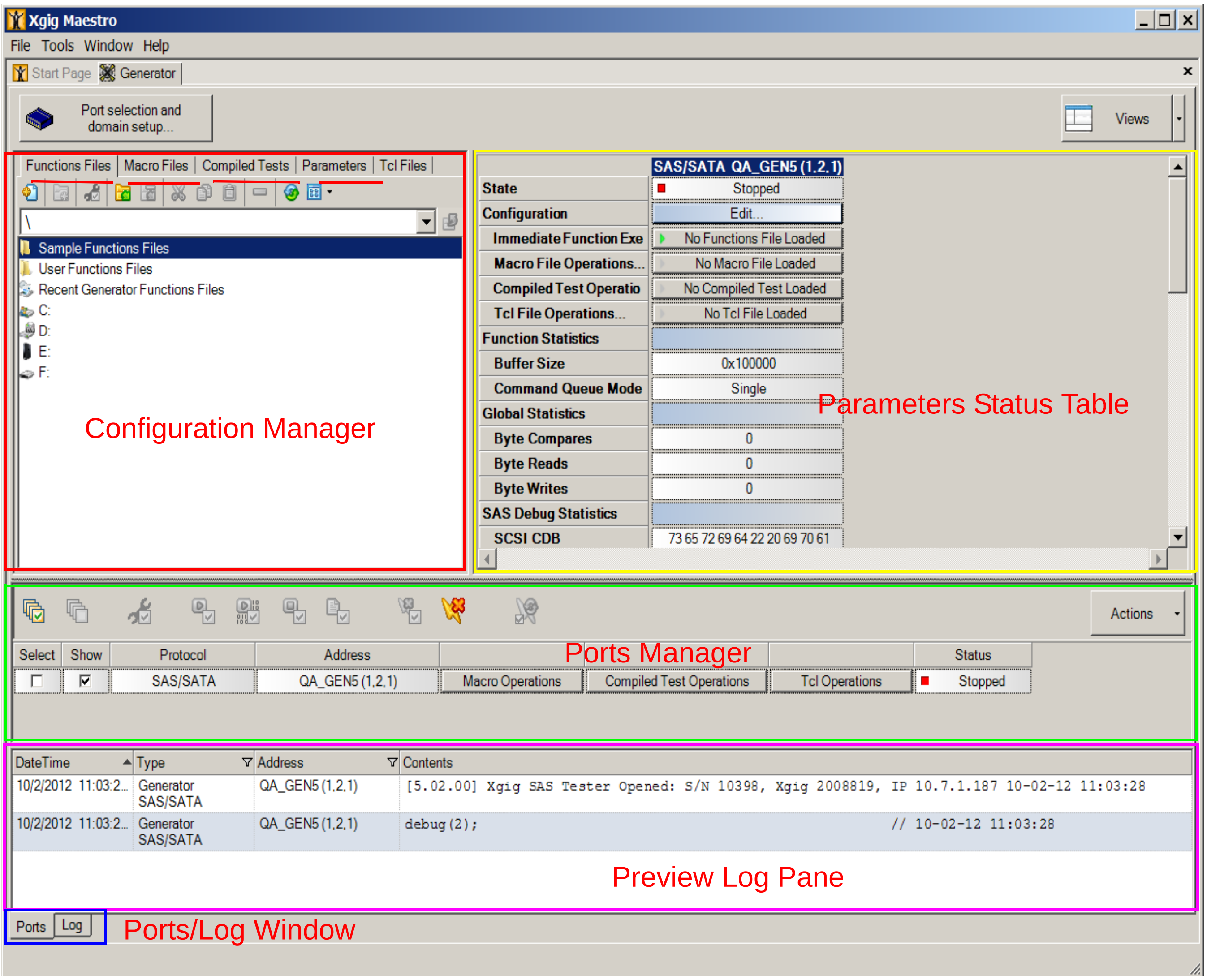
This chapter provides an overview of the Generator tab, on the Xgig Maestro main window, and its functions.

Note: This section refers to specific functions described in the Generator API Help System, which can be accessed by selecting Generator API Help from the main Help drop-down menu in the Maestro main window.

You should have launched Xgig Maestro and locked at least one device as described in *Maestro Introduction Guide*. The Xgig Maestro window is displayed with the Generator tab on the right as shown in Figure 1. This tab is where you operate the Generators you have locked.

The main menu is divided into panes similar to the Windows Explorer structure for ease of navigation and viewing. Right-click menus enable you to perform the desired task within a specific pane

Figure 1: Xgig Generator Tab on Xgig Maestro Window



Configuration Manager

Parameters Status Table

	SAS/SATA QA_GEN5 (1,2,1)
State	Stopped
Configuration	Edit...
Immediate Function Exe	No Functions File Loaded
Macro File Operations...	No Macro File Loaded
Compiled Test Operatio	No Compiled Test Loaded
Tcl File Operations...	No Tcl File Loaded
Function Statistics	
Buffer Size	0x100000
Command Queue Mode	Single
Global Statistics	
Byte Compares	0
Byte Reads	0
Byte Writes	0
SAS Debug Statistics	
SCSI CDB	73 65 72 69 64 22 20 69 70 61

Ports Manager

Select	Show	Protocol	Address				Status
□	☑	SAS/SATA	QA_GEN5 (1,2,1)	Macro Operations	Compiled Test Operations	Tcl Operations	Stopped

DateTime	Type	Address	Contents
10/2/2012 11:03:2...	Generator SAS/SATA	QA_GEN5 (1,2,1)	`[5.02.00] Xgig SAS Tester Opened: S/N 10398, Xgig 2008819, IP 10.7.1.187 10-02-12 11:03:28`
10/2/2012 11:03:2...	Generator SAS/SATA	QA_GEN5 (1,2,1)	`debug(2);                                  // 10-02-12 11:03:28`

Preview Log Pane

Ports/Log Window

The Generator tab of the Maestro window contains the following components as shown:

Configuration Manager, outlined in red, contains the function tabs and the parameters tab. These tabs are described in “Using the Configuration Manager Function Tabs in Generator” on page 12 and “Displaying and Hiding Parameters in Generator” on page 38.

The Parameters Status table, outlined in yellow, provides status information for the ports you have locked. This table is described in “Using the Parameters Status Table in Generator” on page 13.

The Ports Manager, outlined in green, allows you to control settings and operations for the ports you have locked. This pane is described in “Using the Ports Manager in Generator” on page 28.

The Preview Log pane, outlined in magenta, is a quick reference for the Log Manager. This pane is described in “Using Log Manager in Generator” on page 31.

The Ports and Log window tabs, outlined in blue, switch between the Ports () and Log () windows.

Using the Configuration Manager Function Tabs in Generator

There are five function tabs in the Configuration Manager: **Function Files**, **Macro File**, **Compiled Tests**, **Parameters**, and **Tcl Files**. The files displayed in the manager are specific to the function tab displayed. The Configuration Manager provides a list of files organized in the following three folders:

- **Sample Files** includes configurations that are provided with the application.

Note: The two sample compiled test files provided are described in "Sample Compiled Test Files" on page 65.

- **User Configurations** is the default location where files you create are saved.
- **Most Recently Used** allows you easy access to those configuration files most recently used.

The Configuration Manager also includes a list of all the drives on your system allowing you to locate any configuration files on your system quickly.

The tool bar in the Configuration Manager contains the following icons and functionality:

New File allows you to create a new functions file from the **Functions File** tab or to create a new macro file from the **Macro Files** tab. This icon is not present on the Compile Test tab because you cannot create compiled test files from the GUI.

Load allows you to load the selected functions, macro, or compiled test file into a port. This icon is only active when a port has been locked and is selected in the Port Manager. You can also drag-and-drop a file from the Configuration Manager into a port.

Map allows you to select a folder that you want to be listed in the Configuration Manager for the selected function tab.

Unmap allows you to unmap a folder that has been mapped.

Cut allows you to cut a file from its current location. This is not the same as deleting a file.

Copy allows you to copy a file.

Paste allows you to paste a file you have cut or copied.

Path allows you to go up one level in the folder tree. This field next to this icon displays the current folder name.

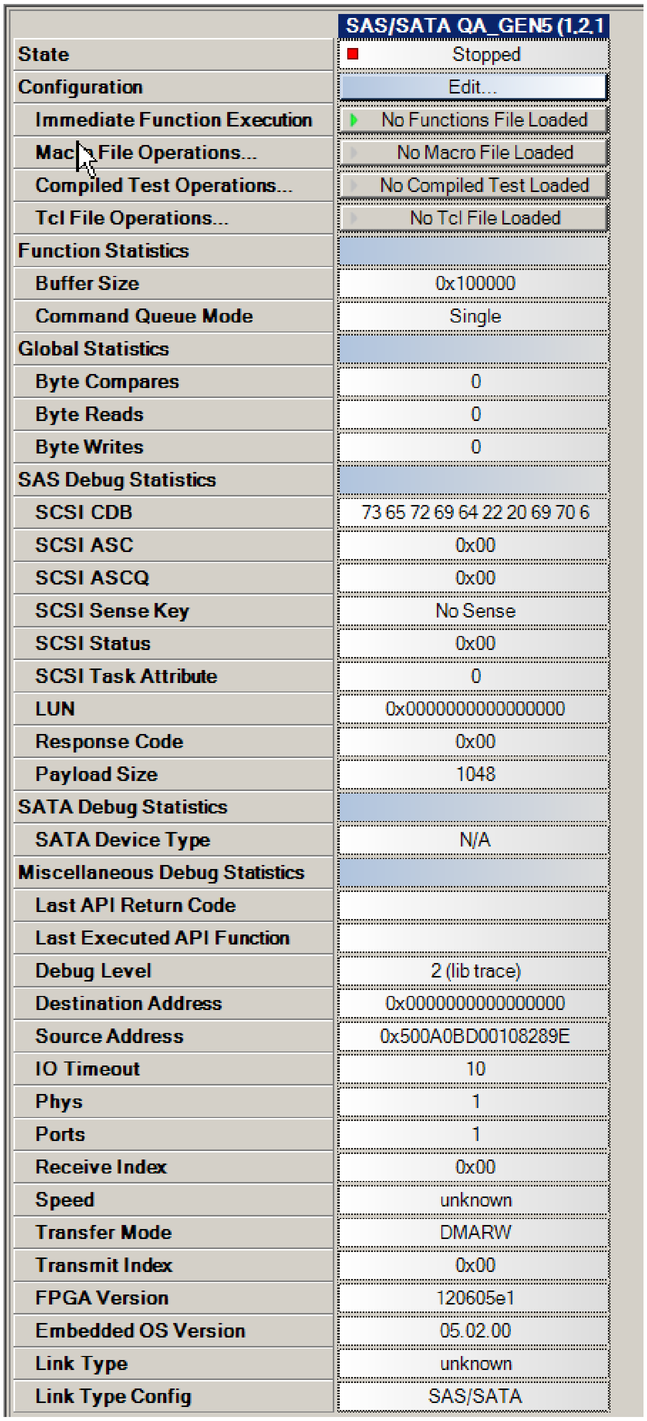
Show/Hide allows you to show or hide the details for the selected file. These details appear in the lower section of the Configuration Manager pane.

View allows you to view the files in the Configuration Manager as a list including only the file name or to view the details of the files in the list.

Using the Parameters Status Table in Generator

The ports you have locked are displayed as vertical columns in the Generator Parameters Status table. At the top of each column is the protocol, the chassis name, and in parenthesis the chassis number, the slot number and port numbers in parenthesis.

Figure 2: Generator Parameters Status Table



	SAS/SATA QA_GEN5 (1,2,1
State	🟥 Stopped
Configuration	Edit...
Immediate Function Execution	▶ No Functions File Loaded
Macro File Operations...	No Macro File Loaded
Compiled Test Operations...	No Compiled Test Loaded
Tcl File Operations...	No Tcl File Loaded
Function Statistics	
Buffer Size	0x100000
Command Queue Mode	Single
Global Statistics	
Byte Compares	0
Byte Reads	0
Byte Writes	0
SAS Debug Statistics	
SCSI CDB	73 65 72 69 64 22 20 69 70 6
SCSI ASC	0x00
SCSI ASCQ	0x00
SCSI Sense Key	No Sense
SCSI Status	0x00
SCSI Task Attribute	0
LUN	0x0000000000000000
Response Code	0x00
Payload Size	1048
SATA Debug Statistics	
SATA Device Type	N/A
Miscellaneous Debug Statistics	
Last API Return Code	
Last Executed API Function	
Debug Level	2 (lib trace)
Destination Address	0x0000000000000000
Source Address	0x500A0BD00108289E
IO Timeout	10
Phys	1
Ports	1
Receive Index	0x00
Speed	unknown
Transfer Mode	DMARW
Transmit Index	0x00
FPGA Version	120605e1
Embedded OS Version	05.02.00
Link Type	unknown
Link Type Config	SAS/SATA

The first column on the tab is the legend for each row in the device columns and contains the following categories:

- State
- Configuration
- Function Statistics
- Global Statistics
- SAS Debug Statistics
- SATA Debug Statistics
- Miscellaneous Debug Statistics

Parameters Category Descriptions for Generator

State

This category displays the current state of the port, i.e. stopped, stopping, running, disconnected, connecting, etc.

Configuration

This category opens the edit window for macros, functions files, compiled tests, SAS Topology, and Tcl scripts. From this edit window you can create or edit macro files and functions files, define parameters for an existing compiled test file, view and configure SAS Zoning topology, and edit Tcl scripts.

Immediate Function Execution

Shows the currently loaded functions file. If no functions file is loaded, the field will read, “`No Functions File Loaded`”. Clicking this field, with or without a functions file loaded, displays the function library and allows you to select a single function and execute it using the **Start** ▷ button. See “Executing a Function on a Generator Device” on page 51” for more information on this functionality.

Macro File Operations

Shows the currently loaded macro file. Clicking the file name allows you to either execute the macro or edit the macro file. If no macro file is loaded, the field will read, “`No Macro File Loaded`”.

Compiled Test Operations

Shows the currently loaded compiled test file. Clicking the file name allows you to either execute the compiled test or edit the compiled test by defining parameters for the test. If no compiled test file is loaded, the field will read, “`No Compiled Test Loaded`”.

➡ **Note:** To remove the configuration for a port, right-click the device column in the Parameters Status table, and select **Reset Configuration**. This effectively “unloads” the functions and macro, and compiles test files that you have loaded into the port.

Tcl File Operations

Shows the currently loaded Tcl file. Clicking the file name allows you to either execute the Tcl file or edit the Tcl file. If no Tcl file is loaded, the field will read, “`No Tcl File Loaded`”.

Function Statistics

This category provides information about the settings that define how functions are executed.

Buffer Size

Displays the size of the separate transmit and receive buffers. When a data transfer reaches the end of a buffer, it wraps to the beginning and overwrites previous data (receive) or resends previous data (transmits). The buffer size is user configurable and can be changed by using the `resize_buf()` function.

Command Queue Mode

Displays the current command queue mode for SCSI I/O transfers, i.e. Single, Auto, or Stack. Refer to the `cmdque_mode()` function. The default setting is Single.

Global Statistics

This category provides information about whether bytes are being processed within specified limits.

Byte Compares

Displays the total number of bytes compared since being reset by the `stats_reset()` function or since the start of the current test session.

Byte Reads

Displays the total number of bytes read since being reset by the `stats_reset()` function or since the beginning of the current tester session.

Byte Writes

Displays the total number of bytes written since being reset by the `stats_reset()` function or since the beginning of the current Generator testing session.

SAS Debug Statistics

This category provides information to help determine the cause of errors occurring during Generator testing.

SCSI CDB

These 16 bytes are the current values contained in the most recently sent SCSI command.

SCSI ASC

Sense key information (according to the SCSI-3 specification) returned in the latest response frame.

SCSI ASCQ

Sense code qual information (according to the SCSI-3 specification) returned in the latest response frame.

SCSI Sense Key

Sense key information (according to the SCSI-3 specification) returned in the latest response frame.

SCSI Status

The value of the SCSI status byte returned in the latest response frame. When a command is issued to a target, the status value is set to 0xFF, which is changed when the command is complete.

SCSI Task Attribute

The SCSI queuing mode or task attribute used in SCSI command frames. This can be changed with the `queue_xx()` functions. Refer to the `queue_aca`, `queue_head`, `queue_ordered`, `queue_simple`, and `queue_untagged` functions for more information.

LUN

A decimal value indicating the current logical unit number (LUN) to be used in command frames during SCSI operations.

Response Code

Response information code from the response frame of the most recently completed command.

Payload Size

The maximum payload (or frame) size for SAS.

SATA Debug Statistics

SATA Device Type

SATA device type, either SATA host or SATA device. Refer to the `get_g_info` function for more information about `sata_device_type`.

Miscellaneous Debug Statistics

Last API Return Code

Indicator to determine the success of a function. Refer to the *Generator API Help* system for a list of possible returns for a specific function and the definitions for each return.

Last Executed API Function

Indicates the most recent API function that was executed. This includes single functions as well as functions in functions files and macro files.

Debug Level

Indicates the current debug level. The default value is 2- Display trace of library calls.

To change the debug level:

1 Select **Debug level** from the context menu.

2 Select the debug level you want: 0 (no tracing), 2 (lib trace), 3 (driver trace), 4 (lib and driver trace), or 6 (lib trace and return codes).

Destination Address

A hexadecimal value indicating the current destination/target ID to be used during SAS operations.

Source Address

A hexadecimal value indicating the current source/initiator ID to be used during SAS operations.

IO Timeout

The length of time to be completed by an operation before an I/O timeout is generated. The timeout value (ioto) is displayed in seconds. At the beginning of an operation, such as the start of a SCSI command, an internal timer starts. While waiting for the operation to progress, the code checks the timer and returns to operator control after ioto seconds have elapsed. For long operations, (such as a FORMAT command), set the timeout value to an appropriately high value.

Phys

The number of phys available on or supported by the tester in use.

Ports

The number of logical ports currently configured on the tester in use. This number can never be more than the number of phys. Currently, Generator can only set port width to 1. Refer to the `sas_set_port_width()` function for more information.

Receive Index

The current index (or offset) in the receive buffer.

Speed

The data transfer rate of the Generator: 1.5G, 3.0G or 6.0G SATA or 1.5G, 3.0G, 6.0G or 12.0G SAS. Refer to the `set_speed()` and `set_speed_ex()` functions for more information.

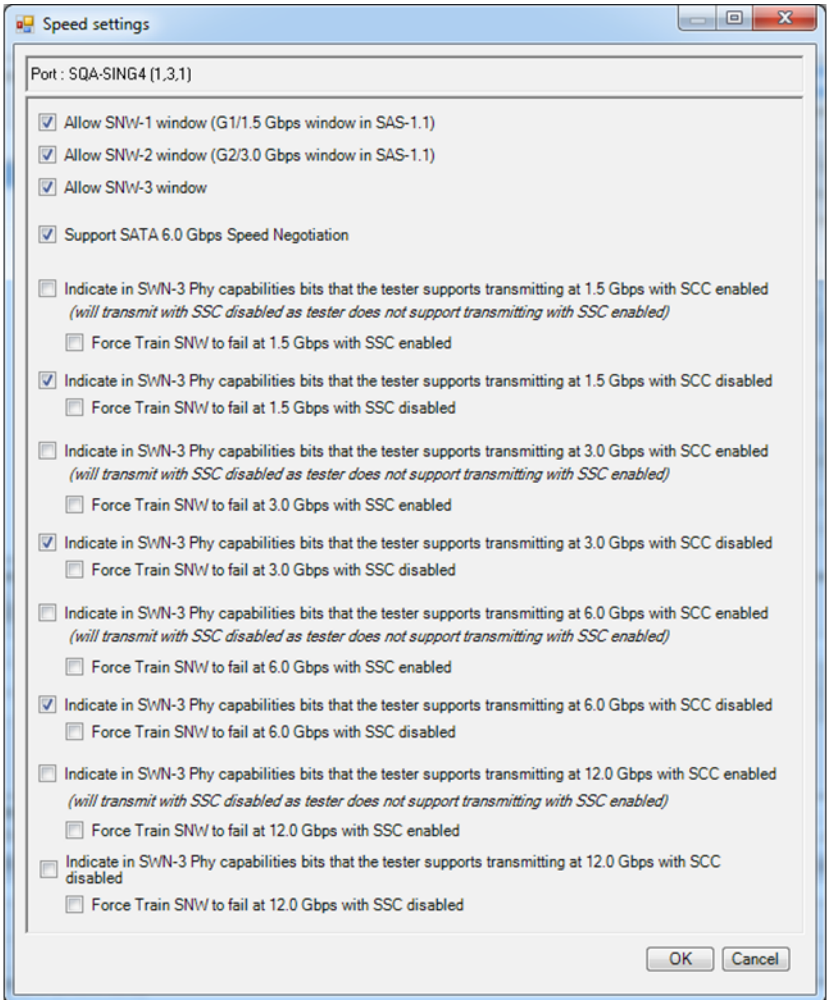
Note: The 1.5G for SAS ports on the 12G blade is not a supported selection, and the results will be unpredictable. Therefore, you should not set this value.

To change the speed settings:

1 Select **Edit speed settings** from the context menu.

The Speed Settings window appears.

Figure 3: Speed Settings Window



Speed settings

Port : SQA-SING4 (1,3,1)

☑ Allow SNW-1 window (G1/1.5 Gbps window in SAS-1.1)
☑ Allow SNW-2 window (G2/3.0 Gbps window in SAS-1.1)
☑ Allow SNW-3 window

☑ Support SATA 6.0 Gbps Speed Negotiation

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 1.5 Gbps with SCC enabled
(will transmit with SSC disabled as tester does not support transmitting with SSC enabled)
☐ Force Train SNW to fail at 1.5 Gbps with SSC enabled

☑ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 1.5 Gbps with SCC disabled
☐ Force Train SNW to fail at 1.5 Gbps with SSC disabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 3.0 Gbps with SCC enabled
(will transmit with SSC disabled as tester does not support transmitting with SSC enabled)
☐ Force Train SNW to fail at 3.0 Gbps with SSC enabled

☑ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 3.0 Gbps with SCC disabled
☐ Force Train SNW to fail at 3.0 Gbps with SSC disabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 6.0 Gbps with SCC enabled
(will transmit with SSC disabled as tester does not support transmitting with SSC enabled)
☐ Force Train SNW to fail at 6.0 Gbps with SSC enabled

☑ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 6.0 Gbps with SCC disabled
☐ Force Train SNW to fail at 6.0 Gbps with SSC disabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 12.0 Gbps with SCC enabled
(will transmit with SSC disabled as tester does not support transmitting with SSC enabled)
☐ Force Train SNW to fail at 12.0 Gbps with SSC enabled
☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 12.0 Gbps with SCC disabled
☐ Force Train SNW to fail at 12.0 Gbps with SSC disabled

2 Check the SAS-1.1 or SATA 1.5G link rate to force the link to negotiate to 1.5 Gbps.

3 Check the SAS-1.1 or SATA 3.0G link rate to force the link to negotiate to 3.0 Gbps.

4 Check both of the two check boxes above to allow the link to auto-negotiate to the highest available speed.

5 Select Allow SNW-3 Window (SAS-2) if you have a SAS device that you want to run at 6.0 or 12.0 Gbps. This type of speed negotiation allows for adaptation at both 1.5G and 3G link rates.

6 Select Support SATA 6.0 Gbps Speed Negotiation to allow the SATA link to auto-negotiate to the highest available speed.

➡ **Note:** If you select the “Support SAS-2 speed negotiation setting”, then the Phy capabilities and Fail train sections become active and allow you to change the settings. Otherwise, these sections are inactive as these settings are not applicable.

The phy capabilities are transmitted by the tester during the SAS-2 SNW3 window at the selected speed. Refer to the `get_speed_ex` and `set_speed_ex` functions in the *Generator API Help*.

This setting defines what you want to convey to a device in the SNW-3 Phy capabilities bits. You can choose to indicate that the tester supports transmitting at a given speed with SSC enabled or with SSC disabled. We do not currently support transmitting with SSC enabled. Therefore, the tester will transmit with SSC disabled even if you set the tester to indicate otherwise.

7 If applicable, set the desired Phy capabilities.

The Fail train is a bitmask that forces the corresponding Train Speed Negotiation Windows (SNW) to fail at the selected speed. Refer to the `get_speed_ex, sas_set_timers`, `set_speed_ex` functions in the *Generator API Help*.

This setting forces the train SNW to fail at a given speed with SSC enabled or with SSC disabled. setting indicates fail train SNW can be set with SSC disabled. We do not currently support forcing the train SNW to fail with SSC enabled. Therefore, the tester will force the train SNW to fail with SSC disabled even if the settings indicate otherwise.

8 If applicable, set the desired Fail train SNW settings.

9 Click **OK** to close the dialog and apply the settings.

Transfer Mode

The type of data transfer for I/O functions. Refer to the hgen_mode() function for more information.

Transmit Index

The current index (or offset) in the transmit buffer.

FPGA Version

Represents a bit file revision code for the Xgig blade’s FPGA.

Embedded OS Version

Indicates the version of Maestro software running on the Xgig chassis.

Link Type

Current link type: Unknown, SAS link, or SATA link. Refer to the `get_g_info` function for more information about `link_type`.

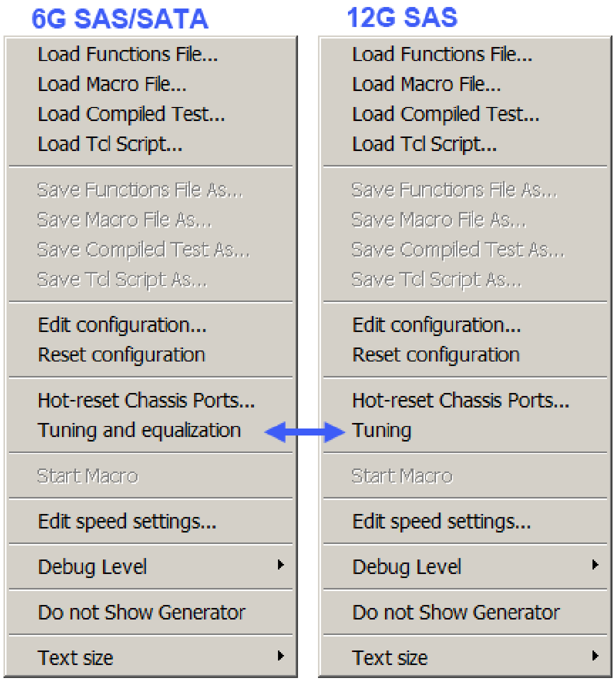
Link Type Config

Current link type configuration: SAS only, SATA only, or SAS and SATA. Refer to the `get_g_info` function for more information about `link_type_cfg`.

Using the Parameters Status Table Context Menu

Right-click on a parameters status column to display the context menu. The available operations are displayed for the column in which you click. This menu provides controls for configuration files, operation, setting clock speed, setting Generator mode of operation (SAS to SATA), removing a Generator from the Parameters Status table, and changing the appearance of the status table.

Figure 4: Parameters Status Table Context Menu for Generator



Loading Files

To load a Functions File, click **Load Functions File...**

To load Macro File, click **Load Macro File...**

To load a Compiled Test, click **Load Compiled Test...**

To load a Tcl Script, click **Load Tcl Script...**

Saving Files

To save the current Functions file, click **Save Functions File As...**

To save the current Macro File, click **Save Macro File As...**

To save the current Compiled Test, click **Save Compiled Test As...**

To save the current Tcl Script, click **Save Tcl Script As...**

Editing Configuration

To edit the current port configuration, click **Edit configuration...**

To reset the port configuration to the default settings, click **Reset configuration**.

Resetting Chassis Ports

To do a hot-reset on the selected port, click on **Hot-reset Chassis Ports...**

This will reset all locked Generator ports and clear the settings applied to the port. You need to reconnect to the port again to use it.

Tuning and Equalization (*6G SAS/SATA only*)

Tuning and Equalization are procedures to optimize ports on both wide blade (Xgig SAS/SATA Wide-Port /4x Blade) and narrow blade (Xgig 6 Gigabit SAS/SATA Blade) for your environment. When you connect the Xgig SAS Analyzer, Jammer, or Generator in a new environment (for example, after changing devices or cable length) we recommend that you run the Tuning procedure detailed in the *Xgig Tuning and Equalization* document.

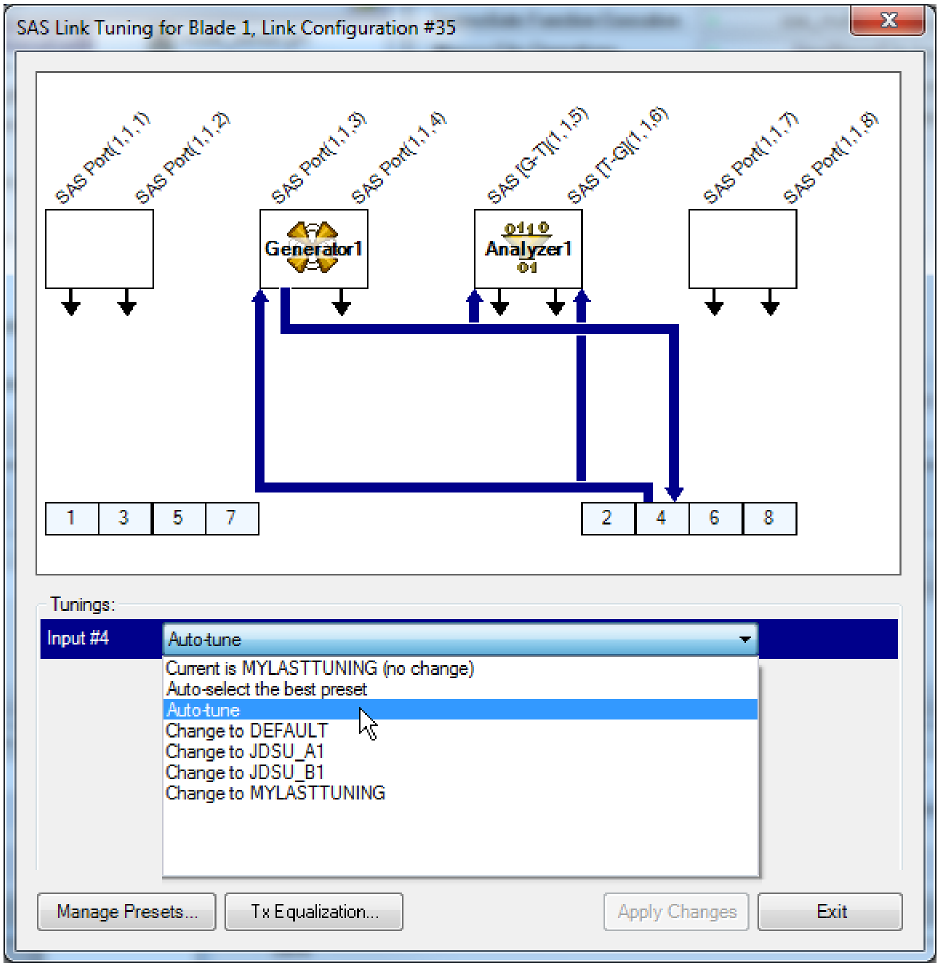
Tuning (12G SAS only)

This option is for 12G SAS only. Click **Tuning** to open the **SAS Link Tuning** dialog (see Figure 4 on page 20.) If two blades are being used, you must tune each blade independently by launching the **SAS Link Tuning** dialog twice.

The 12G **SAS Link Tuning** dialog introduces a new concept of port groups. It still focuses on the tuning of a single 12G blade, but the ports on the blade are aggregated into port groups. The port groups are named "Input #n" or "Output #n", referring to ports on the mini-SAS connectors. The dialog shows the port group for the selected configuration and allows you to load a tuning preset, auto-select the best preset, or auto-tune.

The top portion of the **SAS Link Tuning** dialog displays the lane steering diagram for the current link configuration. It is the same dialog as visible inside the **Discovery** dialog's Link Configuration, except that the wire colors uniquely identify port groups. See Figure 5.

Figure 5: SAS Link Tuning Dialog



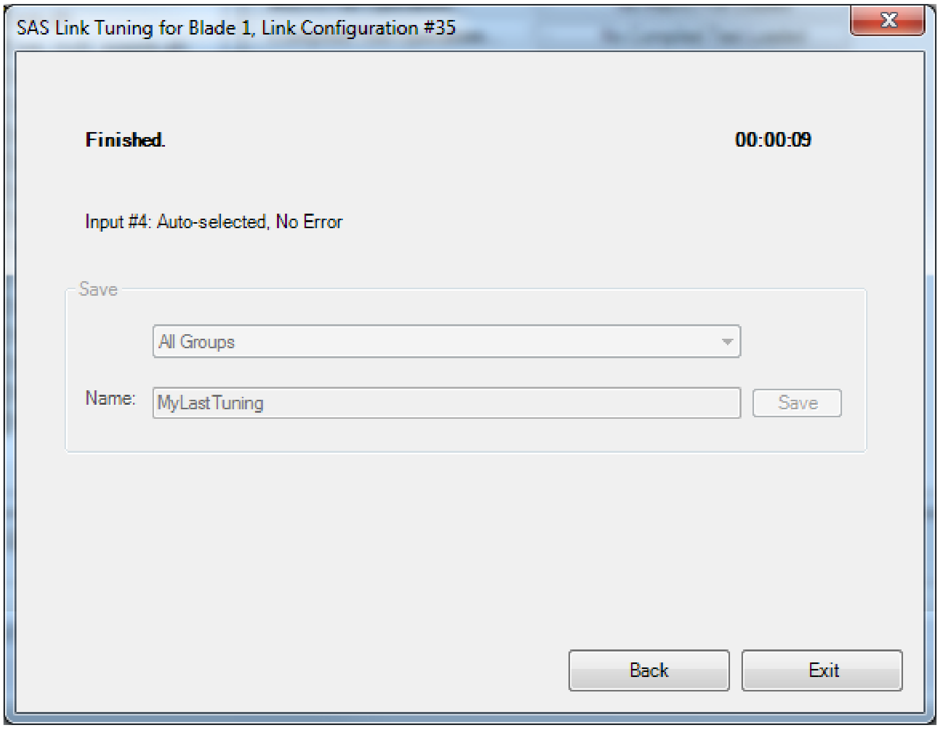
Use the drop-down menu to select a tuning method. You can choose from the following options:

- Auto select the best preset
- Auto-tune
- Change to [preset name]

Auto select the best preset

To automatically select the best preset, select **Auto-select the best preset** from the Tunings drop-down menu. Then, click the **Apply Changes** button. The software selects the best preset from the list of saved presets and applies that tuning preset to the hardware. This selection is based on the preset with the lowest bit error rate (BER). When the preset selection is complete, the following dialog is displayed showing the selected preset.

Figure 6: Auto-Select Complete Dialog



SAS Link Tuning for Blade 1, Link Configuration #35

Finished. **00:00:09**

Input #4: Auto-selected, No Error

Save

All Groups

Name: MyLastTuning

The **Save** area is reserved for future use.

Click the **Back** button to close the dialog and return to the **SAS Link Tuning** dialog where the name of the preset selected is shown as the current tuning.

Click the **Exit** button to close the dialog and return to the **Generator** tab on **Xgig Maestro** window.

Auto-tune

Auto-tuning in Generator is supported by the following configurations:

#35 – G-A

#41 – G-J-A,A

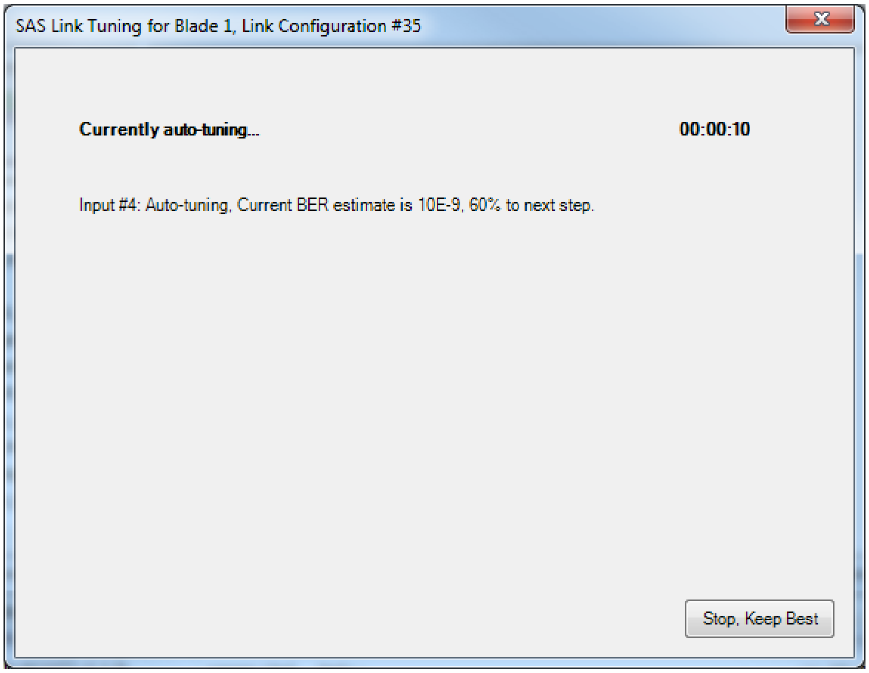
#49 – G-A,G,G

#55 – G-A, G-A

Before starting the auto-tuning process, try applying the best preset. If you are not sure which one is best, perform the **Auto-Select the best preset** process. This may shorten the time it takes to auto-tune. To auto-tune all the ports in the configuration loaded on the hardware for the ports you have selected, select **Auto-tune** from the Tunings drop-down menu. Then, click the **Apply Changes** button to begin the auto-tuning process. Note that when a configuration is selected in the **Discovery** dialog, auto-tuning will tune all the ports in that configuration regardless of whether the other ports in the configuration are locked or not.

A dialog appears showing the tuning progress. It includes the duration of the tuning and the bit error rate (BER).

Figure 7: Auto-tuning In Progress Dialog



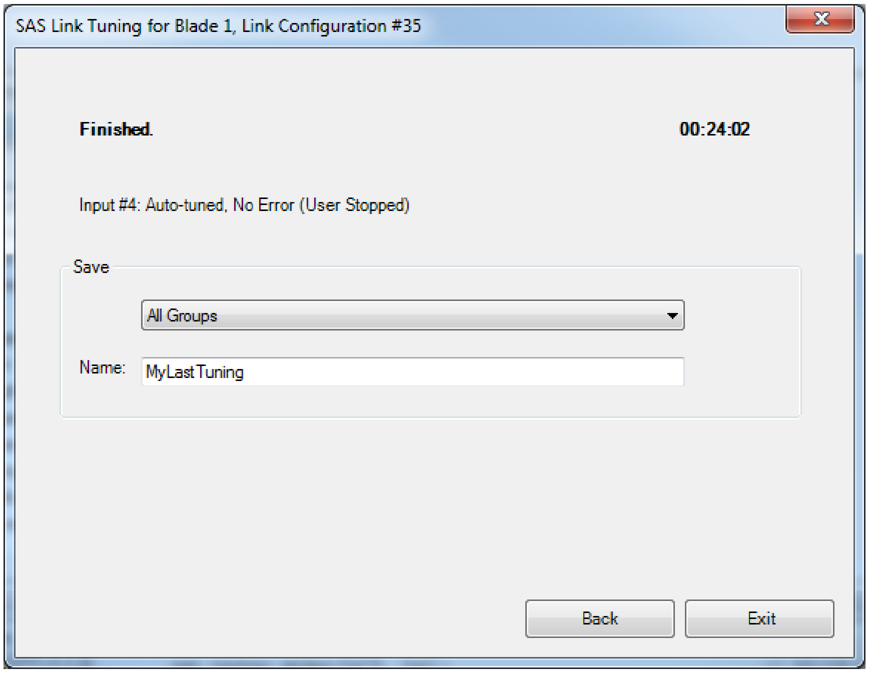
SAS Link Tuning for Blade 1, Link Configuration #35

Currently auto-tuning... 00:00:10

Input #4: Auto-tuning, Current BER estimate is 10E-9, 60% to next step.

The auto-tuning process concludes only when the ports are error free for the longest of a series of intervals. This may take six hours or more. You can stop the auto-tuning process at any time and keep the best settings found thus far by clicking the **Stop, Keep Best** button. In this case, the dialog shown below appears.

Figure 8: Auto-tuning Stopped Dialog



SAS Link Tuning for Blade 1, Link Configuration #35

Finished. 00:24:02

Input #4: Auto-tuned, No Error (User Stopped)

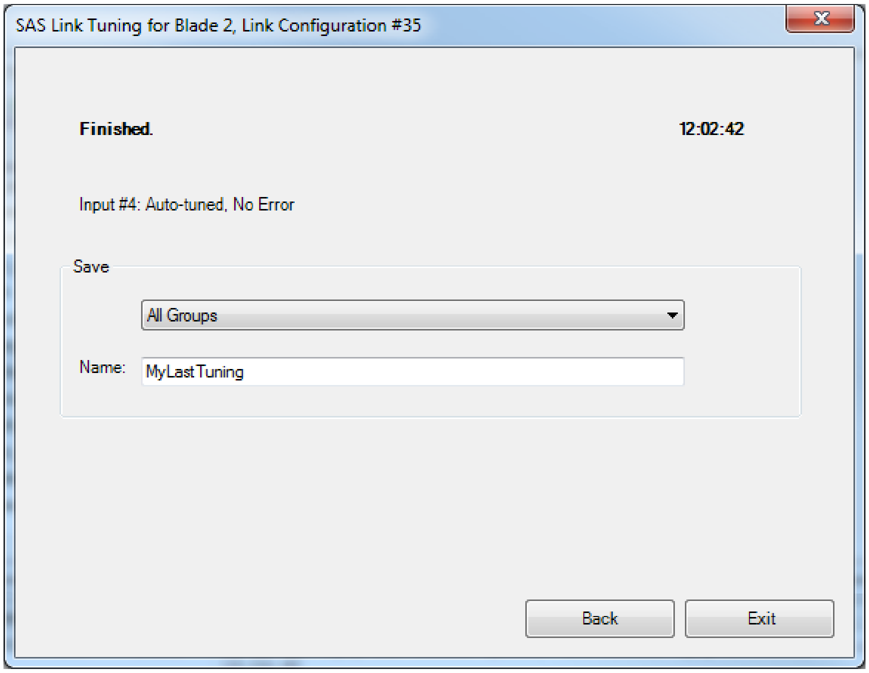
Save

All Groups

Name: MyLastTuning

When the auto-tuning process has completed, the following dialog appears showing the duration of the process.

Figure 9: Auto-tuning Completed



SAS Link Tuning for Blade 2, Link Configuration #35

Finished. **12:02:42**

Input #4: Auto-tuned, No Error

Save

All Groups

Name: MyLastTuning

In the **Save** area, the group and the name of the tuning is displayed. The group is provided by default and is not editable. The default name of the auto-tuning is *MyLastTuning*. You can change the name by typing another name into the text field. When you enter a new name, a **Save** button is displayed. Click the **Save** button to save the auto-tuning with the new name.

Click the **Back** button to close the dialog and returns to the **SAS Link Tuning** dialog.

Click the **Exit** button to close the dialog and return to the **Generator** tab on the **Xgig Maestro** window.

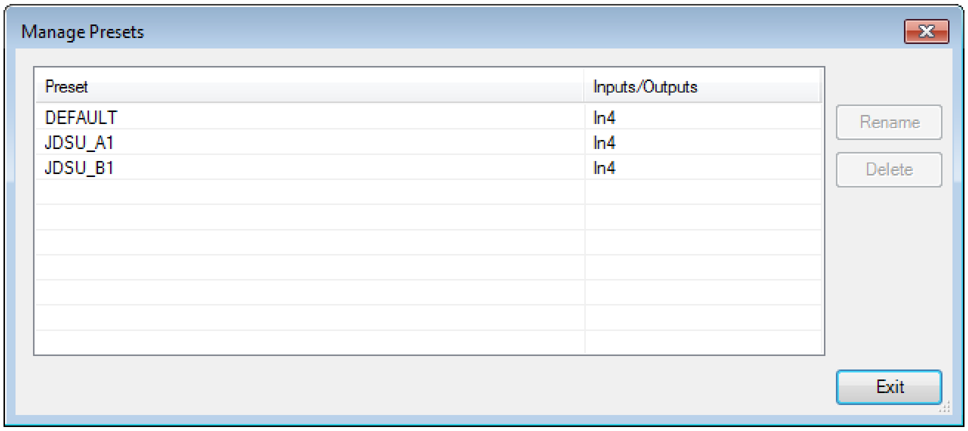
Change to [preset name]

To load a preset on the ports in a configuration, click **Change to [preset name]** from the Tunings drop-down list. Then, click the **Apply Changes** button. The software applies the tuned settings of the preset configuration hardware and returns to the **Generator** tab on the **Xgig Maestro** window.

In addition to the default settings, the tuned settings that you have saved after auto-tuning are also available.

The **Manage Presets...** button at the bottom of the **SAS Link Tuning** dialog opens a dialog allowing you to delete or rename presets.

Figure 10: Manage Presets Dialog



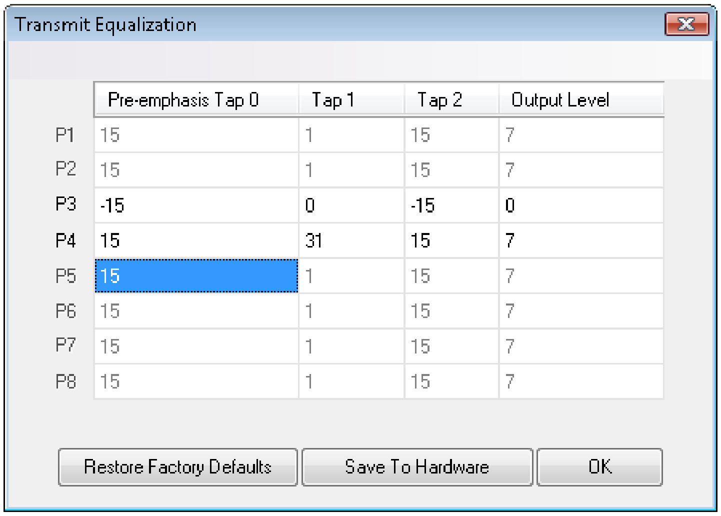
Preset	Inputs/Outputs
DEFAULT	In4
JDSU_A1	In4
JDSU_B1	In4

All the presets are listed in the right column. The left column shows the port groups affected by each preset. You cannot rename or delete the factory provided presets, nor any preset currently in use.

Tx Equalization

Tx Equalization allows you to configure transmitter settings for the Jammer and Generator. Click the **Tx Equalization** button to open the dialog box. For each port, you can specify pre-emphasis and output levels. Tx Equalization is an advanced feature. These settings should normally be left at their defaults, unless changes are recommended by the Technical Assistance team.

Figure 11: Transmit Equalization Dialog



	Pre-emphasis Tap 0	Tap 1	Tap 2	Output Level
P1	15	1	15	7
P2	15	1	15	7
P3	-15	0	-15	0
P4	15	31	15	7
P5	15	1	15	7
P6	15	1	15	7
P7	15	1	15	7
P8	15	1	15	7

The pre-emphasis module in each transmitter boosts high frequencies in the transmit data signal to compensate for attenuation in the transmission media. Three pre-emphasis taps are provided: Tap 0 is the pre-tap, Tap 1 is the first post tap, and Tap 2 is the second post tap.

Tap 0 sets the pre-emphasis on the data bit before each transition. Tap 1 and Tap 2 set the pre-emphasis on the transition bit and the following bit, respectively. Tap 0 and Tap 2 also provide inversion, which is set by using a negative value for the tap setting.

Allowable ranges are -15 to 15 for Tap 0 and Tap 2 and 0 to 31 for Tap 1.

The output level sets the transmitter amplitude. The allowable range is 0 to 7, which corresponds to an amplitude range of approximately 200 – 1200mV.

Note that these settings are applied on internal signal paths in the hardware, and the full effect will not normally be seen at the cable outputs.

Starting Macro

To start a macro file that was loaded, click **Start Macro**.

Editing Speed Settings

To set the speed for the port, click on **Edit speed settings...**

This option is for both SAS and SATA speed settings.

Setting the Debug Level

To set the debug level, click **Debug Level** then select the level from the pop-up menu.

Do not Show Generator

To hide the Parameters Status Table, click **Do not Show Generator**.

Text Size

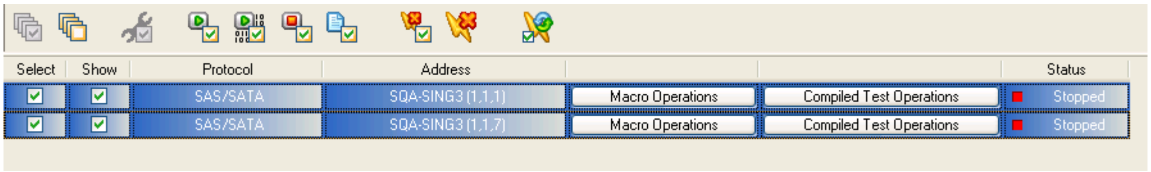
To change the font size of the labels and text used in the Parameters Status Table, click on **Text Size** and then select the font size from the pop-up menu. By default, the text size is set to the smallest.

Using the Ports Manager in Generator

The Ports manager (Figure 12) displays details about the ports you have locked. It shows the protocol, chassis name (with chassis number, slot number, and port numbers), and other port specific information. You can sort the rows from first to last or last to first by clicking the column heading.

Note: If you are disconnected due to a network problem, you can reconnect to the ports by using the Port Selection and Domain Setup window.

Figure 12: Ports Manager



Select	Show	Protocol	Address			Status
☑	☑	SAS/SATA	SQA-SING3 (1,1,1)	Macro Operations	Compiled Test Operations	🟥 Stopped
☑	☑	SAS/SATA	SQA-SING3 (1,1,7)	Macro Operations	Compiled Test Operations	🟥 Stopped

The following icons are displayed on the Ports Manager menu bar:

Select All Ports
Selects all ports in the Ports Manager.

Unselect All Ports
Unselects all ports in the Ports Manager.

Load Configuration to Selected Ports
Loads the selected functions, macro, or compiled test file to all selected ports in the Ports Manager.

Start Selected Ports
Starts loaded macro file on the selected ports in the Ports Manager.

Start Compiled Test on Selected Ports (Generator Only)
Executes the loaded compiled test on the selected ports in the Ports Manager.

Stop Selected Ports
Stops Generator operation on the selected ports in the Ports Manager.

Show Properties of Selected Ports
Displays the Port Properties dialog box, with information on the selected function, protocol, and clock rate (speed). It also shows the chassis name, IP address, slot and ports.

Disconnect Selected Ports
Disconnects the selected ports in the Ports Manager.

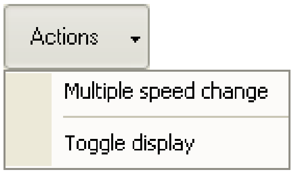
Disconnect All Ports
Disconnects all ports in the Ports Manager.

Hot-Reset Selected Ports
Resets Generator or Target Emulator ports running in chassis.

Actions Button

The **Actions** button contains menus for **Multiple speed change** and **Toggle display**. See Figure 13.

Figure 13: Actions Button Options

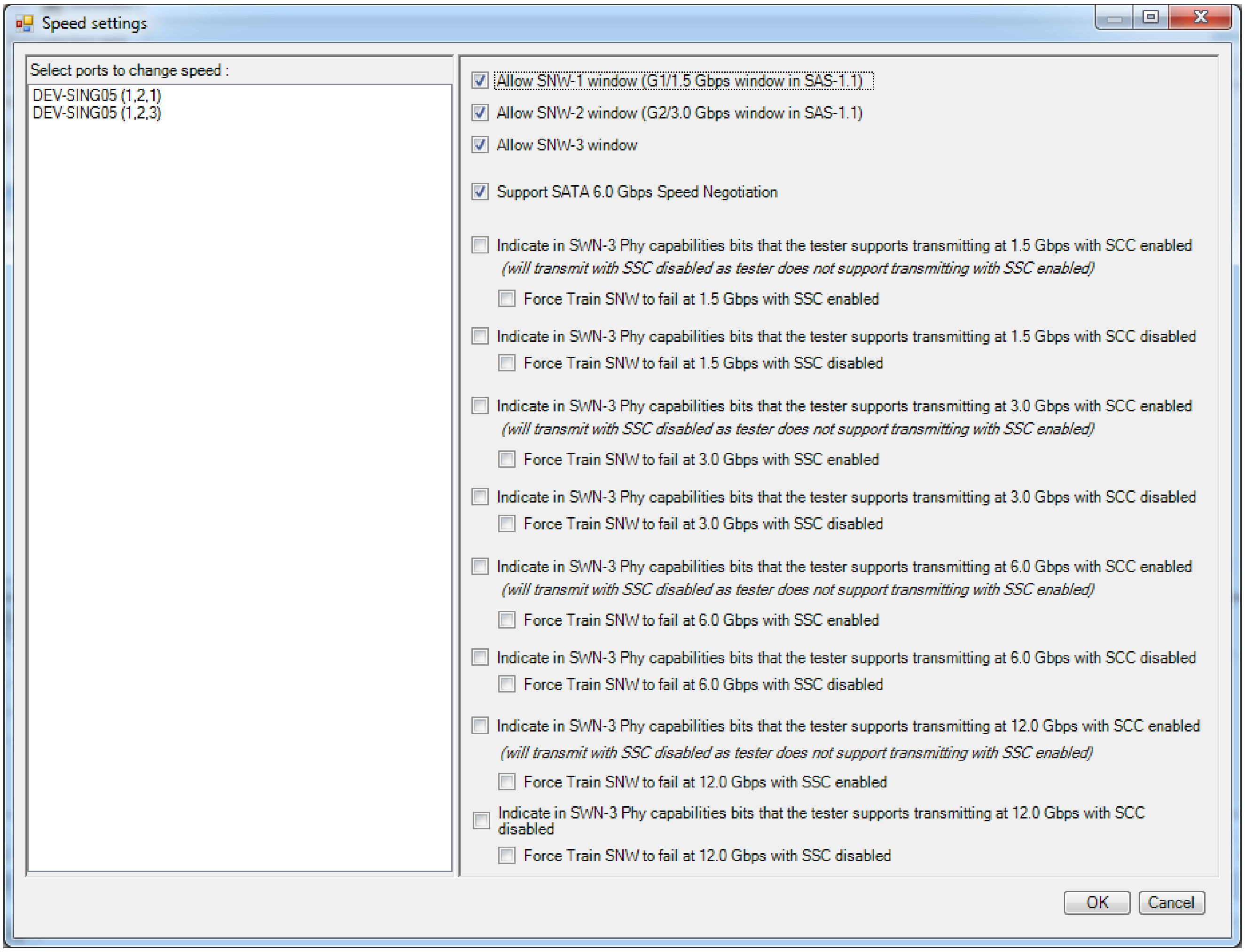


Multiple Speed Change for Generator

To change **Multiple speed change** for the **Generator** tab:

First, select the **Actions** button and then select **Multiple speed change** as shown in Figure 13. The **Speed Settings** window appears as shown in Figure 14. This menu differs from the main speed settings window in that it includes an additional pane that allows you to select the ports for which you want to change the speed.

Figure 14: Multiple Speed Settings Window for Generator



Speed settings

Select ports to change speed :

DEV-SING05 (1,2,1)
DEV-SING05 (1,2,3)

☑ Allow SNW-1 window (G1/1.5 Gbps window in SAS-1.1)

☑ Allow SNW-2 window (G2/3.0 Gbps window in SAS-1.1)

☑ Allow SNW-3 window

☑ Support SATA 6.0 Gbps Speed Negotiation

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 1.5 Gbps with SCC enabled *(will transmit with SSC disabled as tester does not support transmitting with SSC enabled)*

☐ Force Train SNW to fail at 1.5 Gbps with SSC enabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 1.5 Gbps with SCC disabled

☐ Force Train SNW to fail at 1.5 Gbps with SSC disabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 3.0 Gbps with SCC enabled *(will transmit with SSC disabled as tester does not support transmitting with SSC enabled)*

☐ Force Train SNW to fail at 3.0 Gbps with SSC enabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 3.0 Gbps with SCC disabled

☐ Force Train SNW to fail at 3.0 Gbps with SSC disabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 6.0 Gbps with SCC enabled *(will transmit with SSC disabled as tester does not support transmitting with SSC enabled)*

☐ Force Train SNW to fail at 6.0 Gbps with SSC enabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 6.0 Gbps with SCC disabled

☐ Force Train SNW to fail at 6.0 Gbps with SSC disabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 12.0 Gbps with SCC enabled *(will transmit with SSC disabled as tester does not support transmitting with SSC enabled)*

☐ Force Train SNW to fail at 12.0 Gbps with SSC enabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 12.0 Gbps with SCC disabled

☐ Force Train SNW to fail at 12.0 Gbps with SSC disabled

10 Select the ports that you want to change the speed for.

11 For instructions on filling out the main pane of the window, see “Speed” on page 18.

12 Click **OK** to apply the changes.

Toggle Display

This option toggles between a horizontal and vertical view of the ports.

