

Optical Test System OTS Product Family

OTS *Toolkit*[™]

JANUARY 2003

Document No. 071-1134-01



15550 Lightwave Drive • Clearwater, Florida 33760 • United States
T: 727.442.6677 • F: 727.442.5660 • Toll Free: 800.548.9283 or 877.275.3445
info@lightwave.com • <http://www.lightwave.com>

Copyright

Copyright © 2003 Digital Lightwave, Inc.

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

This publication, or parts thereof, may not be reproduced in any form, by any method, for any purpose. For conditions of use and permission to use these materials for publication in other than the English language, contact Digital Lightwave, Inc.

Digital Lightwave, Inc. reserves the right to revise and improve its product as it sees fit. This publication describes the state of this product at the time of its publication and may not represent the product at all times in the future.

OTS and Optical Test System are trademarks of Digital Lightwave, Inc.

Portions of the licensed software products and this publication copyright © Tektronix, Inc. 1999 - 2002

Microsoft® Windows 2000®: Copyright © Microsoft Corporation 1985 - 1996

Welcome

The *OTS Toolkit™ User Manual* describes the capabilities of this companion software package to the OTS system. The manual provides detailed descriptions of the user interface and remote commands. For detailed information about other features and capabilities of the OTS system, refer to the individual product manuals provided on CDrom with the system. The user interface also provides Windows Help files for further information on specific topics.

Table of Contents

General Safety Summary.....	iii
Preface.....	vii
Getting Started	
Product Description	1-1
Features	1-2
Operating Basics	
OTS Toolkit™ Main Menu	2-1
OTS Script Creator	2-2
Learning to Script.....	2-3
How to Create a Script.....	2-3
Modifying a Script.....	2-5
Inserting Commands Into the Script.....	2-5
Moving Commands in the Script.....	2-5
Copying Commands in the Script	2-5
Deleting Commands in the Script	2-5
How to Create a Script Template.....	2-6
Identifying Resources	2-6
Delete a Resource	2-7
Add a Resource	2-7
Modify a Resource	2-8
Managing Variables	2-8
Creation and Editing of Variables	2-9
Define Variables Dialog Menu	2-11
OTS Script Creator Control Statements	2-12
ASSIGN.....	2-12
Comments.....	2-12
FINDSTR.....	2-13
FOR.....	2-15
GOTO.....	2-16
IF.....	2-17
LOG.....	2-18
POW.....	2-19
REPEAT.....	2-20
RANGE.....	2-21
SQR.....	2-22
SQRT	2-22
WAIT	2-23
Script Debugger	2-24
To Start Script Debugger	2-24
Script Interpreter	2-26
To Run a Script using Script Interpreter	2-27

OTS Test Scheduler	2-28
Deleting Scheduled Tests	2-29
Adding a Test to the Schedule	2-29
Viewing and Clearing the Test Log	2-30
Test Scheduler Status Display Messages	2-30
OTS <i>Toolkit</i> [™] Applications	2-31
Custom Script	2-31
ABER (Accelerated BER)	2-31
Scheduling the ABER test	2-31

Reference

General Scripting Notes	3-1
Explanation of the server timeout feature	3-2
Read-only script file attribute	3-3
Script Compile function	3-3
Copying and Pasting script lines	3-3
Modifying scripts for use with multiple OTS instruments	3-3
Searching data strings using the FINDSTR function	3-4
ABER Error Messages	3-4

Appendices

Sample Script	A-1
Sample Script Description	B-1
Digital Lightwave Supplied Sample Scripts	C-1

General Safety Summary

Review the following safety precautions to avoid injury and prevent damage to this product or any equipment connected to it.

To avoid potential hazards, use this product only as specified.

Only qualified personnel should perform service procedures.

While using this product, you may need to access other parts of the system. Read the *General Safety Summary* in other system manuals for warnings and cautions related to operating the system.

How to Avoid Fire or Personal Injury

Use Proper Power Cord. To avoid fire hazard, use only the power cord specified for this product.