Using Log Manager in Generator

To access the Log Manager, click the Log tab at the bottom of the Generator main window.

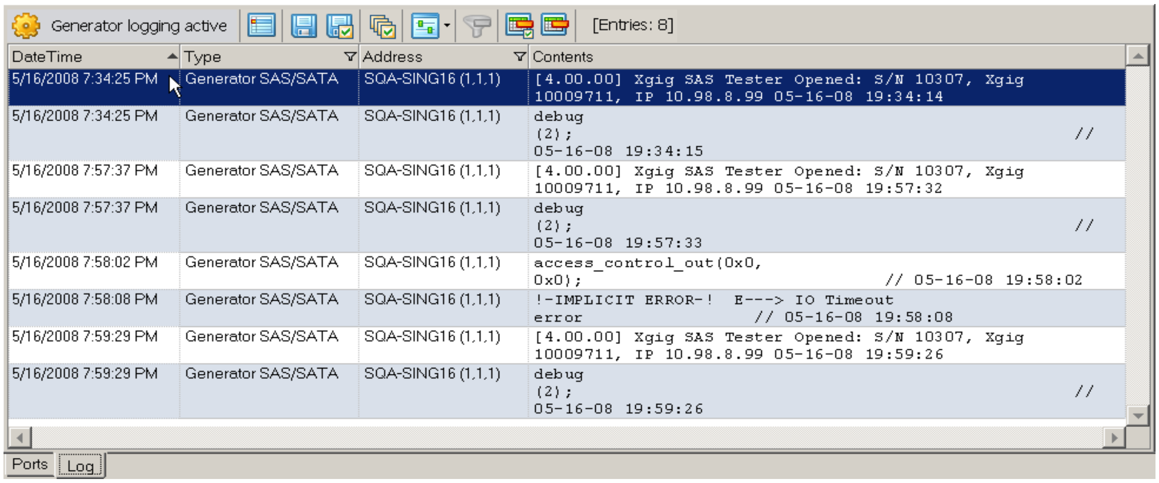
Generator creates logs that report which operations have occurred and which ones encountered errors during execution. These logs are contained within the Log Manager. To access the Log Manager, select the Log tab at the bottom of the Maestro window.

The Log Manager shows the status of each operation and test executed.

Note: The Ports Manager contains a small preview log pane at the bottom of the window for quick reference of log entries.

The Log Manager continues to list the logs and display them depending on the Log Manager options you have selected.

Figure 15: Log Manager



DateTime	Type	Address	Contents
5/16/2008 7:34:25 PM	Generator SAS/SATA	SQA-SING16 (1,1,1)	[4.00.00] Xgig SAS Tester Opened: S/N 10307, Xgig 10009711, IP 10.98.8.99 05-16-08 19:34:14
5/16/2008 7:34:25 PM	Generator SAS/SATA	SQA-SING16 (1,1,1)	debug (2); // 05-16-08 19:34:15
5/16/2008 7:57:37 PM	Generator SAS/SATA	SQA-SING16 (1,1,1)	[4.00.00] Xgig SAS Tester Opened: S/N 10307, Xgig 10009711, IP 10.98.8.99 05-16-08 19:57:32
5/16/2008 7:57:37 PM	Generator SAS/SATA	SQA-SING16 (1,1,1)	debug (2); // 05-16-08 19:57:33
5/16/2008 7:58:02 PM	Generator SAS/SATA	SQA-SING16 (1,1,1)	access_control_out(0x0, 0x0); // 05-16-08 19:58:02
5/16/2008 7:58:08 PM	Generator SAS/SATA	SQA-SING16 (1,1,1)	!-IMPLICIT ERROR-! E---> IO Timeout error // 05-16-08 19:58:08
5/16/2008 7:59:29 PM	Generator SAS/SATA	SQA-SING16 (1,1,1)	[4.00.00] Xgig SAS Tester Opened: S/N 10307, Xgig 10009711, IP 10.98.8.99 05-16-08 19:59:26
5/16/2008 7:59:29 PM	Generator SAS/SATA	SQA-SING16 (1,1,1)	debug (2); // 05-16-08 19:59:26

The Filter icons next to the **Type** and **Address** column labels allow you to choose how you want to display device types, by protocol or **All**, and device addresses, by single address or all devices.

The following icons are displayed on the Log Manager menu bar:

Display Entry Contents
This button is not currently being used and is always disabled.

Save All Entries As
Lets you save all the logs in the Log Manager to a file.

Save selected entries as
Lets you save the selected logs to a file.

Select All
Selects all the logs in the Log Manager.

Options
Opens the Log Manager Options dialog where you can enter your preferences for the information you want displayed in the Log Manager (Figure 16).

Clear Filtering
Removes filtering by Type and Address and displays all the devices you have locked.

Clear Selected Entries
Deletes the log entry you have highlighted.

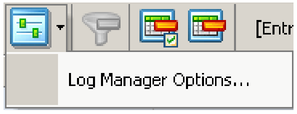
Clear All Entries
Deletes all the log entries in the Log Manager.

Setting Log Options

To set up default naming and locations of Generator log files:

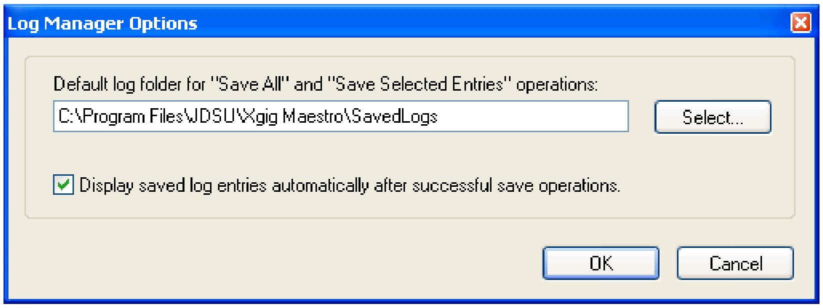
1 Click the Log Manager Options icon to open the drop-down menu, and select **Log Manager Options**.

Figure 16: Log Manager Options Menu



The Generator Log Options dialog is displayed. This dialog allows you to set up how you want the log files to be automatically named and the default location you want them saved to. You can also change the filename and folder when you save the log.

Figure 17: Log Options Dialog



Log Manager Options

Default log folder for "Save All" and "Save Selected Entries" operations:

C:\Program Files\JDSU\Xgig Maestro\SavedLogs

☑ Display saved log entries automatically after successful save operations.

2 Enter the path to the folder where you want to save log files.

You can browse to where you want to save the log file by clicking Select... .

3 Click **OK**.

4 To display save log entries automatically after successful save operations, select **Display saved log entries automatically after successful save operations**.

The right-click menu in the Log Manager contains the same options as those shown in the Log Manager tool bar and described in this section.

Saving a Log

You can save a log from the Log Manager.

To save a log from the Log Manager as an HTML file:

1 Highlight the log(s) you want to save.

2 Click the **Save Selected Entries As** button.

The **Save Log Manager Contents As** window is displayed. This window lets you name and save the log as an HTML file to the Saved Logs folder or to a location you prefer.

Customizing the Appearance of the Maestro/Generator Main Window

You can move and rearrange the individual device tabs on your monitor screen as you prefer. You can also select the information you want to display in the Parameters Status table. In addition, you can select the size of displayed text.

Using the Window Menu

The **Window** menu allows you to arrange the window display:

Layout

Layout offers choices for rearranging the Maestro device tab windows. The Ports manager and Log Manager windows are not affected by this menu selection.

Internal Tabs

Allows you to restore the Maestro main window to its default format.

Internal MDI

Internal multiple document interface (MDI) lets you isolate each device tab as a separate window that you can activate by clicking anywhere on the window.

Show Hidden Windows

Restores any windows you have closed while using the Internal Tabs or Internal MDI display arrangement.

Arrange Icons

Arranges the icons for minimized windows at the bottom of the screen. If an open document window is at the bottom of the screen, some or all of the icons will be underneath this document window and will not be visible.

Cascade

Arranges all open windows in a cascade style from the top, left corner of the window.

BERT - Bit error rate testing

Brings the BERT device tab to the front as the active window

BERT - Latency measurement

Brings the BERT Latency measurement device tab to the front as the active window

Jammer

Brings the Jammer device tab to the front as the active window

Generator

Brings the Generator device tab to the front as the active window

Target Emulator

Brings the Target Emulator device tab to the front as the active window

Delay Emulator

Brings the Delay Emulator device tab to the front as the active window

Load Tester

Brings the Load Tester device tab to the front as the active window

Tile Horizontally

After selecting **Layout > Internal MDI,** this command opens and aligns the BERT, BERT Latency measurement, Jammer, and Generator device windows horizontally one over the other.

Tile Vertically

After selecting **Layout > Internal MDI**, this command opens and aligns the BERT, BERT Latency measurement, Jammer, and Generator device windows vertically one next to the other. You can view the individual device windows by dragging the sides out to resize them.

Minimize All Windows

After selecting **Layout > Internal MDI**, this command minimizes all the open windows. You can restore each window individually, by using the standard window controls on the header or select **Internal Tabs** to restore the Maestro main window to its default format.

Customizing the Parameters Status table in Generator

You can change the appearance of the Parameters Status table by:

- Removing columns
- Resizing columns
- Changing the text size
- Displaying and hiding status table parameters

Removing columns

If you have a number of devices locked, you can remove a column that you do not need to view without unlocking the port or affecting the operation.

To remove a column:

1 Right-click on the specific column you want and open the Parameters context menu.

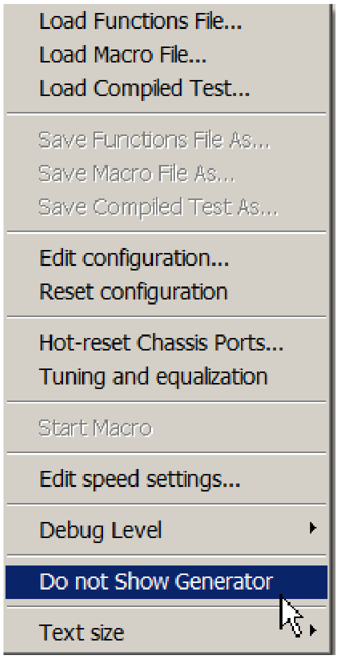
2 Select **Do not show Generator** (Figure 18).

Or:

In the Ports Manager, clear the **Show** check box next to the device you want to remove.

The column is removed, but the device is still locked (“in use”) and displayed in the Ports Manager.

Figure 18: Removing a Device Column



To replace the device column on the Parameters Status table:

>> Click the **Show** check box for the device in the Ports Manager.

Resizing columns

You can resize any of the columns by placing the mouse at the right edge at the top of a column, next to the chassis name. A resizing cursor appears. Hold down the mouse button, and drag the mouse to resize the column.

Changing Text Size

You can choose the text size you want the Parameters Status table or the Ports Manager to display.

Note: You cannot change the text size in the Log Manager.

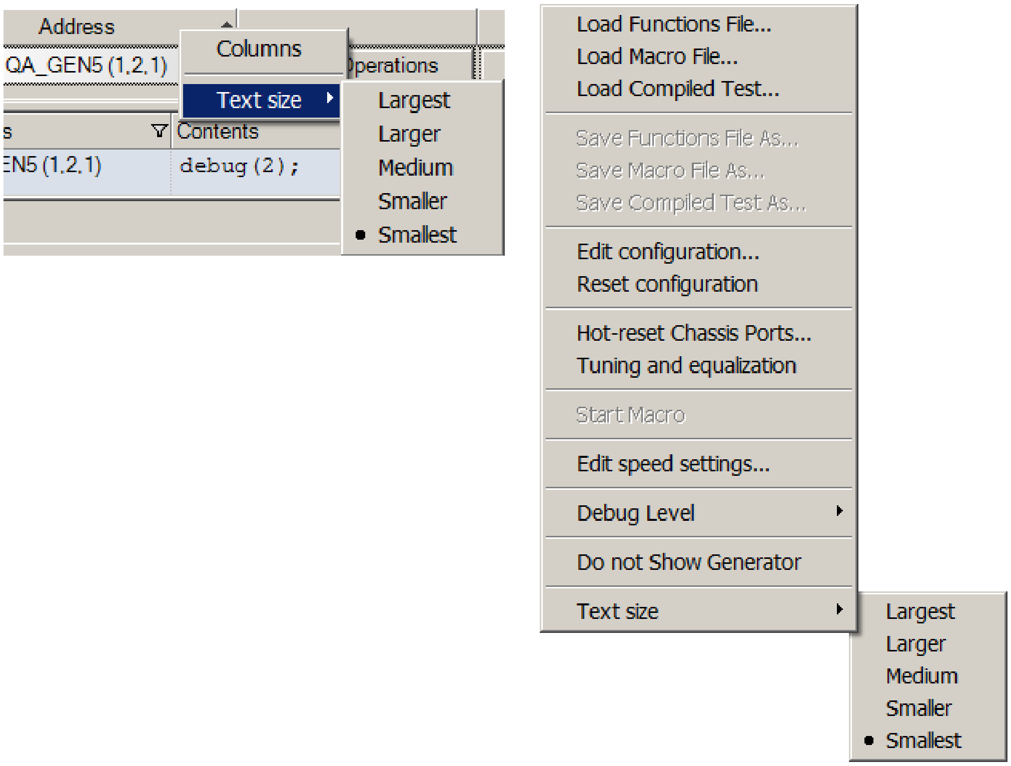
To change text size:

1. Open the context menu by clicking the right mouse button while the cursor is on the Parameters Status table or the Ports Manager.
2. Select **Text size** (Figure 19).

   You have five choices (Largest, Larger, Medium, Smaller, Smallest) from which to select. Smallest is the default size.
3. Select the text size you want.

   A bullet is displayed next to the current selection.

Figure 19: Text Size Menu



Context menu over Ports manager

Context menu over device parameters status table

Displaying and Hiding Parameters in Generator

The **Parameters** tab next to the functions tabs allows you to select the parameters you want to display in the Parameters Status table. You can hide specific parameters to simplify the status tables and show only the information in which you are interested.

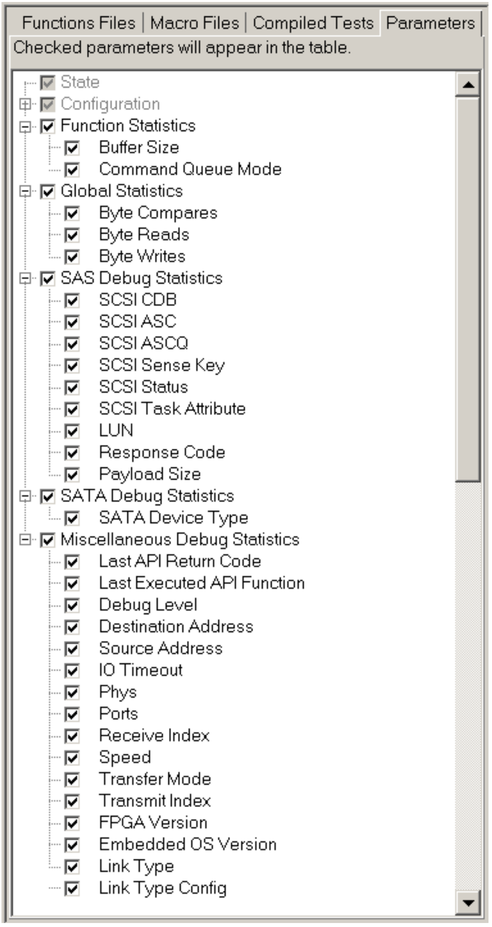
To display or hide parameters on the Parameters Status table:

1 Click the **Parameters** tab.

You can expand the categories by clicking the plus signs on the left.

The default displays all of the parameters.

Figure 20: Parameters Tab



Checked parameters will appear in the table.

- ☑ State
- ⊞ ☑ Configuration
- ⊟ ☑ Function Statistics
  - ☑ Buffer Size
  - ☑ Command Queue Mode
- ⊟ ☑ Global Statistics
  - ☑ Byte Compares
  - ☑ Byte Reads
  - ☑ Byte Writes
- ⊟ ☑ SAS Debug Statistics
  - ☑ SCSI CDB
  - ☑ SCSI ASC
  - ☑ SCSI ASCQ
  - ☑ SCSI Sense Key
  - ☑ SCSI Status
  - ☑ SCSI Task Attribute
  - ☑ LUN
  - ☑ Response Code
  - ☑ Payload Size
- ⊟ ☑ SATA Debug Statistics
  - ☑ SATA Device Type
- ⊟ ☑ Miscellaneous Debug Statistics
  - ☑ Last API Return Code
  - ☑ Last Executed API Function
  - ☑ Debug Level
  - ☑ Destination Address
  - ☑ Source Address
  - ☑ IO Timeout
  - ☑ Phys
  - ☑ Ports
  - ☑ Receive Index
  - ☑ Speed
  - ☑ Transfer Mode
  - ☑ Transmit Index
  - ☑ FPGA Version
  - ☑ Embedded OS Version
  - ☑ Link Type
  - ☑ Link Type Config

2 Click the check box next to a parameter to set or clear the check mark for that parameter.

Checked parameters are displayed in the Parameters Status table.

For details about each of the parameters in the Parameters Status table, see Using the Parameters Status Table in Generator”.

Chapter 3

Using Generator with Ports

In this chapter:

After you have discovered and locked the Generator ports you want to use and also set up your capturing and monitoring applications, such as Xgig Analyzer or Bus Doctor, you are ready to run the Xgig Generator application. This chapter explains the process of managing and executing all of the configuration files for your device.

Loading a Configuration File

The Generator Device window represents the device selected in the ports manager on the Generator tab. This window contains tabs for functions, macros and compiles tests. This chapter discusses the functionality of each tab.

To access the Generator Device window you need to load a functions file, macro file, or compiled test file to a device.

To load a functions, macro, or compiled test file to a port, locate the file you want to load in the Configuration window, then drag the file to the port, and drop it.

Launching the Generator Device Window

You can open the Generator Device window from either the Parameters Status table or the Ports Manager.

To open the Generator Device window for a device from the **Parameters Status table**:

1 Click the **Start** ▷ button on the Parameters Status table for the functions file, macro file, or compiled test file for your device.

The respective Operation window is displayed.

2 Click the **Edit** Edit button to open the Generator Device window.

To open the Generator Device window for a device from the **Ports Manager**:

1 Check the Select check box next to your device in the **Ports Manager**.

2 Click either the Macro Operations Macro Operations button or the Compiled Test Operations button for the device.

The respective Operations window is displayed.

3 Click the **Edit** Edit button to open the Generator Device window.

Once the window is open, select the tab you want to use.

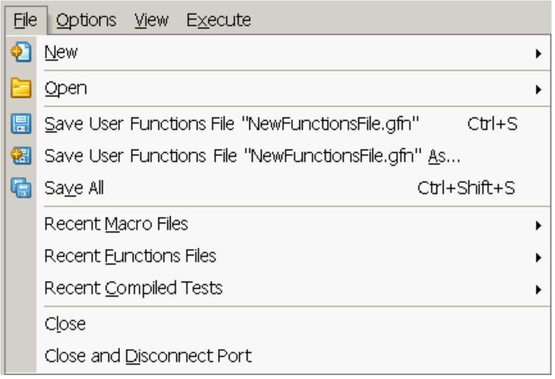
Generator Device Menu Bar

The menu bar at the top of the Generator Device window contains the following menus.

File

The File menu (Figure 21) allows you to either create a new configuration or open an existing/recently-used configuration file for editing. You can also close the application from this menu.

Figure 21: File Menu


File
Options
View
Execute
New
Open
Save User Functions File "NewFunctionsFile.gfn"
Ctrl+S
Save User Functions File "NewFunctionsFile.gfn" As...
Save All
Ctrl+Shift+S
Recent Macro Files
Recent Functions Files
Recent Compiled Tests
Close
Close and Disconnect Port

To create a new macro or functions file:

>> Select **New**, then select:
- New Tcl File
- New Functions File
- New Macro File

A blank file is opened on the tab for the file type you selected.

To open an existing functions, macro, or compiled test file:

>> Select **Open**, then select:
- Open Functions File...
- Open Macro File...
- Open Compiled Test...
- Open Tcl File...

The file is opened on the tab for the file type you selected.

To Save the currently displayed file:

>> Select Save *Macro* file "..."

To Save the currently displayed file as another name or to another location:

>> Select Save *Macro* file "..." As

To Save all open files:

>> Select Save All.

To open a recently-used functions, macro, or compiled test file:

1 Select one of the following:
- Recent Macro File
- Recent Functions File
- Recent Compiled Test File

2 Click on the file you want to open.

The file is opened on the tab for the file type you selected.

To close the Generator Device window, click **Close**.

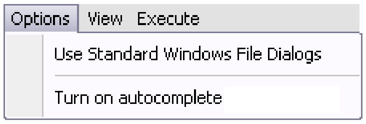
To close the Generator Device window and disconnect the port for the device, click **Close and Disconnect Port**.

Options

The Options menu (Figure 22) allows you to use standard Windows file dialog boxes for **Open** and **Save As...** commands instead of the dialog boxes customized for Generator. For example, if the Macro tab is active and the **Use Standard Windows File Dialogs** is selected, then when you select **Save As...** from the **File** menu, the title of the resulting dialog box will read, "*Save Macro File As...*" However, if the **Use Standard Windows File Dialogs** is not selected, the same dialog box will read, "*Save Generator Macro File for Protocol SAS/SATA as*".

The **Turn on autocomplete** selection enables the Macro and Tcl editors to display a popup box with a list of the available choices that match the first letter of the command that you entered. You can make your selection from the list. However, if you continue to enter additional letters of the command, the first choice of the commands that match your entry is highlighted.

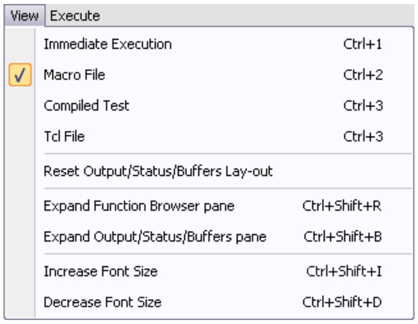
Figure 22: Options Menu



View

The View menu allows you to switch between tabs in the Generator Device window by clicking the entry for the desired tab. This menu also allows you to collapse/expand the Function Browser pane, which is the left pane containing the function library on the left, or to collapse/expand the Output/Status/Buffers pane which is the bottom pane. There is also a selection that allows you to reset the Output/Status/Buffers pane, These two panes are not shown in the Compiled Test window.

Figure 23: View Menu

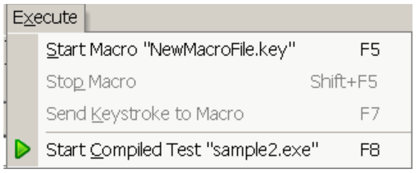

View
Execute
Immediate Execution Ctrl+1
Macro File Ctrl+2
Compiled Test Ctrl+3
Tcl File Ctrl+3
Reset Output/Status/Buffers Lay-out
Expand Function Browser pane Ctrl+Shift+R
Expand Output/Status/Buffers pane Ctrl+Shift+B
Increase Font Size Ctrl+Shift+I
Decrease Font Size Ctrl+Shift+D

Execute

The Execute menu provides the following choices for macros:

- **Start Macro** executes the currently loaded macro file.
- **Stop Macro** stops the execution of a macro.
- **Send Keystroke to Macro** simulates a keystroke to stop looping during macro execution. This selection is active only if the `loopk` function is invoked in your macro. Refer to the description for `loopk` in the function library.
- **Start Compiled Test** executes the currently loaded compiled test file.

Figure 24: Execute Menu


Execute
Start Macro "NewMacroFile.key" F5
Stop Macro Shift+F5
Send Keystroke to Macro F7
Start Compiled Test "sample2.exe" F8

Output/Status/Buffers Pane

This pane is located at the bottom of the **Immediate Execution** and **Macro File** tabs. It allows you to set the debug level, view the output and status of executed functions and macros, edit buffers, and set status viewing options.

Setting the Debug Level

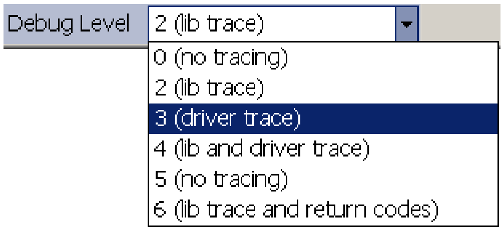
To view the debug level control, make sure the bottom pane of the window is expanded. If it is not, select **Expand Output/Status/Buffers pane** from the **View** menu. The debug control is located on the Output/Status tab.

The following debug levels are available:

- 0 (no tracing) does not display any traces
- 2 (lib trace) displays library traces only (default debug level)
- 3 (driver trace) displays driver traces only
- 4 (lib and driver trace) displays both library and driver traces
- 6 (lib trace and return codes) displays library traces and return codes

To set the debug level, click the Debug Level drop-down menu and select the appropriate debug level.

Figure 25: Setting the Debug Level



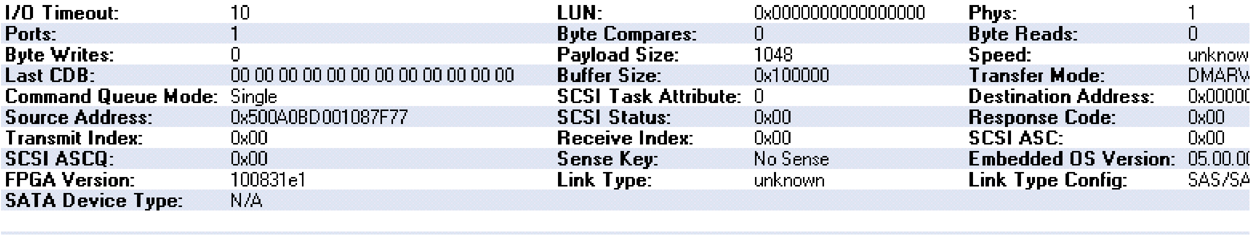
Debug Level: 2 (lib trace)

Setting the Status Viewing Information

The status of the device selected in the Parameters Status table is displayed in the Status pane of the Generator Device window.

To view the status of the device, make sure the bottom pane of the window is expanded. If it is not, select **Expand Output/Status/Buffers** pane from the **View** menu. The status information is located on the **Output/Status** tab.

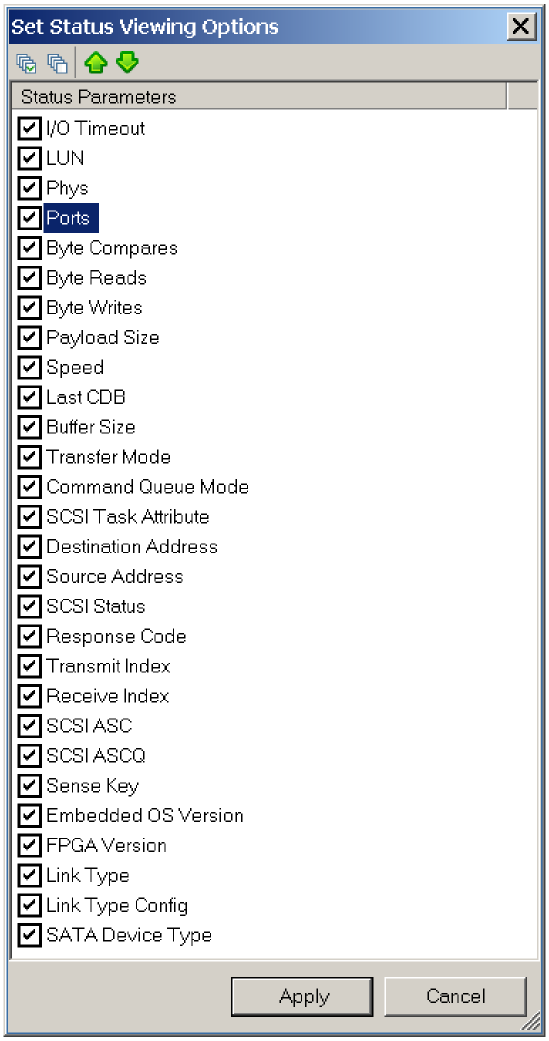
Figure 26: Status Information for Selected Device



I/O Timeout:	10	LUN:	0x0000000000000000	Phys:	1
Ports:	1	Byte Compares:	0	Byte Reads:	0
Byte Writes:	0	Payload Size:	1048	Speed:	unknow
Last CDB:	00 00 00 00 00 00 00 00 00 00 00 00	Buffer Size:	0x100000	Transfer Mode:	DMARV
Command Queue Mode:	Single	SCSI Task Attribute:	0	Destination Address:	0x0000(
Source Address:	0x500A0BD001087F77	SCSI Status:	0x00	Response Code:	0x00
Transmit Index:	0x00	Receive Index:	0x00	SCSI ASC:	0x00
SCSI ASCQ:	0x00	Sense Key:	No Sense	Embedded OS Version:	05.00.0(
FPGA Version:	100831e1	Link Type:	unknown	Link Type Config:	SAS/SA
SATA Device Type:	N/A				

The parameters listed in the Status pane correspond to those shown in the Parameters Status table for the device. For a definition of these categories, refer to “Using the Parameters Status Table in Generator” on page 13”.

Figure 27: Set Status Viewing Options Window



Set Status Viewing Options

Status Parameters

☑ I/O Timeout
☑ LUN
☑ Phys
☑ Ports
☑ Byte Compares
☑ Byte Reads
☑ Byte Writes
☑ Payload Size
☑ Speed
☑ Last CDB
☑ Buffer Size
☑ Transfer Mode
☑ Command Queue Mode
☑ SCSI Task Attribute
☑ Destination Address
☑ Source Address
☑ SCSI Status
☑ Response Code
☑ Transmit Index
☑ Receive Index
☑ SCSI ASC
☑ SCSI ASCQ
☑ Sense Key
☑ Embedded OS Version
☑ FPGA Version
☑ Link Type
☑ Link Type Config
☑ SATA Device Type

To select which parameters you want to display in the Status pane:

1 Click the Set Status Viewing Options button.

2 The Set Status Viewing Options window appears.

3 Select which parameters you want to view in the Status pane.

4 Use the **Move Up**/**Move Down** buttons to arrange the order of the parameters.

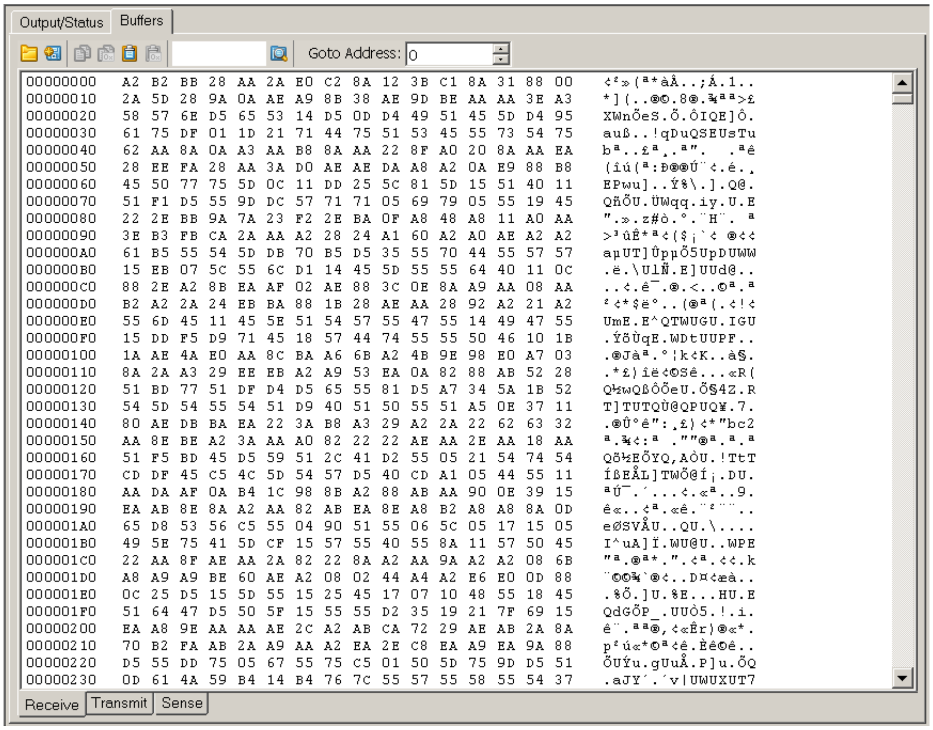
5 Click **Apply**.

Accessing the Buffer Editor

The Buffer Editor enables you to view and manipulate the transmit, receive, and sense buffers. A buffer is a `.bin` file that displays data used in SCSI data frames. SCSI data is sent from the transmit buffer and to the receive buffer during SCSI I/O operations. The sense buffer displays the sense data of the most recent commands containing that data. Refer to the `dmaset()`, `dmarst()`, `get_g_info("bufsz")`, `get_g_info("tx_ptr")`, and `get_g_info("rx_ptr")` functions for more information.

To access the buffer editor, open the main configuration window for a port. Select the **Macro File** tab, then select the **Buffers** tab.

Figure 28: Buffer Editor



```
00000000   A2 B2 BB 28 AA 2A E0 C2 8A 12 3B C1 8A 31 88 00   ¢²»(ª*àÂ..;Á.1..
00000010   2A 5D 28 9A 0A AE A9 8B 38 AE 9D BE AA AA 3E A3   *](..®©.8®.¾ªª>£
00000020   58 57 6E D5 65 53 14 D5 0D D4 49 51 45 5D D4 95   XWnÕeS.Õ.ÔIQE]Ô.
00000030   61 75 DF 01 1D 21 71 44 75 51 53 45 55 73 54 75   auß..!qDuQSEUsTu
00000040   62 AA 8A 0A A3 AA B8 8A AA 22 8F A0 20 8A AA EA   bª..£ª¸.ª".  .ªê
00000050   28 EE FA 28 AA 3A D0 AE AE DA A8 A2 0A E9 88 B8   (îú(ª:Ð®®Ú¨¢.é.¸
00000060   45 50 77 75 5D 0C 11 DD 25 5C 81 5D 15 51 40 11   EPwu]..Ý%\.].Q@.
00000070   51 F1 D5 55 9D DC 57 71 71 05 69 79 05 55 19 45   QñÕU.ÜWqq.iy.U.E
00000080   22 2E BB 9A 7A 23 F2 2E BA 0F A8 48 A8 11 A0 AA   ".».z#ò.º.¨H¨. ª
00000090   3E B3 FB CA 2A AA A2 28 24 A1 60 A2 A0 AE A2 A2   >³ûÊ*ª¢($¡`¢ ®¢¢
000000A0   61 B5 55 54 5D DB 70 B5 D5 35 55 70 44 55 57 57   aµUT]ÛpµÕ5UpDUWW
000000B0   15 EB 07 5C 55 6C D1 14 45 5D 55 55 64 40 11 0C   .ë.\UlÑ.E]UUd@..
000000C0   88 2E A2 8B EA AF 02 AE 88 3C 0E 8A A9 AA 08 AA   ..¢.ê¯.®.<..©ª.ª
000000D0   B2 A2 2A 24 EB BA 88 1B 28 AE AA 28 92 A2 21 A2   ²¢*$ëº..(®ª(.¢!¢
000000E0   55 6D 45 11 45 5E 51 54 57 55 47 55 14 49 47 55   UmE.E^QTWUGU.IGU
000000F0   15 DD F5 D9 71 45 18 57 44 74 55 55 50 46 10 1B   .ÝõÙqE.WDtUUPF..
00000100   1A AE 4A E0 AA 8C BA A6 6B A2 4B 9E 98 E0 A7 03   .®Jàª.º¦k¢K..à§.
00000110   8A 2A A3 29 EE EB A2 A9 53 EA 0A 82 88 AB 52 28   .*£)îë¢©Sê...«R(
00000120   51 BD 77 51 DF D4 D5 65 55 81 D5 A7 34 5A 1B 52   Q½wQßÔÕeU.Õ§4Z.R
00000130   54 5D 54 55 54 51 D9 40 51 50 55 51 A5 0E 37 11   T]TUTQÙ@QPUQ¥.7.
00000140   80 AE DB BA EA 22 3A B8 A3 29 A2 2A 22 62 63 32   .®Ûºê":¸£)¢*"bc2
00000150   AA 8E BE A2 3A AA A0 82 22 22 AE AA 2E AA 18 AA   ª.¾¢:ª .""®ª.ª.ª
00000160   51 F5 BD 45 D5 59 51 2C 41 D2 55 05 21 54 74 54   Qõ½EÕYQ,AÒU.!TtT
00000170   CD DF 45 C5 4C 5D 54 57 D5 40 CD A1 05 44 55 11   ÍßEÅL]TWÕ@Í¡.DU.
00000180   AA DA AF 0A B4 1C 98 8B A2 88 AB AA 90 0E 39 15   ªÚ¯.´...¢.«ª..9.
00000190   EA AB 8E 8A A2 AA 82 AB EA 8E A8 B2 A8 A8 8A 0D   ê«..¢ª.«ê.¨²¨¨..
000001A0   65 D8 53 56 C5 55 04 90 51 55 06 5C 05 17 15 05   eØSVÅU..QU.\....
000001B0   49 5E 75 41 5D CF 15 57 55 40 55 8A 11 57 50 45   I^uA]Ï.WU@U..WPE
000001C0   22 AA 8F AE AA 2A 82 22 8A A2 AA 9A A2 A2 08 6B   "ª.®ª*.".¢ª.¢¢.k
000001D0   A8 A9 A9 BE 60 AE A2 08 02 44 A4 A2 E6 E0 0D 88   ¨©©¾`®¢..D¤¢æà..
000001E0   0C 25 D5 15 5D 55 15 25 45 17 07 10 48 55 18 45   .%Õ.]U.%E...HU.E
000001F0   51 64 47 D5 50 5F 15 55 55 D2 35 19 21 7F 69 15   QdGÕP_.UUÒ5.!.i.
00000200   EA A8 9E AA AA AE 2C A2 AB CA 72 29 AE AB 2A 8A   ê¨.ªª®,¢«Êr)®«*.
00000210   70 B2 FA AB 2A A9 AA A2 EA 2E C8 EA A9 EA 9A 88   p²ú«*©ª¢ê.Èê©ê..
00000220   D5 55 DD 75 05 67 55 75 C5 01 50 5D 75 9D D5 51   ÕUÝu.gUuÅ.P]u.ÕQ
00000230   0D 61 4A 59 B4 14 B4 76 7C 55 57 55 58 55 54 37   .aJY´.´v|UWUXUT7
```

Data is shown in both hexadecimal and ASCII formats.

Opening a Buffer File

To open a buffer file:

1 Select the tab for the type of buffer you want to open, i.e. **Receive**, **Transmit**, or **Sense**.

2 Click the **Folder** 📁 icon in the top left corner of the pane.

3 Browse to the buffer file you want to open.

4 Click OK.

The buffer file is displayed in the buffer pane.

Editing a Buffer

Both the hexadecimal and ASCII fields in the read and write buffers can be edited. The Address field indicates the selected position.

To edit a buffer file:

1. Move to the desired cell using the **GotoAddress** field or by typing text into the **Search** field.
2. Enter the appropriate value. If you are editing a hexadecimal value, the ASCII value changes accordingly, and if you are editing an ASCII value, the hexadecimal value changes accordingly.
3. Use the **Copy** icon to copy an selection of the buffer, and then use the **Paste** icon to paste the selected contents into another buffer, into the current buffer, or into a text editor as ASCII.
4. Alternatively, you can use the **Copy as a hex string** icon to copy a selection of the buffer as a hex string and use the **Paste as a hex string** icon to paste the selected contents into another buffer, into the current buffer, or into a text editor as a hex sting.
5. You can also do the following:
    - Copy a section from another buffer file and paste it into the current file or visa versa using the right-click menu. This menu also contains the option to copy/paste as ASCII or as a hex string.
    - Paste another buffer file into the current file at an insertion point by using the right-click menu.
6. Save the buffer file by selecting the **Save** icon in the top, left corner of the pane. Saving a buffer, saves the data on all three tabs into a single file.

Immediate Execution Tab

After you have configured a Generator device, you can use the Immediate Execute tab to execute individual functions, or to create, edit, and execute functions files.

The **Immediate Execution** window is comprised of two panels. The left panel is the contents pane. It lists the functions present in the functions file. See “Creating a New Functions File” on page 53.

The right pane is the function library. Each of the functions in the library are represented in the API Help System on the Maestro USB drive.The functions in the library are listed in the contents pane by name; current parameters in hexadecimal; and the parameter types: b (byte), d (dword), q (quad), s (string parameter), w (word), and o (optional parameter), which currently cannot be edited or added; and the function category.

You can sort the functions in either pane in the following ways:

- Alphabetically in ascending or descending order by toggling the Function Name heading.
- By category with the categories alphabetically in ascending or descending order by toggling the Function Type heading.
- By parameter types with the parameter types listed by number of parameters from most to least or from least to most by toggling the Parameter heading.

Click the **Clear Filter** icon to remove the filters you have applied by typing text into the search field and revert to viewing all functions in the functions library.

To view the details of the functions in the library or the contents pane, click the **Show/Hide Function Info** button at the top of the pane. The details for each function correspond to an entry in the *Generator API Help* available from the Help menu in the Maestro main window. To display the contents of the information pane for the selected function in a popup window, click the **Display Help Contents in Popup Window** button.

As you begin to click various functions in the function library or the contents pane, while you have the Show Function Info pane open, you create a history of the functions you’ve selected. You can navigate back and forth through the functions you’ve selected by using the **Go Back** and **Go Forward** buttons at the top of the Function Information pane.You can execute a function from the Immediate Execution tab of the Generator Device window or from the Parameters Status table in the Maestro window.

Editing Function Parameters

To edit a function’s parameters inline:

1. Right-click the function, and select Edit Parameters Inline.
2. Type directly in the parameters field to edit inline, or click the arrow at the end of the field to open a drop-down window.

   This window is the same as the Popup Parameters Editor.

To edit a function’s parameters using the Popup Parameters Editor:

1. Right-click the function, and select Popup Parameters Editor.
2. The Popup Parameters Editor appears with the following options in the menu:
   - Apply and execute
   - Apply and close
   - Discard and close
   - Reset parameter values
   - Show parameters info

Depending on the function, the following parameters may be available for editing:

Display - Displays the values in hexadecimal or decimal, depending on the field selected. Use the drop-down menu to change the display.

Byte - Byte- or bit-specific data. A byte is equal to 8 bits.

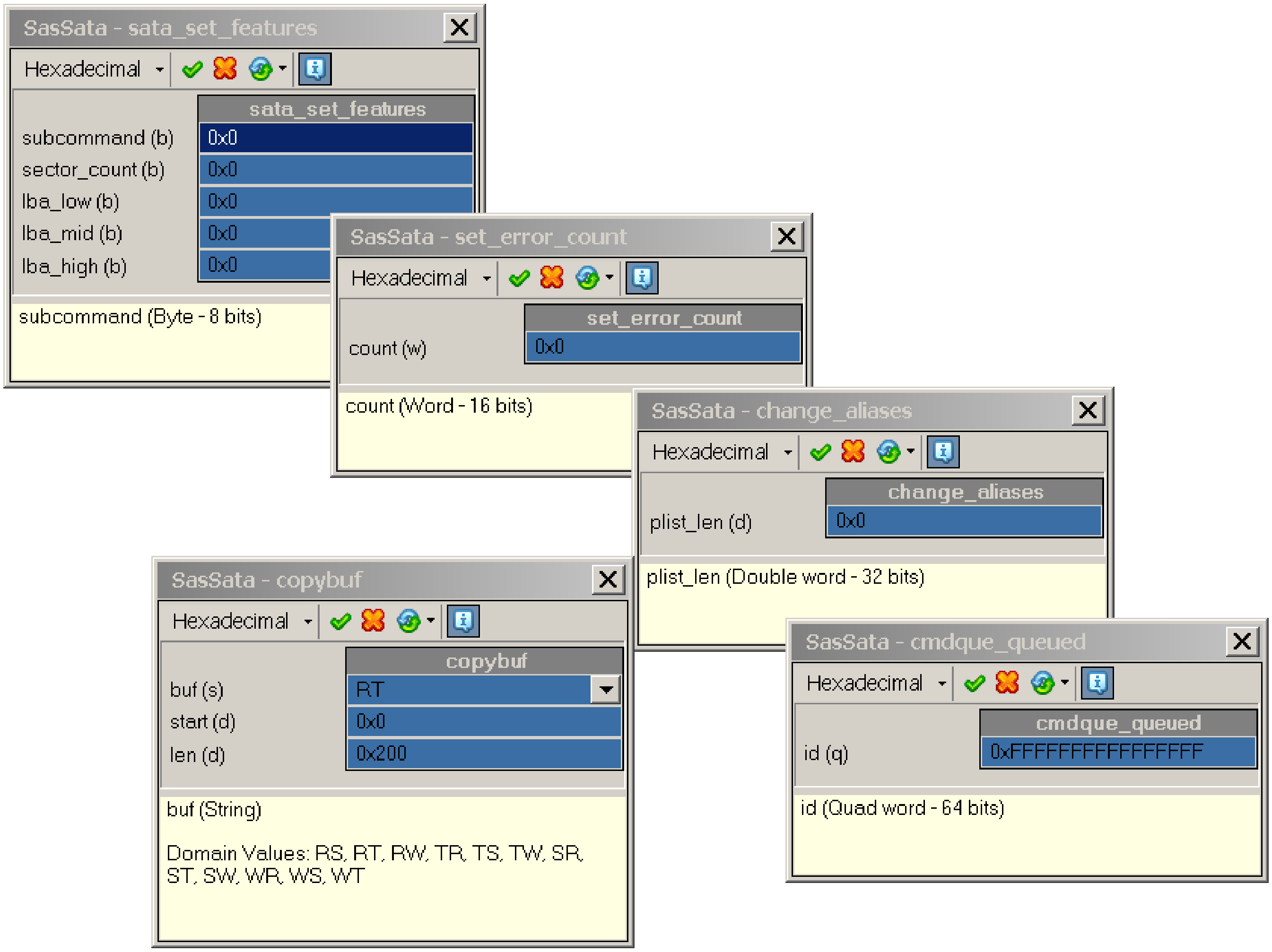
Word - Word or short integer data. A word is equal to 16 bits.

Double - Double word or long integer data. A double word is equal to 32 bits.

Quad - A quad is equal to 64 bits or 4 words.

String - Constant text or string data. Enter data without quotation marks.

Figure 29: Popup Parameters Editor Window



SasSata - sata_set_features

Hexadecimal

sata_set_features

subcommand (b) 0x0
sector_count (b) 0x0
lba_low (b) 0x0
lba_mid (b) 0x0
lba_high (b) 0x0

subcommand (Byte - 8 bits)

SasSata - set_error_count

Hexadecimal

set_error_count

count (w) 0x0

count (Word - 16 bits)

SasSata - change_aliases

Hexadecimal

change_aliases

plist_len (d) 0x0

plist_len (Double word - 32 bits)

SasSata - copybuf

Hexadecimal

copybuf

buf (s) RT
start (d) 0x0
len (d) 0x200

buf (String)

Domain Values: RS, RT, RW, TR, TS, TW, SR, ST, SW, WR, WS, WT

SasSata - cmdque_queued

Hexadecimal

cmdque_queued

id (q) 0xFFFFFFFFFFFFFFFF

id (Quad word - 64 bits)

Executing a Function on a Generator Device

You can execute an individual function from the Contents pane or from the Function Library.

To execute a function from the Generator Device window:

1 Locate the function you want to execute in the function library or contents pane of the window by using the scroll bar or by typing the name of the function into the **Search Function** field.

2 Set the parameters as necessary for the selected function. The right-click menu provides editing options. These options are described in "Editing Function Parameters".

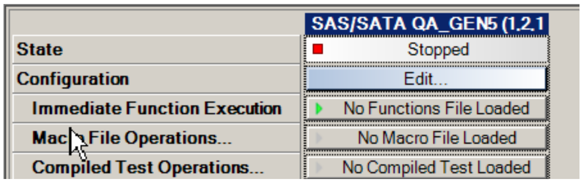
3 Click the **Start** ▷ button to execute the function.

You can also perform immediate function execution from the Parameter status table in the Maestro window.

To execute a function from the Parameters Status table:

1 Click the **Start** ▷ button on the Parameters Status table next to the Immediate Function Execution field.

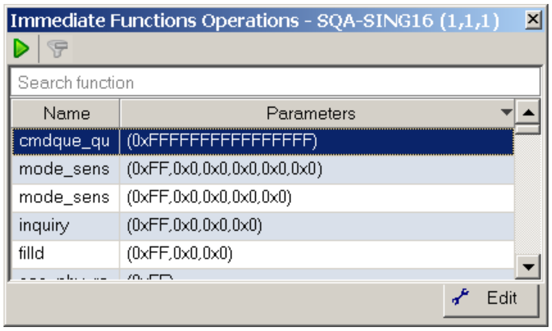
Figure 30: Parameters Status Table Start Function Button



	SAS/SATA QA_GEN5 (1,2,1
State	Stopped
Configuration	Edit...
Immediate Function Execution	▷ No Functions File Loaded
Macro File Operations...	No Macro File Loaded
Compiled Test Operations...	No Compiled Test Loaded

The Immediate Functions Operations window is displayed.

Figure 31: Immediate Functions Operations Window



Immediate Functions Operations - SQA-SING16 (1,1,1)

Search function

Name	Parameters
cmdque_qu	(0xFFFFFFFFFFFFFFFF)
mode_sens	(0xFF,0x0,0x0,0x0,0x0,0x0)
mode_sens	(0xFF,0x0,0x0,0x0,0x0)
inquiry	(0xFF,0x0,0x0,0x0)
filld	(0xFF,0x0,0x0)

2 Locate the function you want to execute by using the scroll bar or by typing the name of the function into the **Search Function** field.

3 Set the parameters as necessary for the selected function. The right-click menu provides editing options. These options are described in "Editing Function Parameters" on page 49".

4 Click the **Start** ▷ button to execute the function.

Opening a Functions File

To open a recently used Xgig Generator functions file:

1 Click the arrow next to the **Open Functions file** 📂▾ button in the menu bar.

2 Click the functions file you want to open.

An **Immediate Execution** window appears with the functions file loaded.

To browse to a functions file, and open it:

1 Click the **Open Functions file** 📂▾ button in the menu bar.

2 Browse to the functions file you want to open.

3 Click the **Open** button.

An **Immediate Execution** window appears with the functions file.

Creating a New Functions File

A functions file is a `.gfn` file that contains a list of functions. You can use a functions file as a repository to group a specified set of functions with a certain set of parameters. You can also use a functions file to define and maintain the order of a group of functions for use in a macro file. In addition, you can select, edit parameters for, and execute individual functions from a function file.

To create a new macro or functions file using the **File** menu:

>> Select **New**, then select **Functions File**

To create a new functions file from the Contents pane, click the **New Generator Functions File** icon in the Contents pane menu bar.

A blank file is opened on the tab.

Adding Functions to a Functions File

To add a function from the function library to the functions file:

1 Click the index position in the Contents pane where you want to add a function.

2 Click the function in the function library that you want to add to the functions file.

3 Click the **Send Selected Function to Functions File** button.

The function appears in the Contents pane at the selected index position.

Note: A convenient shortcut for adding functions to a file is to hold down the *Ctrl* key on your keyboard, then click and drag the function file to the Contents pane.

4 Click the Save or Save as button at the top of the window to save the functions file.

5 Click the Done button to close the window.

Moving Functions within a Functions File

To move a function up or down in within the functions file:

1 Click the function you want to move.

2 Click the **Move Up**/**Move Down** buttons at the top of the Contents pane.

To remove a function you have added to your functions file:

1 Click the function you want to remove from the file.

2 Click the **Delete Selected User Function** button at the top of the Contents pane.

You can also interchange one function with another, move a function for another index location, or copy a function to another index location by right-clicking the function and selecting the appropriate action.

Saving a Functions File

To save a functions file, click the **Save** button.

To save the file as another name or to another location:

1. Click the **Save As** button.

   The **Save**/**Save As** dialog appears.

Note: If you want to use the standard Windows file dialogs instead of the dialogs customized for Generator, select **Use Standard Windows File Dialogs** from the **Options** menu in the Generator Device window menu bar.

2. Navigate to the location where you want to save the file.
3. Type a name for the file.
4. Click **Save.**
5. Click the **Done** button to close the window. If you make changes to the file and then click try to close the window, you will receive prompt asking you if you want to save the file before closing it.

Macro File Tab

Opening a Macro File

To open a recently used Xgig Generator macro file:

1 Click the arrow next to the **Open Macro file** 📂▾ button in the menu bar.

2 Click the macro file you want to open.

A macro window appears with the macro loaded.

To browse to a macro file and open it:

1 Click the **Open Macro file** 📂▾ button in the menu bar.

2 Browse to the macro file you want to open.

3 Click the **Open** button.

A macro window appears with the macro loaded.

Editing a Macro File

The macro editor enables you to easily create macros for automating a test sequence. Macros are ideal for non programmers because they are quick and easy to create and don’t require a C compiler or knowledge of C programming. The macro language, however, has limitations. You can’t write a macro test that takes pointers or addresses as function parameters.

A macro file is a `.key` file that can be created with any ASCII editor or by using the Macro File Editor. A macro file can contain up to 62 user-defined macro tests (0–9, A–Z, and a–z). Each macro test contains a sequence of functions to be called upon when invoking the macro test. In addition to functions, macro tests can also include variables, branches, jumps, loops, and macro functions.

➡ **Note:** This section pertains to individual macro files, which can be created and edited independently of a device. You do not need to lock ports to perform these actions. The Macro Editor window described in this chapter is separate from the Generator Device window, which allows a user to manage and execute all configuration files for a selected device from a central window.

Creating a New Macro File

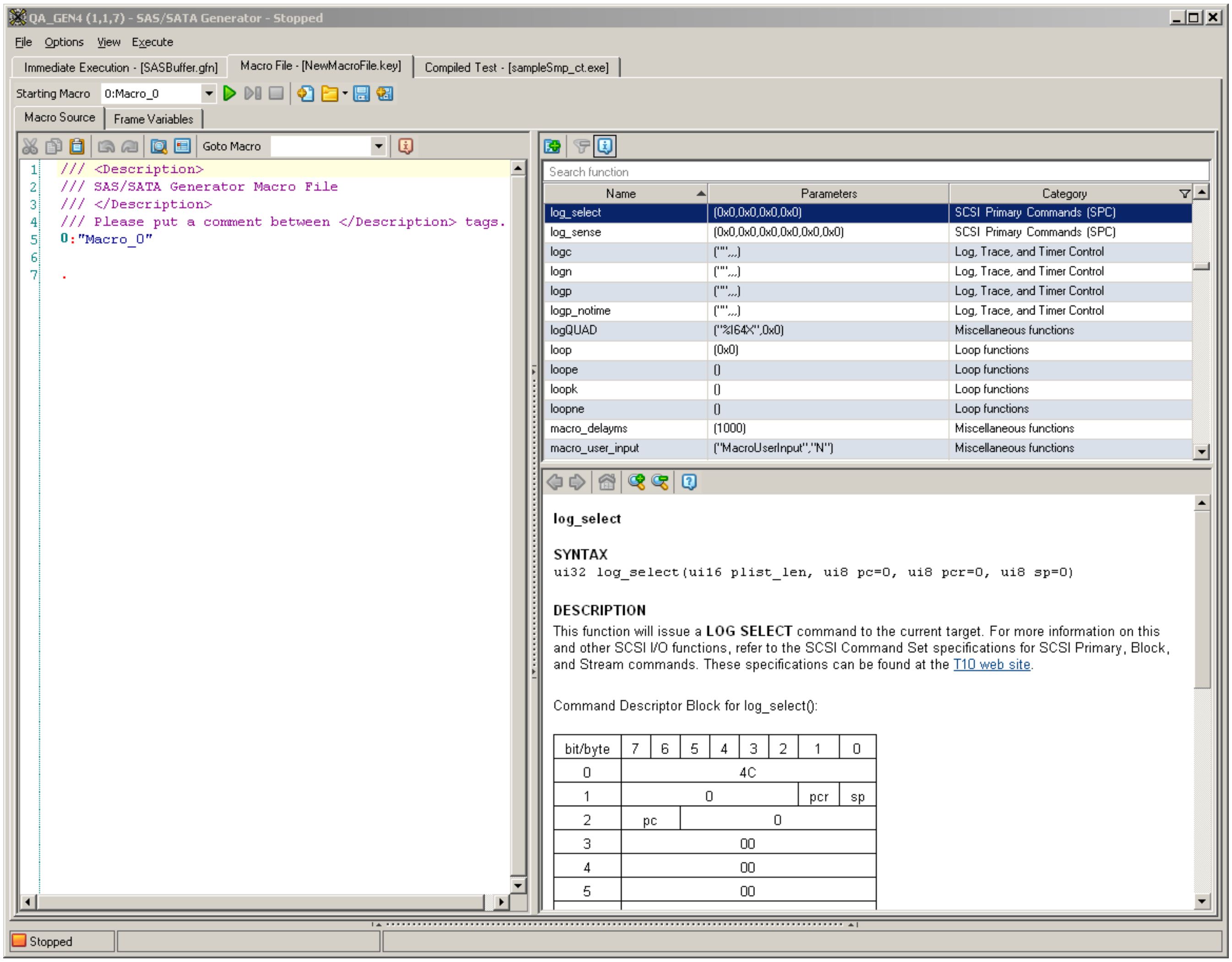
To create a new Xgig Generator Macro file:

1 Select the Macro Files tab in the Configuration window.

2 Click the **New Macro File** icon in the Configuration menu.

A new Macro Editor window appears.