Use Proper Power Source. Do not operate this product from a power source that applies more than the voltage specified.

Connect and Disconnect Properly. Do not connect or disconnect test leads while they are connected to a voltage source.

Avoid Electric Overload. To avoid electric shock or fire hazard, do not apply a voltage to a terminal that is outside the range specified for that terminal.

Ground the Product. This product is grounded through the grounding conductor of the power cord. To avoid electric shock, the grounding conductor must be connected to earth ground. Before making connections to the input or output terminals of the product, ensure that the product is properly grounded.

Observe All Terminal Ratings. To avoid fire or shock hazard, observe all ratings and markings on the product. Consult the product manual for further ratings information before making connections to the product.

The common terminal is at ground potential. Do not connect the common terminal to elevated voltages.

Do not apply a potential to any terminal, including the common terminal, that exceeds the maximum rating of that terminal.

Use Proper AC Adapter. Use only the AC adapter specified for this product.

Do Not Look into the End of a Fibreglass Cable. Never look into the end of a fibreglass cable or a single fibre which could be connected to a laser source. Laser radiation can damage your eyes because it is invisible and your pupils do not contract instinctively as with normal bright light. If you think your eyes have been exposed to laser radiation, you should have your eyes checked immediately by an eye doctor. The optical output's radiation power corresponds to the laser class in accordance with IEC 825-1, 11.93.

Use Proper Fuse. To avoid fire hazard, use only the fuse type and rating specified for this product.

Avoid Exposed Circuitry. Do not touch exposed connections and components when power is present.

Do Not Operate With Suspected Failures. If you suspect there is damage to this product, have it inspected by qualified service personnel.

Do not operate in Wet/Damp Conditions. To avoid electric shock, do not operate this product in wet or damp conditions.

Do Not Operate in Explosive Atmosphere. To avoid injury or fire hazard, do not operate this product in an explosive atmosphere.

Wear Eye Protection. To avoid eye injury, wear eye protections if there is a possibility of exposure to high-intensity rays.



Keep Product Surfaces Clean and Dry.

Provide Proper Ventilation. Refer to the manual's installation instructions for details on installing the product so it has proper ventilation.

Safety Terms and Symbols

Terms in this Manual

These terms may appear in this manual:

Icon	Label	Meaning
	WARNING!	Warning statements identify conditions or practices that could result in injury or loss of life.
	CAUTION!	Caution statements identify conditions or practices that could result in damage to this product or other property.

Terms on the Product

These terms may appear on the product:

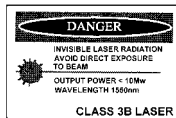
DANGER indicates an injury hazard immediately accessible as you read the marking.

WARNING indicates an injury hazard not immediately accessible as you read the marking.

CAUTION indicates a hazard to property including the product.

Symbols on the Product

The following symbols may appear on the product:



CAUTION
Laser Radiation



Protective Ground
(Earth) Terminal



ATTENTION
Refer to Manual



Electrostatically
hazardous

Preface

This manual describes how to use the OTS *Toolkit*[™] software feature set. This manual is your primary source of information about how the OTS *Toolkit*[™] functions.

The user interface also provides Windows Help files for further information on specific topics.

How This Manual is Organized

This manual is divided into four sections: *Getting Started*, *Operating Basics*, *Reference*, and *Appendices*.

- *Getting Started* provides an overview of the OTS *Toolkit*[™].
- *Operating Basics* explains the basic principles of operating the OTS *Toolkit*[™].
- *Reference* provides other general information in using the OTS *Toolkit*[™] to help the user gain the most use from this tool.
- The *Appendices* provide forms, sample script listings, and other useful information.