Figure 32: Macro File Editor Window with Macro File Loaded


QA_GEN4 (1,1,7) - SAS/SATA Generator - Stopped
File Options View Execute
Immediate Execution - [SASBuffer.gfn]
Macro File - [NewMacroFile.key]
Compiled Test - [sampleSmp_ct.exe]
Macro Source
Frame Variables
Goto Macro
Stopped

Starting Macro 0:Macro_0

```
/// <Description>
/// SAS/SATA Generator Macro File
/// </Description>
/// Please put a comment between </Description> tags.
0:"Macro_0"
```

Search function

Name	Parameters	Category
log_select	(0x0,0x0,0x0,0x0)	SCSI Primary Commands (SPC)
log_sense	(0x0,0x0,0x0,0x0,0x0,0x0)	SCSI Primary Commands (SPC)
logc	("",...)	Log, Trace, and Timer Control
logn	("",...)	Log, Trace, and Timer Control
logp	("",...)	Log, Trace, and Timer Control
logp_notime	("",...)	Log, Trace, and Timer Control
logQUAD	("%I64X",0x0)	Miscellaneous functions
loop	(0x0)	Loop functions
loope	()	Loop functions
loopk	()	Loop functions
loopne	()	Loop functions
macro_delayms	(1000)	Miscellaneous functions
macro_user_input	("MacroUserInput","N")	Miscellaneous functions

log_select

SYNTAX

```
ui32 log_select(ui16 plist_len, ui8 pc=0, ui8 pcr=0, ui8 sp=0)
```

DESCRIPTION

This function will issue a **LOG SELECT** command to the current target. For more information on this and other SCSI I/O functions, refer to the SCSI Command Set specifications for SCSI Primary, Block, and Stream commands. These specifications can be found at the T10 web site.

Command Descriptor Block for log_select():

bit/byte	7	6	5	4	3	2	1	0
0	4C							
1	0						pcr	sp
2	pc		0					
3	00							
4	00							
5	00							

Macro File Structure

When creating macro tests, you must strictly adhere to the correct syntax.

- Extraneous spaces are generally not allowed.
- Numeric variables are decimal values unless they are preceded by 0x, in which case, they are hexadecimal.

In a new macro file, the macro content begins with description tags. This is a placeholder for the description of your file. The description you type here will appear in the Details pane when the file is selected in the Macro Files tab in Generator.

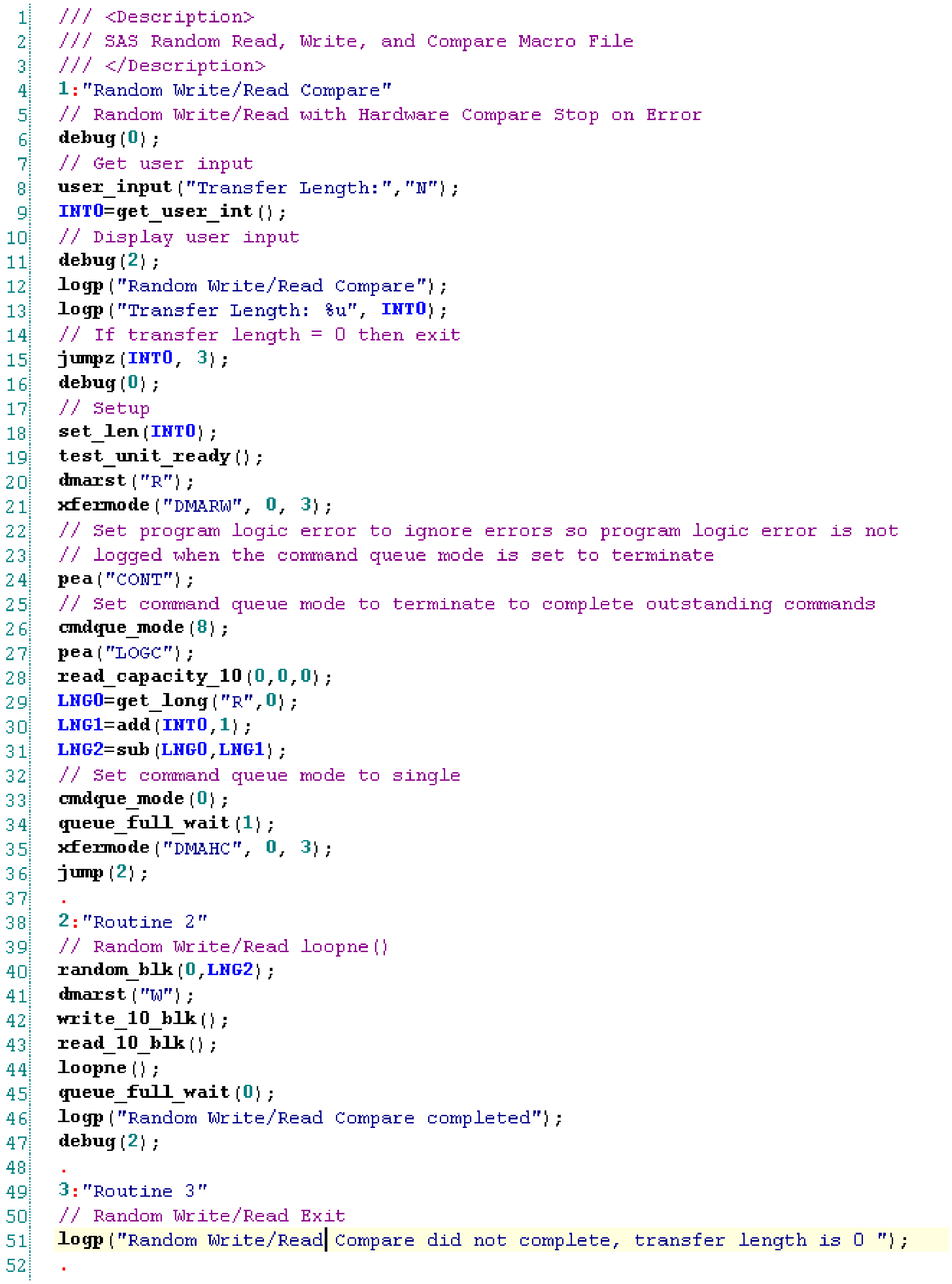
The list of functions in a macro always starts with a number (default = 0) followed by a red colon and ends with a red period. When you open a new macro, the beginning of the macro is indicated by a “`0:`”, and the end is indicated by a “`.`” on a new line as shown in Figure 32 on page 56.

You can either choose to build your macro as one string of functions, or you can separate it into routines. Essentially, each routine is a complete macro, beginning and ending with the symbols listed above. Each routine should be numbered sequentially. As mentioned above, the default number for the beginning of a macro is zero. You can change this number to one or another number if you choose.

You need to give each macro or routine a name. The name should be enclosed in quotation marks and be placed immediately after the red colon indicating the beginning of the macro (no space).

Begin populating your macro with functions. Place a semicolon after each function. Place comments after // on a new line. A single comment that extends over multiple lines in the macro file must have a comment indicator // at the beginning of each line. Figure 33 on page 58 shows a completed macro containing three routines.

Figure 33: Sample Macro File Contents



```
/// <Description>
/// SAS Random Read, Write, and Compare Macro File
/// </Description>
1:"Random Write/Read Compare"
// Random Write/Read with Hardware Compare Stop on Error
debug(0);
// Get user input
user_input("Transfer Length:","N");
INT0=get_user_int();
// Display user input
debug(2);
logp("Random Write/Read Compare");
logp("Transfer Length: %u", INT0);
// If transfer length = 0 then exit
jumpz(INT0, 3);
debug(0);
// Setup
set_len(INT0);
test_unit_ready();
dmarst("R");
xfermode("DMARW", 0, 3);
// Set program logic error to ignore errors so program logic error is not
// logged when the command queue mode is set to terminate
pea("CONT");
// Set command queue mode to terminate to complete outstanding commands
cmdque_mode(8);
pea("LOGC");
read_capacity_10(0,0,0);
LNG0=get_long("R",0);
LNG1=add(INT0,1);
LNG2=sub(LNG0,LNG1);
// Set command queue mode to single
cmdque_mode(0);
queue_full_wait(1);
xfermode("DMAHC", 0, 3);
jump(2);
.
2:"Routine 2"
// Random Write/Read loopne()
random_blk(0,LNG2);
dmarst("W");
write_10_blk();
read_10_blk();
loopne();
queue_full_wait(0);
logp("Random Write/Read Compare completed");
debug(2);
.
3:"Routine 3"
// Random Write/Read Exit
logp("Random Write/Read Compare did not complete, transfer length is 0 ");
.
```

Adding Functions to a Macro File

To add a function from the function library to a macro file:

1. Place the cursor at the location where you want to add the function.
2. Click the function in the function library that you want to add to the macro file.
3. Click the Add Function to Macro button.

   The function appears in the Contents pane at the selected position.

Note: A convenient shortcut for adding functions to a macro file is to hold down the **Ctrl** button on your keyboard, then click and drag the function file to the Contents pane.

4. Click the Save or Save as button at the top of the window to save the macro file.
5. Set the parameters as necessary for each function. The right-click menu provides editing options. These options are described in “Editing Function Parameters” on page 49.
6. Refer to Appendix A, “Macro Variables and Functions and the Adding Frame Variables to a Macro section below for function and variable types and syntax as well as examples.
7. Indicate the end of the macro with a period on a line by itself.
8. Follow steps 1–3 to add the desired macros to the macro file, sequentially numbering the start of each one.

Adding Frame Variables to a Macro

The frame editor is an easy-to-use feature for customizing both legal and illegal SAS and SATA frames to send to a device. You can create and edit frame variables in the Macro File Editor window.

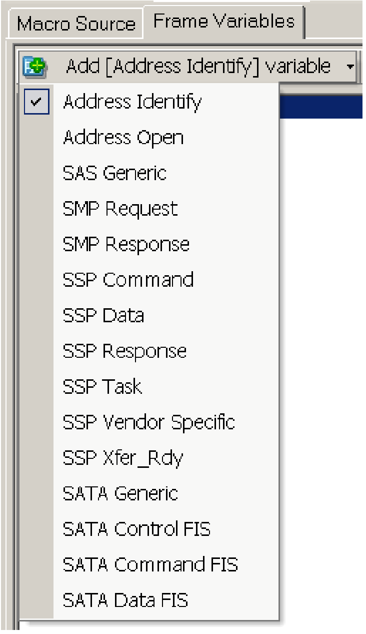
To access the Frame Editor, open the macro file that you want to add frame variables to, then select the Frame Variables tab.

To add a frame to a macro:

1 Select a frame variable, i.e. FRM0, FRM1, etc.

2 Click the **Add [...] variable** drop-down menu, and select the frame type that you want to use.

Figure 34: Frame Variables Drop-Down Menu



Macro Source | Frame Variables

Add [Address Identify] variable

☑ Address Identify
Address Open
SAS Generic
SMP Request
SMP Response
SSP Command
SSP Data
SSP Response
SSP Task
SSP Vendor Specific
SSP Xfer_Rdy
SATA Generic
SATA Control FIS
SATA Command FIS
SATA Data FIS

Generator will fill in the fields in the right pane depending on which frame type you selected.

3 Type a name for the frame variable in the **Name** field.

4 To set the destination address for a SAS variable, click the **Get** button next to the **Dest Addr** field. Once you’ve set the address, click **Set**.

This address indicates where the frame is being sent. If the frame type is not Identify or Open Address, this is the value that will set the hashed address for the frame and will be used in the automatically generate open address frame that will precede the specified frame.

5 To get the source address for a SAS variable, click the **Get** button next to the **Src Addr** field.

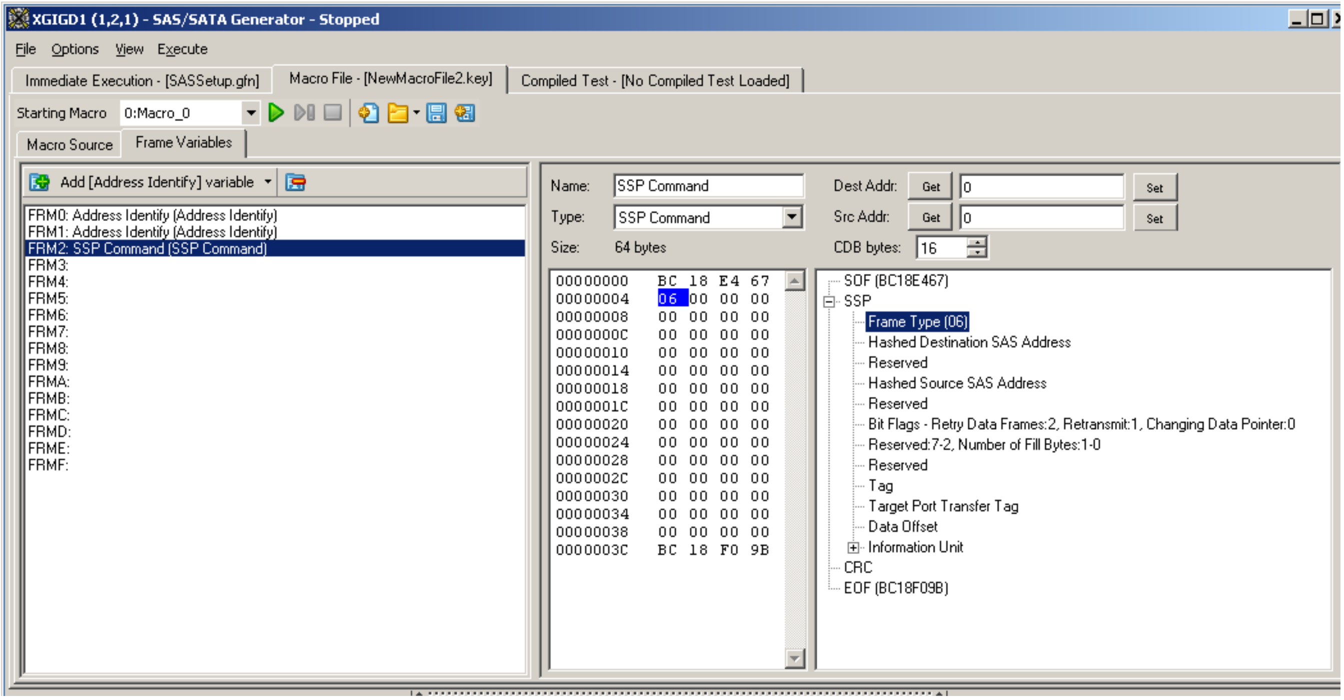
This address indicates where the frame is coming from. If the frame type is not Identify or Open Address, this is the value that will set the hashed address for the frame and will be used in the automatically generate open address frame that will precede the specified frame.

➡ **Note:** SATA variables use a logical block address (LBA) setting instead of source and destination addresses.

6 Type an LBA address for the variable in the **LBA** field. This address indicates the logical block address used to identify the frame. Once you've set the address, click **Set**.

The contents of the frame is listed in the following two panes as shown in Figure 34.

Figure 35: Frame Variables Window


XGIGD1 (1,2,1) - SAS/SATA Generator - Stopped
File
Options
View
Execute
Immediate Execution - [SASSetup.gfn]
Macro File - [NewMacroFile2.key]
Compiled Test - [No Compiled Test Loaded]
Macro Source
Frame Variables
Add [Address Identify] variable
Get
Set
Get
Set

Starting Macro: 0:Macro_0

FRM0: Address Identify (Address Identify)
FRM1: Address Identify (Address Identify)
FRM2: SSP Command (SSP Command)
FRM3:
FRM4:
FRM5:
FRM6:
FRM7:
FRM8:
FRM9:
FRMA:
FRMB:
FRMC:
FRMD:
FRME:
FRMF:

Name: SSP Command
Type: SSP Command
Size: 64 bytes
Dest Addr: 0
Src Addr: 0
CDB bytes: 16

```
00000000   BC 18 E4 67
00000004   06 00 00 00
00000008   00 00 00 00
0000000C   00 00 00 00
00000010   00 00 00 00
00000014   00 00 00 00
00000018   00 00 00 00
0000001C   00 00 00 00
00000020   00 00 00 00
00000024   00 00 00 00
00000028   00 00 00 00
0000002C   00 00 00 00
00000030   00 00 00 00
00000034   00 00 00 00
00000038   00 00 00 00
0000003C   BC 18 F0 9B
```

```
SOF (BC18E467)
SSP
    Frame Type (06)
    Hashed Destination SAS Address
    Reserved
    Hashed Source SAS Address
    Reserved
    Bit Flags - Retry Data Frames:2, Retransmit:1, Changing Data Pointer:0
    Reserved:7-2, Number of Fill Bytes:1-0
    Reserved
    Tag
    Target Port Transfer Tag
    Data Offset
    Information Unit
CRC
EOF (BC18F09B)
```

Frame Decode Pane

This pane lists the contents of the specific frame. This information pane tracks with the frame hex data pane. When you click on node in the frame, the frame hex data size pane shows the number of dwords/bytes that make up that node. When you double-click a node with a defined value in the information pane, such as "Frame Type [06]", the corresponding hex value is set. The main components of a frame are:

- SOF
- Header (Expand this to show nodes)
- CRC
- EOF

Just above this pane are two fields. The first indicates the total size of the frame. The second, which allows you to add or remove certain types of bytes, such as FIS or CDB bytes from a frame, is only available for certain frame variable types. To add bytes to or remove bytes from the frame, click the up/down arrows next to the field.

Frame Hex Data Pane

This pane lists the bytes for a specific frame component. To edit the byte value:

— Select the byte value, then type another value.

You can add bytes to specific types of frames to increase the size of the frame. If the selected frame type allows the addition of bytes, a numeric field will be displayed above the **Frame Hex Data** pane.

Running a Macro on a Generator Device

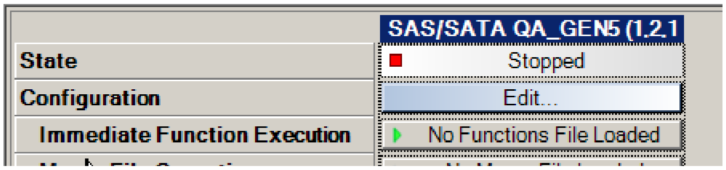
You can run a macro file from the Macro File tab of the Generator Device window or from the Parameters Status table in the Maestro window. When running macro files from the Parameters Status table, you can run the files on multiple devices simultaneously

To run a macro file for a device from the Generator Device window, click the **Start** button on the menu bar of the **Macro File** tab.

To run a single macro for a device from the Parameters Status table:

1 Click the **Start** button on the Parameters Status table for Macro File Operations. You need to have a macro file loaded to perform this action.

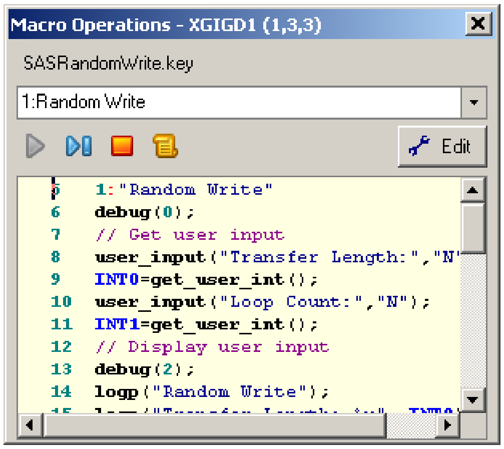
Figure 36: Parameters Status table Start Macro Button



	SAS/SATA QA_GEN5 (1,2,1
State	Stopped
Configuration	Edit...
Immediate Function Execution	No Functions File Loaded

The Macro Operations window is displayed.

Figure 37: Macro Operations Window



Macro Operations - XGIGD1 (1,3,3)

SASRandomWrite.key

1:Random Write

```
1:"Random Write"
debug(0);
// Get user input
user_input("Transfer Length:","N"
INT0=get_user_int();
user_input("Loop Count:","N");
INT1=get_user_int();
// Display user input
debug(2);
logp("Random Write");
```

2 Click the Preview Macro button if you want to preview the macro contents.

3 Click the **Start** button to run the macro.

4 Click the **Abort Macro** button to stop the execution of a macro.

5 Click the **Send Keystroke** button to simulate a keystroke to stop looping during macro execution. This selection is active only if the `loopk` function is invoked in your macro. Refer to the description for `loopk` in the function library.

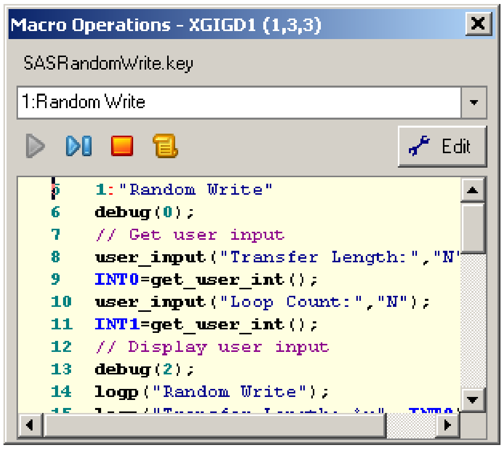
6 A log entry is generated.

To execute a macro for a device from the Ports Manager:

1 Click the Macro Operations Macro Operations button for the device.

The Macro Operations window is displayed.

Figure 38: Macro Operations Window



```
1:"Random Write"
debug(0);
// Get user input
user_input("Transfer Length:","N"
INT0=get_user_int();
user_input("Loop Count:","N");
INT1=get_user_int();
// Display user input
debug(2);
logp("Random Write");
```

2 Click the Preview Macro button if you want to preview the macro contents.

3 Click the **Start** button to run the macro.

4 A log entry is generated.

To simultaneously run macros on multiple devices from the Port Manager:

1 Select each device that you want to run macros on by checking the **Select** check box for the device.

2 Select the macro file you want to load from the Configuration Manager.

3 Click the **Load Configuration into Selected Ports** button.

4 Click the **Start Macro on Selected Ports** button.

5 A log entry is generated.

To stop a macro file in progress, right click the device column in the Parameter status table, and select **Stop Macro**.

Saving a Macro File

To save a macro file, click the **Save** button.

To save the file as another name or to another location:

1 Click the **Save As** button.

The **Save**/**Save As** dialog appears.

➡ **Note:** If you want to use the standard Windows file dialogs instead of the dialogs customized for Generator, select **Use Standard Windows File Dialogs** from the **Options** menu in the Generator Device window menu bar.

2 Navigate to the location where you want to save the file.

3 Type a name for the file.

4 Click **Save.**

5 Click the **Done** button to close the window. If you make changes to the file and then try to close the window, you will receive prompt asking you if you want to save the file before closing it.

Compiled Test Tab

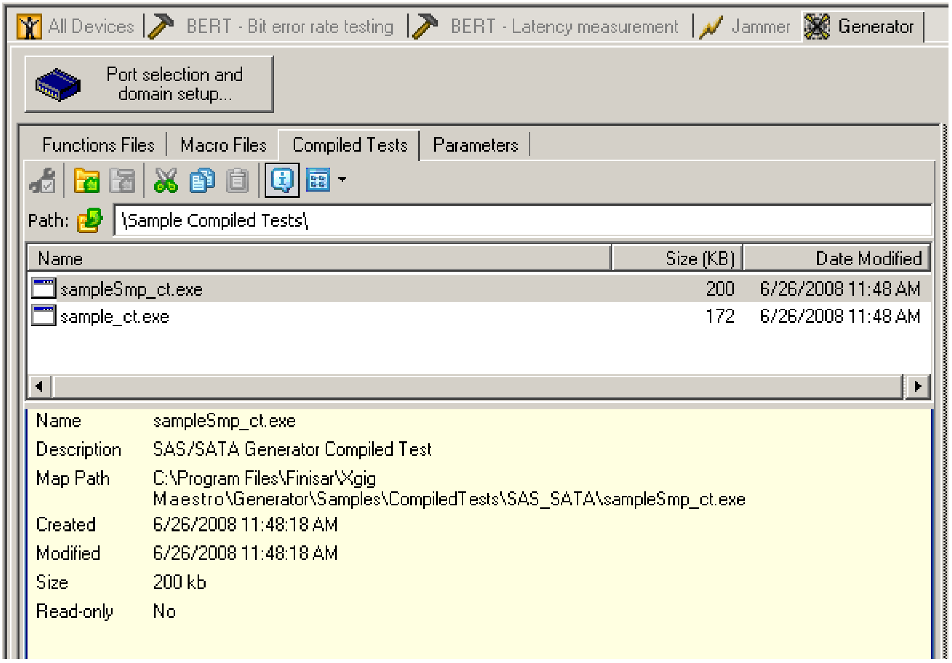
A compiled test file is a .exe file written using the SAS/SATA Generator API. Refer to the Generator API Help for instructions on creating compiled tests. The Generator GUI does not allow you to modify the .exe file. However, the GUI does allow you to add parameters to a complied test, which, when saved, creates a .gen file of the same name.

Using this section, you can add and edit parameters for a compiled test file, essentially, creating .gen files.

Sample Compiled Test Files

The Generator software includes sample compiled test files. These files are located in the Sample Files folder of the Configuration Manager in the Maestro Window as shown Figure 39.

Figure 39: Sample Compiled Test Folder


All Devices
BERT - Bit error rate testing
BERT - Latency measurement
Jammer
Generator
Port selection and domain setup...
Functions Files
Macro Files
Compiled Tests
Parameters

Path: \Sample Compiled Tests\

Name	Size (KB)	Date Modified
sampleSmp_ct.exe	200	6/26/2008 11:48 AM
sample_ct.exe	172	6/26/2008 11:48 AM

Name	sampleSmp_ct.exe
Description	SAS/SATA Generator Compiled Test
Map Path	C:\Program Files\Finisar\Xgig Maestro\Generator\Samples\CompiledTests\SAS_SATA\sampleSmp_ct.exe
Created	6/26/2008 11:48:18 AM
Modified	6/26/2008 11:48:18 AM
Size	200 kb
Read-only	No

This section gives a briefly describes the operations performed when running each of the sample tests.

Sample_ct

Upon execution, this test file opens the tester, requests the fpga version string then displays the returns in the command window listing the bit file version found to be running on your tester. The test then pauses and prompts you to press any key to continue. When you press a key, the test completes and closes the command window.

Samplesmp_ct

This test assumes that your tester is attached to an expander.

Upon execution, this test queries the expander, performs discover smp requests, retrieves the manufacturing information, performs report general requests, records report general responses, and lists the number of phys. For each phy, the test issues and smp discover request and displays the returns in the command window. The test then pauses and prompts you to press any key to continue. When you press a key, the test completes and closes the command window.

Opening a Compiled Test File

To open a recently used Xgig Generator compiled test file:

1. Click the arrow next to the **Open Compiled Test** 📂▾ button in the menu bar.
2. Click the compiled test file you want to open.

   A compiled test window appears with the compiled test file loaded.

To browse to a compiled test and open it:

1. Click the **Open Compiled Test** 📂▾ button in the menu bar.
2. Browse to the compiled test file you want to open.
3. Click the **Open** button.

   A compiled test window appears with the compiled test loaded.

Adding Parameters to a Compiled Test File

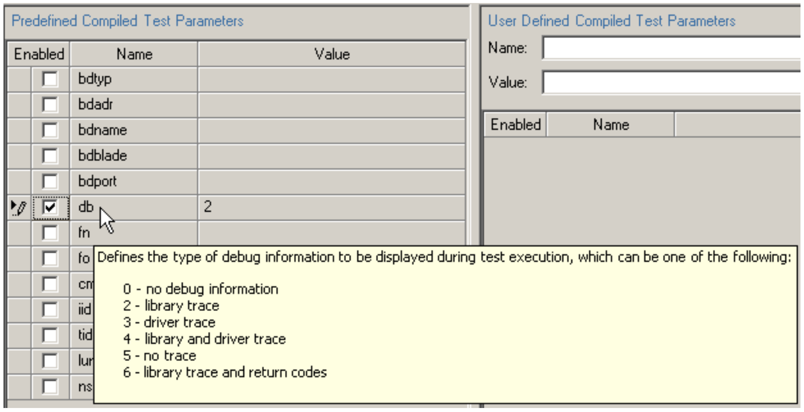
The **Description** field allows you to type a description for the compiled test. This description will appear in information panes when the file is selected.

The **Predefined Compiled Test Parameters** pane lists common command line parameters. These parameters are listed in the **Common Command Line Parameters** section of the API Help. Holding the mouse over one of these parameters brings up a tool tip window containing information about the parameter such as the definition, use, and defined or suggested values.

Example:

Holding the mouse over the `bdtyp` parameter displays the following tool tip.

Figure 40: Bdtyp Compiled Test Parameter Tool Tip


Predefined Compiled Test Parameters
User Defined Compiled Test Parameters
Name:
Value:
Enabled
Name

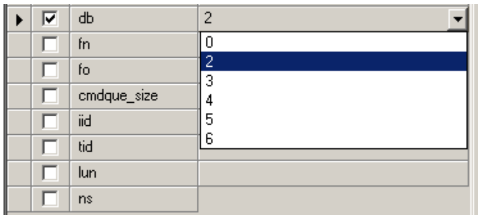
Enabled	Name	Value
☐	bdtyp	
☐	bdadr	
☐	bdname	
☐	bdblade	
☐	bdport	
☑	db	2
☐	fn	
☐	fo	
☐	cm	
☐	iid	
☐	tid	
☐	lur	
☐	ns	

Defines the type of debug information to be displayed during test execution, which can be one of the following:

0 - no debug information
2 - library trace
3 - driver trace
4 - library and driver trace
5 - no trace
6 - library trace and return codes

The tool tip describes the parameter and lists the possible values. Selecting this row and clicking the value drop-down menu, the possible values as shown in the following figure.

Figure 41: Bdtyp Compiled Test Parameter Values



☑	db	2
☐	fn	
☐	fo	
☐	cmdque_size	
☐	iid	
☐	tid	
☐	lun	
☐	ns	

Click the row with the desired parameter to make it editable. The ✎ icon appears next to the enabled field to indicate that this is the parameter you are editing.

To select a value for this parameter, click the value you want from the drop-down list.

➡ **Note:** Not all parameter values fields have a drop-down menu for the value. Some tool tips indicate a value range or define the type of information required and the format required.

To enable a parameter for the compiled test, select the **Enable** check box next to the parameter.

The **User Defined Compiled Test Parameters** pane allows you to create your own parameters.

To create a parameter:

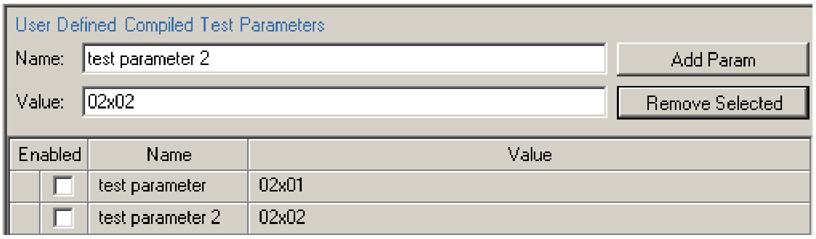
1 Type a name in the **Name** field.

2 Type a value in the **Value** field.

3 Click the **Add Parameter** button.

The parameter appears in the list below as shown in the following figure.

Figure 42: User Defined Parameter



User Defined Compiled Test Parameters

Name: test parameter 2

Value: 02x02

Enabled	Name	Value
□	test parameter	02x01
□	test parameter 2	02x02

4 Click the row with the desired parameter to make it editable. The ✎ icon appears next to the enabled field to indicate that this is the parameter you are editing.

To enable a parameter for the compiled test, select the **Enable** check box next to the parameter.

To delete the parameter, select the entire row, and click the **Remove Selected** button.

Running a Compiled Test on a Generator Device

After you have configured a Generator device and loaded a compiled test file, you can run the file.

Note: Before launching a compiled test, the GUI disconnects from the port then reconnects to the port when the test is complete.

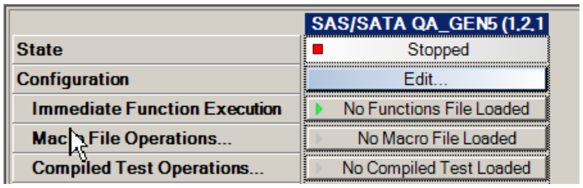
You can run a compiled test file from the Compiled Test tab of the Generator Device window or from the Parameters Status table in the Maestro window. When running compiled tests from the Parameters Status table, you can run the test files on multiple devices simultaneously.

To run a compiled test for a device from the Generator Device window, click the **Start** ▷ button on the menu bar.

To run a single compiled test file for a device from the Parameters Status table:

1 Click the **Start** ▷ button on the Parameters Status table next to the Compiled Test Operations field. You need to have a compiled test file loaded to perform this action.

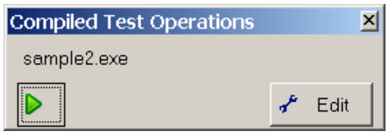
Figure 43: Parameters Status table Start Compiled Test Button



	SAS/SATA QA_GEN5 (1,2,1
State	🟥 Stopped
Configuration	Edit...
Immediate Function Execution	▷ No Functions File Loaded
Macro File Operations...	No Macro File Loaded
Compiled Test Operations...	No Compiled Test Loaded

The Compiled Test Operations window is displayed.

Figure 44: Compiled Test Operations Window



Compiled Test Operations

sample2.exe

2 Click the **Start** ▷ button to run the compiled test.

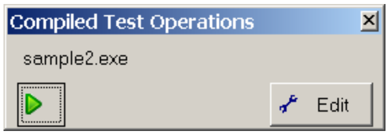
3 A log entry is generated.

To execute a compiled test for a device from the Ports Manager:

1 Click the Compiled Test Operations Compiled Test Operations button for the device.

The Compiled Test Operations window is displayed.

Figure 45: Compiled Test Operations Window



2 Click the **Start** ▷ button to run the compiled test.

3 A log entry is generated.

To simultaneously run compiled tests on multiple devices from the Port Manager:

1 Select each device that you want to run compiled tests on by checking the **Select** check box for the device.

2 Select the compiled test file you want to load.

3 Click the **Load Configuration into Selected Ports** button.

4 Click the **Start Compiled Tests on Selected Ports** button.

5 A log entry is generated.

Saving a Compiled Test File

To save a compiled test file, click the **Save** button.

To save the file as another name or to another location:

1 Click the **Save As** button.

The **Save**/**Save As** dialog appears.

Note: If you want to use the standard Windows file dialogs instead of the dialogs customized for Generator, select **Use Standard Windows File Dialogs** from the **Options** menu in the Generator Device window menu bar.

If you click the **Save as** selection to save a `.gen file` as a new file name, a copy of the existing compiled test `.exe` will be created with along with the new `.gen` file.

2 Navigate to the location where you want to save the file.

3 Type a name for the file.

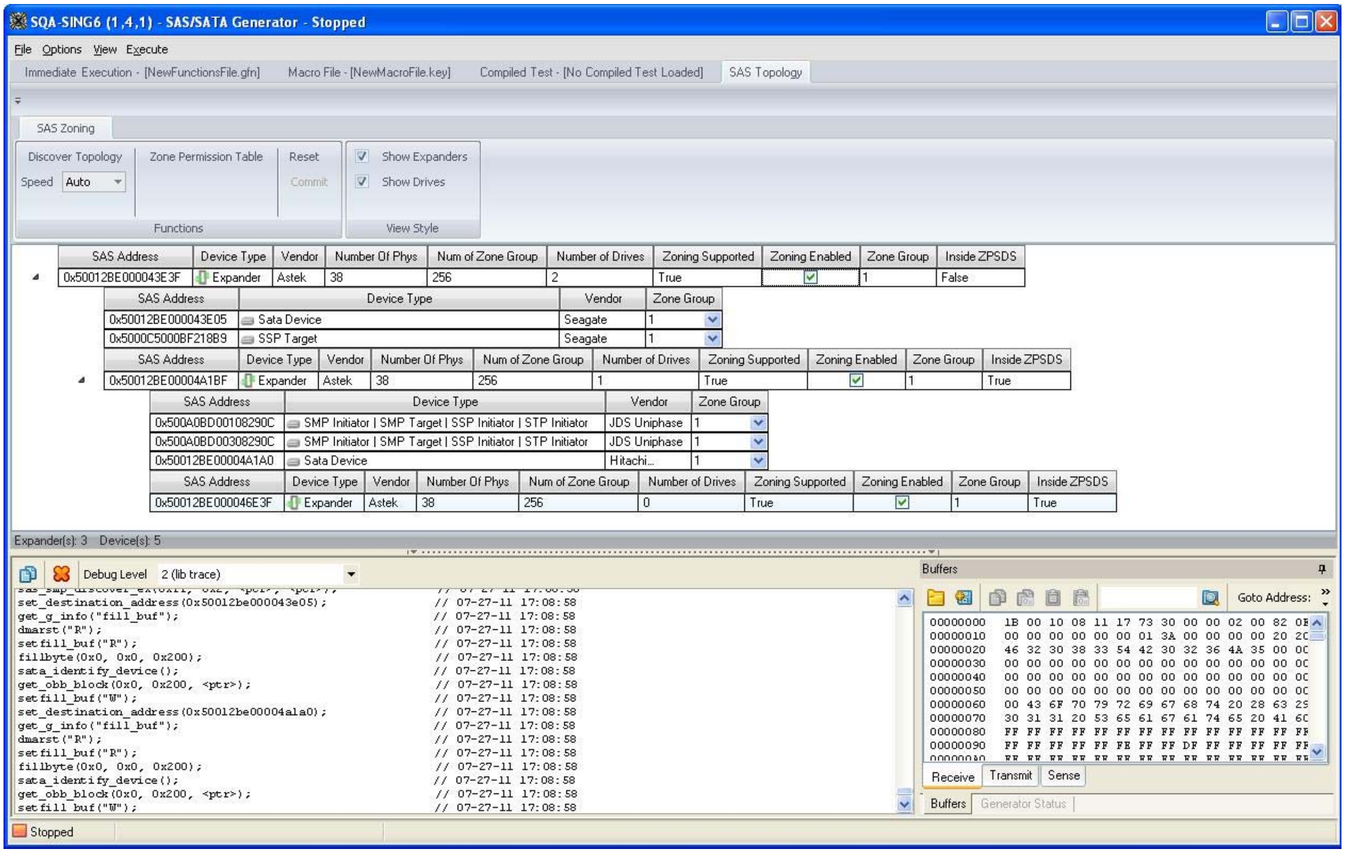
4 Click **Save.**

5 Click the **Done** button to close the window. If you make changes to the file and then try to close the window, you will receive prompt asking you if you want to save the file before closing it.

Using SAS Zoning Topology

Maestro Generator lets you display and manage the SAS Zoning topology of connected expanders and drives. It also allows you to modify the targets' zoning properties such as enable/disable zoning, set a zone group and modify the zone permission table and allows you to commit changes made on the targets' zoning properties.

Figure 46: SAS Zoning Topology Window


SQA-SING6 (1,4,1) - SAS/SATA Generator - Stopped
File Options View Execute
Immediate Execution - [NewFunctionsFile.gfn]
Macro File - [NewMacroFile.key]
Compiled Test - [No Compiled Test Loaded]
SAS Topology
SAS Zoning
Discover Topology
Zone Permission Table
Reset
Commit
Functions
View Style
Buffers
Goto Address:
Receive
Transmit
Sense
Buffers
Generator Status

Speed: Auto

☑ Show Expanders

☑ Show Drives

SAS Address	Device Type	Vendor	Number Of Phys	Num of Zone Group	Number of Drives	Zoning Supported	Zoning Enabled	Zone Group	Inside ZPSDS
0x50012BE000043E3F	Expander	Astek	38	256	2	True	☑	1	False

SAS Address	Device Type	Vendor	Zone Group
0x50012BE000043E05	Sata Device	Seagate	1
0x5000C5000BF218B9	SSP Target	Seagate	1

SAS Address	Device Type	Vendor	Number Of Phys	Num of Zone Group	Number of Drives	Zoning Supported	Zoning Enabled	Zone Group	Inside ZPSDS
0x50012BE00004A1BF	Expander	Astek	38	256	1	True	☑	1	True

SAS Address	Device Type	Vendor	Zone Group
0x500A0BD00108290C	SMP Initiator | SMP Target | SSP Initiator | STP Initiator	JDS Uniphase	1
0x500A0BD00308290C	SMP Initiator | SMP Target | SSP Initiator | STP Initiator	JDS Uniphase	1
0x50012BE00004A1A0	Sata Device	Hitachi...	1

SAS Address	Device Type	Vendor	Number Of Phys	Num of Zone Group	Number of Drives	Zoning Supported	Zoning Enabled	Zone Group	Inside ZPSDS
0x50012BE000046E3F	Expander	Astek	38	256	0	True	☑	1	True

Expander(s): 3 Device(s): 5

Debug Level: 2 (lib trace)

```
set_destination_address(0x50012be000043e05);        // 07-27-11 17:08:58
get_g_info("fill_buf");                             // 07-27-11 17:08:58
dmarst("R");                                        // 07-27-11 17:08:58
setfill_buf("R");                                   // 07-27-11 17:08:58
fillbyte(0x0, 0x0, 0x200);                          // 07-27-11 17:08:58
sata_identify_device();                             // 07-27-11 17:08:58
get_obb_block(0x0, 0x200, <ptr>);                   // 07-27-11 17:08:58
setfill_buf("W");                                   // 07-27-11 17:08:58
set_destination_address(0x50012be00004a1a0);        // 07-27-11 17:08:58
get_g_info("fill_buf");                             // 07-27-11 17:08:58
dmarst("R");                                        // 07-27-11 17:08:58
setfill_buf("R");                                   // 07-27-11 17:08:58
fillbyte(0x0, 0x0, 0x200);                          // 07-27-11 17:08:58
sata_identify_device();                             // 07-27-11 17:08:58
get_obb_block(0x0, 0x200, <ptr>);                   // 07-27-11 17:08:58
setfill_buf("W");                                   // 07-27-11 17:08:58
```

```
00000000   1B 00 10 08 11 17 73 30 00 00 02 00 82 0E
00000010   00 00 00 00 00 00 01 3A 00 00 00 00 20 20
00000020   46 32 30 38 33 54 42 30 32 36 4A 35 00 00
00000030   00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000040   00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000050   00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000060   00 43 6F 70 79 72 69 67 68 74 20 28 63 29
00000070   30 31 31 20 53 65 61 67 61 74 65 20 41 6C
00000080   FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000090   FF FF FF FF FF FE FF FF DF FF FF FF FF FF
```

Stopped

Functional Bar

The Functional Bar provides all functions that you can access. There are 4 functions:

Discover Topology

This function discovers the SAS Zoning topology used by the system. You must execute this function before other functions are accessible.

Speed

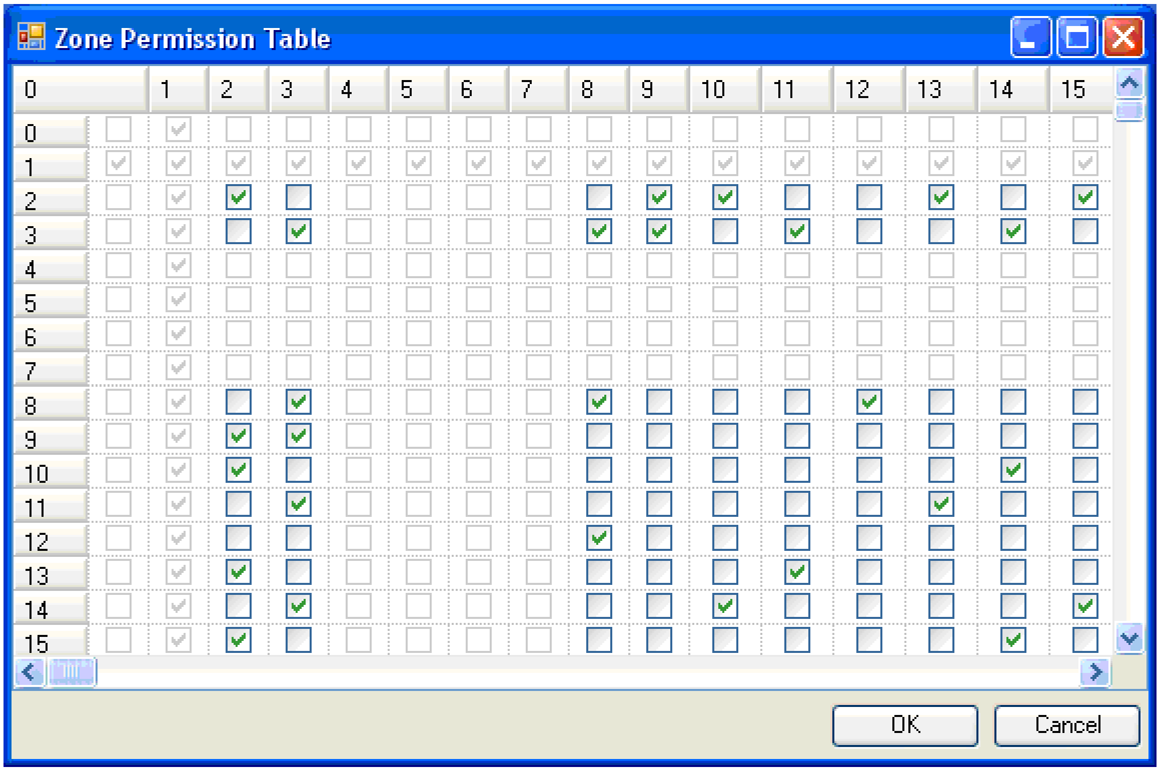
Displays available speeds. Currently SAS topology supports 1.5G, 3G, 6G and 12G. (12G is only available for 12G SAS blade or X1K chassis.)

Zone Permission Table

This function opens up the **Zone Permission Table** window where you can view a grid where each column represents a source group and each row represents a destination group.

Zone Permission Table Window

Figure 47: Zone Permission Table



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
1	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
2	☐	☑	☑	☐	☐	☐	☐	☐	☐	☑	☑	☐	☐	☑	☐	☑
3	☐	☑	☐	☑	☐	☐	☐	☐	☑	☑	☐	☑	☐	☐	☑	☐
4	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
5	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
6	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
7	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
8	☐	☑	☐	☑	☐	☐	☐	☐	☑	☐	☐	☐	☑	☐	☐	☐
9	☐	☑	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
10	☐	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☐
11	☐	☑	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☐	☐
12	☐	☑	☐	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☐	☐
13	☐	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐
14	☐	☑	☐	☑	☐	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☑
15	☐	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☐

The Zone Permission Table is either a 128 by 128 or a 256 by 256 grid with check box in each cell. The NUMBER OF ZONE GROUPS field returned by REPORT GENERAL Function indicates the number of zone groups (for example, the number of entries in the zone group permission table) supported by the expander device and is defined in table 256(refer to *Specification Information technology - Serial Attached SCSI - 2 (SAS-2), Revision 16, 18 April 2009, Section 10.4.3.4 REPORT GENERAL function*). In the grid, each column represents a source group and each row represents a destination group. Because as per the specification (refer to *4.9.3.2 Zone groups*), group 0 and 1 are not configurable and group 4-7 is reserved, corresponding columns and rows should be disabled.

Reset

This function resets the group of each device to group 1, which has all access to other groups.

Commit

This function commits all changes to expanders.

Status Bar

The Status Bar displays useful and read-only information to users such as the number of expanders and number of devices.

Device View

The Device View has two modes:

Device Mode:

When you select the **Show Expanders** check box, the device view will enter **Device Mode. I**n this mode, Sas Zoning topology is displayed in a tree hierarchy. This mode is editable. You can modify properties of expanders and drives, such as enable/disable zoning and zoning group. To apply any of the changes you made, click **Commit**.

Drive only Mode:

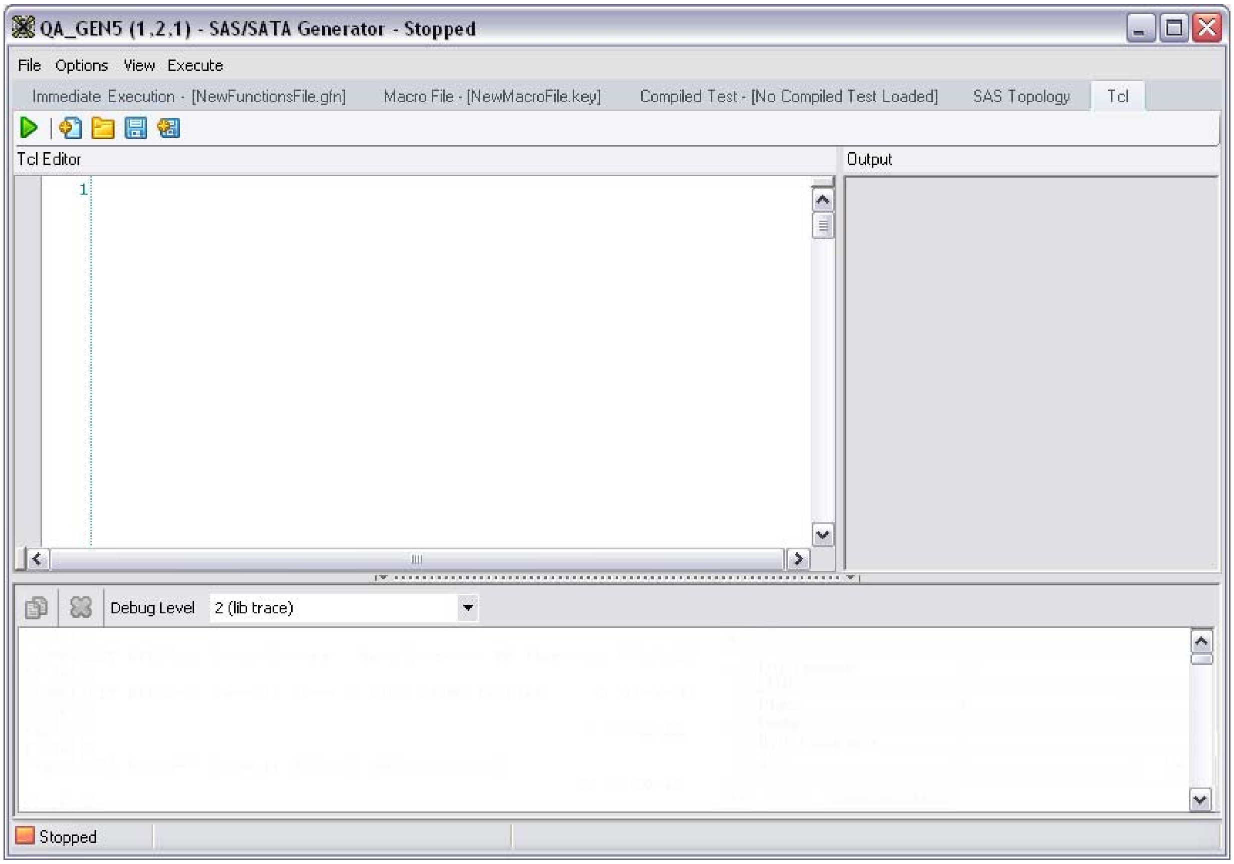
When you select the **Show Drives** check box, the device view will enter **Drive only Mode**. This mode is read-only. The main purpose of this mode is to report the state of function calls on an individual drive. Information shown in drive mode can help you locate the source of an error.

Using the Tcl Script Editor

The Tcl script editor (shown in Figure 48) provides Tcl (Tool Command Language) programming support for all existing Generator APIs in the Generator GUI. It makes writing, reading, running and debugging TCL scripts easier and more intuitive. When the Tcl scripts are run, the GUI passes the Tcl commands to a wrapper which interprets them and calls the corresponding C/C++ Generator API functions. The output results are returned to the GUI and displayed in the Output pane.

Note: In this version of Generator, the Tcl API is a synchronized API. For best performance, you should not run time-consuming functions. As a workaround, you can break down the long functions into shorter functions and execute these shorter functions separately.

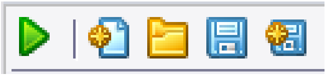
Figure 48: Tcl Script Editor


QA_GEN5 (1,2,1) - SAS/SATA Generator - Stopped
File
Options
View
Execute
Immediate Execution - [NewFunctionsFile.gfn]
Macro File - [NewMacroFile.key]
Compiled Test - [No Compiled Test Loaded]
SAS Topology
Tcl
Tcl Editor
1
Output
Stopped

Debug Level: 2 (lib trace)

The Tcl Editor has its own tool bar which is shown in Figure 49.

Figure 49: Tcl Editor Tool Bar



The tool bar has five buttons that you can use to enter, open, run, or save the Tcl commands.

Start Tcl runs the Tcl commands that are displayed in the Tcl Editor pane.

New Tcl File clears any entries in the Tcl Editor pane so that you can open an existing Tcl command file or enter new Tcl commands directly into the Tcl Editor pane.

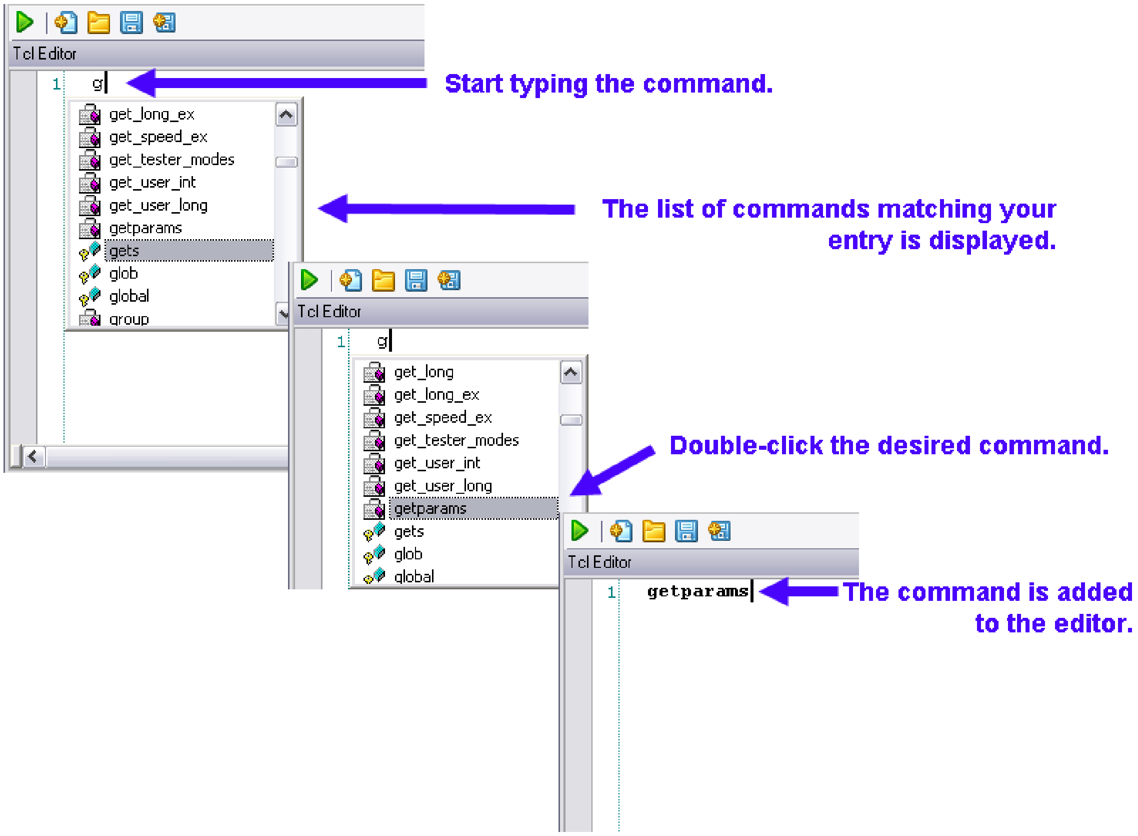
Open Tcl File opens saved Tcl files. You will be prompted to browse to the location and select the saved Tcl file.