Conventions

This manual uses the following conventions:

- ❖ The names of front-panel connectors and LEDs appear in the manual in the same format as found on the front panel label, for example, OPTICAL IN and Rx DATA OUT.
- ❖ When the user interface is discussed, all menus, names tags, and button appear in the manual in the same format as found in the user interface, for example, Enable COM2 and Output Pulse Trigger.
- ❖ In reference to terminology, the user interface may be set to either SDH or SONET references. The user manual provides SDH terminology with SONET terminology in parenthesis immediately following the SDH version. If no second terminology is present, the terminology is the same for both SDH and SONET.
- ❖ In reference to the instrument, the following conventions apply:
 - When referring to the four card 10Gb/s system (Optics, Transmit, Receive, and Clock), the name OTS9100 is used.
 - When referring to each individual card, the card name is used, for example, Optics and Transmit.

NOTE: *Some of the content found in this manual does not pertain to some instruments. Depending on the software revision and the options installed, some of the features described in these pages may not be available.*

Getting Started

This chapter provides an overview of the OTS *Toolkit*[™], its main features and functions.

Product Description

The OTS *Toolkit*[™] is a software enhancement family comprised of feature sets containing individual tools that assist in the automation of test and measurement testing for the OTS product family.

The OTS *Toolkit*[™] is a family of "easy-to-use" programmable GUI based software enhancement feature sets that are designed to automate test and measurement procedures for OTS Product Family systems. OTS *Toolkit*[™] virtually eliminates repetitive set-up time and decreases the time-to-test on optical equipment within the R&D and manufacturing test floor environments.

The OTS *Toolkit*[™] includes the following feature sets:

- o **OTS Test Scheduler** - A scheduler tool that can execute one, or more previously created OTS scripts.
- o **OTS Script Creator** – A scripting tool that provides an intuitive user interface for scripting OTS commands and programming functions used for crafting practical OTS testing applications.
- o **OTS Script Debugger** – A scripting tool that permits single stepping the execution of an OTS script. Useful for finding and correcting OTS script execution and/or programming errors.
- o **OTS Script Interpreter** – A scripting tool that is used to execute and monitor progress of previously created OTS scripts.
- o **OTS Q-Factor (Accelerated BER)** – An automated ABER test, that quickly estimates the quality of an optical signal. This OTS *Q-Factor* auto-test runs for approximately 30 minutes. It quickly determines and identifies DUT's that would have a high probability of not passing long- term performance testing cycles. OTS *Q-Factor* eliminates wasted test time during BER Soak Test procedures, which commonly last 24 to 48 hours.

Features

The key features of the OTS *Toolkit*[™] software package are:

- o Custom OTS scripts and Digital Lightwave-supplied test modules may be executed either from a client PC or directly on an OTS server (test instrument)
- o Robust execution of test scripts and modules over limited bandwidth (dial-up) network connections may be expected
- o A single OTS script may be used to control and monitor multiple OTS servers
- o Useful script variable evaluation and program control functions are provided to aid in creating intelligent scripts
- o Script debugging utility permits single step execution of created OTS scripts to aid in verifying proper script operation
- o Script Creator and Script Interpreter applications permit modifying OTS server configurations prior to execution
- o Script and Digital Lightwave-supplied test modules include automatic creation of test result files (test reports)
- o Test Scheduler may be used to schedule single or multiple scripts and test modules once or at regular intervals
- o All OTS *Toolkit*[™] applications may be executed or scheduled from an integrated user interface

Operating Basics

This section describes the operation and functionality of the OTS *Toolkit*TM user interface. The OTS *Toolkit*TM is a software utility designed to assist in the setup and running of tests on OTS product family systems.

OTS *Toolkit*TM Main Menu

The OTS *Toolkit*TM is a collection of feature sets, all accessible from the main menu. Most of these feature sets open a separate window containing its own program. All feature sets can also be accessed directly from the Start > Programs > Digital Lightwave menu.

NOTE: The user interface is a Microsoft Windows 2000® application. Information regarding standard Windows 2000 functions is beyond the scope of this document. For further information on basic commands and functions of Windows 2000, refer to the Windows 2000 manual.

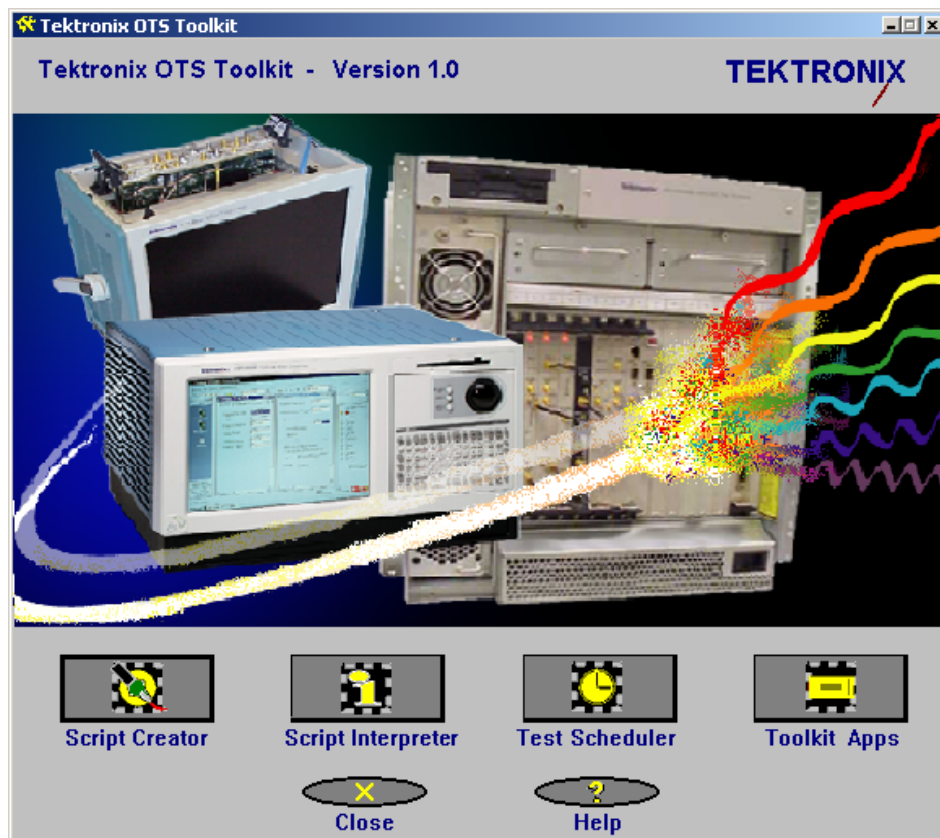


Figure 2-1: Main Menu of OTS *Toolkit*TM software

OTS Script Creator

The OTS Script Creator provides a simple drag and drop tool for creating test scripts. Figure 2-2 shows an example of the main screen. Completed scripts may also be executed from the Script Creator.