Save Tcl File saves the Tcl commands in the editor pane as a Tcl file. If the file was opened previously, a re-save is performed. If these are new commands, you will be prompted to provide the filename and location of the Tcl file.

Save Tcl File As... saves the Tcl commands as a new Tcl file. You will be prompted to provide the filename and location of the Tcl file.

Figure 50, "Entering Commands in the Editor", shows the steps required to enter a Tcl command in the editor pane.

Figure 50: Entering Commands in the Editor



Start typing the command.

The list of commands matching your entry is displayed.

Double-click the desired command.

The command is added to the editor.

When viewing the commands, you will notice that each listed command has an icon at its left. The three icons are:

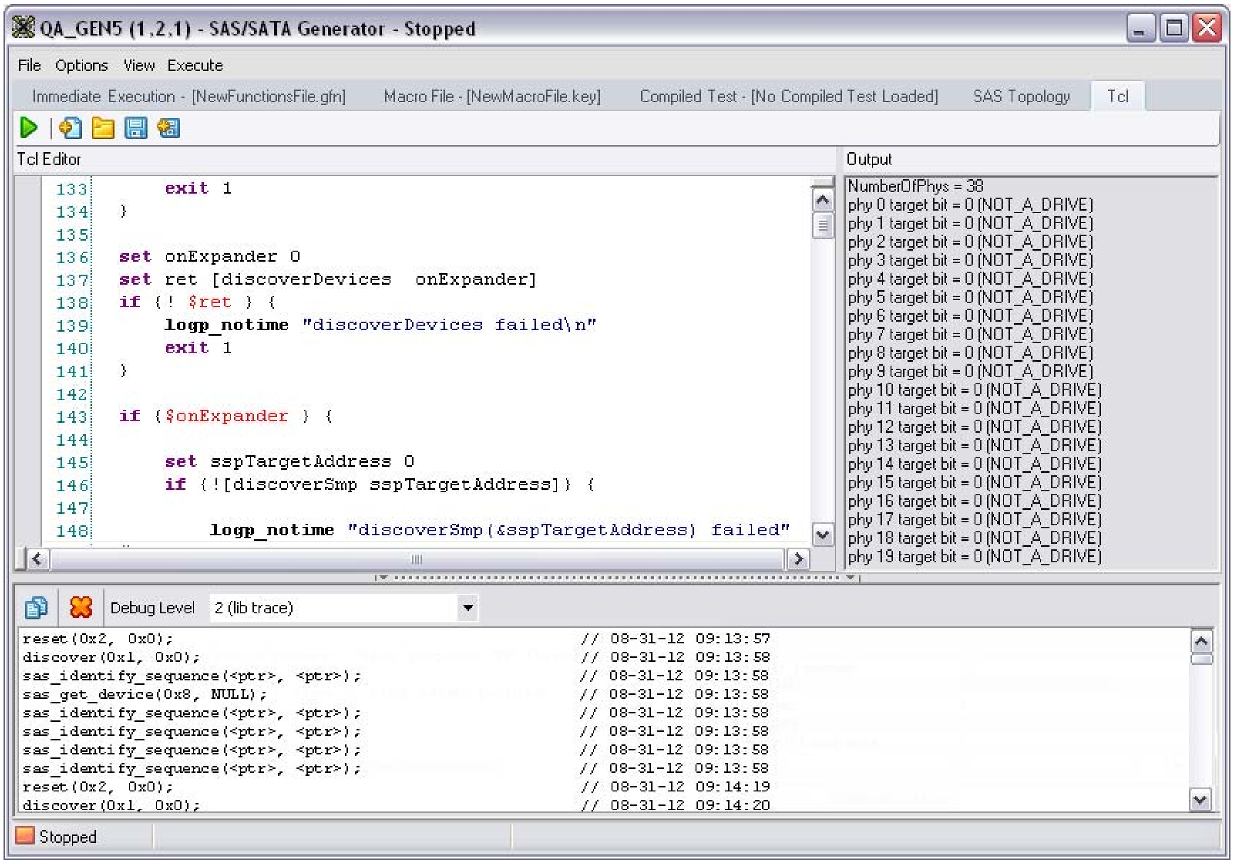
Shows that the command is a Function.

Shows that the command is a Structure.

Shows that the command is a Tcl Keyword.

A sample Tcl file is provided for Generator with the Maestro application. Figure 51, "Sample Tcl File" shows an example of what to expect when you run a Tcl file.

Figure 51: Sample Tcl File


QA_GEN5 (1.2.1) - SAS/SATA Generator - Stopped
File
Options
View
Execute
Immediate Execution - [NewFunctionsFile.gfn]
Macro File - [NewMacroFile.key]
Compiled Test - [No Compiled Test Loaded]
SAS Topology
Tcl
Tcl Editor
Output
Stopped

```
    exit 1
}

set onExpander 0
set ret [discoverDevices  onExpander]
if {! $ret } {
    logp_notime "discoverDevices failed\n"
    exit 1
}

if {$onExpander } {

    set sspTargetAddress 0
    if {![discoverSmp sspTargetAddress]} {

        logp_notime "discoverSmp(&sspTargetAddress) failed"
```

```
NumberOfPhys = 38
phy 0 target bit = 0 (NOT_A_DRIVE)
phy 1 target bit = 0 (NOT_A_DRIVE)
phy 2 target bit = 0 (NOT_A_DRIVE)
phy 3 target bit = 0 (NOT_A_DRIVE)
phy 4 target bit = 0 (NOT_A_DRIVE)
phy 5 target bit = 0 (NOT_A_DRIVE)
phy 6 target bit = 0 (NOT_A_DRIVE)
phy 7 target bit = 0 (NOT_A_DRIVE)
phy 8 target bit = 0 (NOT_A_DRIVE)
phy 9 target bit = 0 (NOT_A_DRIVE)
phy 10 target bit = 0 (NOT_A_DRIVE)
phy 11 target bit = 0 (NOT_A_DRIVE)
phy 12 target bit = 0 (NOT_A_DRIVE)
phy 13 target bit = 0 (NOT_A_DRIVE)
phy 14 target bit = 0 (NOT_A_DRIVE)
phy 15 target bit = 0 (NOT_A_DRIVE)
phy 16 target bit = 0 (NOT_A_DRIVE)
phy 17 target bit = 0 (NOT_A_DRIVE)
phy 18 target bit = 0 (NOT_A_DRIVE)
phy 19 target bit = 0 (NOT_A_DRIVE)
```

Debug Level: 2 (lib trace)

```
reset(0x2, 0x0);                                  // 08-31-12 09:13:57
discover(0x1, 0x0);                               // 08-31-12 09:13:58
sas_identify_sequence(<ptr>, <ptr>);              // 08-31-12 09:13:58
sas_get_device(0x8, NULL);                        // 08-31-12 09:13:58
sas_identify_sequence(<ptr>, <ptr>);              // 08-31-12 09:13:58
sas_identify_sequence(<ptr>, <ptr>);              // 08-31-12 09:13:58
sas_identify_sequence(<ptr>, <ptr>);              // 08-31-12 09:13:58
sas_identify_sequence(<ptr>, <ptr>);              // 08-31-12 09:13:58
reset(0x2, 0x0);                                  // 08-31-12 09:14:19
discover(0x1, 0x0);                               // 08-31-12 09:14:20
```

Tcl Script Helper Functions

In addition, the following the following Tcl script helper functions have been created.

Array Functions:

```
ui8 *new_ui8Array(int nelements)

ui8 *delete_ui8Array (ui8 *ary)

ui8 ui8Array_getitem(ui8 *ary, int index)

ui8 ui8Array_setitem(ui8 *ary, int index, ui8 value)


ui32 *new_ui32Array(int nelements)

ui32 *delete_ui32Array (ui32 *ary)

ui32 ui32Array_getitem(ui32 *ary, int index)

ui32 ui32Array_setitem(ui32 *ary, int index, ui32 value)
```

Pointer Functions:

```
ui32 *new_ui32p()

ui32 *copy_ui32p(ui32 value)

ui32 *delete_ui32p(ui32 *obj)

void ui32p_assign(ui32 *obj, ui32 value)

ui32 ui32p_value(ui32 *obj)
```

String Functions:

```
string GetTclString (ui8* s, unsigned int len)
```

Chapter 4

Using Generator in Edit Only Mode

In this chapter:

This chapter explains the process of creating and editing functions files or macro files independent of any device. These actions do not require any ports to be locked.

Note: This section refers to specific functions described in the Generator API Help System, which can be accessed by selecting Generator API Help from the main Help drop-down menu in the Maestro main window.

Creating/Editing a Functions File

A functions file is a `.gfn` file that contains a list of functions. Using this section, you can create a new functions file, edit an existing functions file, or use an existing functions file as a template to create another file. You can also change the parameters of any function. Each of the functions in the Generator GUI is represented in the API Help System on the product USB drive.

Note: This section pertains to individual functions files, which can be created and edited independently of a device. You do not need to lock ports to perform these actions. The User Functions Editor window described in this chapter is separate from the Generator Device window, which allows a user to manage and execute all configuration files for a selected device from a central window. Refer to Chapter 3, “Using Generator with Ports” for more information.

Creating a New Functions File

To create a new Xgig Generator Functions file:

1 Select the Functions Files tab in the Configuration window.

2 Click the **New Generator Functions File** icon in the Configuration menu.

A new User Functions Editor window appears.

Opening a Functions File

To open an Xgig Generator Functions file:

1 Select the Functions Files tab in the Configuration window.

The following three folders are shown:

- Sample Functions Files - Functions files provided by Viavi
- User Functions Files - Functions files created by the user and saved here
- Most Recently Used - Functions files used recently

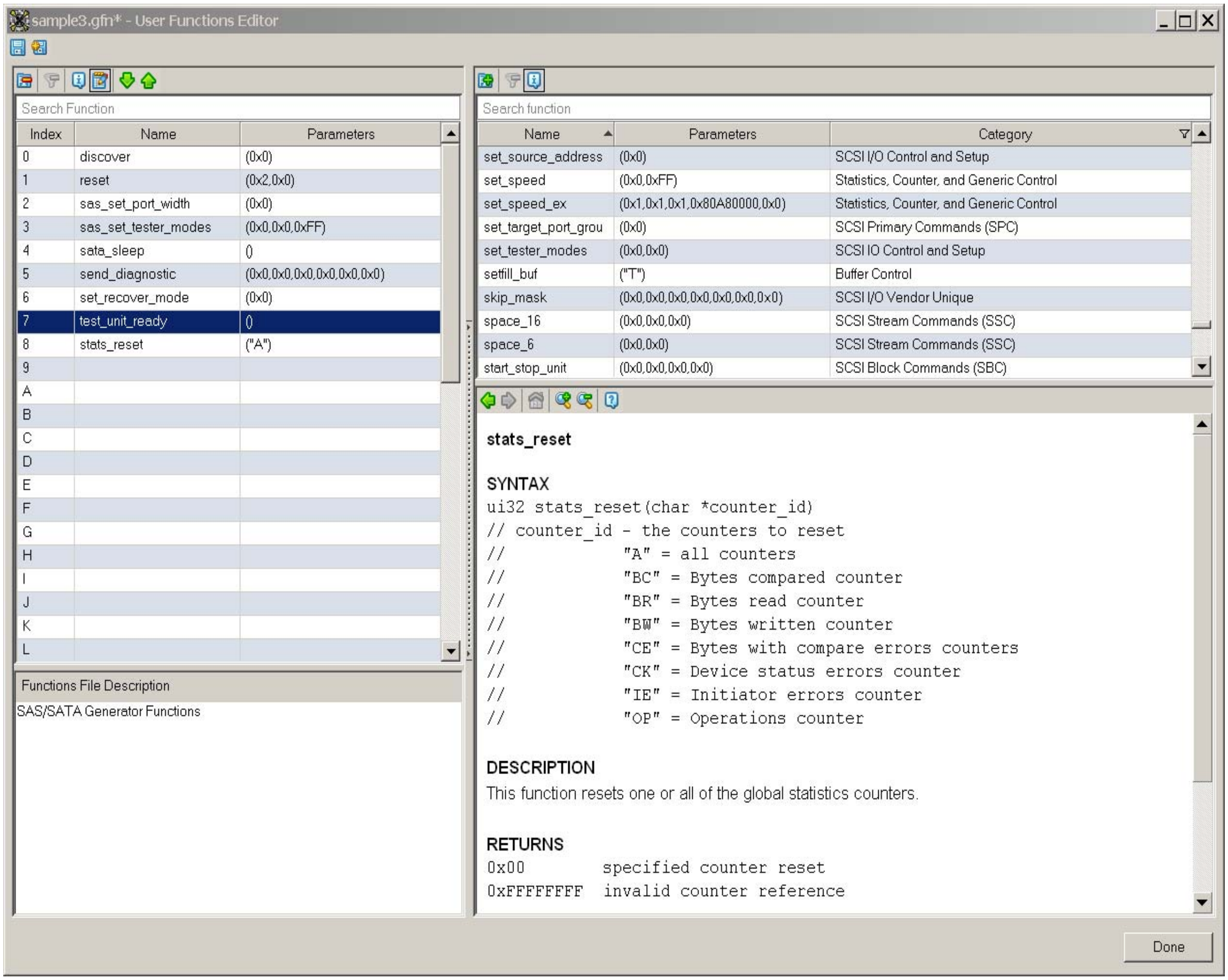
2 Double-click one of the folders to open it, or click the Path icon, and navigate to the folder that contains the functions file you want to open.

Note: To map a folder, click the **Map folder** icon. Browse to the folder you want to map, type a folder name, then click **OK**.
To unmap a folder, click the **Unmap** icon and the folder is removed.

3 Double-click the functions file you want to open.

The file is opened in the User Functions Editor window as shown in the following figure.

Figure 52: User Functions File Editor Window



sample3.gfn* - User Functions Editor
Search Function
Search function
Functions File Description
Done

Index	Name	Parameters
0	discover	(0x0)
1	reset	(0x2,0x0)
2	sas_set_port_width	(0x0)
3	sas_set_tester_modes	(0x0,0x0,0xFF)
4	sata_sleep	()
5	send_diagnostic	(0x0,0x0,0x0,0x0,0x0,0x0)
6	set_recover_mode	(0x0)
7	test_unit_ready	()
8	stats_reset	("A")
9		
A		
B		
C		
D		
E		
F		
G		
H		
I		
J		
K		
L		

SAS/SATA Generator Functions

Name	Parameters	Category
set_source_address	(0x0)	SCSI I/O Control and Setup
set_speed	(0x0,0xFF)	Statistics, Counter, and Generic Control
set_speed_ex	(0x1,0x1,0x1,0x80A80000,0x0)	Statistics, Counter, and Generic Control
set_target_port_grou	(0x0)	SCSI Primary Commands (SPC)
set_tester_modes	(0x0,0x0)	SCSI IO Control and Setup
setfill_buf	("T")	Buffer Control
skip_mask	(0x0,0x0,0x0,0x0,0x0,0x0,0x0)	SCSI I/O Vendor Unique
space_16	(0x0,0x0,0x0)	SCSI Stream Commands (SSC)
space_6	(0x0,0x0)	SCSI Stream Commands (SSC)
start_stop_unit	(0x0,0x0,0x0,0x0)	SCSI Block Commands (SBC)

stats_reset

SYNTAX

```
ui32 stats_reset(char *counter_id)
// counter_id - the counters to reset
//             "A" = all counters
//             "BC" = Bytes compared counter
//             "BR" = Bytes read counter
//             "BW" = Bytes written counter
//             "CE" = Bytes with compare errors counters
//             "CK" = Device status errors counter
//             "IE" = Initiator errors counter
//             "OP" = Operations counter
```

DESCRIPTION

This function resets one or all of the global statistics counters.

RETURNS

```
0x00        specified counter reset
0xFFFFFFFF  invalid counter reference
```

This window is comprised of two panels. The left panel is the contents pane. It lists the functions present in the functions file.

The right pane is the function library. The functions in the library are listed in the contents pane by name; current parameters in hexadecimal; and the parameter types: b (byte), d (dword), q (quad), s (string parameter), w (word), and o (optional parameter), which currently cannot be edited or added; and the function category.

You can sort the functions in either pane in the following ways:

- Alphabetically in ascending or descending order by toggling the Function Name heading.
- By category with the categories alphabetically in ascending or descending order by toggling the Function Type heading.
- By parameter types with the parameter types listed by number of parameters from most to least or from least to most by toggling the Parameter heading.

Click the **Clear Filter** icon to remove the filters you have applied by typing text into the search field and revert to viewing all functions in the functions library.

To view the details of the functions in the library or the contents pane, click the **Show/Hide Function Info** button at the top of the pane. The details for each function correspond to an entry in the Generator API Help available from the Help menu in the Maestro main window. To display the contents of the information pane for the selected function in a popup window, click the **Display Help Contents in Popup Window** button.

As you begin to click various functions in the function library or the contents pane, while you have the Show Function Info pane open, you create a history of the functions you've selected. You can navigate back and forth through the functions you've selected by using the **Go Back** and **Go Forward** buttons at the top of the Function Information pane.

Adding Functions to a Functions File

To add a function from the function library to the functions file:

1. Click the index position in the Contents pane where you want to add a function.
2. Click the function in the function library that you want to add to the functions file.
3. Click the **Send Selected Function to Functions File** button.

   The function appears in the Contents pane at the selected index position.

Note: A convenient shortcut for adding functions to a file is to hold down the Ctrl button on your keyboard, then click and drag the function file to the Contents pane.

4. Click the Save or Save as button at the top of the window to save the functions file.
5. Click the Done button to close the window.

Moving Functions within a Functions File

To move a function up or down in within the functions file:

1. Click the function you want to move.
2. Click the Move Up/Move Down buttons at the top of the Contents pane.

To remove a function you have added to your functions file:

1. Click the function you want to remove from the file.
2. Click the **Delete Selected User Function** button at the top of the Contents pane.

You can also interchange one function with another, move a function for another index location, or copy a function to another index location by right-clicking the function and selecting the appropriate action.

Editing Function Parameters

To edit a function's parameters inline:

1. Right-click the function and select Edit Parameters Inline.
2. Type directly in the parameters field to edit inline, or click the arrow at the end of the field to open a drop-down window.

   This window is the same as the Popup Parameters Editor.

To edit a function's parameters using the Popup Parameters Editor:

1. Right-click the function and select Popup Parameters Editor.
2. The Popup Parameters Editor appears.

Depending on the function, the following parameters may be available for editing:

Display - Displays the values in hexadecimal or decimal, depending on the field selected. Use the drop-down menu to change the display.

Byte - Byte- or bit-specific data. A byte is equal to 8 bits.

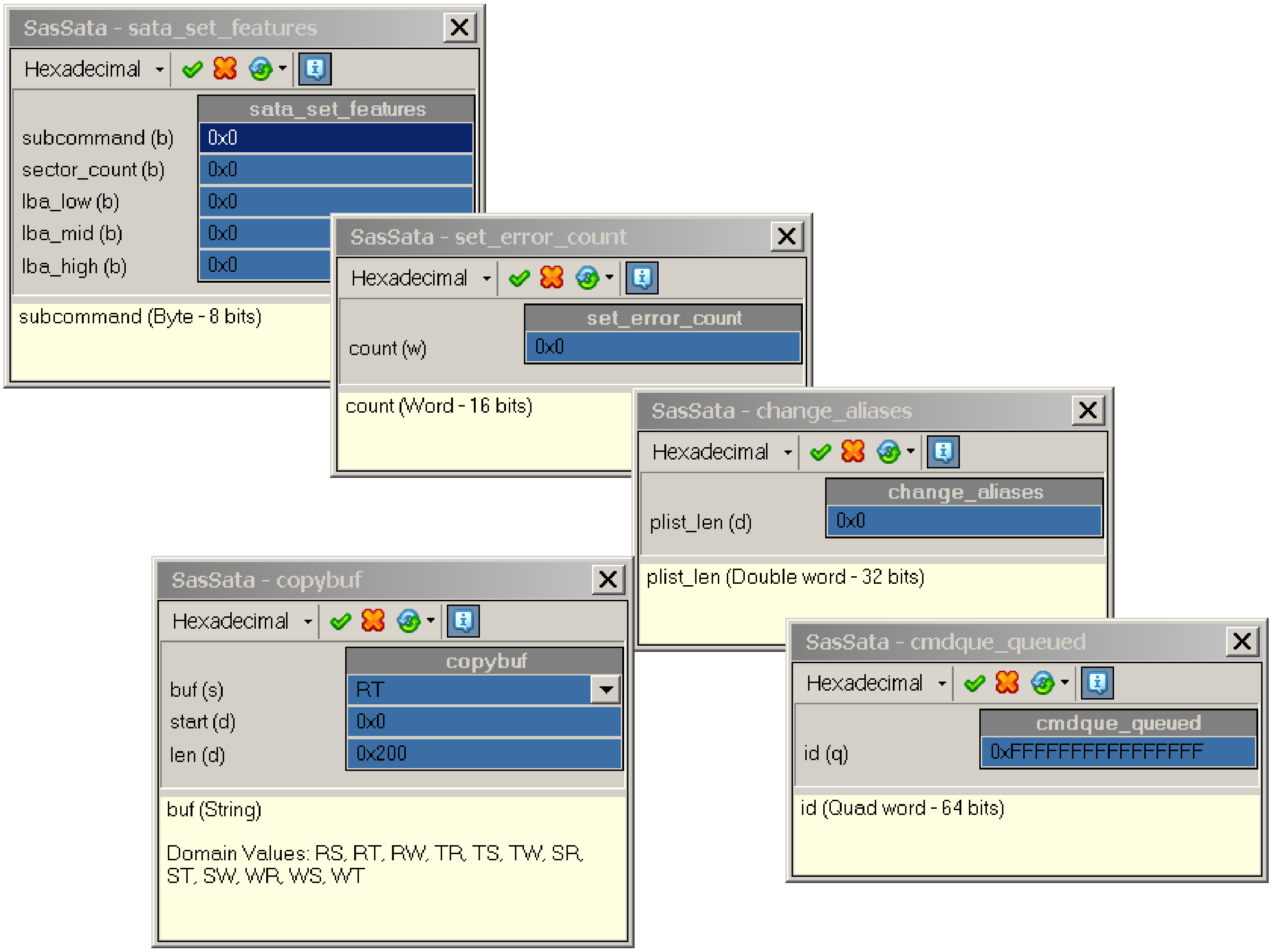
Word - Word or short integer data. A word is equal to 16 bits.

Double - Double word or long integer data. A double word is equal to 32 bits.

Quad - A quad is equal to 64 bits or 4 words.

String - Constant text or string data. Enter data without quotation marks.

Figure 53: Popup Parameters Editor Window



subcommand (b) 0x0
sector_count (b) 0x0
lba_low (b) 0x0
lba_mid (b) 0x0
lba_high (b) 0x0

subcommand (Byte - 8 bits)

count (w) 0x0

count (Word - 16 bits)

plist_len (d) 0x0

plist_len (Double word - 32 bits)

buf (s) RT
start (d) 0x0
len (d) 0x200

buf (String)

Domain Values: RS, RT, RW, TR, TS, TW, SR, ST, SW, WR, WS, WT

id (q) 0xFFFFFFFFFFFFFFFF

id (Quad word - 64 bits)

Functions File Description Editor

To add a description to your functions file:

1 Click the Show/Hide Functions File Descriptions Editor button at the top of the contents pane.

2 Type a description in the Functions File Description pane.

Saving a Functions File

To save a functions file, click the **Save** button.

To save the file as another name or to another location:

1 Click the **Save As** button.

The **Save**/**Save As** dialog appears.

Note: If you want to use the standard Windows file dialogs instead of the dialogs customized for Generator, select **Use Standard Windows File Dialogs** from the **Options** menu in the Generator Device window menu bar.

2 Navigate to the location where you want to save the file.

3 Type a name for the file.

4 Click **Save.**

5 Click the **Done** button to close the window. If you make changes to the file and then click try to close the window, you will receive prompt asking you if you want to save the file before closing it.

Creating/Editing a Macro File

The macro editor enables you to easily create macros for automating a test sequence. Macros are ideal for non-programmers because they are quick and easy to create and don't require a C compiler or knowledge of C programming. The macro language, however, has limitations. You can't write a macro test that takes pointers or addresses as function parameters.

A macro file is a `.key` file that can be created with any ASCII editor or by using the Macro File Editor. A macro file can contain up to 62 user-defined macro tests (0–9, A–Z, and a–z). Each macro test contains a sequence of functions to be called upon when invoking the macro test. In addition to functions, macro tests can also include variables, branches, jumps, loops, and macro functions.

➡ **Note:** This section pertains to individual macro files, which can be created and edited independently of a device. You do not need to lock ports to perform these actions. The Macro Editor window described in this chapter is separate from the Generator Device window, which allows a user to manage and execute all configuration files for a selected device from a central window. Refer to Chapter 3, "Using Generator with Ports" for more information.

Creating a New Macro File

To create a new Xgig Generator Macro file:

1 Select the Macro Files tab in the Configuration window.

2 Click the **New Generator Macro File** icon in the Configuration menu.

A new Macro Editor window appears.

Opening a Macro File

To open an Xgig Generator Macro file:

1 Select the Macro Files tab in the Configuration window.

The following three folders are shown:

- Sample Macro Files - Macro files provided by Viavi
- User Macro Files - Macro files created by the user and saved here
- Most Recently Used - Macro files used recently

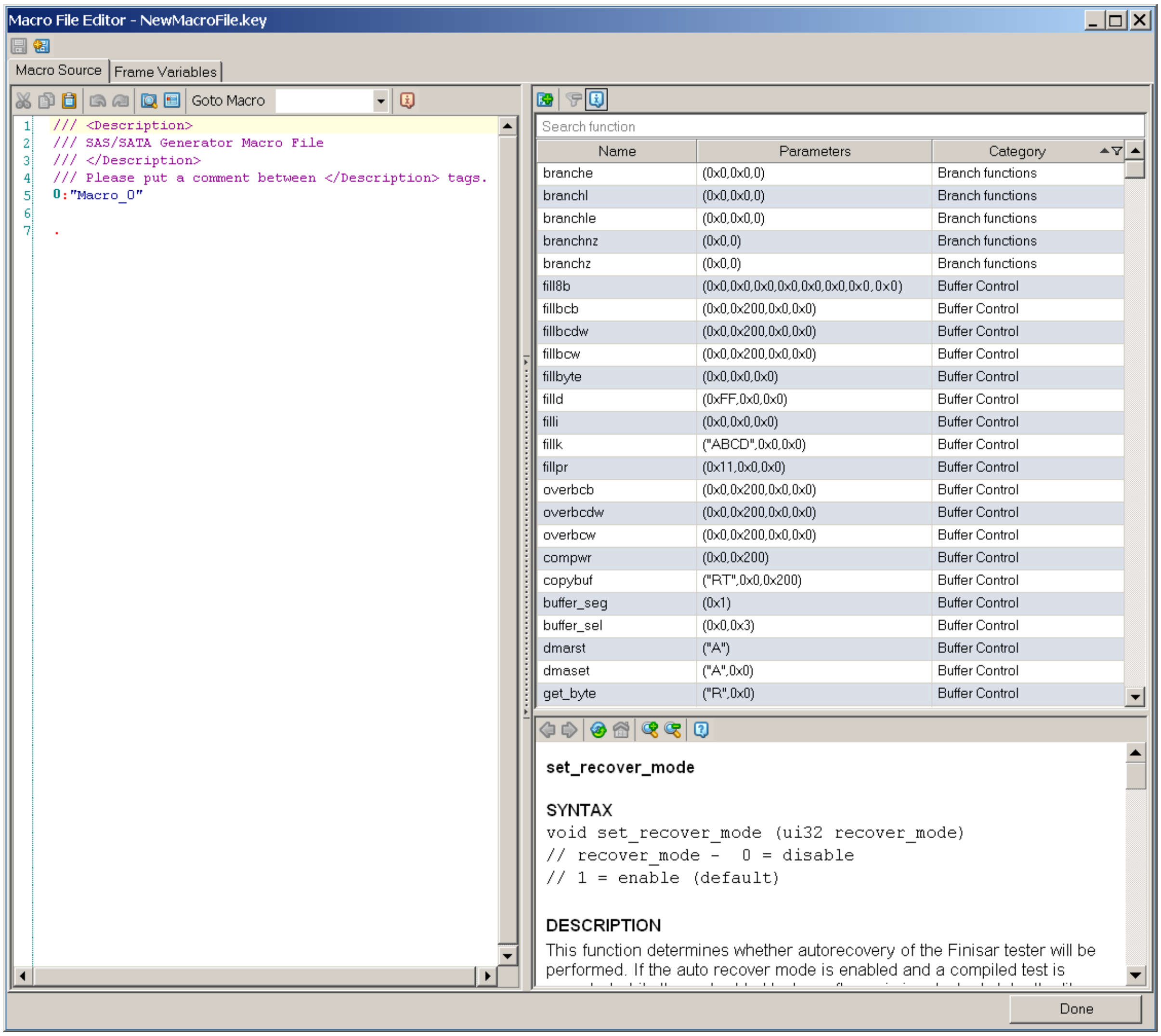
2 Double click one of the folders to open it, or click the Path icon, and navigate to the folder that contains the macro file you want to open.

➡ **Note:** To map a folder, click the **Map folder** icon. Browse to the folder you want to map, type a folder name, then click **OK**.
To unmap a folder, click the **Unmap** icon and the folder is removed.

3 Double-click the macro file you want open.

The file is opened in the Macro File Editor window as shown in the following figure.

Figure 54: Macro File Editor Window - Blank



```
/// <Description>
/// SAS/SATA Generator Macro File
/// </Description>
/// Please put a comment between </Description> tags.
0:"Macro_0"

.
```

Name	Parameters	Category
branche	(0x0,0x0,0)	Branch functions
branchl	(0x0,0x0,0)	Branch functions
branchle	(0x0,0x0,0)	Branch functions
branchnz	(0x0,0)	Branch functions
branchz	(0x0,0)	Branch functions
fill8b	(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0)	Buffer Control
fillbcb	(0x0,0x200,0x0,0x0)	Buffer Control
fillbcdw	(0x0,0x200,0x0,0x0)	Buffer Control
fillbcw	(0x0,0x200,0x0,0x0)	Buffer Control
fillbyte	(0x0,0x0,0x0)	Buffer Control
filld	(0xFF,0x0,0x0)	Buffer Control
filli	(0x0,0x0,0x0)	Buffer Control
fillk	("ABCD",0x0,0x0)	Buffer Control
fillpr	(0x11,0x0,0x0)	Buffer Control
overbcb	(0x0,0x200,0x0,0x0)	Buffer Control
overbcdw	(0x0,0x200,0x0,0x0)	Buffer Control
overbcw	(0x0,0x200,0x0,0x0)	Buffer Control
compwr	(0x0,0x200)	Buffer Control
copybuf	("RT",0x0,0x200)	Buffer Control
buffer_seg	(0x1)	Buffer Control
buffer_sel	(0x0,0x3)	Buffer Control
dmarst	("A")	Buffer Control
dmaset	("A",0x0)	Buffer Control
get_byte	("R",0x0)	Buffer Control

set_recover_mode

SYNTAX

```
void set_recover_mode (ui32 recover_mode)
// recover_mode -  0 = disable
// 1 = enable (default)
```

DESCRIPTION

This function determines whether autorecovery of the Finisar tester will be performed. If the auto recover mode is enabled and a compiled test is

Macro File Structure

When creating macro tests, you must adhere strictly to the correct syntax. Extraneous spaces are generally not allowed. Numeric variables are decimal values unless they are preceded by “0x”, in which case, they are hexadecimal.

In a new macro file, the macro content begins with description tags. This is a placeholder for the description of your file. The description you type here will appear in the Details pane when the file is selected in the Macro Files tab in Generator.

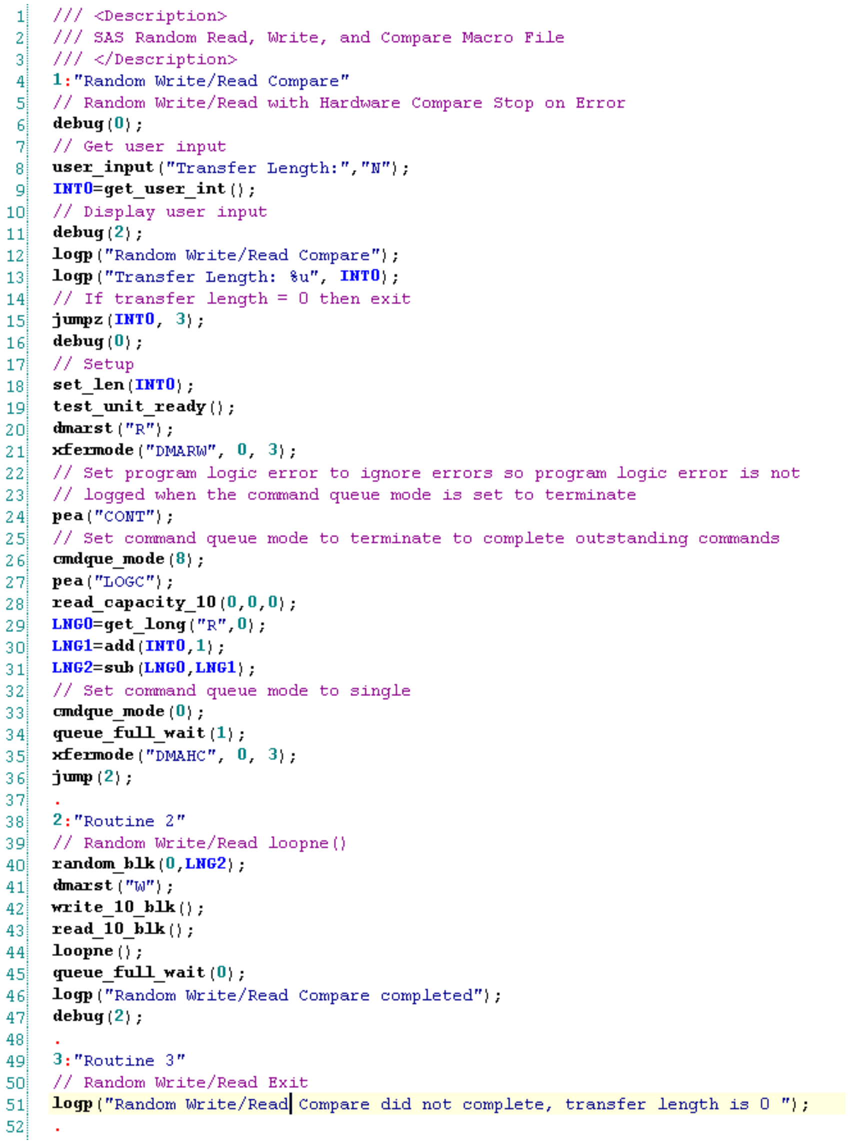
The list of functions in a macro always starts with a number (default = 0) followed by a red colon and ends with a red period. When you open a new macro, the beginning of the macro is indicated by a “`0:`”, and the end is indicated by a “`.`” on a new line as shown in the figure above.

You can either choose to build your macro as one string of functions, or you can separate it into routines. Essentially, each routine is a complete macro, beginning and ending with the symbols listed above. Each routine should be numbered sequentially. As mentioned above, the default number for the beginning of a macro is zero. You can change this number to one or another number if you choose.

Each macro or routine needs a name. The name should be enclosed in quotation marks and be placed immediately after the red colon indicating the beginning of the macro (no space).

Begin populating your macro with functions. Place a semicolon after each function. Place comments after // on a new line. A single comment that extends over multiple lines in the macro file must have a comment indicator // at the beginning of each line. The following illustration shows a completed macro containing three routines.

Figure 55: Sample Macro File Contents



```
/// <Description>
/// SAS Random Read, Write, and Compare Macro File
/// </Description>
1:"Random Write/Read Compare"
// Random Write/Read with Hardware Compare Stop on Error
debug(0);
// Get user input
user_input("Transfer Length:","N");
INT0=get_user_int();
// Display user input
debug(2);
logp("Random Write/Read Compare");
logp("Transfer Length: %u", INT0);
// If transfer length = 0 then exit
jumpz(INT0, 3);
debug(0);
// Setup
set_len(INT0);
test_unit_ready();
dmarst("R");
xfermode("DMARW", 0, 3);
// Set program logic error to ignore errors so program logic error is not
// logged when the command queue mode is set to terminate
pea("CONT");
// Set command queue mode to terminate to complete outstanding commands
cmdque_mode(8);
pea("LOGC");
read_capacity_10(0,0,0);
LNG0=get_long("R",0);
LNG1=add(INT0,1);
LNG2=sub(LNG0,LNG1);
// Set command queue mode to single
cmdque_mode(0);
queue_full_wait(1);
xfermode("DMAHC", 0, 3);
jump(2);
.
2:"Routine 2"
// Random Write/Read loopne()
random_blk(0,LNG2);
dmarst("W");
write_10_blk();
read_10_blk();
loopne();
queue_full_wait(0);
logp("Random Write/Read Compare completed");
debug(2);
.
3:"Routine 3"
// Random Write/Read Exit
logp("Random Write/Read Compare did not complete, transfer length is 0 ");
.
```

Adding Functions to a Macro File

To add a function from the function library to a macro file:

1 Place the cursor at the location where you want to add the function.

2 Click the function in the function library that you want to add to the macro file.

3 Click the Add Function to Macro button.

The function appears in the Contents pane at the selected position.

Note: A convenient shortcut for adding functions to a macro file is to hold down the **Ctrl** button on your keyboard, then click and drag the function file to the Contents pane.

4 Click the Save or Save as button at the top of the window to save the macro file.

5 Set the parameters as necessary for each function. The right-click menu provides editing options. These options are described in Editing Function Parameters above.

6 Refer to Appendix A, "Macro Variables and Functions and the Adding Frame Variables to a Macro section below for function and variable types and syntax as well as examples.

7 Indicate the end of the macro with a period on a line by itself.

8 Follow steps 1-3 to add the desired macros to the macro file, sequentially numbering the start of each one.

9 Click the Save or Save as button to save the macro file.

10 Click the Done button to close the window.

Adding Frame Variables to a Macro

The frame editor is an easy-to-use feature for customizing both legal and illegal SAS and SATA frames to send to a device. You can create and edit frame variables in the Macro File Editor window.

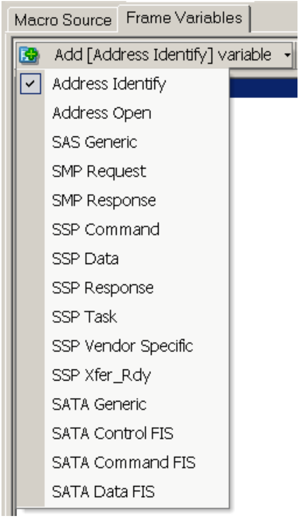
To access the Frame Editor, open the macro file that you want to add frame variables to and then select the Frame Variables tab.

Note: You do not need to have locked any ports to add frame variables to a macro file.

To add a frame to a macro:

1 Select a frame variable, i.e. FRM0, FRM1, etc.

2 Click the **Add [...] variable** drop-down menu, and select the frame type that you want to use.



Generator will fill in the fields in the right pane depending on which frame type you selected.

3 Type a name for the frame variable in the **Name** field.

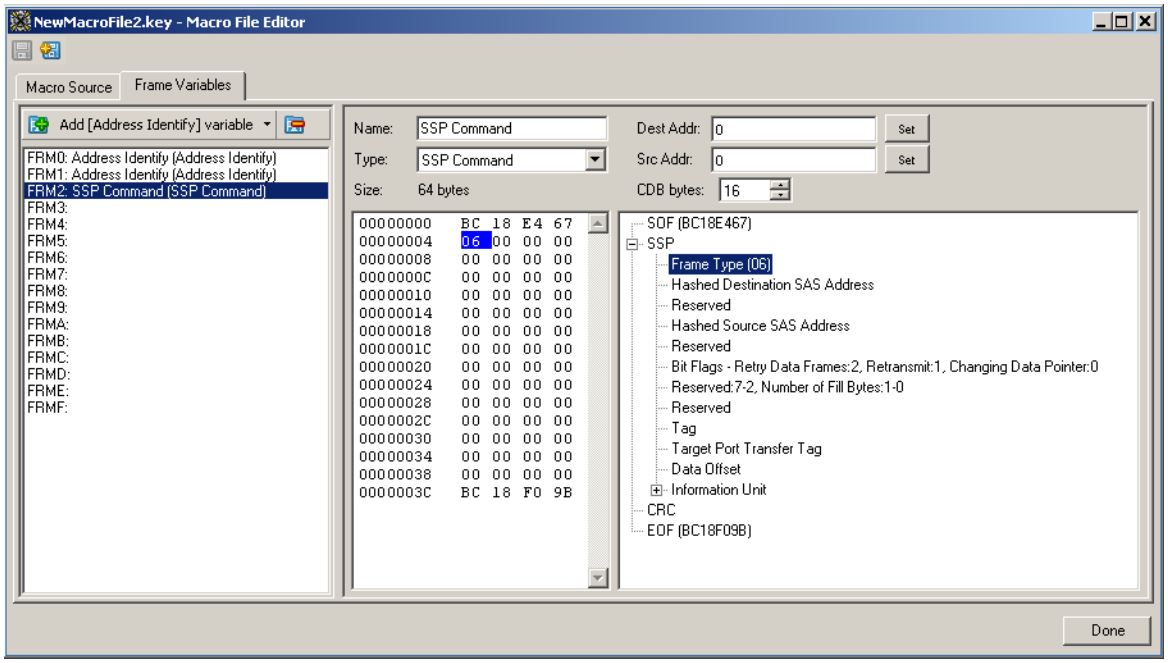
4 Type a destination address for a SAS variable in the **Dest Addr** field. This address indicates where the frame is being sent. If the frame type is not Identify or Open Address, this is the value that will set the hashed address for the frame and will be used in the automatically generate open address frame that will precede the specified frame. Once you've set the address, click **Set**.

5 Type a destination address for a SAS variable in the **Src Addr** field. This address indicates where the frame is coming from. If the frame type is not Identify or Open Address, this is the value that will set the hashed address for the frame and will be used in the automatically generate open address frame that will precede the specified frame. Once you've set the address, click **Set**.

Note: SATA variables use a logical block address (LBA) setting instead of source and destination addresses.

6 To set the LBA for a SATA variable, type an LBA address for the variable in the **LBA** field. This address indicates where the frame is coming from. If the frame type is not Identify or Open Address, this is the value that will set the hashed address for the frame and will be used in the automatically generate open address frame that will precede the specified frame. Once you have set the address, click **Set**.

The contents of the frame are listed in the following two panes as shown below.


NewMacroFile2.key - Macro File Editor
Macro Source
Frame Variables
Add [Address Identify] variable
Set
Set
Done

FRM0: Address Identify (Address Identify)
FRM1: Address Identify (Address Identify)
FRM2: SSP Command (SSP Command)
FRM3:
FRM4:
FRM5:
FRM6:
FRM7:
FRM8:
FRM9:
FRMA:
FRMB:
FRMC:
FRMD:
FRME:
FRMF:

Name: SSP Command
Type: SSP Command
Size: 64 bytes
Dest Addr: 0
Src Addr: 0
CDB bytes: 16

```
00000000   BC 18 E4 67
00000004   06 00 00 00
00000008   00 00 00 00
0000000C   00 00 00 00
00000010   00 00 00 00
00000014   00 00 00 00
00000018   00 00 00 00
0000001C   00 00 00 00
00000020   00 00 00 00
00000024   00 00 00 00
00000028   00 00 00 00
0000002C   00 00 00 00
00000030   00 00 00 00
00000034   00 00 00 00
00000038   00 00 00 00
0000003C   BC 18 F0 9B
```

```
SOF (BC18E467)
SSP
    Frame Type (06)
    Hashed Destination SAS Address
    Reserved
    Hashed Source SAS Address
    Reserved
    Bit Flags - Retry Data Frames:2, Retransmit:1, Changing Data Pointer:0
    Reserved:7-2, Number of Fill Bytes:1-0
    Reserved
    Tag
    Target Port Transfer Tag
    Data Offset
    Information Unit
CRC
EOF (BC18F09B)
```

Frame Decode Pane

The left pane lists the contents of the specific frame. This information pane tracks with the frame hex data pane. When you click on a node in the frame, the frame hex data size pane shows the number of dwords/bytes that make up that node. The components of a frame are:

- SOF
- Header (Expand this to show nodes)
- CRC
- EOF

Just above this pane are two fields. The first indicates the total size of the frame. The second, which allows you to add or remove certain types of bytes, such as FIS or CDB bytes from a frame, is only available for certain frame variable types. To add bytes to or remove bytes from the frame, click the up/down arrows next to the field.

Frame Hex Data Pane

The right pane lists the bytes for a specific frame component.

— To edit the byte value, select the byte value and then type another value.

You can add bytes to specific types of frames to increase the size of the frame. If the selected frame type allows the addition of bytes, a numeric field will be displayed above the **Frame Hex Data** pane.

Save the macro with your changes by clicking the **Save** icon in the upper, left corner of the macro window.

Saving a Macro File

To save a macro file, click the **Save** button.

To save the file as another name or to another location:

1. Click the **Save As** button.

   The **Save**/**Save As** dialog appears.

Note: If you want to use the standard Windows file dialogs instead of the dialogs customized for Generator, select **Use Standard Windows File Dialogs** from the **Options** menu in the Generator Device window menu bar.

2. Navigate to the location where you want to save the file.
3. Type a name for the file.
4. Click **Save.**
5. Click the **Done** button to close the window. If you make changes to the file and then try to close the window, you will receive prompt asking you if you want to save the file before closing it.

Editing Parameters for a Compiled Test

A compiled test file is an executable (`.exe`) file written using the SAS/SATA Generator API. Refer to the Generator API Help for instructions on creating compiled tests. The Generator GUI does not allow you to modify the `.exe` file. However, the GUI does allow you to add parameters to a complied test, which, when saved, creates a `.gen` file of the same name.

Using this section, you can add and edit parameters for a compiled test file, essentially, creating `.gen` files.

Note: This section pertains to individual compiled test files, whose parameters can be created and edited independently of a device. You do not need to lock ports to perform these actions. The Compiled Test Editor window described in this chapter is separate from the Generator Device window, which allows a user to manage and execute all configuration files for a selected device from a central window. Refer to Chapter 3, "Using Generator with Ports" for more information.

Opening a Compiled Test File

To open an Xgig Generator Compiled Test file:

1 Select the Compiled Tests tab in the Configuration window.

The following three folders are shown:

- Sample Compiled Tests - Compiled test files provided by Viavi
- User Compiled Tests - Compiled test files created by the user and saved here
- Most Recently Used - Compiled test files used recently

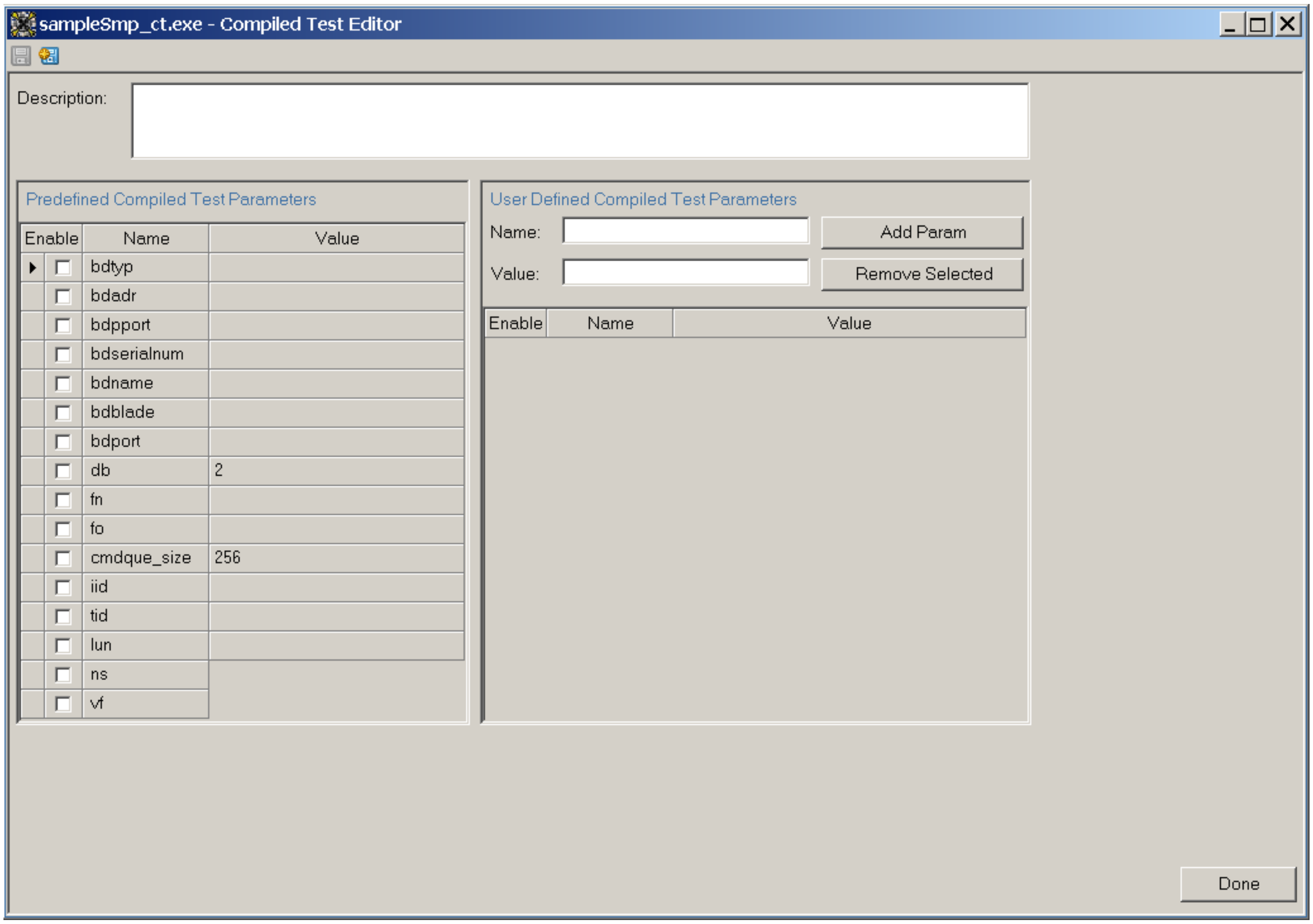
2 Double-click one of the folders to open it, or click the Path icon, and navigate to the folder that contains the compiled test file you want to open.

Note: To map a folder, click the **Map folder** icon. Browse to the folder you want to map, type a folder name, then click **OK**. To unmap a folder, click the **Unmap** icon, and the folder is removed.

3 Double-click the compiled test file you want to open.

The file is opened in the Compiled Test Editor window as shown in the following figure.

Figure 56: Compiled Test Editor Window


sampleSmp_ct.exe - Compiled Test Editor
Description:
User Defined Compiled Test Parameters
Name:
Value:
Add Param
Remove Selected
Done

Predefined Compiled Test Parameters

Enable	Name	Value
□	bdtyp	
□	bdadr	
□	bdpport	
□	bdserialnum	
□	bdname	
□	bdblade	
□	bdport	
□	db	2
□	fn	
□	fo	
□	cmdque_size	256
□	iid	
□	tid	
□	lun	
□	ns	
□	vf	

Enable	Name	Value

Adding Parameters to a Compiled Test File

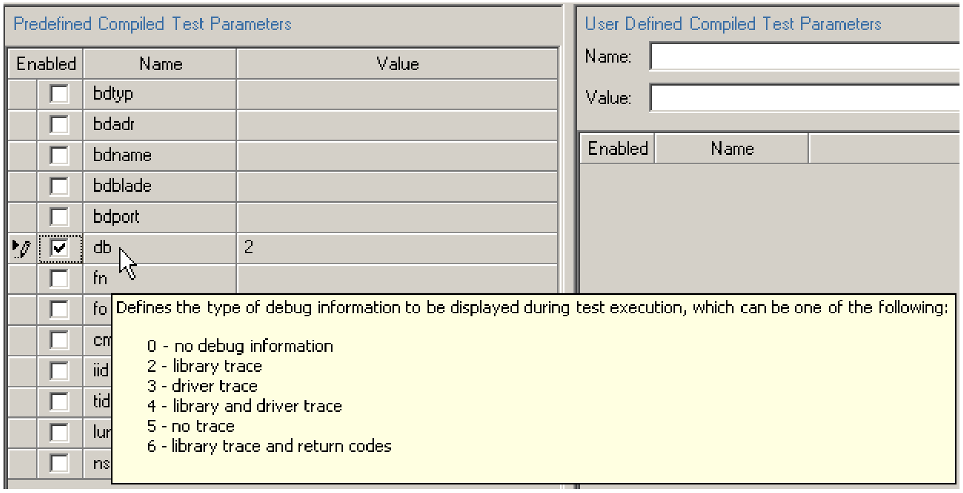
The **Description** field allows you to type a description for the compiled test. This description will appear in information panes when the file is selected.

The **Predefined Compiled Test Parameters** pane lists common command line parameters. These parameters are listed in the **Common Command Line Parameters** section of the API Help. Holding the mouse over one of these parameters brings up a tool tip window containing information about the parameter such as the definition, use, and defined or suggested values.

Example:

Holding the mouse over the `bdtyp` parameter displays the following tool tip.

Figure 57: Bdtyp Compiled Test Parameter Tool Tip


Predefined Compiled Test Parameters
User Defined Compiled Test Parameters
Name:
Value:
Enabled
Name

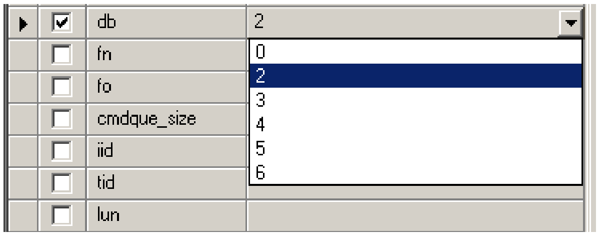
Enabled	Name	Value
□	bdtyp	
□	bdadr	
□	bdname	
□	bdblade	
□	bdport	
☑	db	2
□	fn	
□	fo	
□	cm	
□	iid	
□	tid	
□	lur	
□	ns	

Defines the type of debug information to be displayed during test execution, which can be one of the following:

0 - no debug information
2 - library trace
3 - driver trace
4 - library and driver trace
5 - no trace
6 - library trace and return codes

The tool tip describes the parameter and lists the possible values. Selecting this row and clicking the value drop-down menu, the possible values as shown in the following figure.

Figure 58: Bdtyp Compiled Test Parameter Values



	Name	Value
☑	db	2
□	fn	
□	fo	
□	cmdque_size	
□	iid	
□	tid	
□	lun	

Click the row with the desired parameter to make it editable. The ✎ icon appears next to the enabled field to indicate that this is the parameter you are editing.

To select a value for this parameter, click the value you want from the drop-down list.

➡ **Note:** Not all parameter values fields have a drop-down menu for the value. Some tool tips indicate a value range or define the type of information required and the format required.

To enable a parameter for the compiled test, select the **Enable** check box next to the parameter.

The **User Defined Compiled Test Parameters** pane allows you to create your own parameters. To create a parameter:

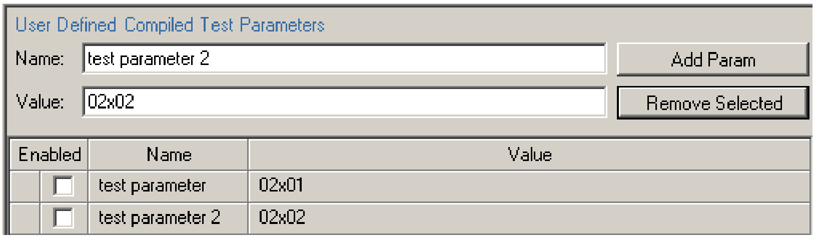
1 Type a name in the **Name** field.