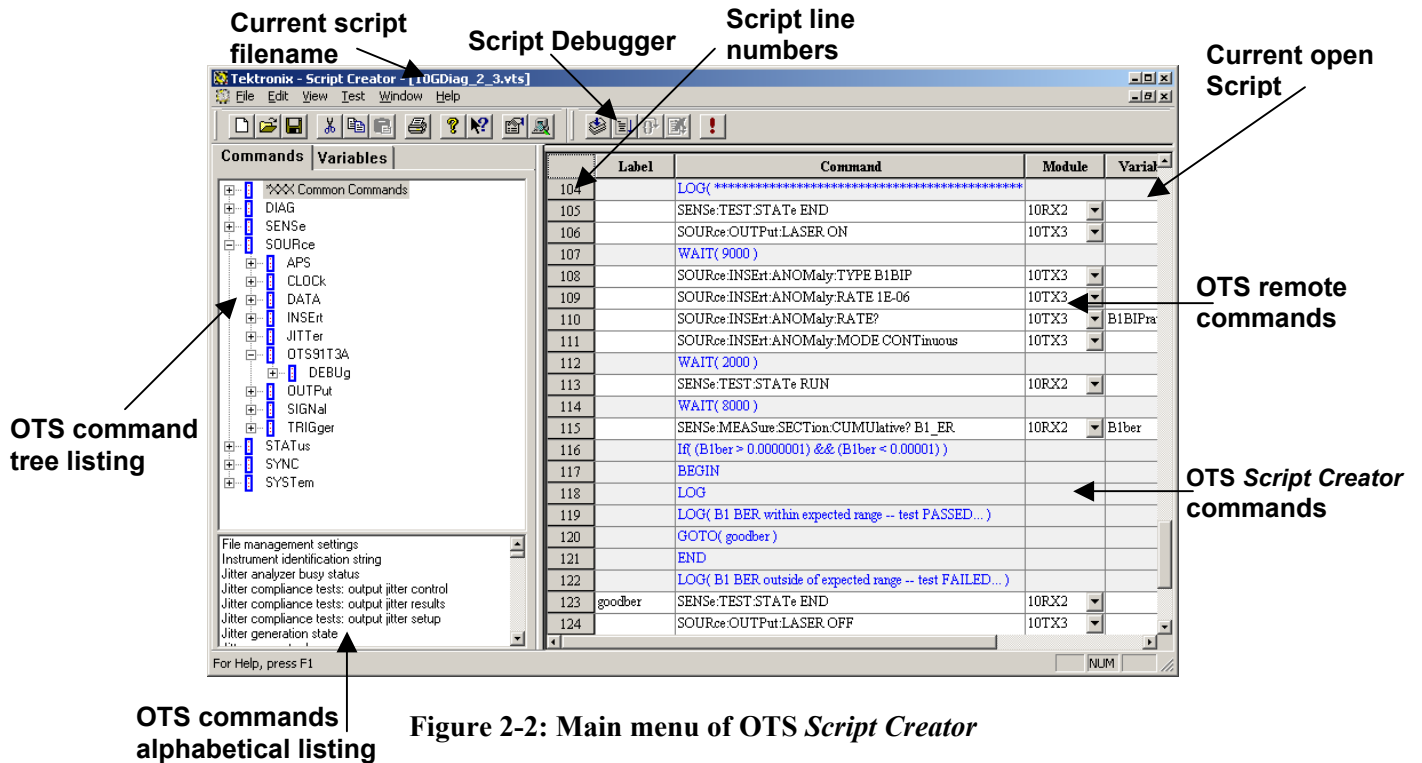


Figure 2-2: Main menu of OTS Script Creator

Commands can be dragged from the left side of the screen into the open script on the right side of the screen. The grey and blue lines are specific commands supported by the OTS Script Creator. The black and white lines are standard remote SCPI commands of the OTS product family. Green lines are remarks.

The following basic OTS script creation and execution procedures are designed to assist users in developing and executing useful OTS scripts used for custom testing applications.

NOTE: The Script Creator uses the command tree of SCPI remote commands from all available OTS systems. Depending upon the system you're using, some of these commands may be applicable to your testing environment. For detailed explanation of all commands in the command tree, refer to the individual product user manuals supplied with the OTS systems.

Learning to Script

The most efficient method of learning how to script OTS test applications is to review a previously created script and apply the programming concepts and OTS commands for similar applications. Therefore, there is a fully functioning test script included in *Appendix A* of this manual for your reference.

In addition, a number of sample scripts are included on the Installation CDrom. Refer to *Appendix C* for more information on these sample scripts.

Completed OTS scripts may be printed, or portions of a script copied and pasted into a document, all of which can assist users with understanding the test methodology used and serve as a training tool for effective OTS use.

How to Create a Script

The basic steps required to create an OTS script are as follows:

1. Decide on a test plan, i.e., what tests do you want the OTS instrument to conduct.
2. Verify the OTS hardware configuration on which you expect to conduct most, if not all, of the tests, i.e., OTS modules equipped and what OTS slots they are installed, etc.

NOTE: *A script does not necessarily have to identify OTS modules and may only include an OTS server network address.*

3. From the File menu, select New or click on the New button on the Toolbar.
4. The New script dialog box, as shown in Figure 2-3, is displayed. Select Test Script.