2 Type a value in the **Value** field.

3 Click the **Add Parameter** button.

The parameter appears in the list below as shown in the following figure.

Figure 59: User Defined Parameter



Name: test parameter 2

Value: 02x02

Enabled	Name	Value
□	test parameter	02x01
□	test parameter 2	02x02

4 Click the row with the desired parameter to make it editable. The ✎ icon appears next to the enabled field to indicate that this is the parameter you are editing.

To enable a parameter for the compiled test, select the **Enable** check box next to the parameter.

To delete the parameter, select the entire row, and click the **Remove Selected** button.

Saving a Compiled Test File

To save a compiled test file, click the **Save** 💾 button.

To save the file as another name or to another location:

1 Click the **Save As** 💾 button.

The **Save**/**Save As** dialog appears.

Note: If you want to use the standard Windows file dialogs instead of the dialogs customized for Generator, select **Use Standard Windows File Dialogs** from the **Options** menu in the Generator Device window menu bar.

If you click the **Save as** selection to save a `.gen file` as a new file name, a copy of the existing compiled test `.exe` will be created with along with the new `.gen` file.

2 Navigate to the location where you want to save the file.

3 Type a name for the file.

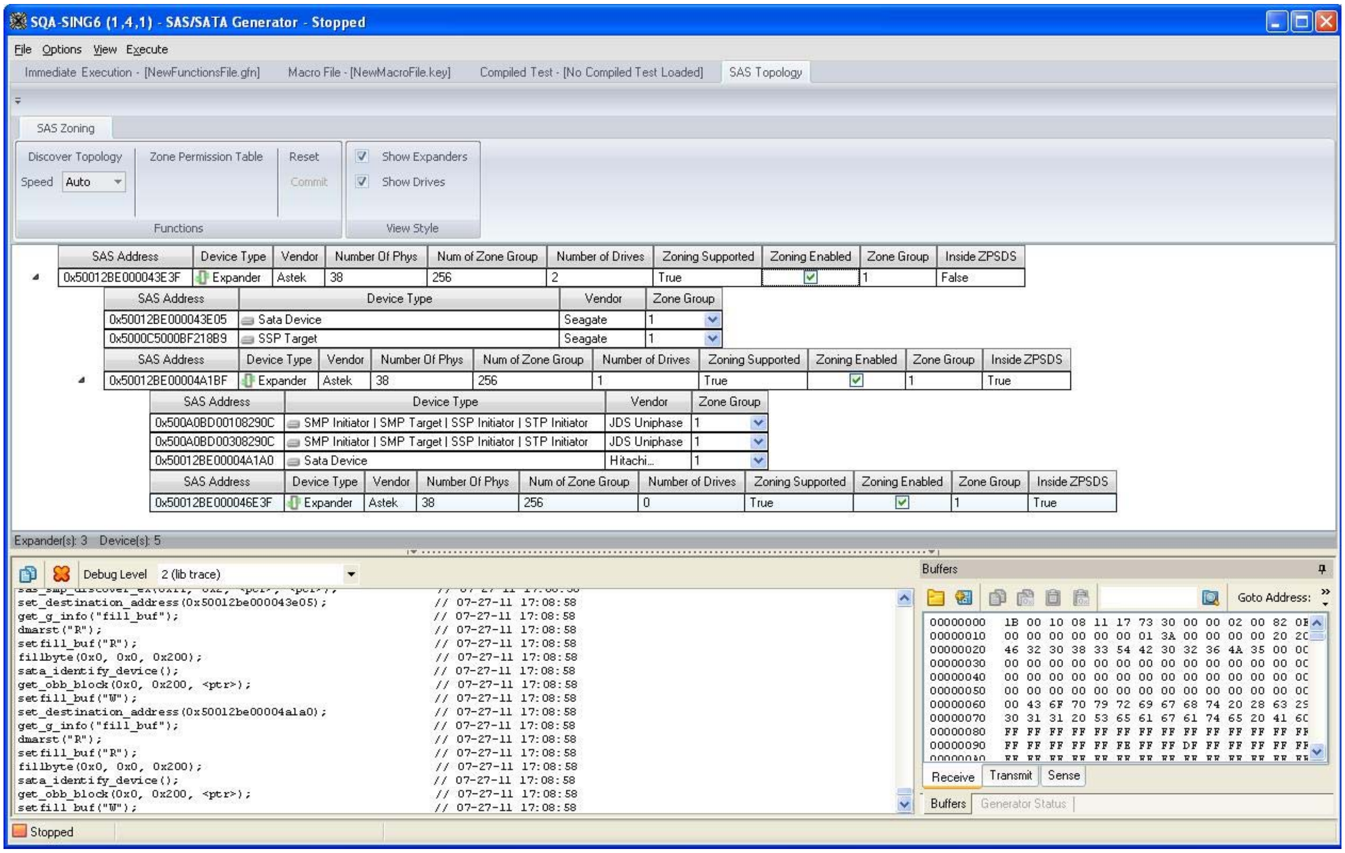
4 Click **Save.**

5 Click the **Done** button to close the window. If you make changes to the file and then try to close the window, you will receive prompt asking you if you want to save the file before closing it.

Using SAS Zoning Topology

Maestro Generator lets you display and manage the SAS Zoning topology of connected expanders and drives. It also allows you to modify the targets' zoning properties such as enable/disable zoning, set a zone group and modify the zone permission table and allows you to commit changes made on the targets' zoning properties.

Figure 60: SAS Zoning Topology Window


SQA-SING6 (1,4,1) - SAS/SATA Generator - Stopped
File Options View Execute
Immediate Execution - [NewFunctionsFile.gfn]
Macro File - [NewMacroFile.key]
Compiled Test - [No Compiled Test Loaded]
SAS Topology
SAS Zoning
Discover Topology
Zone Permission Table
Reset
Commit
Functions
View Style
Buffers
Goto Address:
Receive
Transmit
Sense
Buffers
Generator Status

Speed: Auto

☑ Show Expanders

☑ Show Drives

SAS Address	Device Type	Vendor	Number Of Phys	Num of Zone Group	Number of Drives	Zoning Supported	Zoning Enabled	Zone Group	Inside ZPSDS
0x50012BE000043E3F	Expander	Astek	38	256	2	True	☑	1	False

SAS Address	Device Type	Vendor	Zone Group
0x50012BE000043E05	Sata Device	Seagate	1
0x5000C5000BF218B9	SSP Target	Seagate	1

SAS Address	Device Type	Vendor	Number Of Phys	Num of Zone Group	Number of Drives	Zoning Supported	Zoning Enabled	Zone Group	Inside ZPSDS
0x50012BE00004A1BF	Expander	Astek	38	256	1	True	☑	1	True

SAS Address	Device Type	Vendor	Zone Group
0x500A0BD00108290C	SMP Initiator | SMP Target | SSP Initiator | STP Initiator	JDS Uniphase	1
0x500A0BD00308290C	SMP Initiator | SMP Target | SSP Initiator | STP Initiator	JDS Uniphase	1
0x50012BE00004A1A0	Sata Device	Hitachi...	1

SAS Address	Device Type	Vendor	Number Of Phys	Num of Zone Group	Number of Drives	Zoning Supported	Zoning Enabled	Zone Group	Inside ZPSDS
0x50012BE000046E3F	Expander	Astek	38	256	0	True	☑	1	True

Expander(s): 3 Device(s): 5

Debug Level: 2 (lib trace)

```
sas_smp_discover_ex(0xff, 0x2, <ptr>, <ptr>);        // 07-27-11 17:08:58
set_destination_address(0x50012be000043e05);         // 07-27-11 17:08:58
get_g_info("fill_buf");                              // 07-27-11 17:08:58
dmarst("R");                                         // 07-27-11 17:08:58
setfill_buf("R");                                    // 07-27-11 17:08:58
fillbyte(0x0, 0x0, 0x200);                           // 07-27-11 17:08:58
sata_identify_device();                              // 07-27-11 17:08:58
get_obb_block(0x0, 0x200, <ptr>);                    // 07-27-11 17:08:58
setfill_buf("W");                                    // 07-27-11 17:08:58
set_destination_address(0x50012be00004a1a0);         // 07-27-11 17:08:58
get_g_info("fill_buf");                              // 07-27-11 17:08:58
dmarst("R");                                         // 07-27-11 17:08:58
setfill_buf("R");                                    // 07-27-11 17:08:58
fillbyte(0x0, 0x0, 0x200);                           // 07-27-11 17:08:58
sata_identify_device();                              // 07-27-11 17:08:58
get_obb_block(0x0, 0x200, <ptr>);                    // 07-27-11 17:08:58
setfill_buf("W");                                    // 07-27-11 17:08:58
```

```
00000000   1B 00 10 08 11 17 73 30 00 00 02 00 82 0E
00000010   00 00 00 00 00 00 01 3A 00 00 00 00 20 20
00000020   46 32 30 38 33 54 42 30 32 36 4A 35 00 00
00000030   00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000040   00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000050   00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000060   00 43 6F 70 79 72 69 67 68 74 20 28 63 29
00000070   30 31 31 20 53 65 61 67 61 74 65 20 41 6C
00000080   FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000090   FF FF FF FF FF FE FF FF DF FF FF FF FF FF
000000A0   FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Stopped

Functional Bar

The Functional Bar provides all functions that you can access. There are 4 functions:

Discover Topology

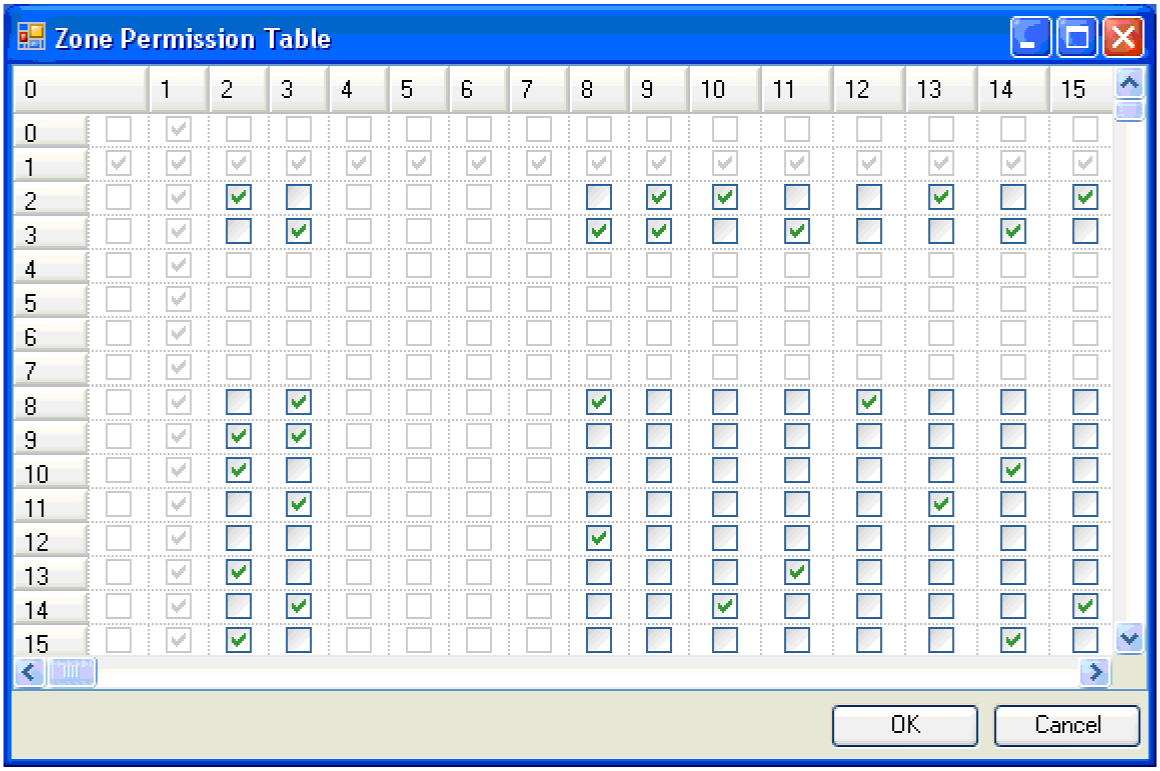
This function discovers the SAS Zoning topology used by the system. You must execute this function before other functions are accessible.

Zone Permission Table

This function opens up the **Zone Permission Table** window where you can view a grid where each column represents a source group and each row represents a destination group.

Zone Permission Table Window

Figure 61: Zone Permission Table



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
1	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
2	☐	☑	☑	☐	☐	☐	☐	☐	☐	☑	☑	☐	☐	☑	☐	☑
3	☐	☑	☐	☑	☐	☐	☐	☐	☑	☑	☐	☑	☐	☐	☑	☐
4	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
5	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
6	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
7	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
8	☐	☑	☐	☑	☐	☐	☐	☐	☑	☐	☐	☐	☑	☐	☐	☐
9	☐	☑	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
10	☐	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☐
11	☐	☑	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☐	☐
12	☐	☑	☐	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☐	☐
13	☐	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐
14	☐	☑	☐	☑	☐	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☑
15	☐	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☐

The Zone Permission Table is either a 128 by 128 or a 256 by 256 grid with check box in each cell. The NUMBER OF ZONE GROUPS field returned by REPORT GENERAL Function indicates the number of zone groups (for example, the number of entries in the zone group permission table) supported by the expander device and is defined in table 256(refer to *Specification Information technology - Serial Attached SCSI - 2 (SAS-2), Revision 16, 18 April 2009, Section 10.4.3.4 REPORT GENERAL function*). In the grid, each column represents a source group and each row represents a destination group. Because as per the specification (refer to *4.9.3.2 Zone groups*), group 0 and 1 are not configurable and group 4-7 is reserved, corresponding columns and rows should be disabled.

Reset

This function resets the group of each device to group 1, which has all access to other groups.

Commit

This function commits all changes to expanders.

Status Bar

The Status Bar displays useful and read-only information to users such as the number of expanders and number of devices.

Device View

The Device View has two modes:

Device Mode:

When you select the **Show Expanders** check box, the device view will enter **Device Mode.** In this mode, SAS Zoning topology is displayed in a tree hierarchy. This mode is editable. You can modify properties of expanders and drives, such as enable/disable zoning and zoning group. To apply any of the changes you made, click **Commit**.

Drive only Mode:

When you select the **Show Drives** check box, the device view will enter **Drive only Mode**. This mode is read-only. The main purpose of this mode is to report the state of function calls on an individual drive. Information shown in drive mode can help you locate the source of an error.

Chapter 5

Trace to Script Feature

In this chapter:

The Trace to Script Feature

The trace to script feature generates test script/program automatically from a captured trace. The generated script will send traffic according to the trace; receive traffic and compare it with the expected version from the trace.

The trace to script feature involves three (3) processes:

1. Conversion: Converting the capture trace to a script
2. Compilation: Compiling the script into a compile test
3. Playback: Running the Compiled Test and comparing the test results

The TraceToScript_Tool command-line utility is used to for the conversion process.

The trace to script feature supports SAS/SATA trace, narrow port trace only. Each port pair in the trace will have individual script and they must be generated separately.

The trace to script feature relies on Xgig Analyzer tool to open and understand Xgig Analyzer trace format. Xgig Analyzer version 4.6 or higher is needed when converting a trace to script.

Hardware and Software Requirements

The table below summarizes the requirements for the Trace to Script Feature.

Table 1: Hardware and Software Requirements

Process	Software	Hardware
Conversion	Xgig Analyzer 4.6 or higher, Xgig Maestro or Xgig Generator SDK	None
Compilation	Xgig Generator SDK	None
Playback	Xgig Generator SDK	Xgig Generator ports

Dependency on Analyzer

The trace to script feature relies on an Xgig Analyzer tool to open and understand Xgig Analyzer trace format.

Analyzer version 4.6 fixed a few critical issues related to trace file parsing. So Analyzer version 4.6 or higher is needed when converting trace to script.

This dependency only applies to the conversion process. Compilation and playback can happen without analyzer installation.

Conversion

Converting the capture trace to a script is done with the Trace to Script tool. To know more about the Trace to Script tool, refer to “TraceToScript_Tool Command-line Utility” on page 103.

Conversion Output

The conversion outputs include one .cpp file and one .h file

The .h file contains constant definitions and trace data

The .cpp file contains the script, calling a series of functions to transmit and receive traffic

Fine Tuning After Conversion

After the script is generated from the trace, you may make necessary modifications to best suit the test setup and requirements.

In the generated script, various functions are called to transmit and receive traffic. Parameters of these functions can be modified to change the behavior of the script.

If a different link speed is desired/needed, the generated `set_speed_ex()` call can be modified.

If the generated script is a function, then it must be declared and included in the main test program, before it is invoked.

After the script is generated from the trace, you may make necessary modifications to best suit the test setup and requirements

In the generated script, various functions are called to transmit and receive traffic. Parameters of these functions can be modified to change the behavior of the script

If a different link speed is desired/needed, the generated `set_speed_ex()` call can be modified

If the generated script is a function, then it must be declared the included in the main test program, before it is invoked.

After the script is generated from the trace, you may make necessary modifications to best suit the test setup and requirements.

In the generated script, various functions are called to transmit and receive traffic. Parameters of these functions can be modified to change the behavior of the script.

If a different link speed is desired/needed, the generated `set_speed_ex()` call can be modified.

If the generated script is a function, then it must be declared the included in the main test program, before it is invoked.

In the generated header file, there is a global variable named DefaultTimeout, which specifies the time-out value for receiving frames in terms of ns.

If this value is set to zero, the script will use the actual timing in the trace as time-out values for each frame.

Actual timing refers to the timestamp differences between adjacent frames.

Compilation Process

The generated script needs compilation before it can be played back.

The script calls a series of functions to transmit and receive traffic. These functions are defined and implemented in `script_playback.h` and `script_playback.cpp`, which are included in the Xgig Generator SDK. These two files must be compiled together with the generated script, along with other libraries in the SDK.

Depending on the script generation option, the script needs to be compiled differently:

For **function script**, it needs to be declared and included in the user-defined main test program to compile.

For **sample script**, it can replace the `sample.cpp` file in the SDK sample project and then it will be compiled into the sample project.

For **sample SMP script**, it can replace the `sampleSmp.cpp` file in the SDK sampleSmp project and then it will be compiled into the sampleSmp project.

Playback Process

Playing back the script is simply running the test program compiled from the script.

If all transmissions are made successfully and all received traffic match the expected version, the playback will succeed. Otherwise, the playback may fail and terminate prematurely.

The playback process tries to reproduce the scenario captured in the trace. But not all environment settings are captured by the trace, these have to be managed manually, such as, state and content of the drive.

To best reproduce the scenario, the playback setup should be kept as identical to the original setting as possible.

Completeness of the trace used sometimes affects the playback result.

TraceToScript_Tool Command-line Utility

The TraceToScript_Tool command-line utility is a tool to convert an Xgig Analyzer trace to a C/C++ script. It makes use of the `trace_to_script()` API function. This tool can be found in the SAS Xgig Generator SDK folder or `C:\Program Files\Viavi\Xgig SAS Generator SDK`.

Note: This software application can be loaded on 32-bit or 64-bit Windows operating systems. The path used above is for a 32-bit operating system. if you are using a 64-bit operating system, the path would be: `C:\Program Files (x86)\Viavi\...`

To use the TraceToScript_Tool command-line utility, run the TraceToScript_Tool.exe executable file with the required parameters.

Syntax:

```
C:\Program Files\Viavi\Xgig SAS Generator SDK\tracetoscript_tool
-input=tracePath -output=scriptPath -tracetype=traceType -tx=txPort
-rx=rxPort -start=startTime -end=endTime -option=options
```

Example:

```
C:\Program Files\Viavi\Xgig SAS Generator SDK\tracetoscript_tool
-input=C:\test.tgp -output=C:\tracetoscript.cpp -tracetype=0 -tx=11
-rx=12 -start=0 -end=0 -option=01010002
```

The parameters are categorized and further discussed below.

File Path

tracePath: The full path of the trace file to be converted (*.tgp file).

scriptPath: The full path of the generated script files.

One .cpp file and one .h file will be generated with the name given

Any extension given will be ignored

For example:

sample.cpp -> sample.cpp + sample.h

sample.h -> sample.cpp + sample.h

sample -> sample.cpp + sample.h

sample.exe -> sample.cpp + sample.h

Trace Type

traceType: The type of trace (*.tgp file) provided in the tracePath. If this parameter is not specified, the default value will be assumed.

Trace Type and Corresponding Values

TRACE TYPE	VALUE
XGIG_TRACE	0x00
HACKY_TRACE	0x01

The following section describes the above mentioned trace types.

XGIG_TRACE: This refers to the standard *.tgp file for trace captured from Xgig Analyzer. (default value)

HACKY_TRACE: This refers to the *.tgp file converted from a Bus Doctor or Itech Monarch trace file. For more information about converting Bus Doctor or Itech Monarch traces, please refer to the Xgig TraceView Help.

Ports

(For XGIG_TRACE)

txPort: The blade and port number in the trace, to which the generated script is expecting to send. The lower nibble indicates the port number and the higher nibble indicates the blade index.

rxPort: The blade and port number in the trace, from which the generated script is expecting to receive.

(For HACKY_TRACE)

txPort: The port number in the trace, to which the generated script is expecting to send.

rxPort: The port number in the trace, to which the generated script is expecting to receive.

The API searches the specified ports in the trace, if the trace doesn't have any items for the ports specified, the generated script will not have any sending/receiving activities.

Duration

startTime: The time (in ns) in the trace where the trace will be converted from.

endTime: The time (in ns) in the trace where the trace will be converted to.

If both startTime and endTime are zero, the whole trace will be converted.

The API searches the timestamp in the trace, if the trace doesn’t have any items between the duration specified, the generated script will not have any sending/receiving activities.

Options

options: A 4-byte value. Each byte controls different aspect of the trace to script feature, and can be ORed together. If any of the 4 bytes are not specified, the default value for that byte will take effect.

Table 2: Options and Corresponding Values

OPTION	VALUE
WAIT_FOR_EXACT_FRAME	0x00000000
WAIT_FOR_EOF	0x00000001
WAIT_FOR_FRAME_TYPE	0x00000002
WAIT_FOR_EXACT_FRAME_EXCEPT_DATA_PAYLOAD	0x00000003
DATA_FRAME_FULL	0x00000000
DATA_FRAME_HEADER_ONLY	0x00000100
FUNCTION_SCRIPT	0x00000000
SAMPLE_SCRIPT	0x00010000
SAMPLE_SMP_SCRIPT	0x00020000
EXACT_DEVICE_ADDRESS	0x00000000
AUTO_DEVICE_ADDRESS	0x01000000

The following sections describe the options mentioned above.

Receiving Frame Options

These options control the compare criterion between received frames and frames in the trace.

WAIT_FOR_EXACT_FRAME: The entire received frame will be compared (default value)

WAIT_FOR_EOF: Only the EOF of the received frame is compared, i.e. only makes sure received traffic is a frame

WAIT_FOR_FRAME_TYPE: Only the frame type of the received frame is compared

WAIT_FOR_EXACT_FRAME_EXCEPT_DATA_PAYLOAD: All parts, except for data payload in DATA frame, will be compared

Transmitting Frame Options

These options control transmitting frame contents for DATA frames.

DATA_FRAME_FULL: When sending DATA frames, the full content from the trace is used (default value)

DATA_FRAME_HEADER_ONLY: When sending DATA frames, the DATA frame payload are not from the trace, but are all zeros

Script Generation Options

These options control the format of the generated script.

FUNCTION_SCRIPT: Generated script will be in a function (default value)

SAMPLE_SCRIPT: Generated script will follow the format of sample.cpp file in Xgig Generator SDK sample project

SAMPLE_SMP_SCRIPT: Generated script will follow the format of the sampleSmp.cpp file in Xgig Generator SDK sample SMP project

The table below summarizes the script generation options.

Table 3: Script Generation Options

Options	**Output**	**Purpose**	**Initialization Content**	**Additional Setup**	**Automatic Link Establishment**
FUNCTION_SCRIPT	A function	To be used with customer test project, where initialization are done in common modules	Tester discovery and port locking NOT included	Tester discovery and port locking, declaration/inclusion of the function, invocation of the function	No
SAMPLE_SCRIPT	Complete cpp source, in sample.cpp format	To be used with Xgig Generator SDK sample project	Tester discovery and port locking included	Change tester name/ IP	Yes
SAMPLE_SMP_SCRIPT	Complete cpp source, in sampleSmp.cpp format	To be used with Xgig Generator SDK sampleSmp project	SMP discovery and port locking included	Change tester name/ IP	Yes

Variable Options

These options control variable settings:

EXACT_DEVICE_ADDRESS: The source and destination addresses used will follow the values in the trace (default value)

AUTO_DEVICE_ADDRESS: The source and destination addresses will be obtained in run time from the actual connected device

Additional Information

Maximum Size of Trace

There is no limit on the size of the trace to be converted. However, if the trace is large, the converted script will also be large and may fail to compile due to memory availability on the build machine

Two options affect the size of the generated script when there are data frames in the trace:

Receiving Frame Options – decides whether expected DATA frame payload need to be recorded in the generated script

Transmitting Frame Options – decides whether transmitting DATA frame payload need to be recorded in the generated script

When the actual data in the DATA frames has no significance to the test result, options can be configured to ignore transmitting and receiving DATA frame payload to reduce the size of the generated script.

AUTO_DEVICE_ADDRESS Option and SMP

The AUTO_DEVICE_ADDRESS option will enable the generated script to automatic discover the connected device and set the source and destination addresses.

This option helps to minimize the modifications needed when you use a different device from the one used in the original trace to playback.

However, in SMP setup, even though the devices connected to the expander can be discovered automatically; there is no way to know the intended one to communicate to for each frame.

Microprogramming Mode and the Implications

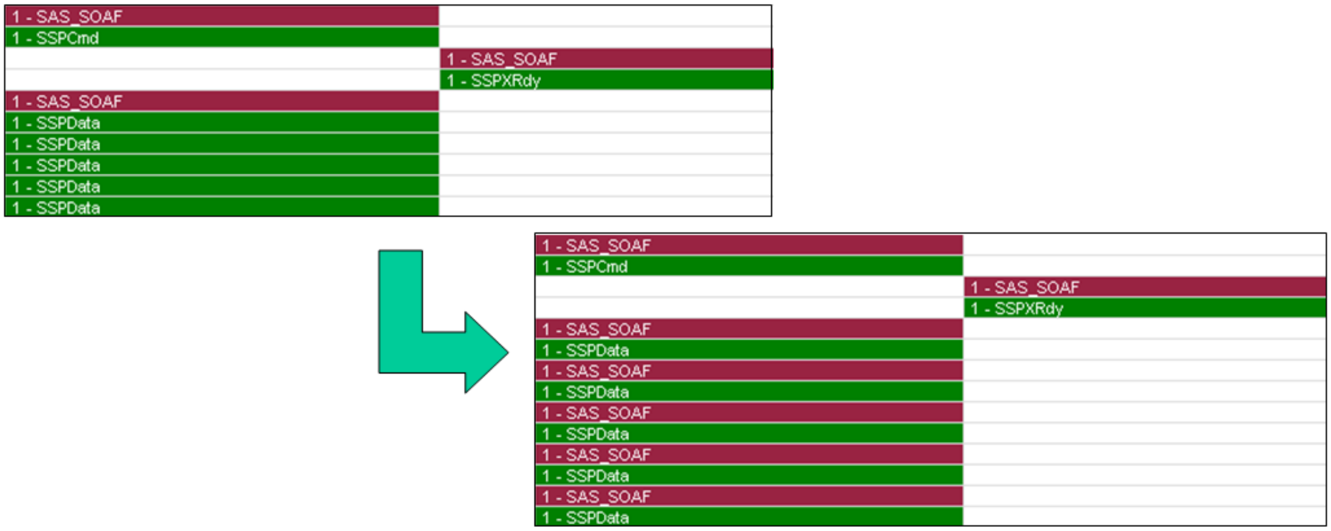
The generated script uses microprogramming mode to transmit and receive traffic which results in slower operation.

Each frame is sent separately. The SAS connection is closed after each frame is sent. Thus, a new connection is opened for every frame sent.

Result: the original trace could have multiple frames sent within one connection, but the playback trace will have new connections for each of these frames and thus, will be slower.

Refer to the following figure.

Figure 62: Resulting Trace from Multiple Frames Sent within One Connection



1 - SAS_SOAF	
1 - SSPCmd	
	1 - SAS_SOAF
	1 - SSPXRdy
1 - SAS_SOAF	
1 - SSPData	
1 - SSPData	
1 - SSPData	
1 - SSPData	
1 - SSPData	

1 - SAS_SOAF	
1 - SSPCmd	
	1 - SAS_SOAF
	1 - SSPXRdy
1 - SAS_SOAF	
1 - SSPData	
1 - SAS_SOAF	
1 - SSPData	
1 - SAS_SOAF	
1 - SSPData	
1 - SAS_SOAF	
1 - SSPData	
1 - SAS_SOAF	
1 - SSPData	

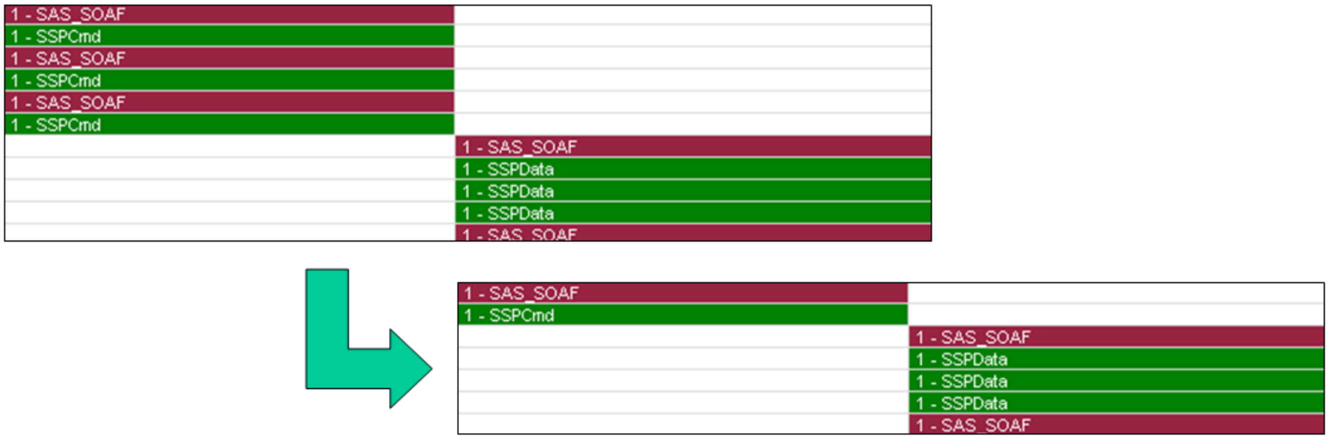
Auto/Stack Mode

In auto/stack mode or in SATA NCQ, the initiator can issue multiple commands to the target without waiting for the responses.

For the generated script, due to the frame-by-frame sending scheme and client software control mechanic, command sending speed is not fast enough compared to the target device. The script controlled initiator will hardly be fast enough to send out multiple commands before responses come back from the target device. So, achieving command stacking is difficult in playback.

Refer to the following figure:

Figure 63: Auto/Stack Mode Trace Comparison



```
1 - SAS_SOAF
1 - SSPCmd
1 - SAS_SOAF
1 - SSPCmd
1 - SAS_SOAF
1 - SSPCmd
                 1 - SAS_SOAF
                 1 - SSPData
                 1 - SSPData
                 1 - SSPData
                 1 - SAS_SOAF
```