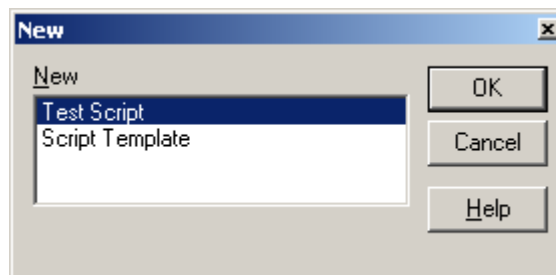


Figure 2-3: New Script dialog box

5. Next you must select the Script Template to be used. The dialog box displays all saved scripts.

NOTE: *Test templates always have a .vtt extension while test scripts always have a .vts extension.*

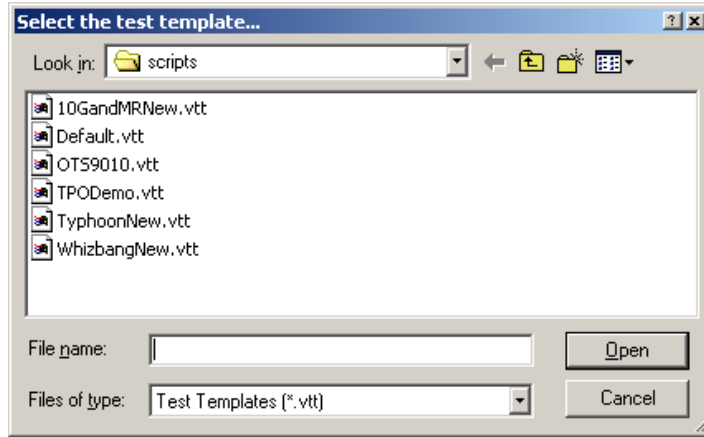


Figure 2-4: Select Test Template Dialog Box

6. Hardware resources must be identified and is the next dialog box to be displayed. Hardware resources identify the OTS network address (if applicable) and specific cards on which the script you're building will be run as well as identifying specific slot numbers for each card.

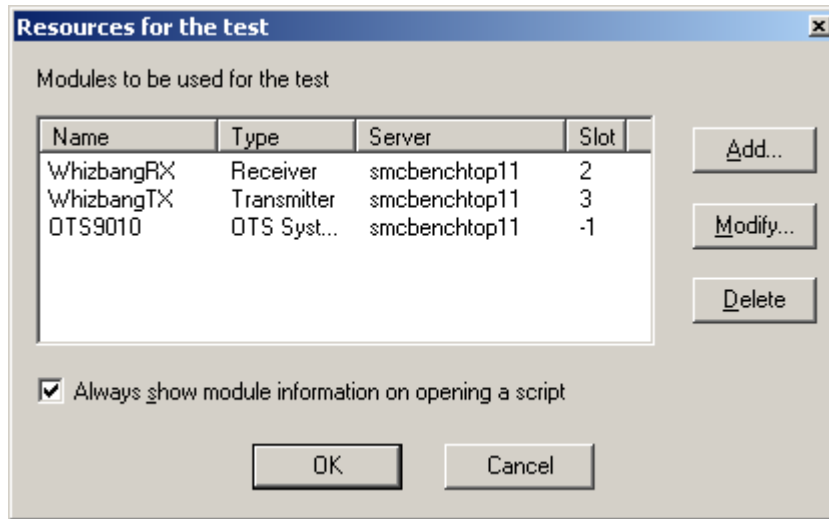


Figure 2-5: Resources for Test dialog box

7. The new script is now displayed on the right side of the screen. The script will already have some information in it, depending upon the Script Template you chose.

Modifying a Script

Modifying a test script is simply a matter of right clicking on the desired script line, including IF, FOR, WAIT, etc., control statements and use the Modify Parameters feature.

Inserting Commands into the script

Click and Drag OTS commands from the left window into the desired script line in the right window.

Moving Commands in the script

To move a command, simply highlight, then click and drag the highlighted line to the new location.

Copying Commands in the script

With the exception of script control statements, i.e., those script lines that contain BEGIN/END blocks, single or multiple script lines may be copied and pasted elsewhere in a script. To use this feature, click on a script line and highlighted area covering the line or lines you wish to copy by dragging the mouse. Release the mouse button and use the Edit menu Copy function to copy the highlighted lines into the text buffer. Then click the mouse button on the script line into which you wish to copy the buffered contents and use the Edit menu Paste function. The copied script lines will be inserted into the space or spaces immediately above the selected script line.

Deleting Commands in the script

There are two ways to delete some script lines in the script:

- o Highlight the line(s) and press the 'Delete' key. The selected line(s) will be deleted after user confirmation and the script will renumber accordingly.
- o Highlight the line, right click on the line number, and select 'Delete'. The selected line(s) will be deleted and the script will renumber accordingly.

NOTE: *Certain script functions, e.g., FOR and IF, may be deleted by highlighting the line containing the function statement, right clicking on this line, and selecting the Delete option (or by using the keyboard DEL key). This action will delete the selected script function statement and the associated BEGIN/END block lines but will not delete script lines previously contained in the function block.*

How to Create a Script Template

The basic steps required to create an OTS test script template are as follows:

1. From the File menu, select New or click on the New button on the Toolbar.
2. The New script dialog box is displayed. Select Script Template.
3. The next dialog box displayed is titled Resources for the test and refers to the hardware resources. Hardware resources identify the specific cards on which the script you're building will be run and it identifies the specific slot number for each card.
4. The new script template is now displayed on the right side of the screen. The script template starts with a default template thus some information is already displayed.

Identifying Resources

Hardware resources identify the specific cards on which the script you're building will be run as well as identifying the specific slot number for each card and one or more OTS servers to which the OTS system commands may be sent, for example, the *RST command. The hardware resources for any script or template can be modified either during the initial setup or via the menu bar. When the dialog box for Resources is displayed, as shown in Figure 2-6, all defined hardware resources for that script are shown. The user can then add, delete, and modify the contents. If the display is empty, then no resources have been previously defined.

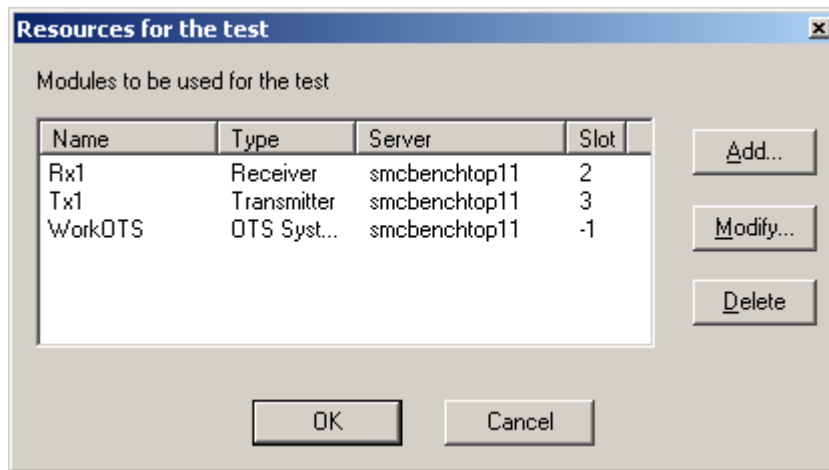


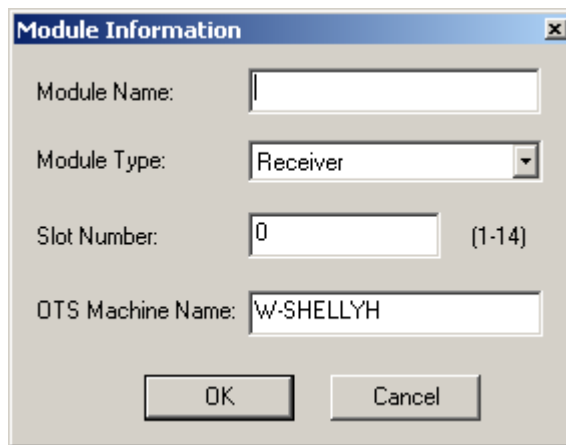
Figure 2-6: Resources for Test dialog box

Delete a Resource

To delete a resource, highlight the resource and click the Delete button. If the resource is not being used, you'll get a dialog box warning that you're about to delete a resource. Click OK and the resource will be deleted. If the resource is being used, you will not be allowed to delete it.