```
1 - SAS_SOAF
1 - SSPCmd
                 1 - SAS_SOAF
                 1 - SSPData
                 1 - SSPData
                 1 - SSPData
                 1 - SAS_SOAF
```

Hardware Receiving Buffer Limit and Competition for Channel

The Xgig Generator has a hardware receiving buffer approximately equal to three SAS frames. Once the buffer is full, the software must receive the data and clear the buffer before more data can be received by the generator.

When the receiving buffer is full, the generator will credit block all incoming transmit requests and the target will keep trying to send. At the same time, if the generator is also trying to transmit, both the generator and the target will compete for the channel. In most cases, the target will win and this results in the generator being unable to transmit without receiving first.

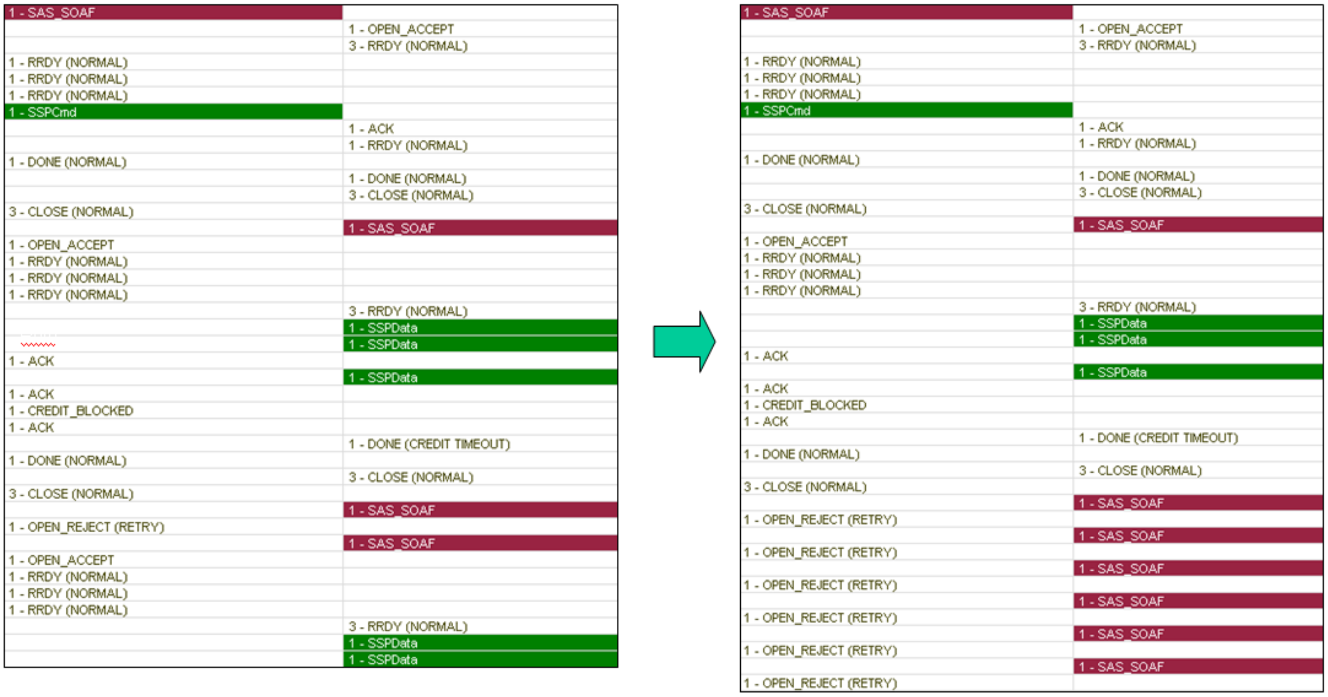
For the generated script, sending and receiving frames follows the original trace. The script has no knowledge of when the receiving buffer is full. To overcome this issue, the script will try to receive frames to clear the receiving buffer if it failed to send out frames and re-try the transmission after receiving.

This will result in the following:

- Contributes to the command stacking issue
- Delays between receiving while trying to transmit frames

Refer to the following figure:

Figure 64: Hardware Receiving Buffer Limit and Competition for Channel



1 - SAS_SOAF	
	1 - OPEN_ACCEPT
	3 - RRDY (NORMAL)
1 - RRDY (NORMAL)	
1 - RRDY (NORMAL)	
1 - RRDY (NORMAL)	
1 - SSPCmd	
	1 - ACK
	1 - RRDY (NORMAL)
1 - DONE (NORMAL)	
	1 - DONE (NORMAL)
	3 - CLOSE (NORMAL)
3 - CLOSE (NORMAL)	
	1 - SAS_SOAF
1 - OPEN_ACCEPT	
1 - RRDY (NORMAL)	
1 - RRDY (NORMAL)	
1 - RRDY (NORMAL)	
	3 - RRDY (NORMAL)
	1 - SSPData
	1 - SSPData
1 - ACK	
	1 - SSPData
1 - ACK	
1 - CREDIT_BLOCKED	
1 - ACK	
	1 - DONE (CREDIT TIMEOUT)
1 - DONE (NORMAL)	
	3 - CLOSE (NORMAL)
3 - CLOSE (NORMAL)	
	1 - SAS_SOAF
1 - OPEN_REJECT (RETRY)	
	1 - SAS_SOAF
1 - OPEN_ACCEPT	
1 - RRDY (NORMAL)	
1 - RRDY (NORMAL)	
1 - RRDY (NORMAL)	
	3 - RRDY (NORMAL)
	1 - SSPData
	1 - SSPData

1 - SAS_SOAF	
	1 - OPEN_ACCEPT
	3 - RRDY (NORMAL)
1 - RRDY (NORMAL)	
1 - RRDY (NORMAL)	
1 - RRDY (NORMAL)	
1 - SSPCmd	
	1 - ACK
	1 - RRDY (NORMAL)
1 - DONE (NORMAL)	
	1 - DONE (NORMAL)
	3 - CLOSE (NORMAL)
3 - CLOSE (NORMAL)	
	1 - SAS_SOAF
1 - OPEN_ACCEPT	
1 - RRDY (NORMAL)	
1 - RRDY (NORMAL)	
1 - RRDY (NORMAL)	
	3 - RRDY (NORMAL)
	1 - SSPData
	1 - SSPData
1 - ACK	
	1 - SSPData
1 - ACK	
1 - CREDIT_BLOCKED	
1 - ACK	
	1 - DONE (CREDIT TIMEOUT)
1 - DONE (NORMAL)	
	3 - CLOSE (NORMAL)
3 - CLOSE (NORMAL)	
	1 - SAS_SOAF
1 - OPEN_REJECT (RETRY)	
	1 - SAS_SOAF
1 - OPEN_REJECT (RETRY)	
	1 - SAS_SOAF
1 - OPEN_REJECT (RETRY)	
	1 - SAS_SOAF
1 - OPEN_REJECT (RETRY)	
	1 - SAS_SOAF
1 - OPEN_REJECT (RETRY)	
	1 - SAS_SOAF
1 - OPEN_REJECT (RETRY)	

Non-support for Truncation of DATA Frame Payload

The input trace file to the TraceToScript_Tool must be a trace that does not contain any truncated DATA frame payload. When converting Bus Doctor or Itech Monarch traces to Hacky Trace Format, it is important to note that the option for full frame conversion without truncation is enabled. For information regarding how this is done, refer to the *Xgig TraceView Help*.

PART TWO: Using Xgig Target Emulator

Chapter 6

Introducing Xgig Target Emulator

In this chapter:

Xgig Target Emulator Overview

The Xgig Target Emulator offers advanced testing and troubleshooting capabilities to storage equipment designers, testers, and manufacturers. Extending the capabilities of the Xgig platform of design and test tools, the Target Emulator connects to users' systems and supports SAS/SATA protocol-level testing at 1.5, 3.0, and 6.0 Gb/s. With full configurability, flexible responsiveness, and automated operation, the Target Emulator is the ideal tool for accelerating the bring-up, regression, and manufacturing testing of SAS/SATA equipment.

Xgig Target Emulator, used with the Xgig Analyzer software, provides a complete test solution including the ability to analyze the data generated and to capture and display SAS/SATA data in a customized trace view.

The Target Emulator builds upon the proven Xgig platform, a flexible chassis-and-blade hardware architecture supporting multiple protocols. Utilizing this architecture and the Xgig 6 Gb/s multi-function 4-link (wide) blades, 2-link (narrow) blades with dual mini-SAS/I-Pass connectors(1), and LXP all-in-one, the Target Emulator allows bidirectional SAS/SATA target emulation up to level-4 functionality (such as SCSI commands) across up to 4 ports per blade or 8 ports per chassis.

New Features

No new features have been released in this version of Xgig Target Emulator.

Xgig Target Emulator Features

Xgig Target Emulator is part of the Xgig family of instruments. Target Emulator has the following features:

- Part of the best-of-breed Xgig platform of design and test tools
- Supports bidirectional SAS/SATA target emulation up to level-4 functionality
- Operates at 1.5, 3.0, and 6.0 Gb/s
- Support for up to 8 simultaneous users leverages test tool investment across an entire development team
- Blade supports 4 SAS/SATA Target Emulators per blade and up to 8 Target Emulators per chassis
- Responds to SATA Gen-3 commands as defined in the Serial ATA Revision 3.0 specification, in addition to all of its current SAS-2 commands and operates within SATA Revision 3.0 (Gold Revision) protocol timeout specifications.
- Provides the ability to inject SATA errors such as the following:
    - CRC manipulations: allow CRC error insertion
    - Line Errors, such as, disparity errors: create disparity errors
    - FIS Manipulations: customize FIS content of the specified FIS types
    - Misplaced Primitives: send SATA primitives upon receipt and/or transmit of specified FIS types
    - EOF error: remove from frame of the specified FIS types
    - Delay injection: set delay in transmitting specified FIS types
    - Delay Drive Readiness: set the time delay the Target Emulator takes to respond after link rest
- Provides the ability to place the DUT into BIST mode (as specified in the SATA Rev.3.0 Gold specification) and specify the various options for this mode. However, the Target Emulator will not support BIST mode.
- For SATA Target Emulator, Packet commands are not supported in this release.

SAS/SATA Target Application Programming Interface (API)

The Target Emulator software allows the user to write and compile tests using C/C++. This functionality requires that the user be familiar with the API function library and the process of writing a compiled test. To this end, the Maestro software includes a Target Emulator API Help system, available in the Help menu of the Maestro main window. Using this API Help system along with this manual will help you understand and control the functionality of the Target Emulator software.

Chapter 7

Using the Target Emulator Tab

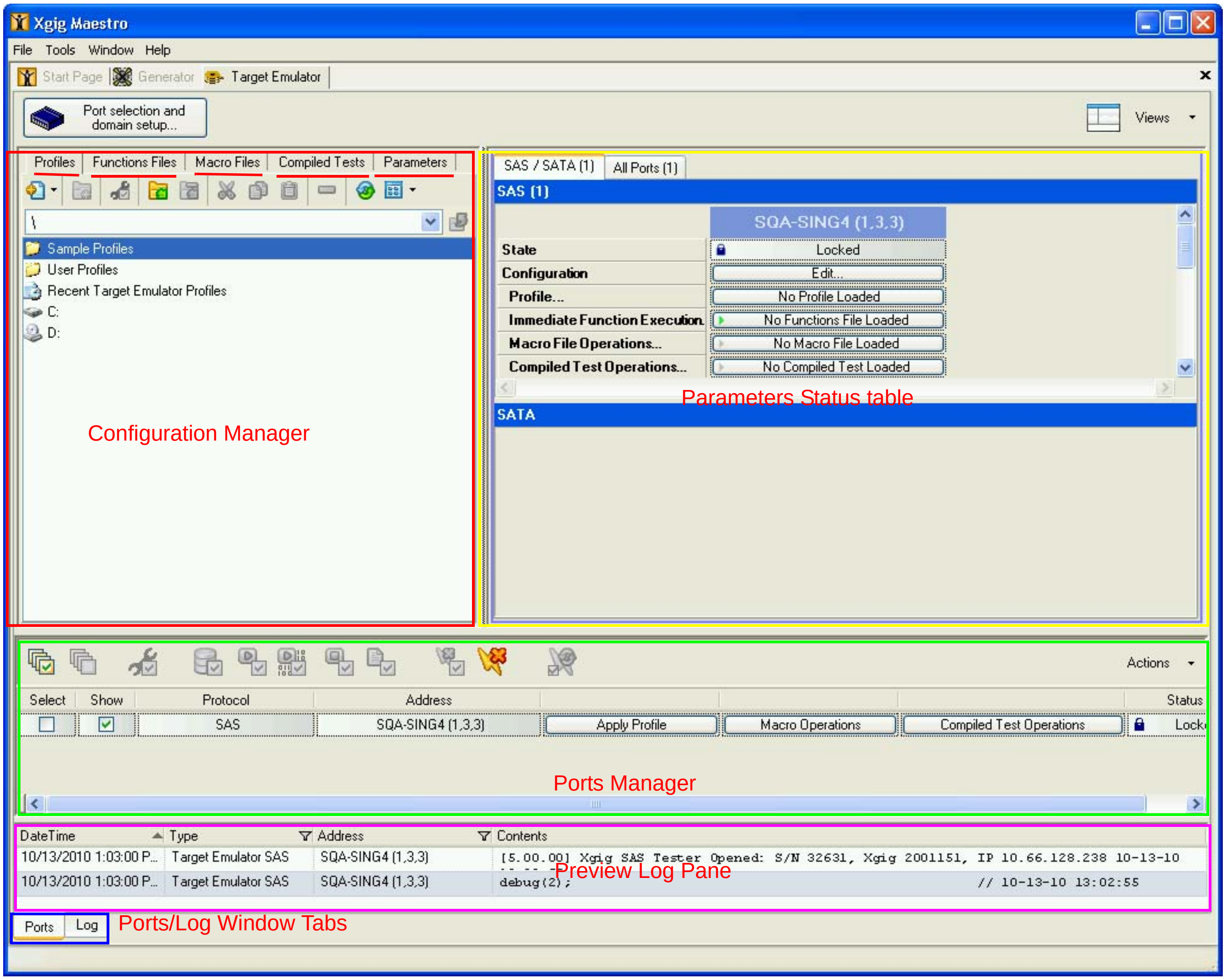
In this chapter:

This chapter provides an overview of the Target Emulator tab, on the Xgig Maestro main window, and its functions. You should have launched Xgig Maestro and locked at least one device as described in the *Maestro Introduction Guide*. The Xgig Maestro window is displayed with the Target Emulator tab on the right as shown in the following figure. This tab is where you operate the Target Emulators you have locked.

Note: This section refers to specific functions described in the Generator API Help System, which can be accessed by selecting Generator API Help from the main Help drop-down menu in the Maestro main window.

The main menu, shown below, is divided into panes similar to the Windows Explorer structure for ease of navigation and viewing. Right-click menus enable you to perform the desired task within a specific pane

Figure 65: Xgig Target Emulator Tab on Xgig Maestro Window



	SQA-SING4 (1,3,3)
State	Locked
Configuration	Edit...
Profile...	No Profile Loaded
Immediate Function Execution	No Functions File Loaded
Macro File Operations...	No Macro File Loaded
Compiled Test Operations...	No Compiled Test Loaded

Select	Show	Protocol	Address				Status
□	☑	SAS	SQA-SING4 (1,3,3)	Apply Profile	Macro Operations	Compiled Test Operations	Lock

DateTime	Type	Address	Contents
10/13/2010 1:03:00 P...	Target Emulator SAS	SQA-SING4 (1,3,3)	[5.00.00] Xgig SAS Tester Opened: S/N 32631, Xgig 2001151, IP 10.66.128.238 10-13-10
10/13/2010 1:03:00 P...	Target Emulator SAS	SQA-SING4 (1,3,3)	debug(2); // 10-13-10 13:02:55

Configuration Manager

Parameters Status table

Ports Manager

Preview Log Pane

Ports/Log Window Tabs

The Target Emulator tab of the Maestro Window contains the following components as shown above:

Configuration Manager, outlined in red, contains the function tabs and the parameters tab. These tabs are described in "Using the Configuration Manager Function Tabs in Target Emulator" on page 120 and "Displaying and Hiding Parameters in Target Emulator" on page 161.

The Parameters Status table, outlined in yellow, provides status information for the port(s) you have locked. This table is described in "Using the Parameters Status Table in Target Emulator" on page 121.

The Ports Manager, outlined in green, allows you to control settings and operations for the port(s) you have locked. This pane is described in "Using the Ports Manager in Target Emulator" on page 152.

The Preview Log pane, outlined in magenta, is a quick reference for the Log Manager. This pane is described in "Using Log Manager in Target Emulator" on page 155.

The Ports and Log window tabs, outlined in blue, switch between the Ports and Log windows.

Using the Configuration Manager Function Tabs in Target Emulator

There are five function tabs in the Configuration Manager, **Profiles**, **Function Files**, **Macro File**, **Compiled Tests** and **Parameters**. The files displayed in the manager are specific to the function tab displayed. The Configuration Manager provides a list of files organized in three types of folders:

- **Sample Files** includes configurations that are provided with the application.
- **User Configurations** is the default location where files you create are saved.
- **Most Recently Used** allows you easy access to those configuration files most recently used.

The Configuration Manager also includes a list of all the drives on your system allowing you to locate any configuration files on your system quickly.

The tool bar in the Configuration Manager contains the following icons and functionality:

New File allows you to create a new profile file from the **Profiles File** tab, to create a new functions file from the **Functions File** tab, or to create a new macro file from the **Macro Files** tab. This icon is not present on the Compile Test tab because you cannot create compiled test files from the GUI.

New Folder allows you to create a new folder within the **Sample Profiles** folder or the **User Profiles** folders.

Load allows you to load the selected profiles, functions, macro, or compiled test file into a port. This icon is only active when a port has been locked and is selected in the Port Manager. You can also drag-and-drop a file from the Configuration Manager into a port.

Map allows you to select a folder that you want to be listed in the Configuration Manager for the selected function tab.

Unmap allows you to unmap a folder that has been mapped.

Cut allows you to cut a file from its current location. This is not the same as deleting a file.

Copy allows you to copy a file.

Paste allows you to paste a file you have cut or copied.

Delete allows you to delete a file you have selected.

Refresh allows you to refresh the file list.

Path allows you to go up one level in the folder tree. This field next to this icon displays the current folder name.

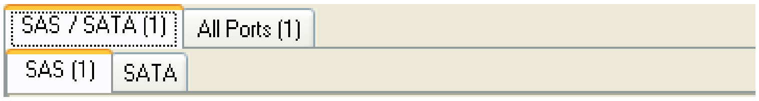
Show/Hide allows you to show or hide the details for the selected file. These details appear in the lower section of the Configuration Manager pane.

View allows you to view the files in the Configuration Manager as a list including only the file name or to view the details of the files in the list.

Using the Parameters Status Table in Target Emulator

The port(s) you have locked are displayed in the Target Emulator Parameters Status table. The ports are arranged into tabs based on protocol type (refer to Figure 66).

Figure 66: Port Tabs



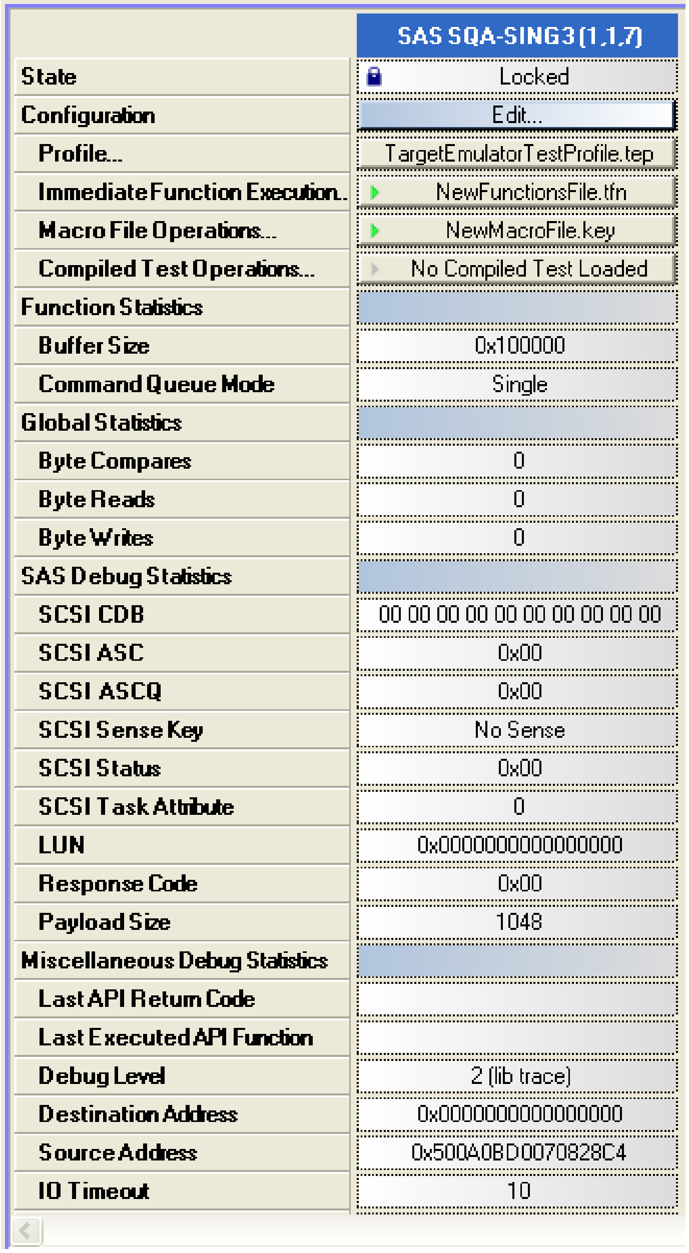
Click on the **SAS/SATA** tab to display the SAS and SATA port tabs. The number beside the protocol name indicates the number of ports using that protocol.

Click the **All Ports** tab to display the all the locked ports under one tab.

Under the tabs, each port is displayed as a column. All the parameters of the port are displayed as rows.

At the top of each column is the protocol, the chassis name, and the chassis number, the slot number and port number(s) in parenthesis.

Figure 67: Target Emulator Parameters Status table



	SAS SQA-SING3 (1,1,7)
State	Locked
Configuration	Edit...
Profile...	TargetEmulatorTestProfile.tep
Immediate Function Execution...	NewFunctionsFile.tfn
Macro File Operations...	NewMacroFile.key
Compiled Test Operations...	No Compiled Test Loaded
Function Statistics	
Buffer Size	0x100000
Command Queue Mode	Single
Global Statistics	
Byte Compares	0
Byte Reads	0
Byte Writes	0
SAS Debug Statistics	
SCSI CDB	00 00 00 00 00 00 00 00 00 00
SCSI ASC	0x00
SCSI ASCQ	0x00
SCSI Sense Key	No Sense
SCSI Status	0x00
SCSI Task Attribute	0
LUN	0x0000000000000000
Response Code	0x00
Payload Size	1048
Miscellaneous Debug Statistics	
Last API Return Code	
Last Executed API Function	
Debug Level	2 (lib trace)
Destination Address	0x0000000000000000
Source Address	0x500A0BD0070828C4
IO Timeout	10

The first column on the tab is the legend for each row in the device columns and contains the following categories:

- State
- Configuration
- Function Statistics
- Global Statistics
- SAS Debug Statistics
- Miscellaneous Debug Statistics

The following categories are only available for SAS:

- Target Characteristics
- Delay Injections
- Frame Field Replacements
- CRC Error Injections
- EOF Error Injections
- Disparity Error Injections
- Data Miscompare Injections
- Data Offset/Length Error Injections
- Status and Sense Injections
- Break Sequence Error Injections
- Hard Reset Sequence Error
- Reset Sequence Error Injections

The following categories are only available for SATA:

- Identify Device Data Configuration
- FIS Delay Injections
- FIS Error Injections
- Custom FIS Injections
- FIS Primitive Injections

Parameters Category Descriptions for Target Emulator

State

The current state of the port, such as stopped, stopping, running, disconnected, connecting, and others.

Configuration

Opens the edit window for macros, functions files, and compiled tests. From this edit window you can create or edit macro files and functions files, and you can define parameters for an existing compiled test file.

Profile

Indicates the currently loaded Profile. If no functions file is loaded, the field will read, “No Profile Loaded”. Clicking this field, with or without a functions file loaded, displays the function library opens the **Target Emulator Device** window.

Immediate Function Execution

Indicates the currently loaded functions file. If no functions file is loaded, the field will read, “No Functions File Loaded”. Clicking this field, with or without a functions file loaded, displays the function library and allows you to select a single function and execute it using the **Start** ▷ button. See “Executing a Function on a Generator Device” on page 51” for more information on this functionality.

Macro File Operations

Indicates the currently loaded macro file. Clicking the file name allows you to either execute the macro or edit the macro file. If no macro file is loaded, the field will read, “No Macro File Loaded”.

Compiled Test Operations

Indicates the currently loaded compiled test file. Clicking the file name allows you to either execute the compiled test or edit the compiled test by defining parameters for the test. If no compiled test file is loaded, the field will read, “No Compiled Test Loaded”.

➡ **Note:** To remove the configuration for a port, right-click the device column in the Parameters Status table, and select **Reset Configuration**. This effectively “unloads” the functions, macro, and compiles test files you have loaded into the port.

Function Statistics

This category provides information about the settings that define how functions are executed.

Buffer Size

The size of the separate transmit and receive buffers. When a data transfer reaches the end of a buffer, it wraps to the beginning and overwrites previous data (receive) or resends previous data (transmits). The buffer size is user configurable and can be changed by using the `resize_buf()` function.

Command Queue Mode

The current command queue mode for SCSI I/O transfers, i.e. Single, Auto, or Stack. Refer to the `cmdque_mode()` function. The default setting is Single.

Global Statistics

This category provides information about whether bytes are being processed within specified limits.

Byte Compares

Total number of bytes compared since being reset by the `stats_reset()` function or since the start of the current test session.

Byte Reads

The total number of bytes read since being reset by the `stats_reset()` function or since the beginning of the current tester session.

Byte Writes

The total number of bytes written since being reset by the `stats_reset()` function or since the beginning of the current Generator testing session.

SAS Debug Statistics

This category provides information to help determine the cause of errors occurring during Generator testing.

SCSI CDB

These 16 bytes are the current values contained in the most recently sent SCSI command.

SCSI ASC

Sense key information (according to the SCSI-3 specification) returned in the latest response frame.

SCSI ASCQ

Sense code qual information (according to the SCSI-3 specification) returned in the latest response frame.

SCSI Sense Key

Sense key information (according to the SCSI-3 specification) returned in the latest response frame.

SCSI Status

The value of the SCSI status byte returned in the latest response frame. When a command is issued to a target, the status value is set to 0xFF, which is changed when the command is complete.

SCSI Task Attribute

The SCSI queuing mode or task attribute used in SCSI command frames. This can be changed with the `queue_xx()` functions. Refer to the `queue_aca`, `queue_head`, `queue_ordered`, `queue_simple`, and `queue_untagged` functions for more information.

LUN

A decimal value indicating the current logical unit number (LUN) to be used in command frames during SCSI operations.

Response Code

Response information code from the response frame of the most recently completed command.

Payload Size

The maximum payload (or frame) size for SAS.

Miscellaneous Debug Statistics

Last API Return Code

Indicator to determine the success of a function. Refer to the Generator API Help system for a list of possible returns for a specific function and the definitions for each return.

Last Executed API Function

Indicates the most recent API function that was executed. This includes single functions as well as functions in functions files and macro files.

Debug Level

Indicates the current debug level. The default value is 2- Display trace of library calls.

To change the debug level:

1. Select **Debug level** from the context menu.
2. Select the debug level you want: 0 (no tracing), 2 (lib trace), 3 (driver trace), 4 (lib and driver trace), or 6 (lib trace and return codes).

Destination Address

A hexadecimal value indicating the current destination/target ID to be used during SAS operations.

Source Address

A hexadecimal value indicating the current source/initiator ID to be used during SAS operations.

IO Timeout

The length of time to be completed by an operation before an I/O timeout is generated. The timeout value (ioto) is displayed in seconds. At the beginning of an operation, such as the start of a SCSI command, an internal timer starts. While waiting for the operation to progress, the code checks the timer and returns to operator control after ioto seconds have elapsed. For long operations, (such as a FORMAT command), set the timeout value to an appropriately high value.

Phys

The number of phys available on or supported by the tester in use.

Ports

The number of logical ports currently configured on the tester in use. This number can never be more than the number of phys. Currently, Generator can only set port width to 1. Refer to the sas_set_port_width() function for more information.

Receive Index

The current index (or offset) in the receive buffer.

Speed

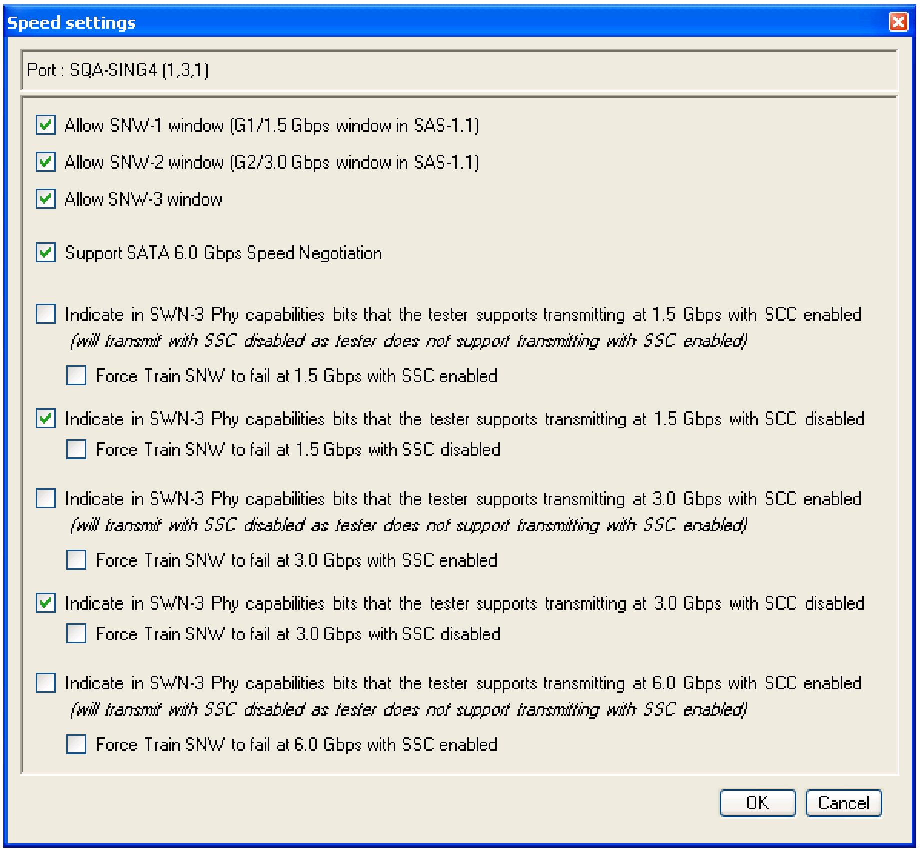
The data transfer rate of the Target Emulator, (1.5G, 3G or 6G SATA or 1.5G, 3G, or 6G SAS). Refer to the `set_speed()` and `set_speed_ex()` functions for more information.

To change the speed settings:

1 Select **Edit speed settings** from the context menu.

The Speed Settings window appears (Only in SAS Mode).

Figure 68: Speed Settings Window



Speed settings

Port : SQA-SING4 (1,3,1)

☑ Allow SNW-1 window (G1/1.5 Gbps window in SAS-1.1)

☑ Allow SNW-2 window (G2/3.0 Gbps window in SAS-1.1)

☑ Allow SNW-3 window

☑ Support SATA 6.0 Gbps Speed Negotiation

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 1.5 Gbps with SCC enabled *(will transmit with SSC disabled as tester does not support transmitting with SSC enabled)*

☐ Force Train SNW to fail at 1.5 Gbps with SSC enabled

☑ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 1.5 Gbps with SCC disabled

☐ Force Train SNW to fail at 1.5 Gbps with SSC disabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 3.0 Gbps with SCC enabled *(will transmit with SSC disabled as tester does not support transmitting with SSC enabled)*

☐ Force Train SNW to fail at 3.0 Gbps with SSC enabled

☑ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 3.0 Gbps with SCC disabled

☐ Force Train SNW to fail at 3.0 Gbps with SSC disabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 6.0 Gbps with SCC enabled *(will transmit with SSC disabled as tester does not support transmitting with SSC enabled)*

☐ Force Train SNW to fail at 6.0 Gbps with SSC enabled

2 Check the SAS-1.1 or SATA 1.5Gig link rate to force the link to negotiate to 1.5Gbps.

3 Check the SAS-1.1 or SATA 3.0Gig link rate to force the link to negotiate to 3.0Gbps.

4 Check both of the two check boxes above to allow the link to auto-negotiate to the highest available speed.

5 Select Allow SNW-3 Window (SAS-2) if you have a SAS device that you want to run at 6.0Gbps. This type of speed negotiation allows for adaptation at both 1.5G and 3G link rates.

➡ **Note:** If you select the “Support SAS-2 speed negotiation setting”, then the Phy capabilities and Fail train sections become active and allow you to change the settings. Otherwise, these sections are inactive as these settings are not applicable.

The phy capabilities are transmitted by the tester during the SAS-2 SNW3 window at the selected speed. Refer to the `get_speed_ex` and `set_speed_ex` functions in the Generator API Help.

This setting defines what you want to convey to a device in the SNW-3 Phy capabilities bits. You can choose to indicate that the tester supports transmitting at a given speed with SSC enabled or with SSC disabled. We do not currently support transmitting with SSC enabled. Therefore, the tester will transmit with SSC disabled even if you set the tester to indicate otherwise.

6 If applicable, set the desired Phy capabilities.

The Fail train is a bitmask that forces the corresponding Train speed negotiation window(s) (SNW) to fail at the selected speed. Refer to the `get_speed_ex, sas_set_timers`, `set_speed_ex` functions in the Generator API Help.

This setting forces the train SNW to fail at a given speed with SSC enabled or with SSC disabled. setting indicates fail train SNW can be set with SSC disabled. We do not currently support forcing the train SNW to fail with SSC enabled. Therefore, the tester will force the train SNW to fail with SSC disabled even if the settings indicate otherwise.

7 If applicable, set the desired Fail train SNW settings.

8 Click **OK** to close the dialog and apply the settings.

Transfer Mode

The type of data transfer for I/O functions. Refer to the hgen_mode() function for more information.

Transmit Index

The current index (or offset) in the transmit buffer.

FPGA Version

Represents a bit file revision code for the Xgig blade’s FPGA.

Embedded OS Version

Indicates the version of software running on the Xgig chassis.

Link Type

Current link type: Unknown, SAS link, or SATA link. Refer to the `get_g_info` function for more information about `link_type`.

Link Type Config

Current link type configuration: SAS only, SATA only, or SAS and SATA. Refer to the `get_g_info` function for more information about `link_type_cfg`.

Parameters in SAS Mode

Target Characteristics

This category provides information about the emulated target.

Vendor ID

Indicates the Vendor ID of the emulated target drive set in the Target Emulator profile.

Product ID

Indicates the Product ID of the emulated target drive set in the Target Emulator profile.

Product Revision

Indicates the Product Revision number of the emulated target drive set in the Target Emulator profile.

Device Serial

Indicates the device serial number of the emulated target drive set in the Target Emulator profile.

Device Capacity (sectors)

Indicates the device capacity of the emulated target drive set in the Target Emulator profile.

SAS and SATA Target Emulator provide default storage capacity of 48MB. Users can change the capacity to more than 48MB. I/O operations attempting to write beyond the 48MB will round buffer and may overwrite the data that was earlier written to the emulator's memory. User using operations that perform data integrity/comparison checks may want to take note of this.

Transfer Size (sectors)

Indicates the transfer size of the emulated target drive set in the Target Emulator profile.

Delay Injections

The following parameters reflect the delay injection settings for the Target Emulator profile applied to the port.

Response Frame Delays

Indicates the number of delays for response frames set in the Target Emulator profile.

DATA Frame Delays

Indicates the number of delays for DATA frames set in the Target Emulator profile.

XFER_RDY Frame Delays

Indicates the number of delays for XFER_RDY frames set in the Target Emulator profile.

Min. Resp. Frame countdown

Indicates the minimum number of times a response frame is detected before emitting the required frame response as set in the Target Emulator profile.

Min. DATA Frame Countdown

Indicates the minimum number of times a DATA frame is detected before emitting the required frame response as set in the Target Emulator profile.

Min. XFER_RDY Frame Countdown

Indicates the minimum number of times a XFER_RDY frame is detected before emitting the required frame response as set in the Target Emulator profile.

Total Injections Counter

Indicates the number of errors that have been injected.

Frame Field Replacements

The following parameters reflect the Frame Field Injections settings for the Target Emulator profile applied to the port.

Enabled for Resp. Frames

Indicates **True** if Frame Fields are enabled and **False** if not.

Freq. for Resp. Frames

Indicates the number of times a request frame is detected before emitting the Response frame as set in the Target Emulator profile.

Countdown for Resp. Frames

Indicates the number of events until the next Frame Field Replacements.

Enabled for DATA Frames

Indicates **True** if Frame Fields are enabled and **False** if not.

Freq. for DATA Frames

Indicates the number of times a request frame is detected before emitting the DATA frame as set in the Target Emulator profile.

Countdown DATA Frames

Indicates the number of events until the next Frame Field Replacements.

Enabled for XFER_RDY Frames

Indicates **True** if Frame Fields are enabled and **False** if not.

Freq. for XFER_RDY Frames

Indicates the number of times a request frame is detected before emitting the XFER_RDY frame as set in the Target Emulator profile.

Countdown for XFER_RDY Frames

Indicates the number of events until the next Frame Field Replacements.

Total Injections Counter

Indicates the number of errors that have been injected.

CRC Error Injections

The following parameters reflect the CRC Error Injection settings for the Target Emulator profile applied to the port.

Enabled for Resp. Frames

Indicates **True** if CRC error injections are enabled for response frames and **False** if not.

Freq. for Resp. Frames

Indicates the number of times a request frame is detected before emitting the Response frame as set in the Target Emulator profile.

Countdown for Resp. Frames

Indicates the number of events until the next error injection (1 means an error will be injected during the next event).

Enabled for DATA Frames

Indicates **True** if CRC error injections are enabled for DATA frames and **False** if not.

Freq. for DATA Frames

Indicates the number of times a request frame is detected before emitting the DATA frame as set in the Target Emulator profile.

Countdown DATA Frames

Indicates the number of events until the next error injection (1 means an error will be injected during the next event).

Enabled for XFER_RDY Frames

Indicates **True** if CRC error injections are enabled for XFER_RDY frames and **False** if not.

Freq. for XFER_RDY Frames

Indicates the number of times a request frame is detected before emitting the XFER_RDY frame as set in the Target Emulator profile.

Countdown for XFER_RDY Frames

Indicates the number of events until the next error injection (1 means an error will be injected during the next event).

Total Injections Counter

Indicates the number of errors that have been injected.

EOF Error Injections

The following parameters reflect the EOF Error Injection settings for the Target Emulator profile applied to the port.

Enabled for Resp. Frames

Indicates **True** if EOF error injections are enabled for response frames and **False** if not.

Freq. for Resp. Frames

Indicates the number of times a request frame is detected before emitting the Response frame as set in the Target Emulator profile.

Countdown for Resp. Frames

Indicates the number of events until the next error injection (1 means an error will be injected during the next event).

Enabled for DATA Frames

Indicates **True** if EOF error injections are enabled for DATA frames and **False** if not.

Freq. for DATA Frames

Indicates the number of times a request frame is detected before emitting the DATA frame as set in the Target Emulator profile.

Countdown for DATA Frames

Indicates the number of events until the next error injection (1 means an error will be injected during the next event).

Enabled for XFER_RDY Frames

Indicates **True** if EOF error injections are enabled for XFER_RDY frames and **False** if not.

Freq. for XFER_RDY Frames

Indicates the number of times a request frame is detected before emitting the XFER_RDY frame as set in the Target Emulator profile.

Countdown for XFER_RDY Frames

Indicates the number of events until the next error injection (1 means an error will be injected during the next event).

Total Injections Counter

Indicates the number of errors that have been injected.

Disparity Error Injections

The following parameters reflect the Disparity Error Injection settings for the Target Emulator profile applied to the port.

Enabled for Resp. Frames

Indicates **True** if disparity error injections are enabled for response frames and **False** if not.

Freq. for Resp. Frames

Indicates the number of times a request frame is detected before emitting the required frame response as set in the Target Emulator profile.

Countdown for Resp. Frames

Indicates the number of events until the next error injection (1 means an error will be injected during the next event).

Enabled for DATA Frames

Indicates **True** if disparity error injections are enabled for DATA frames and **False** if not.

Freq. for DATA Frames

Indicates the number of times a request frame is detected before emitting the required frame response as set in the Target Emulator profile.

Countdown for DATA Frames

Indicates the number of events until the next error injection (1 means an error will be injected during the next event).

Enabled for XFER_RDY Frames

Indicates **True** if disparity error injections are enabled for DATA frames and **False** if not.

Freq. for XFER_RDY Frames

Indicates the number of times a request frame is detected before emitting the required frame response as set in the Target Emulator profile.

Countdown for XFER_RDY Frames

Indicates the number of events until the next error injection (1 means an error will be injected during the next event).

Total Injections Counter

Indicates the number of errors that have been injected.

Data Miscompare Injections

The following parameters reflect the Data Miscompare Injection settings for the Target Emulator profile applied to the port.

Enabled

Indicates **True** if Data Miscompare error injections are enabled and **False** if not.

Start Byte

Indicates the start location of the Byte range where the original data replacement will take place.

Stop Byte

Indicates the stop location of the Byte range where the original data replacement will take place.

Replacement Pattern

Indicates the data pattern that will replace the original data.

Frequency

Indicates the number of times a request frame is detected before emitting the modified data frames.

Countdown

Indicates the number of events until next error injection (1 means an error will be injected during the next event).

Total Injections Counter

Indicates the number of errors that have been injected.

Data Offset/Length Error Injections

The following parameters reflect the Data Miscompare Injection settings for the Target Emulator profile applied to the port.

Enabled for DATA Frames

Indicates **True** if Data Offset/Length Error injections are enabled for DATA frames and **False** if not.

Freq. for DATA Frames

Indicates the number of times the DATA frames are detected before the requested changes are injected into the frame.

Countdown for DATA Frames

Indicates the number of events until next error injection (1 means an error will be injected during the next event).

Enabled for XFER_RDY Frames

Indicates **True** if Data Offset/Length Error injections are enabled for XFER_RDY frames and **False** if not.

Freq. for XFER_RDY Frames

Indicates the number of times the XFER_RDY frames are detected before the requested changes are injected into the frame.

Countdown for XFER_RDY Frames

Indicates the number of events until next error injection (1 means an error will be injected during the next event).

Total Injections Counter

Indicates the number of errors that have been injected.

Status and Sense Injections

The following parameters reflect the Status and Sense Injection settings for the Target Emulator profile applied to the port.

Injections Defined

Indicates the number of customized SCSI commands defined in the Target Emulator Profile.

Min. Frequency

Indicates the minimum frequency value set among all custom defined SCSI command codes in the profile.

Max. Frequency

Indicates the maximum frequency value set among all custom defined SCSI command codes in the profile.

Min. Countdown

Indicates the minimum countdown value set among all custom defined SCSI command codes in the profile.

Max. Countdown

Indicates the maximum countdown value set among all custom defined SCSI command codes in the profile.

Total Injections Counter

Indicates the number of errors that have been injected.

Break Sequence Error Injections

Injections Defined

Indicates the number of customized SCSI commands defined in the Target Emulator Profile.

Min. Frequency

Indicates the minimum frequency value set among all custom defined SCSI command codes in the profile.

Max. Frequency

Indicates the maximum frequency value set among all custom defined SCSI command codes in the profile.

Min. Countdown

Indicates the minimum countdown value set among all custom defined SCSI command codes in the profile.

Max. Countdown

Indicates the maximum countdown value set among all custom defined SCSI command codes in the profile.

Total Injections Counter

Indicates the number of errors that have been injected.

Hard Reset Sequence Error Injections

Injections Defined

Indicates the number of customized SCSI commands defined in the Target Emulator Profile.

Min. Frequency

Indicates the minimum frequency value set among all custom defined SCSI command codes in the profile.

Max. Frequency

Indicates the maximum frequency value set among all custom defined SCSI command codes in the profile.

Min. Countdown

Indicates the minimum countdown value set among all custom defined SCSI command codes in the profile.

Max. Countdown

Indicates the maximum countdown value set among all custom defined SCSI command codes in the profile.

Total Injections Counter

Indicates the number of errors that have been injected.

Reset Sequence Error Injections

Injections Defined

Indicates the number of customized SCSI commands defined in the Target Emulator Profile.

Min. Frequency

Indicates the minimum frequency value set among all custom defined SCSI command codes in the profile.

Max. Frequency

Indicates the maximum frequency value set among all custom defined SCSI command codes in the profile.

Min. Countdown

Indicates the minimum countdown value set among all custom defined SCSI command codes in the profile.

Max. Countdown

Indicates the maximum countdown value set among all custom defined SCSI command codes in the profile.

Total Injections Counter

Indicates the number of errors that have been injected.

Parameters for SATA Mode

Identify Device Data Configuration

Serial Number

Indicates the serial number of the Target Emulator drive. To change this value, right-click on the parameters column and select **Edit Configuration** to open the Target Emulator Device window.

Firmware Revision

Indicates the firmware version of the Target Emulator drive. To change this value, right-click on the parameters column and select **Edit Configuration** to open the Target Emulator Device window.

Model Number

Indicates the model number of the Target Emulator drive. To change this value, right-click on the parameters column and select **Edit Configuration** to open the Target Emulator Device window.

Multiple Count (sectors)

Indicates the maximum number of logical sectors per DRQ data block that the device supports for READ/WRITE MULTIPLE commands. For SATA devices, this is set to 16 or less. Only values 0x01, 0x02, 0x04, 0x08 and 0x10 are allowed.

28 bits addressable user sects

Indicates a value that is one greater than the maximum user addressable LBA. The maximum value that shall be placed in this field is 0FFF_FFFFh. If this field contains 0FFF_FFFFh and the device has user addressable LBAs greater than or equal to 0FFF_FFFFh then words 100..103 contain the total number of user addressable LBAs.

SAS and SATA Target Emulator provide default storage capacity of 48MB. Users can change the capacity to more than 48MB. I/O operations attempting to write beyond the 48MB will round buffer and may overwrite the data that was earlier written to the emulator's memory. User using operations that perform data integrity/comparison checks may want to take note of this.

48 bits addressable user sects

Indicates a value that is one greater than the maximum LBA in user accessible space when the 48-bit Addressing feature set is supported. The maximum value that shall be placed in this field is 0000_FFFF_FFFF_FFFFh. Support of these words is mandatory if the 48-bit Address feature set is supported.

SAS and SATA Target Emulator provide default storage capacity of 48MB. Users can change the capacity to more than 48MB. I/O operations attempting to write beyond the 48MB will round buffer and may overwrite the data that was earlier written to the emulator's memory. User using operations that perform data integrity/comparison checks may want to take note of this.

Supports SATA 6 Gbps

Indicates whether the support for Serial ATA Gen3 signalling speed (6.0Gbps) is enabled in the Identify Device Data Configuration. The SATA Target Emulator attempts speed negotiation at 6.0Gbps, if this Bit is 1.

Supports SATA 3 Gbps

Indicates whether the support for Serial ATA Gen2 signalling speed (3.0Gbps) is enabled in the Identify Device Data Configuration. The SATA Target Emulator attempts speed negotiation at 3.0Gbps, if this Bit is 1 and Word 76 Bit 3 is 0.

Supports SATA 1.5 Gbps

Indicates whether the support for Serial ATA Gen1 signalling speed (1.5Gbps) is enabled in the Identify Device Data Configuration. The SATA Target Emulator attempts speed negotiation at 1.5Gbps, if this Bit is 1, both Word 76 Bit 3 and Word 76 Bit 2 are 0.

Supports dev. auto partial to slumber

Indicates whether the support for Device Automatic Partial to Slumber transitions is enabled in the Identify Device Data Configuration.

Supports host auto partial to slumber

Indicates whether the support for Host Automatic Partial to Slumber transitions is enabled in the Identify Device Data Configuration.

Supports unload while NCQ

Indicates whether the support for Unload while NCQ commands outstanding is enabled in the Identify Device Data Configuration. If this feature is enabled, the Target Emulator supports the Priority field in the READ FPDMA QUEUED and WRITE FPDMA QUEUED commands and optimization based on this information.

Support dev. initiated PMREQ

Indicates whether the support for device initiating interface power management is enabled in the Identify Device Data Configuration.

Supports recep. host initiated PMREQ

Indicates whether the support for Host Automatic Partial to Slumber transitions is enabled in the Identify Device Data Configuration.

Supports NCQ

Indicates whether the support for Native Command Queuing is enabled in the Identify Device Data Configuration.

Supports dev. initiated PMREQ

Indicates whether the support for receipt of device-initiated interface power management requests is enabled in the Identify Device Data Configuration. The SATA Target Emulator will only initiate interface power management when it is in STANDBY mode. STANDBY mode can be achieved by issuing a Standby Immediate ATA command to the target emulator. When initiating interface power management, the target emulator issues PMREQ_S primitives to the host.

Supports host protected area

Indicates whether the Host Protected Area (HPA) feature set is enabled in the Identify Device Data Configuration.

Supports 48 bits addressing

Indicates whether the 48-bit Address feature set is supported.

Supports idle IMM unload

Indicates whether the IDLE IMMEDIATE command with unload feature is supported.

Supports FUA EXT

Indicates whether the WRITE DMA FUA EXT command and WRITE MULTIPLE FUA EXT command is supported.

FIS Delay Injections

Dev. to Host FIS delay (secs)

Indicates the Device to Host FIS delay time (in seconds) as specified in the delay injection section of the Target Emulator Profile.

Dev. to Host FIS delay countdown

Indicates the Device to Host FIS delay countdown before the next Device to Host FIS delay injection.

Dev. to Host FIS Delay Frequency

Indicates the Device to Host FIS delay frequency as specified in the delay injection section of the Target Emulator Profile.

DMA Activate FIS delay (secs)

Indicates the DMA Activate FIS delay time (in seconds) as specified in the delay injection section of the Target Emulator Profile.

DMA Activate FIS delay countdown

Indicates the DMA Activate FIS delay countdown before the next DMA Activate FIS delay injection.

DMA Activate FIS delay frequency

Indicates the DMA Activate FIS delay frequency as specified in the delay injection section of the Target Emulator Profile.

DMA Setup FIS delay (secs)

Indicates the DMA Setup FIS delay time (in seconds) as specified in the delay injection section of the Target Emulator Profile.

DMA Setup FIS delay countdown

Indicates the DMA Setup FIS delay countdown before the next DMA Activate FIS delay injection.

DMA Setup FIS delay frequency

Indicates the DMA Setup FIS delay frequency as specified in the delay injection section of the Target Emulator Profile.

Data FIS delay (secs)

Indicates the Data FIS delay time (in seconds) as specified in the delay injection section of the Target Emulator Profile.

Data FIS delay countdown

Indicates the Data FIS delay countdown before the next Data FIS delay injection.

Data FIS delay frequency

Indicates the Data FIS delay frequency as specified in the delay injection section of the Target Emulator Profile.

PIO Setup FIS delay (secs)

Indicates the PIO Setup FIS delay time (in seconds) as specified in the delay injection section of the Target Emulator Profile.

PIO Setup FIS delay countdown

Indicates the PIO Setup FIS delay countdown before the next PIO Setup FIS delay countdown.

PIO Setup FIS delay frequency

Indicates the PIO Setup FIS delay frequency as specified in the delay injection section of the Target Emulator Profile.

Set Device Bits FIS delay (secs)

Indicates the Set Device Bits FIS delay time (in seconds) as specified in the delay injection section of the Target Emulator Profile.

Set Device Bits FIS delay countdown

Indicates the Set Device Bits FIS delay countdown before the next DMA Activate FIS delay injection.

Set Device Bits FIS delay frequency

Indicates the Set Device Bits FIS delay frequency as specified in the delay injection section of the Target Emulator Profile.

Total injection count

Indicates the total number of delay injections sent.

FIS Error Injections

Dev. to Host FIS errors enabled

Indicates whether there are errors enabled for the Device to Host FIS type.

Dev. to Host FIS errors injected

Indicates the error type injected for the Device to Host FIS type.

Dev. to Host FIS error frequency

Indicates the number of request frames to detect before emitting a modified frame. For example, If you set a frequency of **4**, an error will be injected every **4th frame**.

Dev. to Host FIS error countdown

Indicates the number of request frames remaining before injecting the error.

DMA Activate FIS errors enabled

Indicates whether there are errors enabled for the DMA Activate FIS type.

DMA Activate FIS errors injected

Indicates the error type injected for the DMA Activate FIS type.

DMA Activate FIS error frequency

Indicates the number of request frames to detect before emitting a modified frame. For example, If you set a frequency of **4**, an error will be injected every **4th frame**.

DMA Activate FIS error countdown

Indicates the number of request frames remaining before injecting the error.

DMA Setup FIS errors enabled

Indicates whether there are errors enabled for the DMA Setup FIS type.

DMA Setup FIS errors injected

Indicates the error type injected for the DMA Setup FIS type.

DMA Setup FIS error frequency

Indicates the number of request frames to detect before emitting a modified frame. For example, If you set a frequency of **4**, an error will be injected every **4th frame**.

DMA Setup FIS error countdown

Indicates the number of request frames remaining before injecting the error.

Data FIS errors enabled

Indicates whether there are errors enabled for the Data FIS type.

Data FIS errors injected

Indicates the error type injected for the Data FIS type.

Data FIS error frequency

Indicates the number of request frames to detect before emitting a modified frame. For example, If you set a frequency of **4**, an error will be injected every **4th frame**.

Data FIS error countdown

Indicates the number of request frames remaining before injecting the error.

PIO Setup FIS errors enabled

Indicates whether there are errors enabled for the PIO Setup FIS type.

PIO Setup FIS errors injected

Indicates the error type injected for the PIO Setup FIS type.

PIO Setup FIS error frequency

Indicates the number of request frames to detect before emitting a modified frame. For example, If you set a frequency of **4**, an error will be injected every **4th frame**.

PIO Setup FIS error countdown

Indicates the number of request frames remaining before injecting the error.

Set Device Bits FIS errors enabled

Indicates whether there are errors enabled for the Set Device Bits FIS type.

Set Device Bits FIS errors injected

Indicates the error type injected for the Set Device Bits FIS type.

Set Device Bits error frequency

Indicates the number of request frames to detect before emitting a modified frame. For example, If you set a frequency of **4**, an error will be injected every **4th frame**.

Set Device Bits FIS error countdown

Indicates the number of request frames remaining before injecting the error.

Total injection count

Indicates the total number of FIS error injections sent.

Custom FIS Injections

Enabled for Dev. to Host FIS

Indicates whether there are custom FIS injections enabled for the Device to Host FIS type.

Custom size for Dev. to Host FIS (dword)

Indicates the size (in Dwords) of the Custom FIS.

Freq. for Dev. to Host FIS

Indicates the number of request frames to detect before emitting a Custom FIS. For example, If you set a frequency of **4**, an error will be injected every **4th frame**.

Countdown for Dev. to Host FIS

Indicates the number of request frames remaining before injecting the error.

Enabled for DMA Activate FIS

Indicates whether there are custom FIS injections enabled for the DMA Activate FIS type.

Custom size for DMA Activate FIS (dword)

Indicates the size (in Dwords) of the Custom FIS.

Freq. for DMA Activate FIS

Indicates the number of request frames to detect before emitting a Custom FIS. For example, If you set a frequency of **4**, an error will be injected every **4th frame**.

Countdown for DMA Activate FIS

Indicates the number of request frames remaining before injecting the error.

Enabled for DMA Setup FIS

Indicates whether there are custom FIS injections enabled for the DMA Setup FIS type.

Custom size for DMA Setup FIS (dword)

Indicates the size (in Dwords) of the Custom FIS.

Freq. for DMA Setup FIS

Indicates the number of request frames to detect before emitting a Custom FIS. For example, If you set a frequency of **4**, an error will be injected every **4th frame**.

Countdown for DMA Setup FIS

Indicates the number of request frames remaining before injecting the error.

Enabled for PIO Setup FIS

Indicates whether there are custom FIS injections enabled for the PIO Setup FIS type.

Custom size for PIO Setup FIS (dword)

Indicates the size (in Dwords) of the Custom FIS.

Freq. for DMA PIO FIS

Indicates the number of request frames to detect before emitting a Custom FIS. For example, If you set a frequency of **4**, an error will be injected every **4th frame**.

Countdown for PIO Setup FIS

Indicates the number of request frames remaining before injecting the error.

Enabled for Set Device Bits FIS

Indicates whether there are custom FIS injections enabled for the Set Device Bits FIS type.

Custom size for Set Device Bits FIS (dword)

Indicates the size (in Dwords) of the Custom FIS.

Freq. for Set Device Bits FIS

Indicates the number of request frames to detect before emitting a Custom FIS. For example, If you set a frequency of **4**, an error will be injected every **4th frame**.

Countdown for Set Device Bits FIS

Indicates the number of request frames remaining before injecting the error.

Total injection count

Indicates the total number of Custom FIS injections sent.

FIS Primitive Injections

Enabled for Dev. to Host FIS

Indicates whether there are primitive injections enabled for the Device To Host FIS type.

Prim. injected after Dev. to Host FIS

Indicates the type of primitive injected after the Device to Host FIS.

Prim. count for Dev. to Host FIS

Indicates the number of the specified primitive to be injected.

Freq. for Dev. to Host FIS

Indicates the number of FIS frame occurrence of the specified FIS type before injecting the primitive.

Countdown for Dev. to Host FIS

Indicates the number of request frames remaining before injecting the primitive.

Enabled for DMA Activate FIS

Indicates whether there are primitive injections enabled for the DMA Activate FIS type.

Prim. injected after DMA Activate FIS

Indicates the number of the specified primitive to be injected.

Prim. count for DMA Activate FIS

Indicates the number of the specified primitive to be injected.

Freq. for DMA Activate FIS

Indicates the number of FIS frame occurrence of the specified FIS type before injecting the primitive.

Countdown for DMA Activate FIS

Indicates the number of request frames remaining before injecting the primitive.

Enabled for DMA Setup FIS

Indicates whether there are primitive injections enabled for the DMA Setup FIS type.

Prim. injected after DMA Setup FIS

Indicates the number of the specified primitive to be injected.

Prim. count for DMA Setup FIS

Indicates the number of the specified primitive to be injected.

Freq. for DMA Setup FIS

Indicates the number of FIS frame occurrence of the specified FIS type before injecting the primitive.

Countdown for DMA Setup FIS

Indicates the number of request frames remaining before injecting the primitive.

Enabled for Data FIS

Indicates whether there are primitive injections enabled for the Data FIS type.

Prim. injected after Data FIS

Indicates the number of the specified primitive to be injected.

Prim. count for Data FIS

Indicates the number of the specified primitive to be injected.

Freq. for Data FIS

Indicates the number of FIS frame occurrence of the specified FIS type before injecting the primitive.

Countdown for Data FIS

Indicates the number of request frames remaining before injecting the primitive.

Enabled for PIO Setup FIS

Indicates whether there are primitive injections enabled for the PIO Setup FIS type.

Prim. injected after PIO Setup FIS

Indicates the number of the specified primitive to be injected.

Prim. count for PIO Setup FIS

Indicates the number of the specified primitive to be injected.

Freq. for PIO Setup FIS

Indicates the number of FIS frame occurrence of the specified FIS type before injecting the primitive.

Countdown for PIO Setup FIS

Indicates the number of request frames remaining before injecting the primitive.

Enabled for Set Device Bits FIS

Indicates whether there are primitive injections enabled for the Set Device Bits FIS type.

Prim. injected after Set Device Bits FIS

Indicates the number of the specified primitive to be injected.

Prim. count for Set Device Bits FIS

Indicates the number of the specified primitive to be injected.

Freq. for Set Device Bits FIS

Indicates the number of FIS frame occurrence of the specified FIS type before injecting the primitive.

Countdown for Set Device Bits FIS

Indicates the number of request frames remaining before injecting the primitive.

Enabled for Host to Dev. FIS

Indicates whether there are primitive injections enabled for the Host to Device FIS type.

Prim. injected after Host to Dev. FIS

Indicates the number of the specified primitive to be injected.

Prim. count for Host to Dev. FIS

Indicates the number of the specified primitive to be injected.

Freq. for Set Host to Dev. FIS

Indicates the number of FIS frame occurrence of the specified FIS type before injecting the primitive.

Countdown for Host to Dev. FIS

Indicates the number of request frames remaining before injecting the primitive.

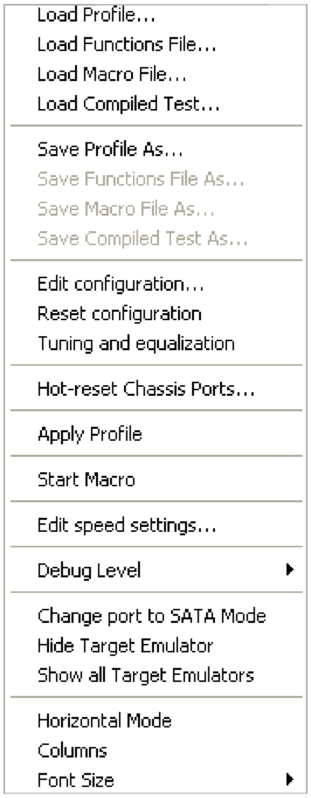
Total injection count

Indicates the total number of primitive injections sent.

Using the Parameters Status Table Context Menu

Right-click on a Parameters status column to display the context menu. The available operations are displayed for the column in which you click. This menu provides controls for configuration files, operation, setting clock speed, setting Target Emulator mode of operation (SAS to SATA), tuning and equalization, removing a Target Emulator from the Parameters Status table, and changing the appearance of the status table.

Figure 69: Parameters Status Table Context Menu for Target Emulator



Loading Files

To load a Profile click **Load Profile...**

To load a Functions File, click **Load Functions File...**

To load Macro File, click **Load Macro File...**

To load a Compiled Test, click **Load Compiled Test...**

Saving Files

To save the current Profile, click **Save Profile As...**

To save the current Functions file, click **Save Functions File As...**

To save the current Macro File, click **Save Macro File As...**

To save the current Compiled Test, click **Save Compiled Test As...**

Editing Configuration

To edit the current port configuration, click **Edit configuration...**

To reset the port configuration to the default settings, click **Reset configuration**.

Tuning and Equalization

Tuning and Equalization are procedures to optimize ports on both wide blade (Xgig SAS/SATA Wide-Port /4x Blade) and narrow blade (Xgig 6 Gigabit SAS/SATA Blade) for your environment. When you connect the Xgig SAS Analyzer, Jammer, Generator, or Target Emulator in a new environment (for example, after changing devices or cable length) we recommend that you run the Tuning procedure detailed in the *Xgig Tuning and Equalization* document.

Resetting Chassis Ports

To do a hot-reset on the selected port, click on **Hot-reset Chassis Ports...**

This will reset all locked Target Emulator ports and clear the settings applied to the port. You need to reconnect to the port again to use it.

Applying Profile Changes

To apply all the changes made to the profile, or to apply the newly loaded profile, click on **Apply Profile**.

Starting Macro

To start a macro file that was loaded, click **Start Macro**.

Editing Speed Settings

To set the speed for the port, click on **Edit speed settings...**

This option is for SAS TE port only. For SATA TE, speed is part of the SATA Identify Device Data settings.

Setting the Debug Level

To set the debug level, click **Debug Level** then select the level from the pop-up menu.

Changing port mode

When you first lock a port as a Target Emulator, it will be in SAS Mode. To change the port to SATA mode, click **Change port to SATA Mode**.

Displaying Target Emulator ports

To hide ports locked as Target emulator from the Parameters Status Table, click on **Hide Target Emulator**. To show the hidden port right-click on the Parameters Stable Table and click **Show all Target Emulators**.

Displaying Ports in the Parameters Status Table

By default locked ports are displayed as columns with the parameters displayed as rows. To reverse this, where the ports are displayed as rows and the parameters are displayed as columns, click **Horizontal Mode**.

To select which parameters will be displayed as columns, click **Columns**. The **Customize Band** window opens where you can select which columns to display.

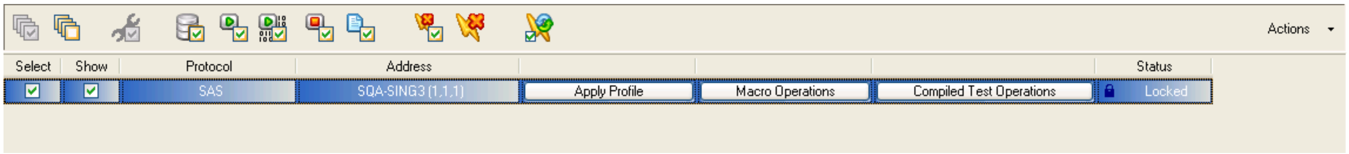
To change the font size of the labels and text used in the Parameters Status Table, click on **Font Size** and then select the font size from the pop-up menu. By default, the font size is set to the smallest.

Using the Ports Manager in Target Emulator

The Ports manager displays details about the ports you have locked (Figure 70). It shows the protocol, chassis name (with chassis number, slot number, and port numbers), and other port specific information. You can sort the rows from first to last or last to first by clicking the column heading.

Note: If you are disconnected due to a network problem, you can reconnect to the ports by using the Port Selection and Domain Setup window.

Figure 70: Ports Manager



Select	Show	Protocol	Address				Status
☑	☑	SAS	SQA-SING3 (1,1,1)	Apply Profile	Macro Operations	Compiled Test Operations	Locked

The following icons are displayed on the menu bar:

Select All Ports
Selects all ports in the Ports Manager.

Unselect All Ports
Unselect all ports in the Ports Manager.

Load Configuration to Selected Ports
Load the selected functions, macro, or compiled test file to all selected ports in the Ports Manager.

Apply Profile to Selected Ports
Load the selected functions, macro, or compiled test file to all selected ports in the Ports Manager.

Start Selected Ports
Start loaded macro file on the selected ports in the Ports Manager.

Start Compiled Test on Selected Ports (Generator Only)
Execute the loaded compiled test on the selected ports in the Ports Manager.

Stop Selected Ports
Stop Generator operation on the selected ports in the Ports Manager.

Show Properties of Selected Ports
Display the Port Properties dialog box, with information on the selected function, protocol, and clock rate (speed). It also shows the chassis name, IP address, slot and port(s).

Disconnect Selected Ports
Disconnect the selected ports in the Ports Manager.

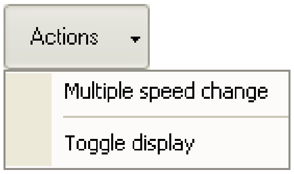
Disconnect All Ports
Disconnect all ports in the Ports Manager.

Hot-reset Selected Ports
Reset Target Emulator running in chassis.

Actions Button

The **Actions** button contains menus for **Multiple speed Change** and **Toggle Display**.

Figure 71: Actions Button Options

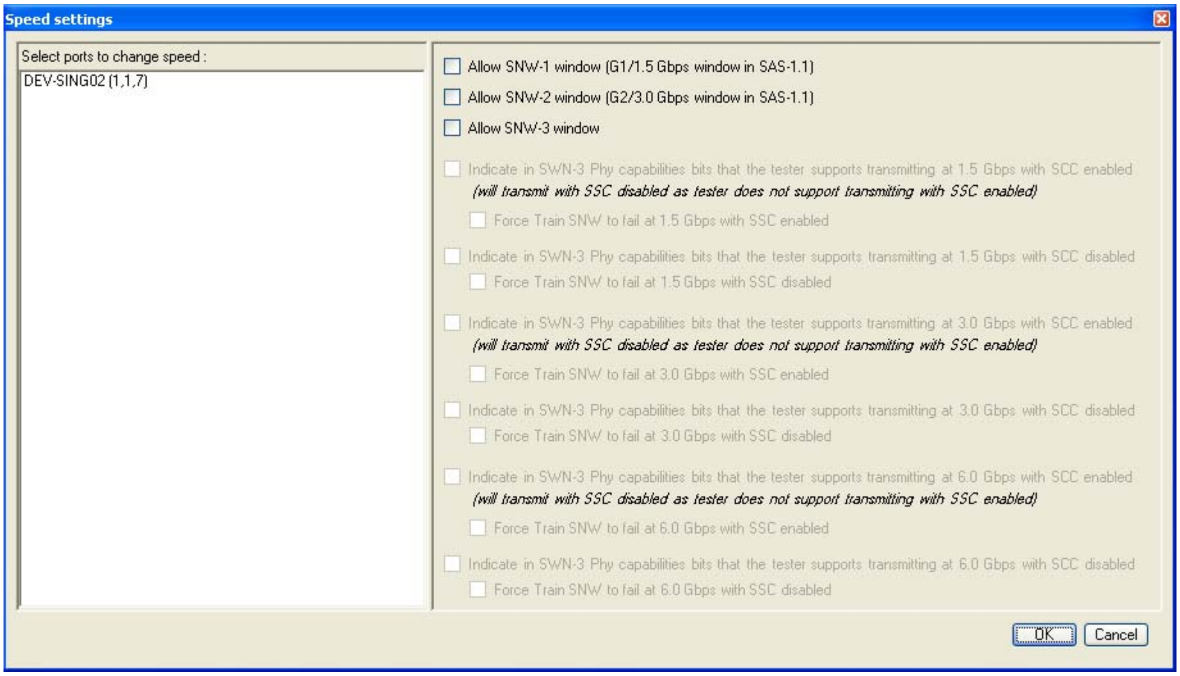


Multiple Speed Change for Target Emulator

To change **Multiple speed change** for the **Target Emulator** tab:

First, select the **Actions** button and then select **Multiple speed change** as shown in Figure 71. The **Speed Settings** window appears as shown in Figure 72. This menu differs from the main speed settings window in that it includes an additional pane that allows you to select the ports for which you want to change the speed.

Figure 72: Multiple Speed Settings Window for Target Emulator



Select ports to change speed :
DEV-SING02 (1,1,7)

☐ Allow SNW-1 window (G1/1.5 Gbps window in SAS-1.1)
☐ Allow SNW-2 window (G2/3.0 Gbps window in SAS-1.1)
☐ Allow SNW-3 window

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 1.5 Gbps with SCC enabled
(will transmit with SSC disabled as tester does not support transmitting with SSC enabled)
☐ Force Train SNW to fail at 1.5 Gbps with SSC enabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 1.5 Gbps with SCC disabled
☐ Force Train SNW to fail at 1.5 Gbps with SSC disabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 3.0 Gbps with SCC enabled
(will transmit with SSC disabled as tester does not support transmitting with SSC enabled)
☐ Force Train SNW to fail at 3.0 Gbps with SSC enabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 3.0 Gbps with SCC disabled
☐ Force Train SNW to fail at 3.0 Gbps with SSC disabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 6.0 Gbps with SCC enabled
(will transmit with SSC disabled as tester does not support transmitting with SSC enabled)
☐ Force Train SNW to fail at 6.0 Gbps with SSC enabled

☐ Indicate in SWN-3 Phy capabilities bits that the tester supports transmitting at 6.0 Gbps with SCC disabled
☐ Force Train SNW to fail at 6.0 Gbps with SSC disabled

1. Select the ports that you want to change the speed for.
2. For instructions on filling out the main pane of the window, see “Speed” on page 127.
3. Click **OK** to apply the changes.

Toggle Display

This option toggles between a horizontal and vertical view of the ports.

Using Log Manager in Target Emulator

To access the Log Manager, click the **Log** tab at the bottom of the Target Emulator main window.

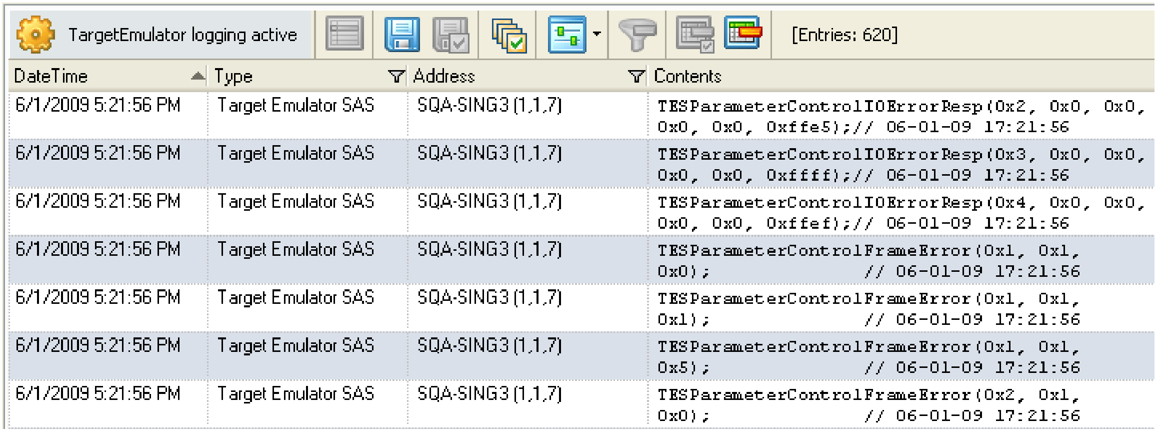
Target Emulator creates logs that report which operations have occurred and which ones encountered errors during execution. These logs are contained within the Log Manager. To access the Log Manager, select the Log tab at the bottom of the Maestro window.

The Log Manager shows the status of each operation and test executed.

➡ **Note:** The Ports Manager contains a small preview log pane at the bottom of the window for quick reference of log entries.

The Log Manager continues to list the logs and display them depending on the Log Manager options you have selected.

Figure 73: Log Manager



DateTime	Type	Address	Contents
6/1/2009 5:21:56 PM	Target Emulator SAS	SQA-SING3 (1,1,7)	TESParameterControlIOErrorResp(0x2, 0x0, 0x0, 0x0, 0x0, 0xffe5);// 06-01-09 17:21:56
6/1/2009 5:21:56 PM	Target Emulator SAS	SQA-SING3 (1,1,7)	TESParameterControlIOErrorResp(0x3, 0x0, 0x0, 0x0, 0x0, 0xffff);// 06-01-09 17:21:56
6/1/2009 5:21:56 PM	Target Emulator SAS	SQA-SING3 (1,1,7)	TESParameterControlIOErrorResp(0x4, 0x0, 0x0, 0x0, 0x0, 0xffef);// 06-01-09 17:21:56
6/1/2009 5:21:56 PM	Target Emulator SAS	SQA-SING3 (1,1,7)	TESParameterControlFrameError(0x1, 0x1, 0x0); // 06-01-09 17:21:56
6/1/2009 5:21:56 PM	Target Emulator SAS	SQA-SING3 (1,1,7)	TESParameterControlFrameError(0x1, 0x1, 0x1); // 06-01-09 17:21:56
6/1/2009 5:21:56 PM	Target Emulator SAS	SQA-SING3 (1,1,7)	TESParameterControlFrameError(0x1, 0x1, 0x5); // 06-01-09 17:21:56
6/1/2009 5:21:56 PM	Target Emulator SAS	SQA-SING3 (1,1,7)	TESParameterControlFrameError(0x2, 0x1, 0x0); // 06-01-09 17:21:56

The Filter icons next to the **Type** and **Address** column labels allow you to choose how you want to display device types, by protocol or All, and device addresses, by single address or All devices.

The following icons are displayed on the Log Manager menu bar:

Display Entry Contents

This button is not currently being used and is always disabled.

Save All Entries As

Lets you save all the logs in the Log Manager to a file.

Save selected entries as

Lets you save the selected logs to a file.

Select All

Selects all the logs in the Log Manager.

Options
Opens the Log Manager Options dialog where you can enter your preferences for the information you want displayed in the Log Manager (Figure 74).

Clear Filtering
Removes filtering by Type and Address and displays all the devices you have locked.

Clear Selected Entries
Deletes the log entry you have highlighted.

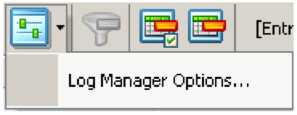
Clear All Entries
Deletes all the log entries in the Log Manager.

Setting Log Options

To set up default naming and locations of Target Emulator log files:

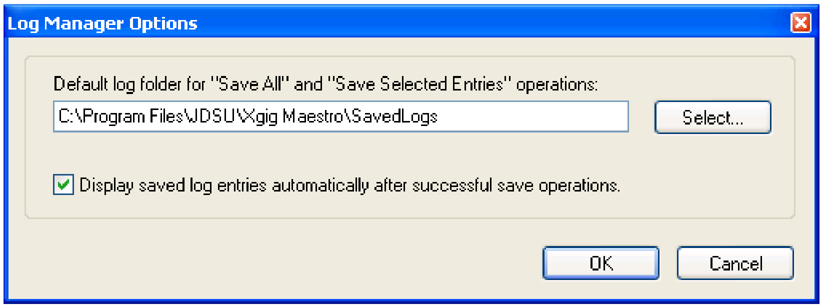
1 Click the **Log Manager Options** icon to open the drop-down menu, and select **Log Manager Options**.

Figure 74: Log Manager Options Menu



The Target Emulator **Log Manager Options** dialog is displayed. This dialog allows you to set up how you want the log files to be automatically named and the default location you want them saved to. You can also change the filename and folder when you save the log.

Figure 75: Log Options Dialog



Log Manager Options

Default log folder for "Save All" and "Save Selected Entries" operations:

C:\Program Files\JDSU\Xgig Maestro\SavedLogs

☑ Display saved log entries automatically after successful save operations.

2 Enter the path to the folder where you want to save log files.

You can browse to where you want to save the log file by clicking Select... .

3 Click **OK**.

4 To display saved log entries automatically after successful save operations, check the box at the bottom of the **Log Options** dialog.

The right-click menu in the Log Manager contains the same options as those shown in the Log Manager tool bar and detailed in this section.

Saving a Log

You can save a log from the Log Manager.

To save a log from the Log Manager as an HTML file:

1. Highlight the log(s) you want to save.
2. Click the **Save Selected Entries As** button.

   The **Save Log Manager Contents As** window is displayed. This window lets you name and save the log as an HTML file to the Saved Logs folder or to a location you prefer.

Customizing the Appearance of the Maestro/Target Emulator Main Window

You can move and rearrange the individual device tabs on your monitor screen as you prefer. You can also select the information you want to display in the Parameters Status table. In addition, you can select the size of displayed text.

Using the Window Menu

The **Window** menu allows you to arrange the window display:

Layout

Layout offers choices for rearranging the Maestro device tab windows. The Ports manager and Log Manager windows are not affected by this menu selection.

Internal Tabs

Allows you to restore the Maestro main window to its default format.

Internal MDI

Internal multiple document interface (MDI) lets you isolate each device tab as a separate window that you can activate by clicking anywhere on the window.

Show Hidden Windows

Restores any windows you have closed while using the Internal Tabs or Internal MDI display arrangement.

Arrange Icons

Arranges the icons for minimized windows at the bottom of the screen. If an open document window is at the bottom of the screen, some or all of the icons will be underneath this document window and will not be visible.

Cascade

Arranges all open windows in a cascade style from the top, left corner of the window.

Tile Horizontally

After selecting **Layout > Internal MDI,** this command opens and aligns the BERT, BERT Latency measurement, Jammer, and Generator device windows horizontally one over the other.

Tile Vertically

After selecting **Layout > Internal MDI**, this command opens and aligns the BERT, BERT Latency measurement, Jammer, Delay Emulator, Generator, Load Tester, and Target Emulator device windows vertically one next to the other. You can view the individual device windows by dragging the sides out to resize them.

Minimize All Windows

After selecting **Layout > Internal MDI**, this command minimizes all the open windows. You can restore each window, individually, by using the standard window controls on the header or select **Internal Tabs** to restore the Maestro main window to its default format.

The following options appear when you select **Layout > Internal Tabs**:.

BERT - Bit error rate testing

Brings the BERT device tab to the front as the active window

BERT - Latency measurement

Brings the BERT Latency measurement device tab to the front as the active window

Jammer

Brings the Jammer device tab to the front as the active window

Generator

Brings the Generator device tab to the front as the active window

Target Emulator

Brings the Target Emulator device tab to the front as the active window

Delay Emulator

Brings the Delay Emulator device tab to the front as the active window

Load Tester

Brings the Load Tester device tab to the front as the active window

Customizing the Parameters Status table in Target Emulator

You can change the appearance of the Parameters Status table by:

- Removing columns
- Resizing columns
- Changing the text size
- Displaying and hiding status table parameters

Removing columns

If you have a number of devices locked, you can remove a column that you do not need to view without unlocking the port or affecting the operation.

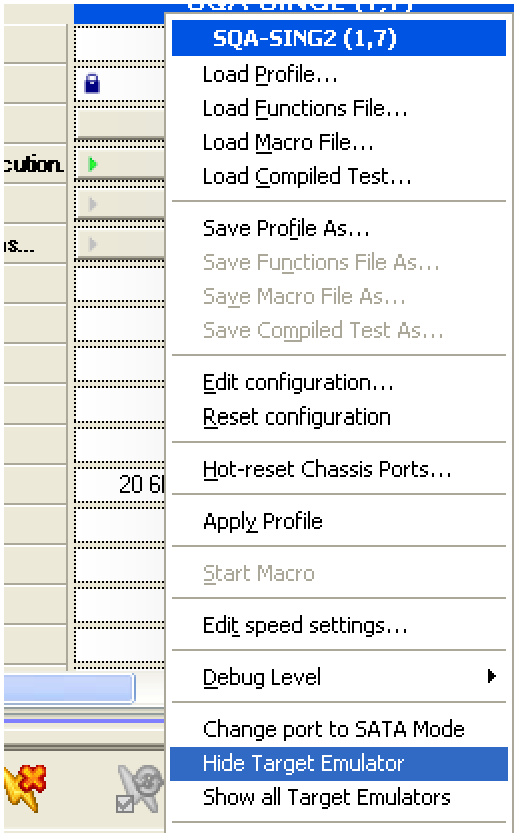
To remove a column:

1. Right-click on the specific column you want and open the Parameters context menu.
2. Select **Hide Target Emulator**(Figure 76).

   Or:

In the Ports Manager, clear the **Show** check box next to the device you want to remove. The column is removed, but the device is still locked (“in use”) and displayed in the Ports Manager.

Figure 76: Removing a Device Column



To replace the device column on the Parameters Status table:

>> Click the **Show** check box for the device in the Ports Manager.

Resizing columns

You can resize any of the columns by placing the mouse at the right edge at the top of a column, next to the chassis name. A resizing cursor appears. Hold down the mouse button, and drag the mouse to resize the column.

Changing Text Size

You can choose the text size you want the Parameters Status table or the Ports Manager to display.

➡ **Note:** You cannot change the text size in the Log Manager.

To change text size:

1 Open the context menu by clicking the right mouse button while the cursor is on the Parameters Status table or the Ports Manager.

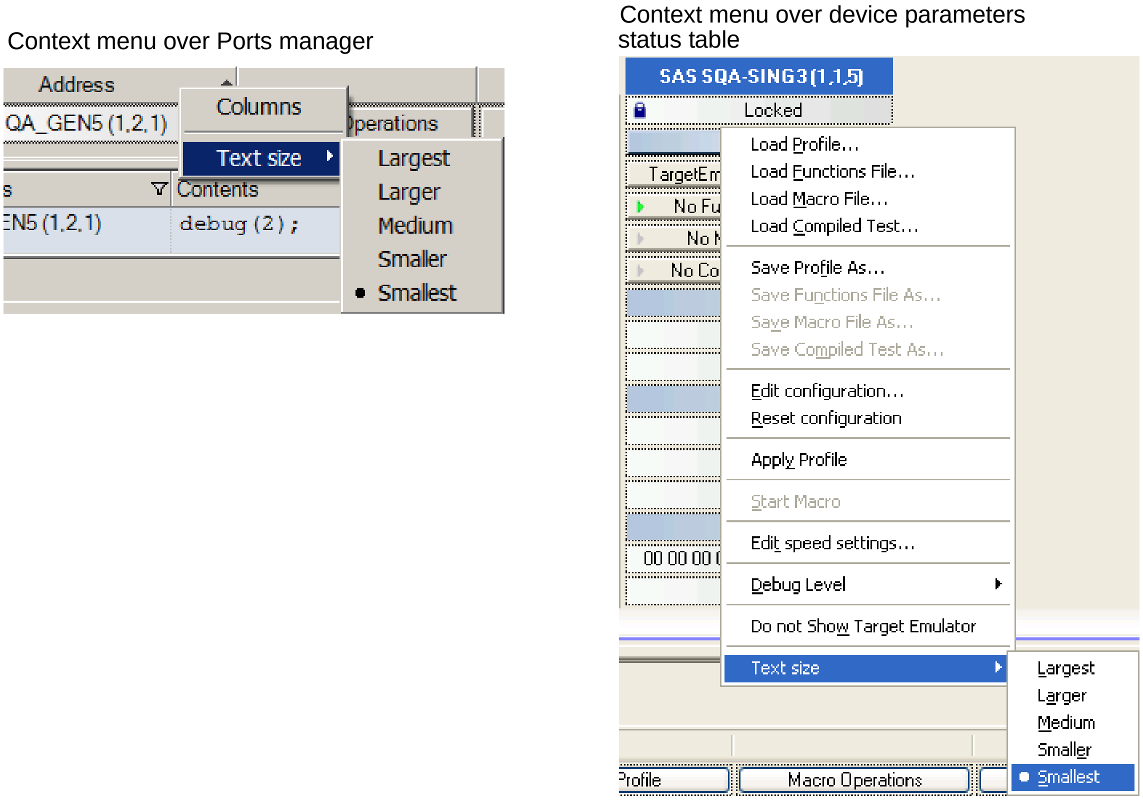
2 Select **Text size** (Figure 77).

You have five choices from which to select. Smallest is the default size.

3 Select the text size you want.

A bullet is displayed next to the current selection.

Figure 77: Text Size Menu



Context menu over Ports manager

Context menu over device parameters status table

Displaying and Hiding Parameters in Target Emulator

The **Parameters** tab next to the functions tabs allows you to select the parameters you want to display in the Parameters Status table. You can hide specific parameters to simplify the status tables and show only the information in which you are interested.

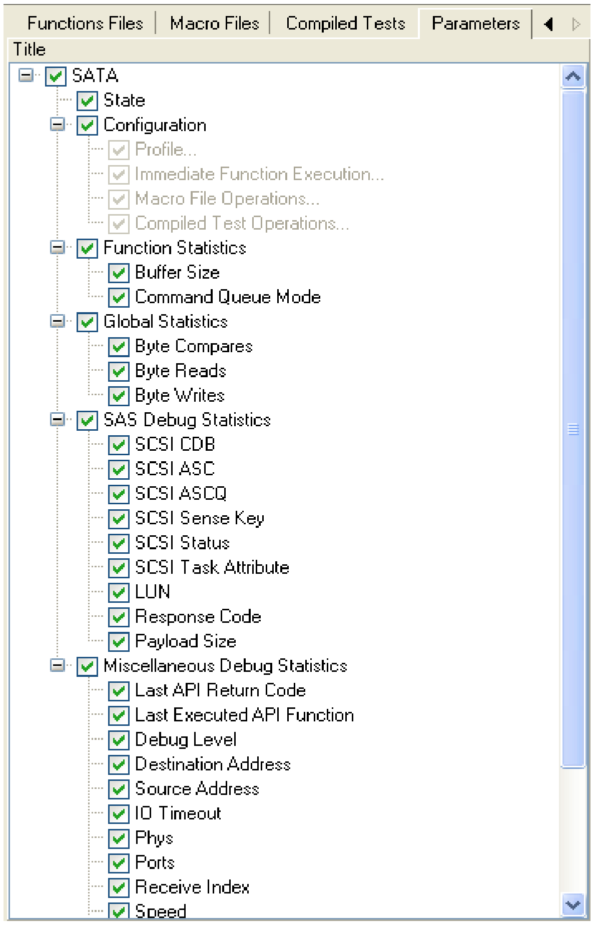
To display or hide parameters on the Parameters Status table:

1 Click the **Parameters** tab.

You can expand the categories by clicking the plus signs on the left.

The default displays all of the parameters.

Figure 78: Parameters Tab



- ☑ SATA
  - ☑ State
  - ☑ Configuration
    - ☑ Profile...
    - ☑ Immediate Function Execution...
    - ☑ Macro File Operations...
    - ☑ Compiled Test Operations...
  - ☑ Function Statistics
    - ☑ Buffer Size
    - ☑ Command Queue Mode
  - ☑ Global Statistics
    - ☑ Byte Compares
    - ☑ Byte Reads
    - ☑ Byte Writes
  - ☑ SAS Debug Statistics
    - ☑ SCSI CDB
    - ☑ SCSI ASC
    - ☑ SCSI ASCQ
    - ☑ SCSI Sense Key
    - ☑ SCSI Status
    - ☑ SCSI Task Attribute
    - ☑ LUN
    - ☑ Response Code
    - ☑ Payload Size
  - ☑ Miscellaneous Debug Statistics
    - ☑ Last API Return Code
    - ☑ Last Executed API Function
    - ☑ Debug Level
    - ☑ Destination Address
    - ☑ Source Address
    - ☑ IO Timeout
    - ☑ Phys
    - ☑ Ports
    - ☑ Receive Index
    - ☑ Speed

2 Click the check box next to a parameter to set or clear the check mark for that parameter.

Checked parameters are displayed in the Parameters Status table.

For details about each of the parameters in the Parameters Status table, see “Using the Parameters Status Table in Target Emulator” on page 121.

Chapter 8

Using Target Emulator

In this chapter:

After you have discovered and locked the Target Emulator ports you want to use and also set up your capturing and monitoring applications, such as Xgig Analyzer or Bus Doctor, you are ready to run the Xgig Target Emulator application. This chapter explains the process of managing and executing all of the configuration files for your device.

Loading a Configuration File

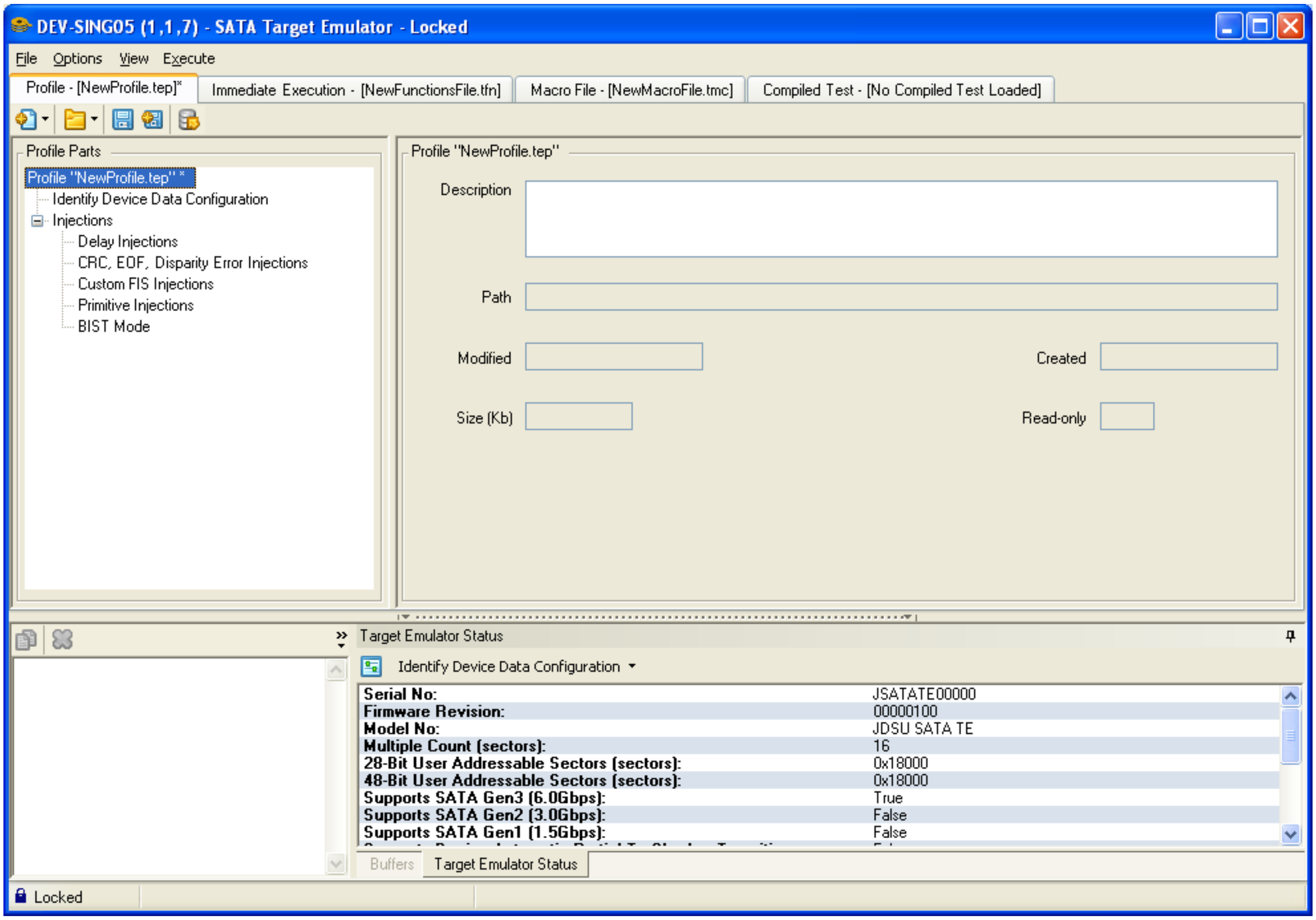
The Target Emulator Device window represents the device selected in the ports manager on the Target Emulator tab. This window contains tabs for profiles, functions, macros and compiled tests. This chapter discusses the functionality of each tab.

To access the Target Emulator Device window you need to load a profile file, functions file, macro file, or compiled test file to a device.

To load a functions, macro, or compiled test file to a port, locate the file you want to load in the Configuration window, then drag the file to the port, and drop it.

Launching the Target Emulator Device Window

Figure 79: Target Emulator Device Window


DEV-SING05 (1,1,7) - SATA Target Emulator - Locked
File Options View Execute
Profile - [NewProfile.tep]*
Immediate Execution - [NewFunctionsFile.tfn]
Macro File - [NewMacroFile.tmc]
Compiled Test - [No Compiled Test Loaded]
Profile Parts
Profile "NewProfile.tep" *
Identify Device Data Configuration
Injections
Delay Injections
CRC, EOF, Disparity Error Injections
Custom FIS Injections
Primitive Injections
BIST Mode
Profile "NewProfile.tep"
Description
Path
Modified
Created
Size (Kb)
Read-only
Target Emulator Status
Identify Device Data Configuration
Buffers
Target Emulator Status
Locked

Serial No:	JSATATE00000
Firmware Revision:	00000100
Model No:	JDSU SATA TE
Multiple Count (sectors):	16
28-Bit User Addressable Sectors (sectors):	0x18000
48-Bit User Addressable Sectors (sectors):	0x18000
Supports SATA Gen3 (6.0Gbps):	True
Supports SATA Gen2 (3.0Gbps):	False
Supports SATA Gen1 (1.5Gbps):	False

You can open the Target Emulator Device window (Figure 79) from either the Parameters status table or the Ports Manager.

To open the Target Emulator Device window for a device from the **Parameters Status table**, click the **Edit** Edit... button on the Parameters Status table.

The **SAS Target Emulator** window is displayed.

To open the Target Emulator Device window for a device from the **Ports Manager**:

1 Check the **Select** check box next to your device in the **Ports Manager**.

2 Click the **Macro Operations** button or the **Compiled Test Operations** button for the device.

The respective Operations window is displayed.

3 Click the **Edit** Edit button to open the Target Emulator Device window.

Once the window is open, select the tab you want to use.

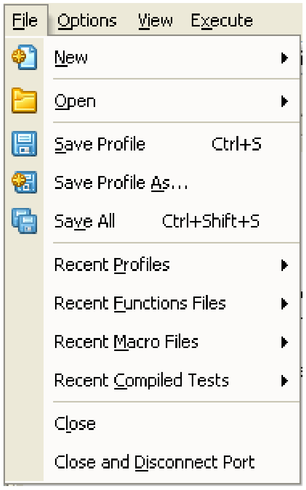
Target Emulator Device Menu Bar

The menu bar at the top of the Target Emulator Device window contains the following menus.

File

The File menu (Figure 80) allows you to create a new configuration or open an existing or recently-used configuration file for editing. You can also close the application from this menu.

Figure 80: File Menu



To create a new profile, macro or functions file:

>> Select **New**, then select:
- New Profile
- New Profile from Port
- New Functions File
- New Macro File

A blank file is opened on the tab for the file type you selected.

To open an existing profile, functions, macro, or compiled test file:

>> Select **Open**, then select:
- Open Profile
- Open Functions File
- Open Macro File
- Open Compiled Test

The file is opened on the tab for the file type you selected.

To save the currently displayed file:

>> Select **Save Macro file "..."**

To Save the currently displayed file as another name or to another location:

>> Select **Save Macro file "..." As**

To Save all open files:

>> Select **Save All**.

To open a recently-used functions, macro, or compiled test file:

1 Select one of the following:
- Recent Profiles
- Recent Macro Files
- Recent Functions Files
- Recent Compiled Test Files

2 Click on the file you want to open.

The file is opened on the tab for the file type you selected.

To close the Target Emulator Device window, click **Close**.

To close the Target Emulator Device window and disconnect the port for the device, click **Close and Disconnect Port**.

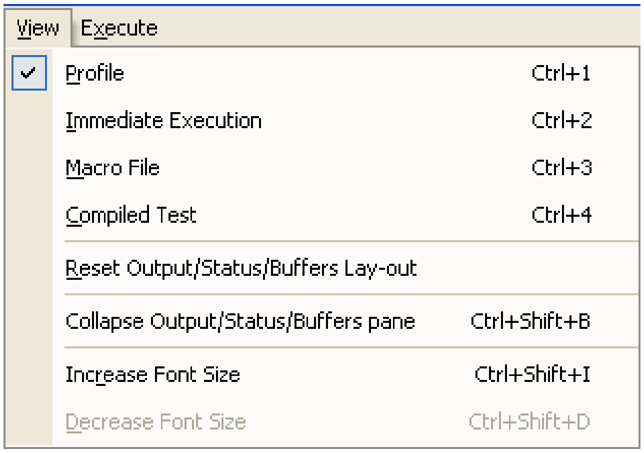
Options

The **Options** menu lets you select or unselect the option to confirm profile item deletion. When the **Confirm Profile Item Deletion** option is selected, a message box will verify first if you are really want to remove a profile like an error injection.

View

The View menu allows you to switch between tabs in the Target Emulator Device window by clicking the entry for the desired tab. This menu also allows you to collapse/expand the Function Browser pane, which is the left pane containing the function library on the left, or to collapse/expand the Output/Status/Buffers pane, which is the bottom pane. These two panes are not shown in the Compiled Test window.

Figure 81: View Menu

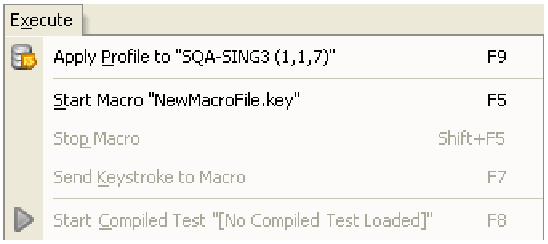

View
Execute
✓ Profile Ctrl+1
Immediate Execution Ctrl+2
Macro File Ctrl+3
Compiled Test Ctrl+4
Reset Output/Status/Buffers Lay-out
Collapse Output/Status/Buffers pane Ctrl+Shift+B
Increase Font Size Ctrl+Shift+I
Decrease Font Size Ctrl+Shift+D

Execute

The Execute menu provides the following choices for macros:

- **Apply Profile** applies the selected profile to the port.
- **Start Macro** executes the currently loaded macro file.
- **Stop Macro** stops the execution of a macro
- **Send Keystroke to Macro** simulates a keystroke to stop looping during macro execution. This selection is active only if the `loopk` function is invoked in your macro. Refer to the description for `loopk` in the function library.
- **Start Compiled Test** executes the currently loaded compiled test file.

Figure 82: Execute Menu


Execute
Apply Profile to "SQA-SING3 (1,1,7)" F9
Start Macro "NewMacroFile.key" F5
Stop Macro Shift+F5
Send Keystroke to Macro F7
Start Compiled Test "[No Compiled Test Loaded]" F8

Output/Status/Buffers Pane

This pane is located at the bottom of the **Profile**, **Immediate Execution**, **Macro File** and **Compiled Test** tabs. It allows you to set the debug level, view the output and status of executed functions and macros, edit buffers, and set status viewing options.

You can also click and drag on the Buffer/Status pane to move it to a different location, float the window, or dock them back to the TE Device Window.

Setting the Debug Level

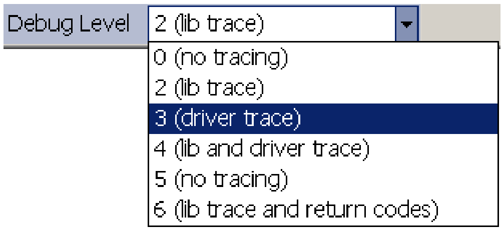
To view the debug level control, make sure the bottom pane of the window is expanded. If it is not, select **Expand Output/Status/Buffers pane** from the **View** menu. The debug control is located on the **Output/Status** tab.

The following debug levels are available:

- 0 (no tracing) does not display any traces
- 2 (lib trace) displays library traces
- 3 (driver trace) displays driver traces
- 4 (lib and driver trace) displays both library and driver traces
- 6 (lib trace and return codes) displays library traces and return codes

The default debug level is 2, which displays library traces only. To set the debug level, click the Debug Level drop-down menu, and select the appropriate debug level.

Figure 83: Setting the Debug Level



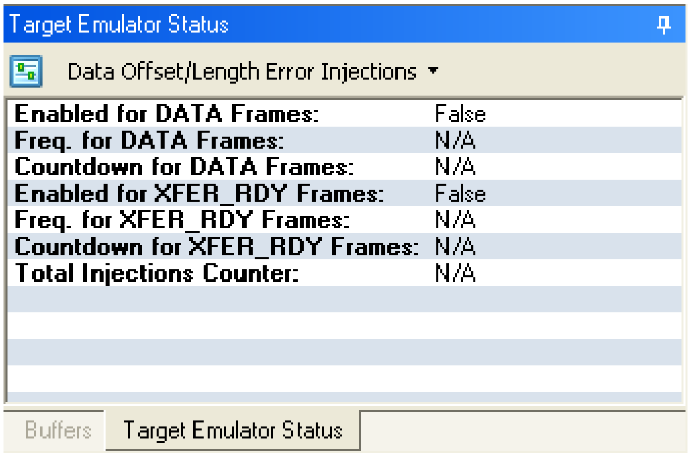
Debug Level 2 (lib trace)

Setting the Status Viewing Information

The status of the device selected in the Parameters Status table is displayed in the Status pane of the Target Emulator Device window.

To view the status of the device, make sure the bottom pane of the window is expanded. If it is not, select **Expand Output/Status/Buffers** pane from the **View** menu. The Status information is located on the **Target Emulator Status** tab (refer to Figure 84).

Figure 84: Status Information for Selected Device



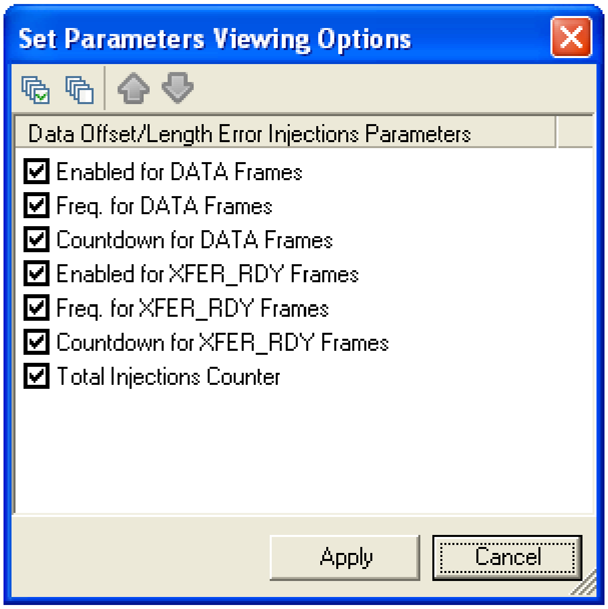
Target Emulator Status

Data Offset/Length Error Injections

Enabled for DATA Frames:	False
Freq. for DATA Frames:	N/A
Countdown for DATA Frames:	N/A
Enabled for XFER_RDY Frames:	False
Freq. for XFER_RDY Frames:	N/A
Countdown for XFER_RDY Frames:	N/A
Total Injections Counter:	N/A

The parameters listed in the Status pane correspond to those shown in the Parameters Status table for the device. For a definition of these categories, refer to “Using the Parameters Status Table in Target Emulator” on page 121.

Figure 85: Set Parameters Viewing Options Window



Set Parameters Viewing Options

Data Offset/Length Error Injections Parameters

- ☑ Enabled for DATA Frames
- ☑ Freq. for DATA Frames
- ☑ Countdown for DATA Frames
- ☑ Enabled for XFER_RDY Frames
- ☑ Freq. for XFER_RDY Frames
- ☑ Countdown for XFER_RDY Frames
- ☑ Total Injections Counter

To select which parameters you want to display in the Status pane:

1 Click the **Set Parameters Viewing Options** button.

2 The **Set Parameters Viewing Options** window appears.

3 Select which parameters you want to view in the Status pane.

4 Use the **Move Up**/**Move Down** buttons to arrange the order of the parameters.

5 Click **Apply**.

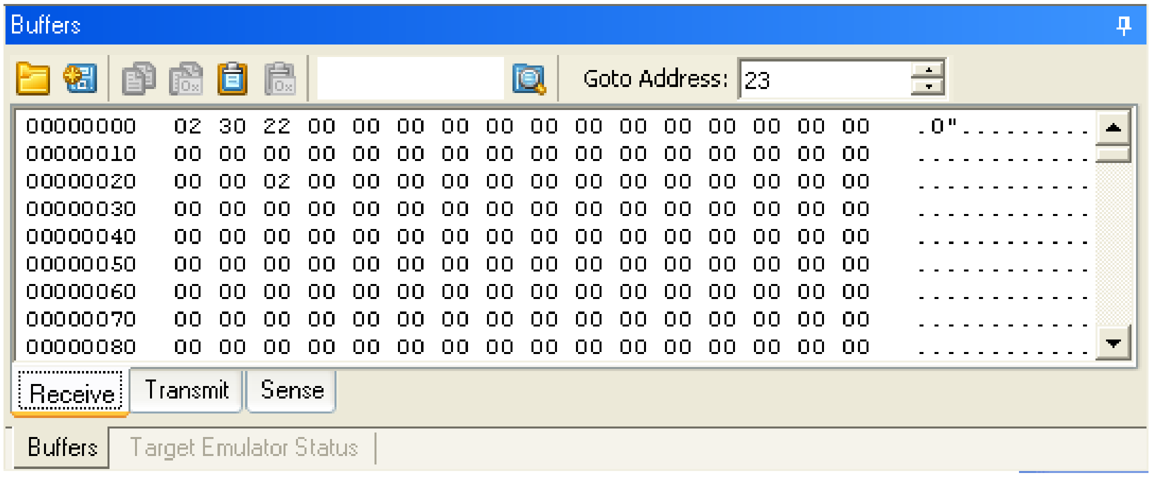
The information displayed on the Target Emulator Status is updated whenever a change in the profile is done and sent to the port.

Accessing the Buffer Editor

The Buffer Editor enables you to view and manipulate the transmit, receive, and sense buffers. A buffer is a `.bin` file that displays data used in SCSI data frames. SCSI data is sent from the transmit buffer and to the receive buffer during SCSI I/O operations. The sense buffer displays the sense data of the most recent commands containing that data. Refer to the `dmaset()`, `dmarst()`, `get_g_info("bufsz")`, `get_g_info("tx_ptr")`, and `get_g_info("rx_ptr")` functions for more information.

To access the buffer editor, open the Target Emulator Device Window. Then, click on the **Buffers** tab on the lower pane (refer to Figure 86).

Figure 86: Buffer Editor



```
00000000   02 30 22 00 00 00 00 00 00 00 00 00 00 00 00 00    .0".........
00000010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ............
00000020   00 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00    ............
00000030   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ............
00000040   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ............
00000050   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ............
00000060   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ............
00000070   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ............
00000080   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ............
```

Data is shown in both hexadecimal and ASCII.

Opening a Buffer File

To open a buffer file:

1. Select the tab for the type of buffer you want to open, specifically **Receive**, **Transmit**, or **Sense**.
2. Click the **Folder** 📁 icon in the top left corner of the pane.
3. Browse to the buffer file you want to open.
4. Click OK.

   The buffer file is displayed in the buffer pane.

Editing a Buffer

Both the hexadecimal and ASCII fields in the read and write buffers can be edited. The Address field indicates the selected position.

To edit a buffer file:

1. Move to the desired cell using the **GotoAddress** field or by typing text into the **Search** field.

2. Type an appropriate value. If you are editing a hexadecimal value, the ASCII value changes accordingly, and if you are editing an ASCII value, the hexadecimal value changes accordingly.
3. Use the **Copy** icon to copy an selection of the buffer, and then use the **Paste** icon to paste the selected contents into another buffer, into the current buffer, or into a text editor as ASCII.
4. Alternatively, you can use the **Copy as a hex string** icon to copy a selection of the buffer as a hex string and use the **Paste as a hex string** icon to paste the selected contents into another buffer, into the current buffer, or into a text editor as a hex sting.
5. You can also do the following:
    - Copy a section from another buffer file and paste it into the current file or visa versa using the right-click menu. This menu also contains the option to copy/paste as ASCII or as a hex string.
    - Paste another buffer file into the current file at an insertion point by using the right-click menu.
6. Save the buffer file by selecting the **Save** icon in the top, left corner of the pane. Saving a buffer, saves the data on all three tabs into a single file.

Profile Tab

After you have configured a Target Emulator device, you can use the Profile tab to create Target Emulator profiles and apply them to the locked Target Emulator ports.

Profile Action buttons

New Profile

There are two options when you create a profile. These options can be accessed by clicking the inverted triangle beside the button. Selecting **New Profile from Port** will create a new profile based on the profile currently loaded to the port. If you select **New Profile**, a new blank profile with all the default settings will be created.

Open Profile

To load a previously saved profile, click **Open Profile**. You can also click on the inverted triangle to show a list of recent profiles opened. Clicking the Open Profile button opens the **Open Target Emulator Profile** window where you can browse for, and select the profile to open.

Save Profile

To save changes to the current profile, click **Save Profile**.

Save Profile As

To create a new profile with the changes to the current profile, click **Save Profile As**. The **Save Target Emulator Profile As** window will open where you can enter a new name for the file and browse for the preferred location of the new profile.

Apply Profile to Port

Once you are done editing the profile, you can now apply it to the port. Click A**pply Profile to Port**.

Profile Parts Pane

This pane lists the parts of the profile. When you select a part name from the pane, the Parts Option pane will show the options available for that part.

Profile Parts in SAS Mode

The following parts are available when the locked port is in SAS Mode.

Profile

This part gives the details of the Profile file. You can enter a description for the current profile to better describe it.

Target Characteristics

The main function of Target emulator is to emulate a target storage device. You can set the device characteristics with these options.

Speed Negotiation

These options lets you set the speed settings to be used between the emulated target and the host or initiator. To know more about speed settings, refer to “Speed” on page 18.

Custom Commands

You can add, edit or delete custom SCSI commands for the profile from these options.

To add a custom command, click **Add Custom Command**. A new entry under the Custom Command pane will be added.

To customize that command, select it. The options on the right side of the pane will become available. Adjust the settings according to your test preferences.

You can add up to sixteen (16) custom commands.

Injections

Aside from creating a virtual storage device or target, Target Emulator also gives you the option to inject different types of errors into the responses sent to the host or initiator. This allows you to create a number of scenarios to fully test your devices.

The following are the different error injection types that can be configured into Target Emulator device profile’s responses.

Break, Hard Reset, Reset Seq. Injections

Inject Reset, Hard Reset, and Break sequences in frames sent by the target depending on the request command.

CRC, EOF, Disparity Error Injections

Inject CRC, EOF, or Disparity errors in frames sent by the target.

Data Miscompare Injections

Inject Data Miscompare errors in frames sent by the target.

Data Offset/Length Error Injections

Inject errors in data offset or data length fields of Data and XFER_RDY frames sent by the target.

Delay Injections

Insert delays for target responses (DATA, Response, or XFER_RDY frames) depending on the request.

Frame Field Replacements

Replace a portion of a frame sent by the target with a given data pattern.

Status and Sense Injections

Inject status and sense data in response frames sent by the target, depending on the response frames.

Profile Parts in SATA Mode

Identify Device Data Configuration

This part lets you customize the IDENTIFY DEVICE data of the Target Emulator Port. Some parameters of the IDENTIFY DEVICE data cannot be changed hence the options for these locked parameters are not displayed in the Parts Options Pane.

The following table lists modifiable Identify Device Data fields supported by the SATA Target Emulator:

Table 4: List of Supported Modifiable Identify Device Data Fields

Word	**Description**	**Default Value**
10-19	Serial Number *- 20 ASCII characters*	VSATATE00000
23-26	Firmware Revision *- 8 ASCII characters*	00000100
27-46	Model Number *- 40 ASCII characters*	VIAVI SATA TE
47	7:0 Multiple Count *- 0x00 = Reserved* *- 0x01 - 0x10 = Maximum number of logical sectors that shall be transferred per DRQ block on READ/WRITE MULTIPLE commands (0x01, 0x02, 0x04, 0x08 or 0x10 only)* *- 0x11 - 0xFF = Reserved*	0x10
60-61	Total Number of User Addressable Logical Sectors *- Maximum value is 0x0FFFFFFF*	0x00018000

Table 4: List of Supported Modifiable Identify Device Data Fields

Word	Description	Default Value
76	14 Supports Device Automatic Partial to Slumber transitions	0
	13 Supports Host Automatic Partial to Slumber transitions	0
	11 Supports Unload while NCQ commands outstanding	1
	9 Supports receipt of host-initiated power management requests	1
	8 Supports Native Command Queuing	1
	3 Supports Serial ATA Gen 3 signalling speed (6.0Gbps) *- The SATA Target Emulator attempts speed negotiation at 6.0Gbps, if this Bit is 1.*	1
	2 Supports Serial ATA Gen 2 signalling speed (3.0Gbps) *- The SATA Target Emulator attempts speed negotiation at 3.0Gbps, if this Bit is 1 and Word 76 Bit 3 is 0.*	0
	1 Supports Serial ATA Gen 1 signalling speed (1.5Gbps) *- The SATA Target Emulator attempts speed negotiation at 1.5Gbps, if this Bit is 1, both Word 76 Bit 3 and Word 76 Bit 2 are 0.*	0
78	3 Device supports initiating interface power management *- The SATA Target Emulator will only initiate interface power management when it is in STANDBY mode.* *- STANDBY mode can be achieved by issuing a Standby Immediate ATA command to the target emulator.* *- When initiating interface power management, the target emulator issues PMREQ_S primitives to the host.*	1
82	10 Host Protected Area feature set supported	1
83	10 48-bit Address feature set supported	1
84	13 IDLE IMMEDIATE with UNLOAD FEATURE supported	1
	6 WRITE DMA FUA EXT and WRITE MULTIPLE FUA EXT commands supported	1
100-103	Total Number of User Addressable Logical Sectors for 48-bit Address feature set *- Maximum value is 0x0000FFFFFFFFFFFF*	0x0000000000018000

Injections

Delay Injections

Inject delays for different FIS Types.

CRC, EOF, Disparity Injections

Inject CRC, EOF, or Disparity errors in FIS sent by the target.

Custom FIS Injections

Inject customized FIS or Frame Information Structure. A FIS is a group of Dwords that convey information between host and device as described previously. The boundaries of the FIS are defined by primitives. The primitives may be also be inserted to control the rate of the information flow. You can customize the FIS Frame from the Parts Options pane.

Primitive Injections

Injects SATA Primitives to FIS Frame. You can select the Primitive type to inject, how many of that primitive is injected, and the frequency.

BIST Mode

The Built In Self Test (BIST) Mode is intended for Inspection/Observation Testing, as well as support for conventional laboratory equipment, rather than for in-system automated testing. Using this part of the profile, you can inject the BIST Activate FIS used to place the receiver in one of several loopback modes. You can also set a user-defined loopback mode pattern definition.

Parts Options Pane

As you select the profile part on the Profile Parts pane, the options for that selected part will appear on the Parts Option Pane. To know the different parameters for the Target Emulator profile, refer to "Using the Parameters Status Table in Target Emulator" on page 121.

When you edit an option for the profile part, you can immediately apply that change into the port by clicking the **Apply Profile Part** button. The Parameter Status table of that port will be updated to reflect the change.

The following parts configures the Target characteristics for ports in SAS Mode:

Target Characteristics

Characteristics for the emulated target are set under this option.

Speed Negotiation

This part lets you set the speed negotiation options for the emulated target.

Custom Commands

This part lets you add, edit or delete custom SCSI commands for the profile from these options.

For the Port in SATA Mode, you have the following part for the device:

Identify Device Data Configuration

The options pane for this part is divided into three columns:

- **Word -** this column lists the Word numbers

- **Description -** this column lists the description for the Word. This column also allows you to customize the word. To do so, select the check box of the particular parameter to enable that parameter. You can customize the value for those parameters that have an optional text entry box.
- **Value -** The Hex values are arranged according to the correct endian format as stated in the ATA specifications (the least significant byte first for 16-bit values, and the least significant word first for greater or equal to 32-bit values) as reflected in the IDENTIFY DEVICE data.

Injections in SAS Mode

Break, Hard Reset, Reset Seq. Injections

There are three (3) tabs for this error type:

Break Sequence Errors

To add a break sequence error do the following:

1 Click the **Add Injection** button.

A new custom command will be added to that list. You can add up to 10 custom commands.

2 Click the custom command in the list to enable the options on the right side of the pane.

3 Edit the SCSI Command Code as needed.

4 Edit the Frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

To delete a break sequence error:

1 Select the custom command to delete from the list.

2 Click **Remove Injection**.

Hard Reset Sequence Errors

To add a hard reset sequence error do the following:

1 Click the **Add Injection** button.

A new custom command will be added to that list. You can add up to 10 custom commands.

2 Click the custom command in the list to enable the options on the right side of the pane.

3 Edit the SCSI Command Code as needed.

4 Edit the Frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

To delete a hard reset sequence error:

1 Select the custom command to delete from the list.

2 Click **Remove Injection**.

Reset Sequence Errors

To add a reset sequence error do the following:

1. Click the **Add Injection** button.

   A new custom command will be added to that list. You can add up to 10 custom commands.
2. Click the custom command in the list to enable the options on the right side of the pane.
3. Edit the SCSI Command Code as needed.
4. Edit the Frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

To delete a reset sequence error:

1. Select the custom command to delete from the list.
2. Click **Remove Injection** button.

CRC, EOF, Disparity Error Injections

You can insert an CRC, EOF, or Disparity Error injection for each of the request frame types.

To insert CRC errors for any of the request frames (Response, DATA, or XFER_RDY frame):

1. Select **Insert CRC Errors** check box under the **Response Frame Errors**, **DATA Frame Errors,** or **XFER_RDY Frame Errors** pane.
2. Edit the frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

To insert EOF errors for any of the request frames (Response, DATA, or XFER_RDY frame):

1. Select **Insert EOF Errors** check box under the **Response Frame Errors**, **DATA Frame Errors,** or **XFER_RDY Frame Errors** pane.
2. Edit the frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

To insert Disparity errors for any of the request frames (Response, DATA, or XFER_RDY frame):

1. Select **Insert Disparity Errors** check box under the **Response Frame Errors**, **DATA Frame Errors,** or **XFER_RDY Frame Errors** pane.
2. Edit the frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

Data Miscompare Injections

You can set Target Emulator to insert data miscompare errors to the frames sent by the target.

To insert data miscompare errors:

1. Select **Enable data miscompare error injection** check box.

2 Set the byte range where the original data will be replaced by setting the start and end bytes of the range.

3 Enter the data pattern that will replace the original data.

4 Enter the frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

Data Offset/Length Error Injections

You can set Target Emulator to inject errors in data offset or data length fields of Data and XFER_RDY frames sent by the target.

To have a DATA frame error injection:

1 Select **Enable data offset and data length error injection** check box in the **DATA Frame Error Injection** pane.

2 Set the value of the “Data Offset” field in the frame header.

3 Set the value of the data payload length.

4 Enter the frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

To have an XFER_RDY frame error injection:

1 Select **Enable data offset and data length error injection** check box in the **XFER_RDY Frame Error Injection** pane.

2 Set the value of the “Data Offset” field in the frame header.

3 Set the value of the data payload length.

4 Enter the frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

Delay Injections

You can insert delays for target responses (DATA, Response, or XFER_RDY frames) depending on the request.

There are three (3) tabs for this error type:

Delays for Response Frames

To add a custom command with delays, do the following:

1 Click the **Add Injection** button.

A new custom command will be added to that list. You can add up to 10 custom commands.

2 Click the custom command in the list to enable the options on the right side of the pane.

3 Edit the SCSI Command Code as needed.

4 Enter the delay time (in milliseconds).

5 Edit the Frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

To delete the delay injection:

1 Select the command with the delay injection to delete.

2 Click **Remove Injection**.

Delays for DATA Frames

To add a custom command with delays, do the following:

1 Click the **Add Injection** button.

A new custom command will be added to that list. You can add up to 10 custom commands.

2 Click the custom command in the list to enable the options on the right side of the pane.

3 Edit the SCSI Command Code as needed.

4 Enter the delay time (in milliseconds).

5 Edit the Frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

To delete the delay injection:

1 Select the command with the delay injection to delete.

2 Click **Remove Injection**.

Delays for XFER_RDY Frames

1 Click the **Add Injection** button.

A new custom command will be added to that list. You can add up to 10 custom commands.

2 Click the custom command in the list to enable the options on the right side of the pane.

3 Edit the SCSI Command Code as needed.

4 Enter the delay time (in milliseconds).

5 Edit the Frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

To delete the delay injection:

1 Select the command with the delay injection to delete.

2 Click **Remove Injection**.

Frame Field Replacements

Frame Field replacement error injection replaces a portion of a frame sent by the target with a given data pattern.

You can set Frame Field replacement injections for Response Frame Field, DATA Frame Payload, and XFER_RDY Frame Field.

To have Frame Field replacement injections for Response Frames:

1. Select **Enable frame field replacement** check box under the **Response Frame Field Replacement**.
2. Set the number of bytes to replace.
3. Set the starting byte number to replace.
4. Enter the pattern to be used for replacement.
5. Set the frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

To have Frame Field replacement injections for DATA Frames:

1. Select **Enable frame payload replacement** check box under the **DATA Frame Payload Replacement**.
2. Set the number of payload bytes to replace.
3. Set the starting payload byte number to replace.
4. Enter the pattern to be used for replacement.
5. Set the frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

To have Frame Field replacement injections for XFER_RDY Frames:

1. Select **Enable frame field replacement** check box under the **XFER_RDY Frame Field Replacement**.
2. Set the number of bytes to replace.
3. Set the starting byte number to replace.
4. Enter the pattern to be used for replacement.
5. Set the frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

Status and Sense Injections

You can inject status and sense data in response frames sent by the target, depending on the response frames.

To add a custom command with status and sense error injections, do the following:

1. Click the **Add Injection** button.

   A new custom command will be added to that list. You can add up to 10 custom commands.
2. Click the custom command in the list to enable the options on the right side of the pane.
3. Edit the SCSI Command Code as needed.
4. Enter the Response Status.
5. Enter the Sense Key.
6. Set the Sense Data Length
7. Set the ASC.

8 Set the ASCQ.

9 Edit the Frequency, or the number of times the request frames are detected before the requested changes are injected into the frame.

To delete the delay injection:

1 Select the command with the delay injection to delete.

2 Click **Remove Injection**.

When you edit an option for the profile part, you can immediately apply that change into the port by clicking the **Apply Profile Part** button. The Parameter Status table of that port will be updated to reflect the change.

Injections in SATA Mode

Delay Injection

You can inject delays on the transmitting frame.

To set delay injections:

1 Select the FIS type from the **Delay Injections pane**.

2 Specify how long the delay is in the Delay field. You can also use the up or down buttons to adjust the value.

3 Specify how frequent the delays will be injected in the **Frequency** field.

Example: *If you set a delay time of* 5 *seconds and a frequency of* 4*, a delay of* 5 seconds *will be injected every **4th frame.***

4 Click **Apply Profile Part** to send the changes to the port. The Target Emulator Status will be updated.

CRC, EOF, Disparity Error Injections

You can insert a CRC, EOF, or Disparity Error injection for each of the FIS types.

To insert a CRC, EOF or Disparity Error for a FIS Type:

1 Select the error type under the specific FIS Errors pane.

2 Specify the number of request frames to detect before emitting a modified frame.

Note that only one type of error is allowed per FIS type. Once you select one, the other error types are disabled.

3 Click **Apply Profile Part** to send the changes to the port. The Target Emulator Status will be updated.

Custom FIS Injections

You can modify a particular FIS with Customized FIS contents.

To create a customized FIS to be injected after a specific FIS type:

1 Select the FIS Type from the **Custom FIS Injections** pane.

2 Specify the size of the custom FIS in the Custom FIS Size field.

3 Set the frequency of injection for the custom FIS.

4 Customize the FIS frame by directly editing the Hex values of the DWords in the **Custom FIS Frame box**.

5 Click **Apply Profile Part** to send the changes to the port. The Target Emulator Status will be updated.

Primitive Injections

You can control the injection of primitives.

To inject SATA primitives:

1 Select the FIS type where SATA primitives will be injected.

2 Select the primitive type from the **Primitive** drop-down menu.

3 Set the number of the specified primitive to be injected in the **Primitive Count** field.

4 Specify the number of FIS frame occurrence of the specified FIS type before injecting the primitive in the **Frequency** field.

Example:

You selected **0x46 Data FIS** from the Primitive Injections pane, selected a primitive type of **PMNAK**, specified a primitive count of **16** and frequency of **2**. This means that for every 2 blocks of 0x46 Data FIS frames transmitted, 16 PMNAK primitives will be injected after the transmission of the second block of 0x46 Data FIS.

5 Click **Apply Profile Part** to send the changes to the port. The Target Emulator Status will be updated.

BIST Mode

You can send out the BIST Activate FIS with your customized parameters. The BIST Activate FIS is transmitted to tell the Host to initiate a Built In Self Test (BIST).

To customize the BIST Activate FIS:

1 Select the pattern definition from the **Patter Definition** drop-down menu. This parameter defines the Pattern Definition byte in the BIST Activate FIS frame. If you select User defined BIST mode, you can set the 1 byte hex value of the pattern definition in the **Value** field.

2 Enter a 4 byte hex value for **Data 1** and **Data 2** field. The **Data 1** and **Data 2** parameters define the data information used to determine what pattern is transmitted as a result of the BIST Activate FIS.

3 Click **Apply Profile Part** to send the changes to the port. The Target Emulator Status will be updated.

Target Emulator can put DUT in Built-In Self Test (BIST) mode by sending out the BIST Activate FIS but it does not support and won't participate in any form of BIST mode activities.

Immediate Execution Tab

After you have configured a Generator device, you can use the Immediate Execute tab to execute individual functions, or to create, edit, and execute functions files.

The **Immediate Execution** window is comprised of two panels. The left panel is the contents pane. It lists the functions present in the functions file. See “Creating a New Functions File” on page 188.

The right pane is the function library. Each of the functions in the library are represented in the API Help System on the product USB drive.The functions in the library are listed in the contents pane by name; current parameters in hexadecimal; and the parameter types: b (byte), d (dword), q (quad), s (string parameter), w (word), and o (optional parameter), which currently cannot be edited or added; and the function category.

You can sort the functions in either pane in the following ways:

- Alphabetically in ascending or descending order by toggling the Function Name heading.
- By category with the categories alphabetically in ascending or descending order by toggling the Function Type heading.
- By parameter types with the parameter types listed by number of parameters from most to least or from least to most by toggling the Parameter heading.

Click the **Clear Filter** icon to remove the filters you have applied by typing text into the search field and revert to viewing all functions in the functions library.

To view the details of the functions in the library or the contents pane, click the **Show/Hide Function Info** button at the top of the pane. The details for each function correspond to an entry in the Generator API Help available from the Help menu in the Maestro main window. To display the contents of the information pane for the selected function in a popup window, click the **Display Help Contents in Popup Window** button.

As you begin to click various functions in the function library or the contents pane, while you have the Show Function Info pane open, you create a history of the functions you’ve selected. You can navigate back and forth through the functions you’ve selected by using the **Go Back** and **Go Forward** buttons at the top of the Function Information pane.You can execute a function from the Immediate Execution tab of the Generator Device window or from the Parameters Status table in the Maestro window.

Editing Function Parameters

To edit a function’s parameters inline:

1. Right-click the function, and select Edit Parameters Inline.
2. Type directly in the parameters field to edit inline, or click the arrow at the end of the field to open a drop-down window.

   This window is the same as the Popup Parameters Editor.

To edit a function’s parameters using the Popup Parameters Editor:

1. Right-click the function, and select Popup Parameters Editor.

2 The Popup Parameters Editor appears wit the following options in the menu:

- Apply and execute
- Apply and close
- Discard and close
- Reset parameter values
- Show parameters info

Depending on the function, the following parameters may be available for editing:

Display - Displays the values in hexadecimal or decimal, depending on the field selected. Use the drop-down menu to change the display.

Byte - Byte- or bit-specific data. A byte is equal to 8 bits.

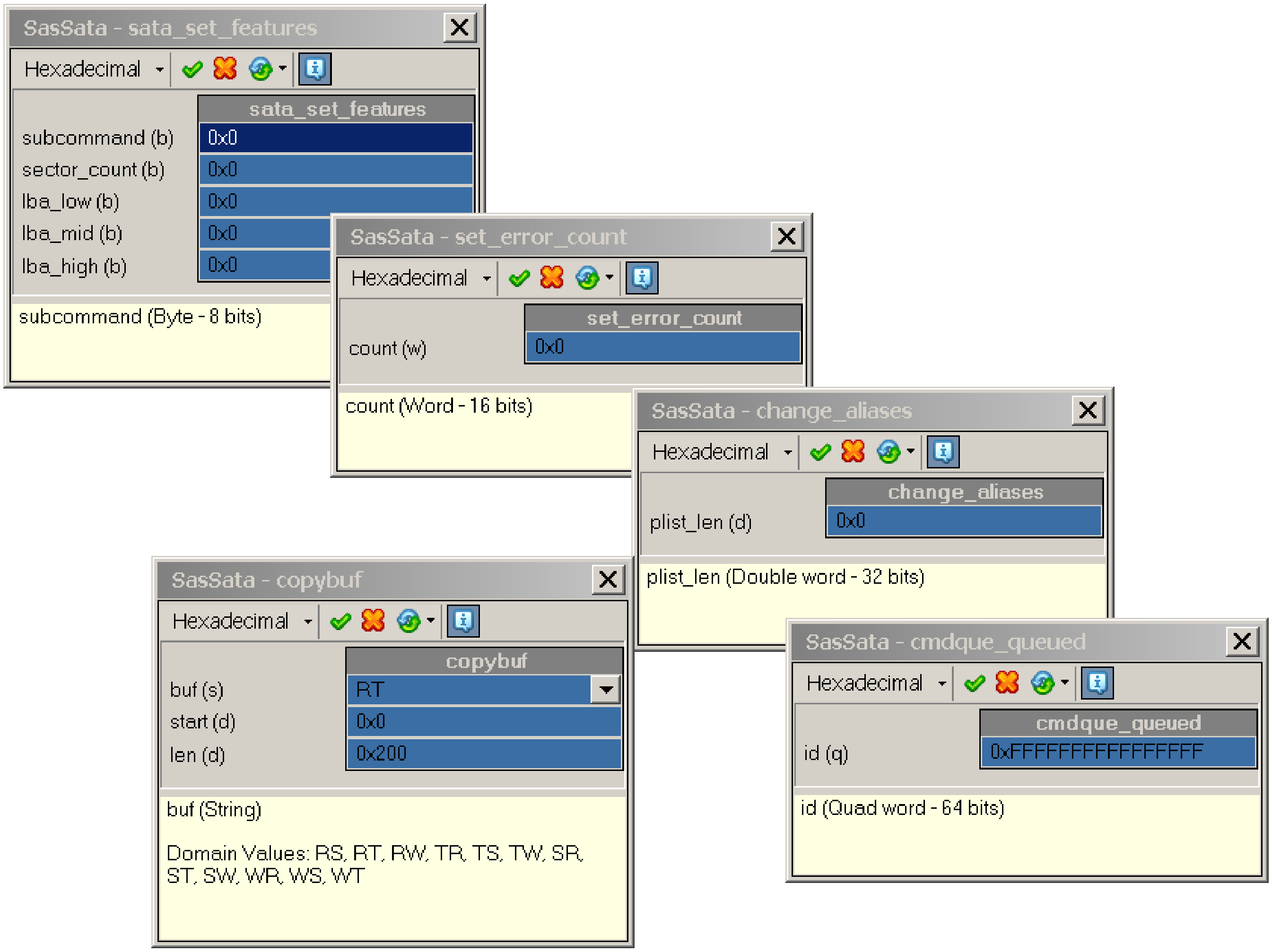
Word - Word or short integer data. A word is equal to 16 bits.

Double - Double word or long integer data. A double word is equal to 32 bits.

Quad - A quad is equal to 64 bits or 4 words.

String - Constant text or string data. Enter data without quotation marks.

Figure 87: Popup Parameters Editor Window



	sata_set_features
subcommand (b)	0x0
sector_count (b)	0x0
lba_low (b)	0x0
lba_mid (b)	0x0
lba_high (b)	0x0

subcommand (Byte - 8 bits)

	set_error_count
count (w)	0x0

count (Word - 16 bits)

	change_aliases
plist_len (d)	0x0

plist_len (Double word - 32 bits)

	copybuf
buf (s)	RT
start (d)	0x0
len (d)	0x200

buf (String)

Domain Values: RS, RT, RW, TR, TS, TW, SR, ST, SW, WR, WS, WT

	cmdque_queued
id (q)	0xFFFFFFFFFFFFFFFF

id (Quad word - 64 bits)

Executing a Function on a Generator Device

You can execute an individual function from the Contents pane or from the Function Library.

To execute a function from the Generator Device window:

1 Locate the function you want to execute in the function library or contents pane of the window by using the scroll bar or by typing the name of the function into the **Search Function** field.

2 Set the parameters as necessary for the selected function. The right-click menu provides editing options. These options are described in “Editing Function Parameters”.

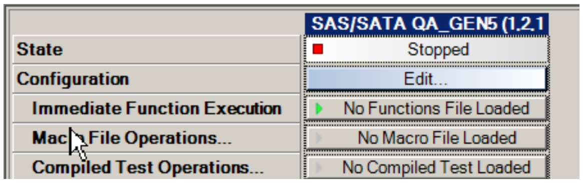
3 Click the **Start** ▷ button to execute the function.

You can also perform immediate function execution from the Parameter status table in the Maestro window.

To execute a function from the Parameters Status table:

1 Click the **Start** ▷ button on the Parameters Status table next to the Immediate Function Execution field.

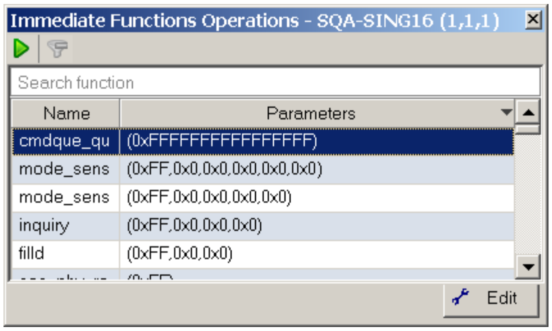
Figure 88: Parameters Status table Start Function Button



	SAS/SATA QA_GEN5 (1,2,1
State	Stopped
Configuration	Edit...
Immediate Function Execution	No Functions File Loaded
Macro File Operations...	No Macro File Loaded
Compiled Test Operations...	No Compiled Test Loaded

The Immediate Functions Operations window is displayed.

Figure 89: Immediate Functions Operations Window



Immediate Functions Operations - SQA-SING16 (1,1,1)

Search function

Name	Parameters
cmdque_qu	(0xFFFFFFFFFFFFFFFF)
mode_sens	(0xFF,0x0,0x0,0x0,0x0,0x0)
mode_sens	(0xFF,0x0,0x0,0x0,0x0)
inquiry	(0xFF,0x0,0x0,0x0)
filld	(0xFF,0x0,0x0)

2 Locate the function you want to execute by using the scroll bar or by typing the name of the function into the **Search Function** field.

3 Set the parameters as necessary for the selected function. The right-click menu provides editing options. These options are described in "Editing Function Parameters".

4 Click the **Start** button to execute the function.

Opening a Functions File

To open a recently used Xgig Generator functions file:

1 Click the arrow next to the **Open Functions file** button in the menu bar.

2 Click the functions file you want to open.

An **Immediate Execution** window appears with the functions file loaded.

To browse to a functions file, and open it:

1 Click the **Open Functions file** button in the menu bar.

2 Browse to the functions file you want to open.

3 Click the **Open** button.

An **Immediate Execution** window appears with the functions file.

Creating a New Functions File

A functions file is a `.gfn` file that contains a list of functions. You can use a functions file as a repository to group a specified set of functions with a certain set of parameters. You can also use a functions file to define and maintain the order of a group of functions for use in a macro file. In addition, you can select, edit parameters for, and execute individual functions from a function file.

To create a new macro or functions file using the **File** menu:

>> Select **New**, then select **Functions File**

To create a new functions file from the Contents pane, click the **New Generator Functions File** icon in the Contents pane menu bar.

A blank file is opened on the tab.

Adding Functions to a Functions File

To add a function from the function library to the functions file:

1. Click the index position in the Contents pane where you want to add a function.
2. Click the function in the function library that you want to add to the functions file.
3. Click the **Send Selected Function to Functions File** button.

   The function appears in the Contents pane at the selected index position.

> ➡ **Note:** A convenient shortcut for adding functions to a file is to hold down the Ctrl button on your keyboard, then click and drag the function file to the Contents pane.

4. Click the Save or Save as button at the top of the window to save the functions file.
5. Click the Done button to close the window.

Moving Functions within a Functions File

To move a function up or down in within the functions file:

1. Click the function you want to move.
2. Click the **Move Up**/**Move Down** buttons at the top of the Contents pane.

To remove a function you have added to your functions file:

1. Click the function you want to remove from the file.
2. Click the **Delete Selected User Function** button at the top of the Contents pane.

You can also interchange one function with another, move a function for another index location, or copy a function to another index location by right-clicking the function and selecting the appropriate action.

Saving a Functions File

To save a functions file, click the **Save** button.

To save the file as another name or to another location:

1. Click the **Save As** button.

The **Save**/**Save As** dialog appears.

➡ **Note:** If you want to use the standard Windows file dialogs instead of the dialogs customized for Generator, select **Use Standard Windows File Dialogs** from the **Options** menu in the Generator Device window menu bar.

2 Navigate to the location where you want to save the file.

3 Type a name for the file.

4 Click **Save.**

5 Click the **Done** button to close the window. If you make changes to the file and then click try to close the window, you will receive prompt asking you if you want to save the file before closing it.

Macro File Tab

Opening a Macro File

To open a recently used Xgig Generator macro file:

1 Click the arrow next to the **Open Macro file** button in the menu bar.

2 Click the macro file you want to open.

A macro window appears with the macro loaded.

To browse to a macro file and open it:

1 Click the **Open Macro file** button in the menu bar.

2 Browse to the macro file you want to open.

3 Click the **Open** button.

A macro window appears with the macro loaded.

Editing a Macro File

The macro editor enables you to easily create macros for automating a test sequence. Macros are ideal for non programmers because they are quick and easy to create and don’t require a C compiler or knowledge of C programming. The macro language, however, has limitations. You can’t write a macro test that takes pointers or addresses as function parameters.

A macro file is a `.key` file that can be created with any ASCII editor or by using the Macro File Editor. A macro file can contain up to 62 user-defined macro tests (0–9, A–Z, and a–z). Each macro test contains a sequence of functions to be called upon when invoking the macro test. In addition to functions, macro tests can also include variables, branches, jumps, loops, and macro functions.

➡ **Note:** This section pertains to individual macro files, which can be created and edited independently of a device. You do not need to lock ports to perform these actions. The Macro Editor window described in this chapter is separate from the Target Emulator Device window, which allows a user to manage and execute all configuration files for a selected device from a central window.

Creating a New Macro File

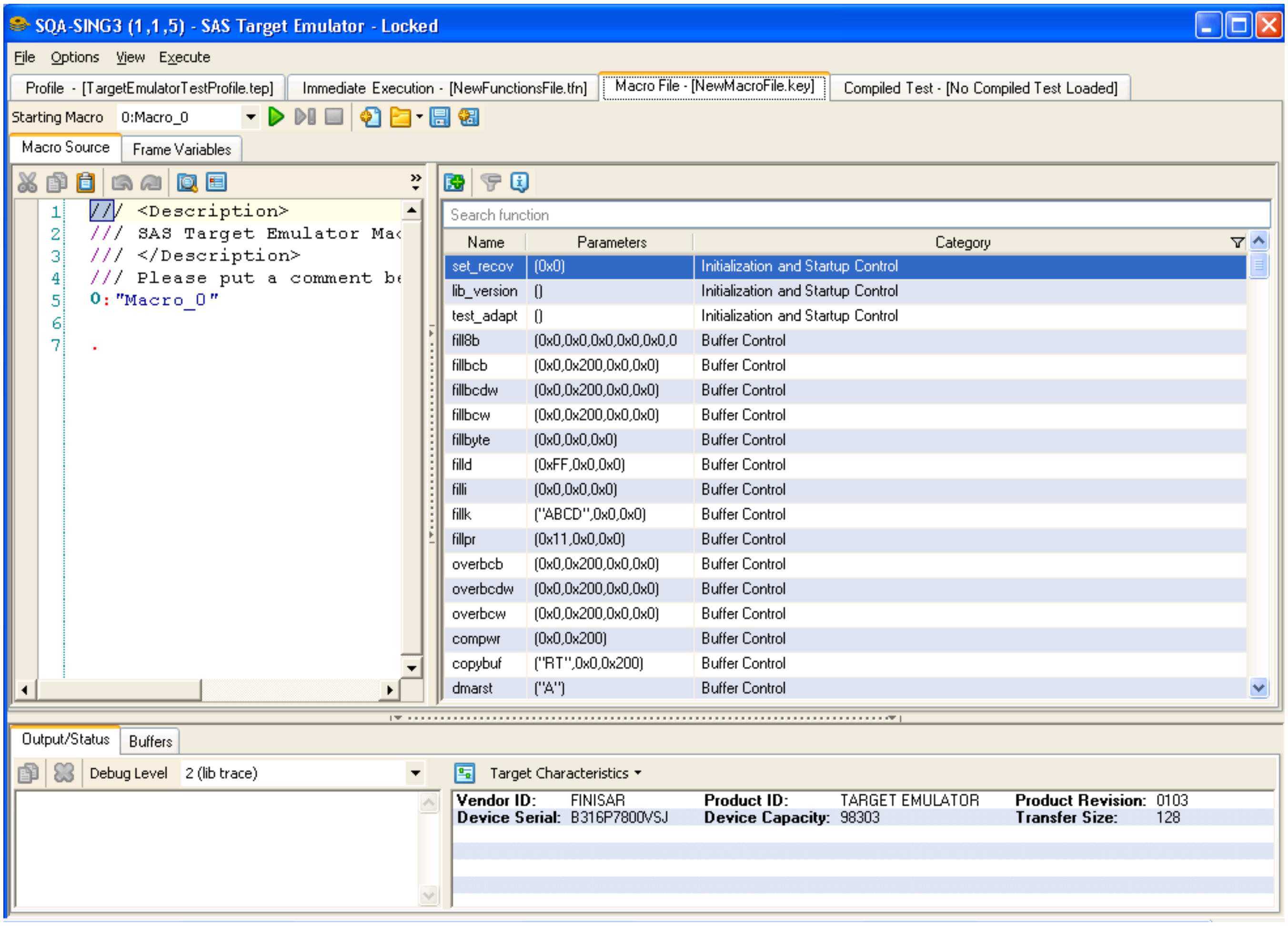
To create a new Xgig Generator Macro file:

1 Select the Macro Files tab in the Configuration window.

2 Click the **New Macro File** icon in the Configuration menu.

A new Macro Editor window appears.

Figure 90: Macro File Editor Window with Macro File Loaded


SQA-SING3 (1,1,5) - SAS Target Emulator - Locked
File
Options
View
Execute
Profile - [TargetEmulatorTestProfile.tep]
Immediate Execution - [NewFunctionsFile.tfn]
Macro File - [NewMacroFile.key]
Compiled Test - [No Compiled Test Loaded]
Macro Source
Frame Variables
Search function
Output/Status
Buffers
Target Characteristics

Starting Macro: 0:Macro_0

```
/// <Description>
/// SAS Target Emulator Mac
/// </Description>
/// Please put a comment be
0:"Macro_0"

.
```

Name	Parameters	Category
set_recov	(0x0)	Initialization and Startup Control
lib_version	()	Initialization and Startup Control
test_adapt	()	Initialization and Startup Control
fill8b	(0x0,0x0,0x0,0x0,0x0,0	Buffer Control
fillbcb	(0x0,0x200,0x0,0x0)	Buffer Control
fillbcdw	(0x0,0x200,0x0,0x0)	Buffer Control
fillbcw	(0x0,0x200,0x0,0x0)	Buffer Control
fillbyte	(0x0,0x0,0x0)	Buffer Control
filld	(0xFF,0x0,0x0)	Buffer Control
filli	(0x0,0x0,0x0)	Buffer Control
fillk	("ABCD",0x0,0x0)	Buffer Control
fillpr	(0x11,0x0,0x0)	Buffer Control
overbcb	(0x0,0x200,0x0,0x0)	Buffer Control
overbcdw	(0x0,0x200,0x0,0x0)	Buffer Control
overbcw	(0x0,0x200,0x0,0x0)	Buffer Control
compwr	(0x0,0x200)	Buffer Control
copybuf	("RT",0x0,0x200)	Buffer Control
dmarst	("A")	Buffer Control

Debug Level: 2 (lib trace)

Vendor ID:	FINISAR	**Product ID:**	TARGET EMULATOR	**Product Revision:**	0103
Device Serial:	B316P7800VSJ	**Device Capacity:**	98303	**Transfer Size:**	128

Macro File Structure

When creating macro tests, you must adhere strictly to the correct syntax. Extraneous spaces are generally not allowed. Numeric variables are decimal values unless they are preceded by 0x, in which case, they are hexadecimal.

In a new macro file, the macro content begins with description tags. This is a placeholder for the description of your file. The description you type here will appear in the Details pane when the file is selected in the Macro Files tab in Target Emulator.

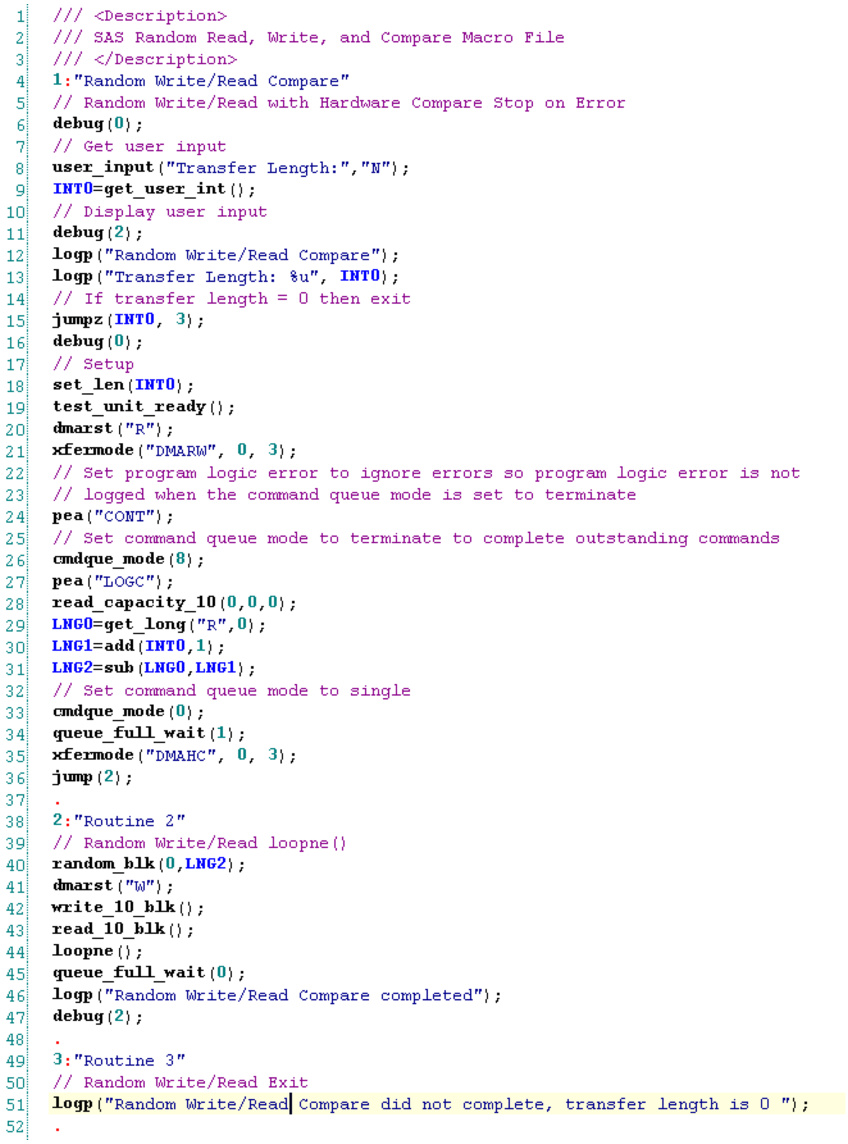
The list of functions in a macro always starts with a number (default = 0) followed by a red colon and ends with a red period. When you open a new macro, the beginning of the macro is indicated by a “0:”, and the end is indicated by a “.” on a new line as shown in the figure above.

You can either choose to build your macro as one string of functions, or you can separate it into routines. Essentially, each routine is a complete macro, beginning and ending with the symbols listed above. Each routine should be numbered sequentially. As mentioned above, the default number for the beginning of a macro is zero. You can change this number to one or another number if you choose.

Each macro or routine needs a name. The name should be enclosed in quotation marks and be placed immediately after the red colon indicating the beginning of the macro (no space).

Begin populating your macro with functions. Place a semicolon after each function. Place comments after // on a new line. A single comment that extends over multiple lines in the macro file must have a comment indicator // at the beginning of each line. The following figure shows a completed macro containing three routines.

Figure 91: Sample Macro File Contents



```
/// <Description>
/// SAS Random Read, Write, and Compare Macro File
/// </Description>
1:"Random Write/Read Compare"
// Random Write/Read with Hardware Compare Stop on Error
debug(0);
// Get user input
user_input("Transfer Length:","N");
INT0=get_user_int();
// Display user input
debug(2);
logp("Random Write/Read Compare");
logp("Transfer Length: %u", INT0);
// If transfer length = 0 then exit
jumpz(INT0, 3);
debug(0);
// Setup
set_len(INT0);
test_unit_ready();
dmarst("R");
xfermode("DMARW", 0, 3);
// Set program logic error to ignore errors so program logic error is not
// logged when the command queue mode is set to terminate
pea("CONT");
// Set command queue mode to terminate to complete outstanding commands
cmdque_mode(8);
pea("LOGC");
read_capacity_10(0,0,0);
LNG0=get_long("R",0);
LNG1=add(INT0,1);
LNG2=sub(LNG0,LNG1);
// Set command queue mode to single
cmdque_mode(0);
queue_full_wait(1);
xfermode("DMAHC", 0, 3);
jump(2);
.
2:"Routine 2"
// Random Write/Read loopne()
random_blk(0,LNG2);
dmarst("W");
write_10_blk();
read_10_blk();
loopne();
queue_full_wait(0);
logp("Random Write/Read Compare completed");
debug(2);
.
3:"Routine 3"
// Random Write/Read Exit
logp("Random Write/Read Compare did not complete, transfer length is 0 ");
.
```

Adding Functions to a Macro File

To add a function from the function library to a macro file:

1 Place the cursor at the location where you want to add the function.

2 Click the function in the function library that you want to add to the macro file.

3 Click the Add Function to Macro button.

The function appears in the Contents pane at the selected position.

Note: A convenient shortcut for adding functions to a macro file is to hold down the **Ctrl** button on your keyboard, then click and drag the function file to the Contents pane.

4 Click the Save or Save as button at the top of the window to save the macro file.

5 Set the parameters as necessary for each function. The right-click menu provides editing options. These options are described in Editing Function Parameters above.

6 Refer to Appendix A, "Macro Variables and Functions and the Adding Frame Variables to a Macro section below for function and variable types and syntax as well as examples.

7 Indicate the end of the macro with a period on a line by itself.

8 Follow steps 1-3 to add the desired macros to the macro file, sequentially numbering the start of each one.

Adding Frame Variables to a Macro

The frame editor is an easy-to-use feature for customizing both legal and illegal SAS and SATA frames to send to a device. You can create and edit frame variables in the Macro File Editor window.

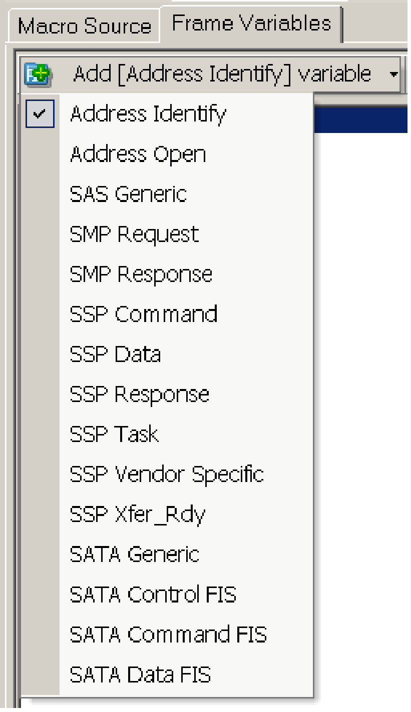
To access the Frame Editor, open the macro file that you want to add frame variables to, then select the Frame Variables tab.

To add a frame to a macro:

1 Select a frame variable, i.e. FRM0, FRM1, etc.

2 Click the **Add [...] variable** drop-down menu, and select the frame type that you want to use.

Figure 92: Frame Variables Drop Down Menu



Generator will fill in the fields in the right pane depending on which frame type you selected.

3 Type a name for the frame variable in the **Name** field.

4 To set the destination address for a SAS variable, click the in the **Get** button next to the **Dest Addr** field. This address indicates where the frame is being sent. If the frame type is not Identify or Open Address, this is the value that will set the hashed address for the frame and will be used in the automatically generate open address frame that will precede the specified frame. Once you've set the address, click **Set**.

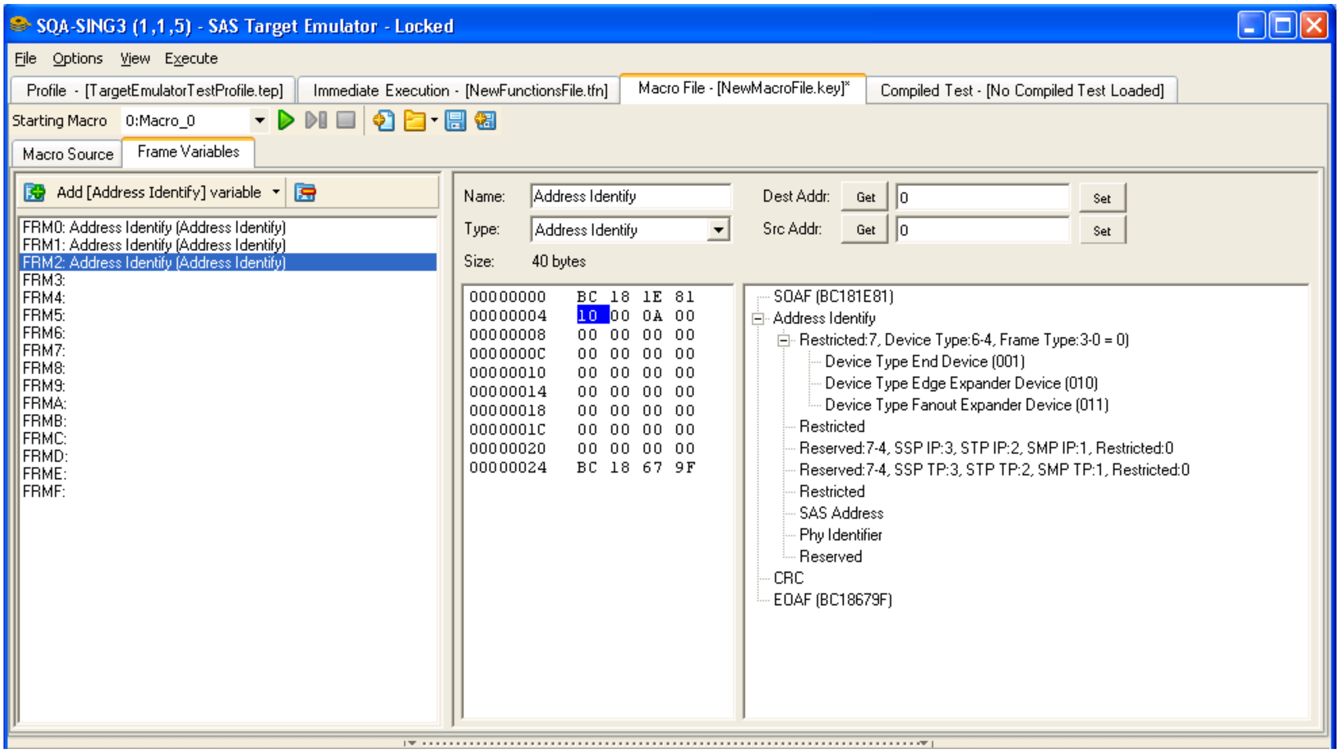
5 To get the source address for a SAS variable, click the Get button next to the **Src Addr** field. This address indicates where the frame is coming from. If the frame type is not Identify or Open Address, this is the value that will set the hashed address for the frame and will be used in the automatically generate open address frame that will precede the specified frame.

Note: SATA variables use a logical block address (LBA) setting instead of source and destination addresses.

6 Type an LBA address for the variable in the **LBA** field. This address indicates the logical block address used to identify the frame. Once you've set the address, click **Set**.

The contents of the frame is listed in the following two panes as shown below.

Figure 93: Frame Variables Window



SQA-SING3 (1,1,5) - SAS Target Emulator - Locked
File Options View Execute
Profile - [TargetEmulatorTestProfile.tep]
Immediate Execution - [NewFunctionsFile.tfn]
Macro File - [NewMacroFile.key]*
Compiled Test - [No Compiled Test Loaded]
Macro Source
Frame Variables
Add [Address Identify] variable
Get
Set
Get
Set

Starting Macro: 0:Macro_0

FRM0: Address Identify (Address Identify)
FRM1: Address Identify (Address Identify)
FRM2: Address Identify (Address Identify)
FRM3:
FRM4:
FRM5:
FRM6:
FRM7:
FRM8:
FRM9:
FRMA:
FRMB:
FRMC:
FRMD:
FRME:
FRMF:

Name: Address Identify
Type: Address Identify
Size: 40 bytes
Dest Addr: 0
Src Addr: 0

```
00000000   BC 18 1E 81
00000004   10 00 0A 00
00000008   00 00 00 00
0000000C   00 00 00 00
00000010   00 00 00 00
00000014   00 00 00 00
00000018   00 00 00 00
0000001C   00 00 00 00
00000020   00 00 00 00
00000024   BC 18 67 9F
```

```
SOAF (BC181E81)
Address Identify
  Restricted:7, Device Type:6-4, Frame Type:3-0 = 0)
    Device Type End Device (001)
    Device Type Edge Expander Device (010)
    Device Type Fanout Expander Device (011)
  Restricted
  Reserved:7-4, SSP IP:3, STP IP:2, SMP IP:1, Restricted:0
  Reserved:7-4, SSP TP:3, STP TP:2, SMP TP:1, Restricted:0
  Restricted
  SAS Address
  Phy Identifier
  Reserved
CRC
EOAF (BC18679F)
```

Frame Decode Pane

This pane lists the contents of the specific frame. This information pane tracks with the frame hex data pane. When you click on node in the frame, the frame hex data size pane shows the number of dwords/bytes that make up that node. When you double-click a node with a defined value in the information pane, such as “Frame Type [06]”, the corresponding hex value is set. The main components of a frame are:

- SOF
- Header (Expand this to show nodes)
- CRC
- EOF

Just above this pane are two fields. The first indicates the total size of the frame. The second, which allows you to add or remove certain types of bytes, such as FIS or CDB bytes from a frame, is only available for certain frame variable types. To add bytes to or remove bytes from the frame, click the up/down arrows next to the field.

Frame Hex Data Pane

This pane lists the bytes for a specific frame component. To edit the byte value:

— Select the byte value, then type another value.

You can add bytes to specific types of frames to increase the size of the frame. If the selected frame type allows the addition of bytes, a numeric field will be displayed above the **Frame Hex Data** pane.

Running a Macro on a Target Emulator Device

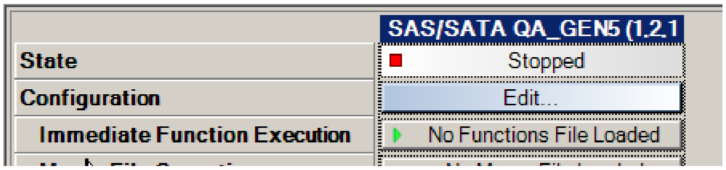
You can run a macro file from the Macro File tab of the Target Emulator Device window or from the Parameters Status table in the Maestro window. When running macro files from the Parameters Status table, you can run the files on multiple devices simultaneously.

To run a macro file for a device from the Target Emulator Device window, click the **Start** ▷ button on the menu bar of the **Macro File** tab.

To run a single macro for a device from the Parameters Status table:

1 Click the **Start** ▷ button on the Parameters Status table for Macro File Operations. You need to have a macro file loaded to perform this action.

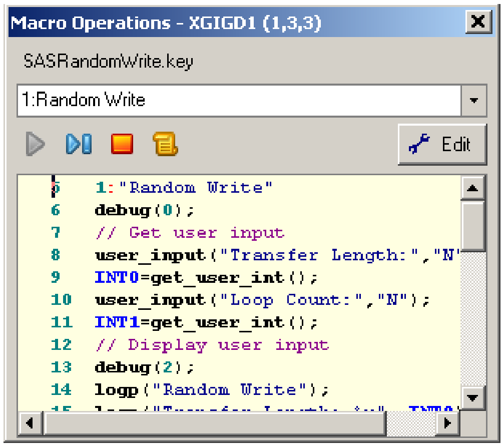
Figure 94: Parameters Status table Start Macro Button



	SAS/SATA QA_GEN5 (1,2,1
State	🟥 Stopped
Configuration	Edit...
Immediate Function Execution	▷ No Functions File Loaded

The Macro Operations window is displayed.

Figure 95: Macro Operations Window



```
1:"Random Write"
debug(0);
// Get user input
user_input("Transfer Length:","N"
INT0=get_user_int();
user_input("Loop Count:","N");
INT1=get_user_int();
// Display user input
debug(2);
logp("Random Write");
```

2 Click the Preview Macro button if you want to preview the macro contents.

3 Click the **Start** ▷ button to run the macro.

4 Click the **Abort Macro** 🟥 button to stop the execution of a macro.

5 Click the **Send Keystroke** button to simulate a keystroke to stop looping during macro execution. This selection is active only if the `loopk` function is invoked in your macro. Refer to the description for `loopk` in the function library.

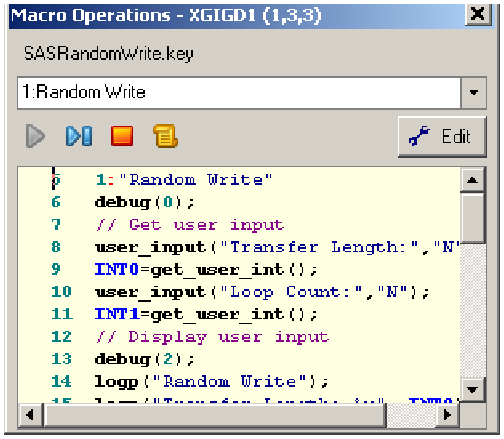
6 A log entry is generated.

To execute a macro for a device from the Ports Manager:

1 Click the Macro Operations [Macro Operations] button for the device.

The Macro Operations window is displayed.

Figure 96: Macro Operations Window



```
1:"Random Write"
debug(0);
// Get user input
user_input("Transfer Length:","N"
INT0=get_user_int();
user_input("Loop Count:","N");
INT1=get_user_int();
// Display user input
debug(2);
logp("Random Write");
```

2 Click the Preview Macro button if you want to preview the macro contents.

3 Click the **Start** button to run the macro.

4 A log entry is generated.

To simultaneously run macros on multiple devices from the Port Manager:

1 Select each device that you want to run macros on by checking the **Select** check box for the device.

2 Select the macro file you want to load from the Configuration Manager.

3 Click the **Load Configuration into Selected Ports** button.

4 Click the **Start Macro on Selected Ports** button.

5 A log entry is generated.

To stop a macro file in progress, right click the device column in the Parameter status table, and select **Stop Macro**.

Saving a Macro File

To save a macro file, click the **Save** button.

To save the file as another name or to another location:

1 Click the **Save As** button.

The **Save**/**Save As** dialog appears.

➡ **Note:** If you want to use the standard Windows file dialogs instead of the dialogs customized for Target Emulator, select **Use Standard Windows File Dialogs** from the **Options** menu in the Target Emulator Device window menu bar.

2 Navigate to the location where you want to save the file.

3 Type a name for the file.

4 Click **Save.**

5 Click the **Done** button to close the window. If you make changes to the file and then try to close the window, you will receive prompt asking you if you want to save the file before closing it.

Compiled Test Tab

A compiled test file is a `.exe` file written using the SAS/SATA Target Emulator API. Refer to the Generator and Target Emulator API Help for instructions on creating compiled tests. The Target Emulator GUI does not allow you to modify the `.exe` file. However, the GUI does allow you to add parameters to a complied test, which, when saved, creates a `.gen` file of the same name.

Using this section, you can add and edit parameters for a compiled test file, essentially, creating `.gen` files.

Opening a Compiled Test File

To open a recently used Xgig Target Emulator compiled test file:

1 Click the arrow next to the **Open Compiled Test** 📂▾ button in the menu bar.

2 Click the compiled test file you want to open.

A compiled test window appears with the compiled test file loaded.

To browse to a compiled test and open it:

1 Click the **Open Compiled Test** 📂▾ button in the menu bar.

2 Browse to the compiled test file you want to open.

3 Click the **Open** button.

A compiled test window appears with the compiled test loaded.

Adding Parameters to a Compiled Test File

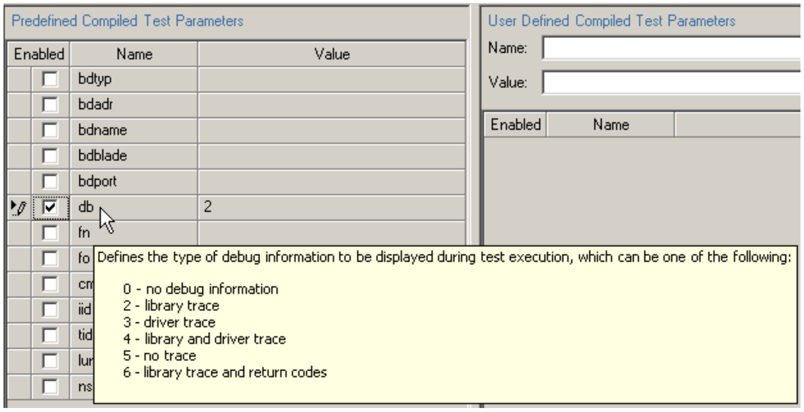
The **Description** field allows you to type a description for the compiled test. This description will appear in information panes when the file is selected.

The **Predefined Compiled Test Parameters** pane lists common command line parameters. These parameters are listed in the **Common Command Line Parameters** section of the API Help. Holding the mouse over one of these parameters brings up a tool tip window containing information about the parameter such as the definition, use, and defined or suggested values.

Example:

Holding the mouse over the `bdtyp` parameter displays the following tool tip.

Figure 97: Bdtyp Compiled Test Parameter Tool Tip

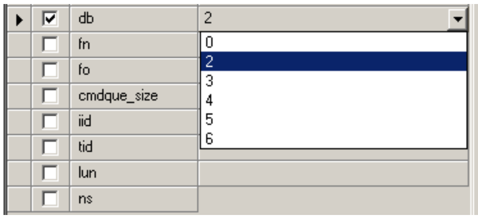


Defines the type of debug information to be displayed during test execution, which can be one of the following:

0 - no debug information
2 - library trace
3 - driver trace
4 - library and driver trace
5 - no trace
6 - library trace and return codes

The tool tip describes the parameter and lists the possible values. Selecting this row and clicking the value drop-down menu, the possible values as shown in the following figure.

Figure 98: Bdtyp Compiled Test Parameter Values



Click the row with the desired parameter to make it editable. The ✎ icon appears next to the enabled field to indicate that this is the parameter you are editing.

To select a value for this parameter, click the value you want from the drop-down list.

Note: Not all parameter values fields have a drop-down menu for the value. Some tool tips indicate a value range or define the type of information required and the format required.

To enable a parameter for the compiled test, select the **Enable** check box next to the parameter.

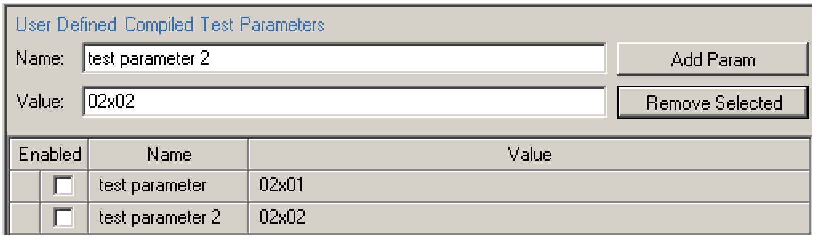
The **User Defined Compiled Test Parameters** pane allows you to create your own parameters.

To create a parameter:

1. Type a name in the **Name** field.
2. Type a value in the **Value** field.
3. Click the **Add Parameter** button.

The parameter appears in the list below as shown in the following figure.

Figure 99: User Defined Parameter



User Defined Compiled Test Parameters

Name: test parameter 2 Add Param

Value: 02x02 Remove Selected

Enabled	Name	Value
□	test parameter	02x01
□	test parameter 2	02x02

4 Click the row with the desired parameter to make it editable. The ✎ icon appears next to the enabled field to indicate that this is the parameter you are editing.

To enable a parameter for the compiled test, select the **Enable** check box next to the parameter.

To delete the parameter, select the entire row, and click the **Remove Selected** button.

Running a Compiled Test on a Target Emulator Device

After you have configured a Target Emulator device and loaded a compiled test file, you can run the file.

Note: Before launching a compiled test, the GUI disconnects from the port then reconnects to the port when the test is complete.

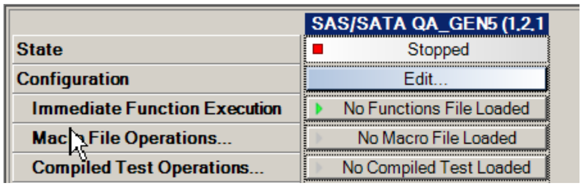
You can run a compiled test file from the Compiled Test tab of the Target Emulator Device window or from the Parameters Status table in the Maestro window. When running compiled tests from the Parameters Status table, you can run the test files on multiple devices simultaneously.

To run a compiled test for a device from the Target Emulator Device window, click the **Start** ▶ button on the menu bar.

To run a single compiled test file for a device from the Parameters Status table:

1 Click the **Start** ▶ button on the Parameters Status table next to the Compiled Test Operations field. You need to have a compiled test file loaded to perform this action.

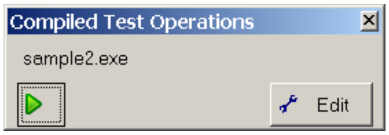
Figure 100: Parameters Status table Start Compiled Test Button



	SAS/SATA QA_GEN5 (1,2,1
State	■ Stopped
Configuration	Edit...
Immediate Function Execution	▶ No Functions File Loaded
Macro File Operations...	No Macro File Loaded
Compiled Test Operations...	No Compiled Test Loaded

The Compiled Test Operations window is displayed.

Figure 101: Compiled Test Operations Window



Compiled Test Operations

sample2.exe

2 Click the **Start** ▷ button to run the compiled test.

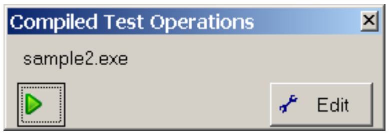
3 A log entry is generated.

To execute a compiled test for a device from the Ports Manager:

1 Click the Compiled Test Operations [Compiled Test Operations] button for the device.

The Compiled Test Operations window is displayed.

Figure 102: Compiled Test Operations Window



2 Click the **Start** ▷ button to run the compiled test.

3 A log entry is generated.

To simultaneously run compiled tests on multiple devices from the Port Manager:

1 Select each device that you want to run compiled tests on by checking the **Select** check box for the device.

2 Select the compiled test file you want to load.

3 Click the **Load Configuration into Selected Ports** button.

4 Click the **Start Compiled Tests on Selected Ports** button.

5 A log entry is generated.

Saving a Compiled Test File

To save a compiled test file, click the **Save** button.

To save the file as another name or to another location:

1 Click the **Save As** button.

The **Save**/**Save As** dialog appears.

➡ **Note:** If you want to use the standard Windows file dialogs instead of the dialogs customized for Generator, select **Use Standard Windows File Dialogs** from the **Options** menu in the Generator Device window menu bar.

If you click the **Save as** selection to save a `.gen file` as a new file name, a copy of the existing compiled test `.exe` will be created with along with the new `.gen` file.

2 Navigate to the location where you want to save the file.

3 Type a name for the file.

4 Click **Save.**

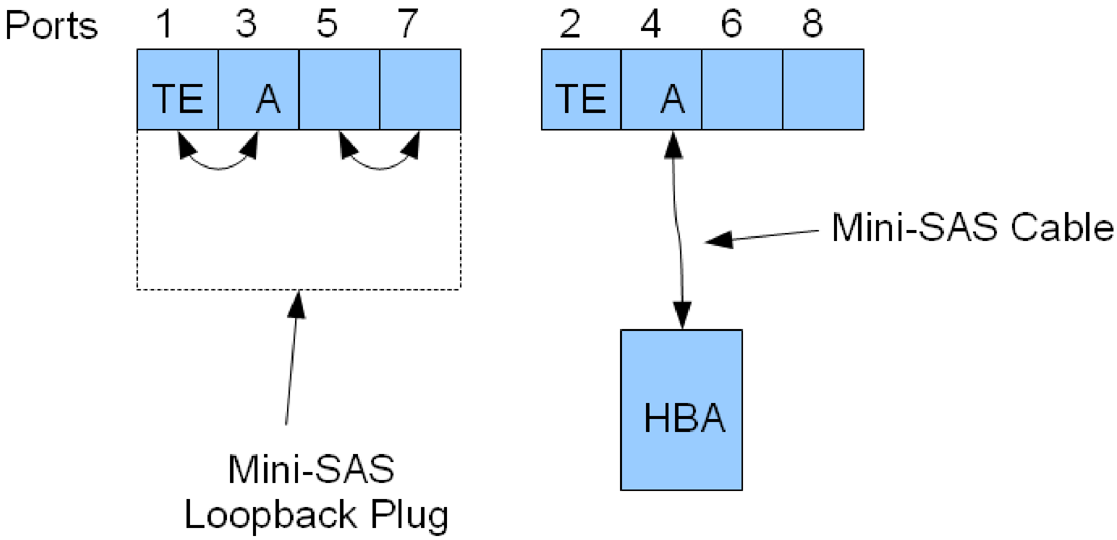
5 Click the **Done** button to close the window. If you make changes to the file and then try to close the window, you will receive prompt asking you if you want to save the file before closing it.

Setting up a Test with Target Emulator and HBA

The following setup process will discuss how to build a test setup with Xgig Target Emulator, Xgig Analyzer, and a Host Bus Adapter (HBA).

Set up the proper connection between the Xgig Chassis and the HBA.

Figure 103: Test setup for Target Emulator, Analyzer and HBA



1 Connect the Host PC’s HBA to the Xgig Blade using a Mini-SAS cable. Refer to Figure 103 for the connection diagram.

2 Use a Mini-SAS loop back plug to create a loop back for the other ports of the blade.

This connection setup promises better signal integrity (shorter cables), and is better if 6G interfacing speed is needed.

Lock the ports to SATA TE on Maestro

1 Launch Xgig Maestro on you PC (ideally not installed in your Host PC).

2 Click on **Port Selection and Domain Setup** button from the **Maestro Global** tab.

3 Select a chassis that has a 8-port wide blade and a port pair with Target Emulator licence.

4 Click on the first port pair. Refer to Figure 103.

5 Right-click on the selected ports and select **Use Port(s) as...>Target Emulator SAS/SATA.**

6 Go to the Parameters Status Table and right click on the locked port.

7 Select **Change port to SATA mode**.

Lock the ports to Xgig Analyzer

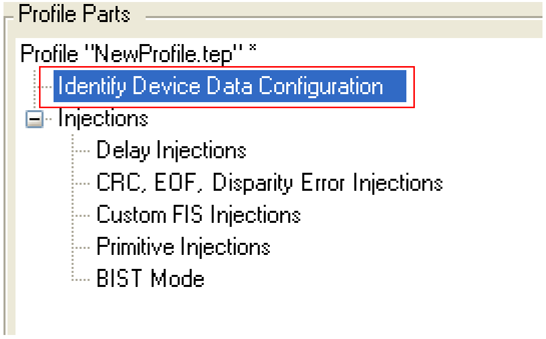
Please refer to the *Xgig Analyzer User’s Guide* on how to lock the ports to Xgig Analyzer.

Set the Signalling Speed from the Identify Device Data Configuration

1 From the Parameters Status Table, click on **Edit** on the **Configuration** row. This will open the Target Emulator Device window.

2 Click the **Identify Device Data Configuration** on **Profile Parts** pane. The Parts Options pane will display the options for the Identify Device Data Configuration.

Figure 104: Identify Device Data Configuration on Profile Parts Pane



Profile Parts

Profile "NewProfile.tep" *
- Identify Device Data Configuration
- Injections
  - Delay Injections
  - CRC, EOF, Disparity Error Injections
  - Custom FIS Injections
  - Primitive Injections
  - BIST Mode

3 Scroll down the Parts Options pane to **Dword 76**.

4 Select the signalling speed that to be used by the Target Emulator.

5 Once done, click on **Apply Profile Part**. This will send the new settings to the Target Emulator port.

The Target Emulator is now ready to be used for the test.

Power-on the PC

Turn on the PC. After the Windows operating system has loaded, you will not see the Target Emulator Drive in Windows Explorer. You need to initialize and format the drive first.

Initialize and Format the Target Emulator drive

To initialize the drive, go to Computer Management:

1 Click **Start**, and then click **Control Panel**.

2 Click **Performance and Maintenance**.

3 Click **Administrative Tools**, and then double-click **Computer Management**.

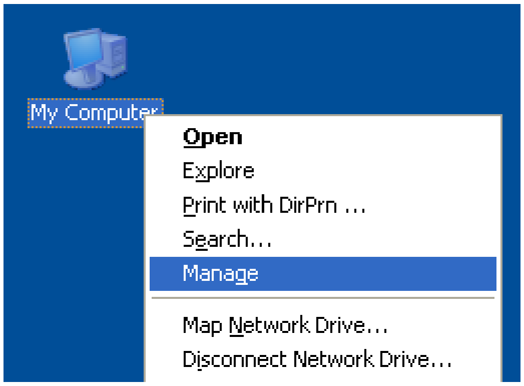
Alternatively, you can do the following:

1 Right-click on the **My Computer** icon.

2 Select **Manage**. This will open the **Computer Management** window (refer to Figure 105).

Note that you must be logged on as an administrator or a member of the Administrators group in order to complete this procedure. If your computer is connected to a network, network policy settings might also prevent you from completing this procedure.

Figure 105: My Computer Context Menu

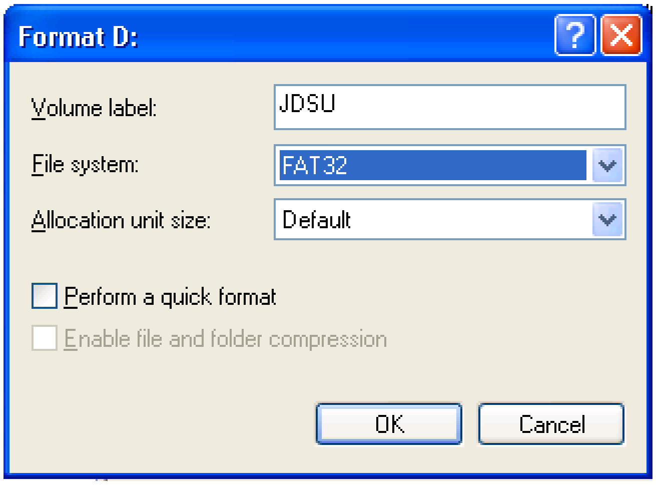


New disks appear as **Not Initialized**. Before you can use a disk, you must first initialize it. If you start **Disk Management** after adding a disk, the **Initialize Disk Wizard** appears so you can initialize the disk.

After initializing, you must format the drive. To format the drive:

1. Click on Disk Management from the **Computer Management** window.
2. Right-click on the drive on the Disk Management window and select **Format**.

Figure 106: Format Drive



Format D:

Volume label: JDSU

File system: FAT32

Allocation unit size: Default

☐ Perform a quick format

☐ Enable file and folder compression

3. Enter the Volume label.
4. Select the file system.

Note that NTFS is not recommended for capacity greater than 48MB because NTFS writes disk and partition information to the start and end sectors of the drive, which in our case, will encounter a circular buffer wraparound, and there will be errors. The FAT32 file system writes disk and partition information only to the starting sectors of the drive, so this can be applied for all cases.

5. Click **OK** to start formatting.

Copy Files to the Drive

Once the Target Emulator drive has been formatted, you can copy files to the drive to test if you can write data into it..

You can reboot the Host PC and verify that the file is still intact.

Modify Device Data

If you have verified that the files you copied into the Target Emulator drive has been properly written and read, you can now try modifying the Device Data of the Target Emulator drive.

1. Go to the Maestro window in the client PC.
2. Under Identify Device Data Configuration in the Target Emulator Device window, change the Model Number to a different model, for example, “JSATA”, and then click **Apply Profile Part.**
3. Change the values in both word 60-61 and word 100-103 to change the drive capacity. You must specify the number of sectors.

   For Example 10Gig = 10x1024x1024x1024 = 10737418240Bytes. You need to divide this value by 512 = 20971520 = 0x140 0000 sectors.
4. Reboot the Host PC so that the changes to the Target Emulator drive will be detected and reflected in the Host PC.

After you have completed this procedure you can now proceed with you other tests.

PART THREE: Appendices

Appendix A

Macro Variables and Functions

This appendix defines the variables used in macros as well as the various types of macro functions. This appendix also shows the syntax for these elements and provides examples of the context in which the variables and functions could be used. Numeric variables are decimal values unless they are preceded by “0x”, in which case, they are hexadecimal.

Macro Variables

Macro tests have 80 variables that can be used. Sixteen are integer variables, sixteen are long variables, sixteen are quadword variables, sixteen are frame variables, and sixteen are string variables. Integer variables are 16 bits long, long variables are 32 bits long, and quadword variables are 64 bits long. String variables can be any length. Frame variables are user-created frames listed in the User Frames pane.

Integer, long, and quadword variables may be freely used in any function with its size converted to the size the function expects. For example, if the variable LNG0 is passed to a function expecting an integer parameter, the value is reduced to integer size by discarding the upper 16 bits before being passed to the function. So if LNG0 = asgn(0x12348001), 0x8001 will be passed to the function, although the variable LNG0 itself will still contain its original value. Note that if the function takes a signed rather than an unsigned integer, 0x8001 will be treated as a negative number even though LNG0 was not negative.

STR variables can be used in any function that has a string as its parameter type.

Only function return values can be stored in variables. If a function returns a long and it is assigned to an integer variable, the value will be truncated to fit.

`INT0, INT1, ... INTF` – integer variables
Example: `INT0=get_int("R", 0L);`

`LNG0, LNG1, ... LNGF` – long variables
Example: `LNG0=get_g_info("bufsz");`

`QUAD0, QUAD1, ... QUADF` – quadword variables
Example: `QUADO=get_info_64("lun");`

Note: A QUAD variable can’t be used as a parameter in arithmetic functions because arithmetic functions work only with INT or LNG variables.

`STR0, STR1, ... STRF` – string variables
Example: `STR0=asgnstr("This is STR0.");`

`FRM0, FRM1, ... FRMF` – frame variables
Example: `INT0=transmit(FRM0);`

`transmit(), receive(), display_receive(), assign_receive()` – frame functions

Use the macro-specific function asgn(xx) to set the variable to a constant value.
Use the macro-specific function asgnstr("") to set the string variable.

Example: `LNG0=asgn(1);`
Example: `STR0=asgnstr("This is STR0.");`

Additional quadword examples:

```
QUAD0=asgn(1);
logQUAD("QUAD0=%I64u",QUAD0);
LNG0=asgn(2);
logp("LNG0(ASGN 2)=%d",LNG0);
QUAD1=add(LNG0,2);
```

```
QUAD0=asgn(1);
logQUAD("QUAD0=0x%I64X",QUAD0);
QUAD1=asgn(0xffffffffffffffff);
logQUAD("QUAD1=0x%I64X",QUAD1);
logQUAD("QUAD1=%I64d",QUAD1);
logQUAD("QUAD1=%I64u",QUAD1);
```

Note: The example below will not work because you can’t use QUAD as a parameter in arithmetic functions.

```
QUAD1=or(QUAD0,0);
logQUAD("QUAD1=%I64u",QUAD1);
LNG1=OR(QUAD0,0);
logp("LNG1=%d",LNG1);
```

Macro Functions

Branches

Branch functions move the execution pointer to a new line in the macro, above or below the current branch function, allowing the use of conditional statements inside a single macro. The variable X is a relative line number, where X=$^{-}$1 is the line before the branch and X=1 is the line after the branch.

`branche(xx,yy,X)`
This function branches if xx=yy. The parameters xx and yy can be constants or variables.

`branchl(xx,yy,X)`
This function branches if xx<yy. The parameters xx and yy can be constants or variables.

`branchle(xx,yy,X)`
This function branches if xx≤yy. The parameters xx and yy can be constants or variables.

`branchnz(xx,X)`
This function branches if xx is nonzero. The parameters xx and yy can be constants or variables.

`branchz(xx,X)`
This function branches if xx=0. The parameters xx and yy can be constants or variables.

Jumps

Jump functions take execution to the beginning of a new macro, allowing multiple macro tests to comprise a larger test sequence. Note that macros that are jumped to must reside in the same macro file.

`jump(X)`
This function jumps unconditionally to a new macro X.

`jumpe(xx,yy,X)`
This function jumps to a new macro X if xx=yy. The parameters xx and yy can be constants or variables.

`jumpl(xx,yy,X)`
This function jumps to a new macro X if xx<yy. The parameters xx and yy can be constants or variables.

`jumple(xx,yy,X)`
This function jumps to a new macro X if xx≤yy. The parameters xx and yy can be constants or variables.

`jumpnz(xx,X)`
This function jumps to a new macro X if xx is nonzero. The parameter xx can be a constant or a variable.

`jumpz(xx,X)`
This function jumps to a new macro X if xx=0. The parameter xx can be a numerical constant or one of the variables provided.

`jumpsub(X)`
This function jumps to a new macro X. Upon completion of macro X, the control jumps back to the calling macro at the line immediately following the jumpsub() line, allowing for subroutine-like functionality.

➡ **Note:** The macro function jumpsub() can’t be nested.

Loops

Loop functions take execution of the macro test back to the beginning of the current macro. If you right-click on the macro and select Macro Keystroke from the menu, the current loop will terminate with execution continuing with the function immediately following the loop function. If you right-click on the macro and select Macro Abort from the menu, macro execution ends entirely.

loop(n)
Returns execution to the beginning of the macro. The function will loop n times after which execution will continue with the function immediately following the loop(n) function, if any.

Note: The macro function loop() can't be nested directly. If it is nested, the outermost loop() will act as a loopk(). This function cannot be nested indirectly. If you use a jumpsub(x) in a loop() where x is a macro that contains another loop, this results in an indirect case of a nested loop().

Example:

```
1:"outer loop"
jumpsub(2)
loop(1)
.
2:"sub"
loop(1)
.
```

loope()
Loops to the beginning of the macro if an error occurred between the beginning of the macro and the loope() function. If no error occurred during a loop, the loop will terminate and fall through to the next function in the macro.

loopk()
Loops until you select **Send Keystroke to Macro**, which is in the Execute menu of the Generator Device window. See Chapter 3, "Using Generator with Ports.

loopne()
Same as loope() except that it loops while there is no error.

Logical Functions

Since only function return codes can be stored in variables, the functions described below are used to perform initialization, simple math, or logical manipulation. In the following variable descriptions, integer refers to 16-bit data types and long refers to 32-bit data types.

`add(xx,yy)`
Returns the sum of xx and yy as a long. The parameters are expected to be longs. Integer parameters will be automatically converted to longs as described above.

`and(xx,yy)`
Returns the bit-wise and of xx and yy as a long. The parameters are expected to be longs.

`asgn(xx)`
Returns the parameter xx as a long and is used to initialize both integer and long variables.

`bit_shift(xx,RIGHT,BITS)`
If RIGHT is zero, returns xx<<BITS (xx shifted to the left by BITS number of bits). If RIGHT is nonzero, returns xx>>BITS (xx shifted to the right by BITS number of bits).

`itoi(0xXX,0xYY)`
Returns the integer 0xXXYY as a long and is used to concatenate two one-byte integers into a single two-byte integer. The results may not be as expected if either parameter exceeds 0xFF in size.

`itol2(0xXXXX,0xYYYY)`
Returns the long 0xXXXXYYYY and is used to concatenate two two-byte integers into a single long.

`itol4(0xWW,0xXX,0xYY,0xZZ)`
Returns the long 0xWWXXYYZZ and is used to concatenate four one-byte integers into a single long. The results may not be as expected if any parameter exceeds 0xFF in size.

`not(xx)`
Returns the bit-wise not of the value xx as a long. The parameter is expected to be a long.

`or(xx,yy)`
Returns the bit-wise or of xx and yy as a long. The parameters are expected to be longs.

`sub(xx,yy)`
Returns xx–yy as a long. The parameters are expected to be longs. The results will not be as expected if xx<yy since the returned value is unsigned.

`xor(xx,yy)`
Returns the bit-wise xor of xx and yy as a long. The parameters are expected to be longs.

Miscellaneous Functions

asgnstr("xx")
Returns the parameter xx as a string used to initialize string variables.

clear_log_view
Clears the log pane.

rtnfilesz("path name")
Returns the size of a binary file as a long. The parameter is a string path name.

copy_user_string(STR0)
Copies the current user string, as set by user_input(), into the string variable. The parameter is a string variable STR0-STRF.

Example:

```
user_input("Enter Name of File", "S");
copy_user_string(STR0);
```

transmit(FRM0)
Transmits a user frame using frame_user(). The return value is TRUE (1) or FALSE (0).

Example:

```
INT0=transmit(FRM0);
logp("INT0 = 0x%04X", INT0);
```

receive()
Receives and stores the frame in the macro receive buffer using get_frame(1) and get_frame_info(). The return value is TRUE (1) or FALSE (0).

Example:

```
INT0=receive();
```

display_receive()
Displays the frame in the macro receive buffer in raw data format in the log window. The return value is TRUE (1) or FALSE (0).

Example:

```
INT1=display_receive();
logp("INT1 = 0x%04X", INT1);
```

asgn_receive (xx)
Assign the receive buffer dword to a variable.

Example:

```
INT0=receive();
logp("INT0 = 0x%04X", INT0);
// LNG0 = receive buffer dword 1
LNG0=asgn_receive(1);
logp("LNG0 = 0x%08X", LNG0);
```

macro_user_input

Displays an input dialog with p_string as prompt. The expected data type for the value to be provided by the user is specified by the second parameter expectedType. To retrieve the value entered by the user, the macro developer must assign the return value to a compatible macro variable (INT, LNG, QUAD or STR). An error message will be shown if the user enters a value that is not of the expected data type or if macro tries to assign the function return to an incompatible macro variable.

macro_user_output

Writes a text string to the Log after appending a timestamp to the string. The string may contain variable formatting codes that should match each of the optional parameters provided. The acceptable formatting codes are the same used by the sprintf standard C function.

macro_delayms

Delays the macro execution for about ms_delay milliseconds.

transmit_smp (FRM0)

Transmits an SMP user frame using sas_smp_user(). The received SMP response is also retrieved to the macro receive buffer. Thus, the receive() should not be called after transmit_SMP. The return value is TRUE (1) or FALSE (0).

Example:

```
INT0=transmit_SMP(FRM0);
logp("INT0 = 0x%04X", INT0);
```

Index

November 2015
Version 8.1
English

Viavi Solutions

North America:	1.844.GO VIAVI / 1.844.468.4284
Latin America	+52 55 5543 6644
EMEA	+49 7121 862273
APAC	+1 512 201 6534
All Other Regions:	viavisolutions.com/contacts
email	customer.care@viavisolutions.com