Add a Resource

To add a Resource, click the Add button. A dialog box is displayed requesting module information, as displayed in Figure 2-7. Once this information is filled out, click OK to add it to the resources list.



The screenshot shows a dialog box titled "Module Information". It contains the following fields and controls:

- Module Name:
- Module Type:
- Slot Number: (1-14)
- OTS Machine Name:
- Buttons: OK, Cancel

Figure 2-7: Add Resources Dialog Box

Modify a Resource

To modify a resource, highlight the resource and click the Modify button. The Module Information dialog box is displayed with the resource information filled in, as shown in Figure 2-8. Change the appropriate information and click OK to complete your modifications.

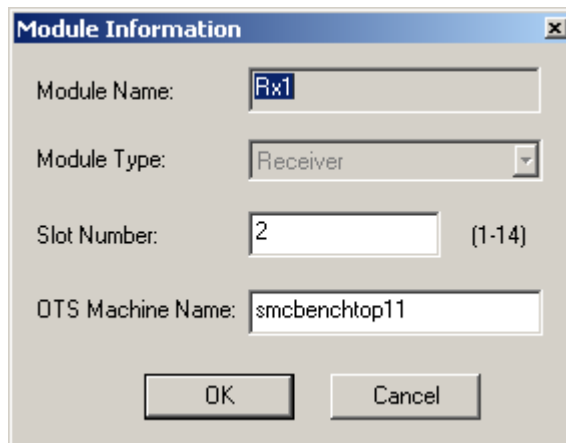


Figure 2-8: Modify Resources Dialog Box

Managing Variables

To manage variables in the current script, click the Variables tab. The listing will show any variables currently defined. If the listing is empty, no variables have yet been assigned to this script. Use the buttons along the bottom of the screen to add a new variable, delete an existing variable, or modify an existing variable.

Creation and Editing of Variables

Variables for a given script are defined and edited in the Variables tab of the docking window, as shown in Figure 2-9. Use the “New” button to create a new variable. To modify / delete an existing variable, select the variable from the list and choose the appropriate button. A variable that is in use in the active script cannot be deleted and the user will be presented with an error message.

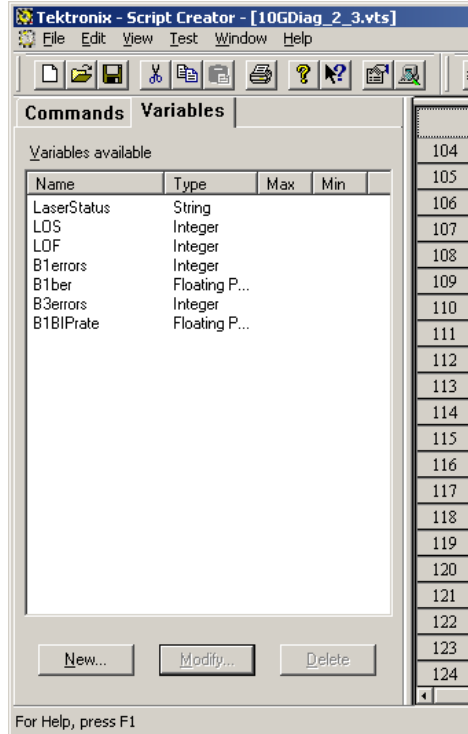


Figure 2-9: Variables Tab Display

The variable creation dialog allows the user to specify action items associated with each of the variables. Variables in OTS scripts could be of one of the following types: Integer, String, Boolean, Floating point. Also, for Integer and floating point variables, maximum and minimum values can be specified, based on which the above-mentioned actions will be triggered. Refer to Figure 2-10, as needed.

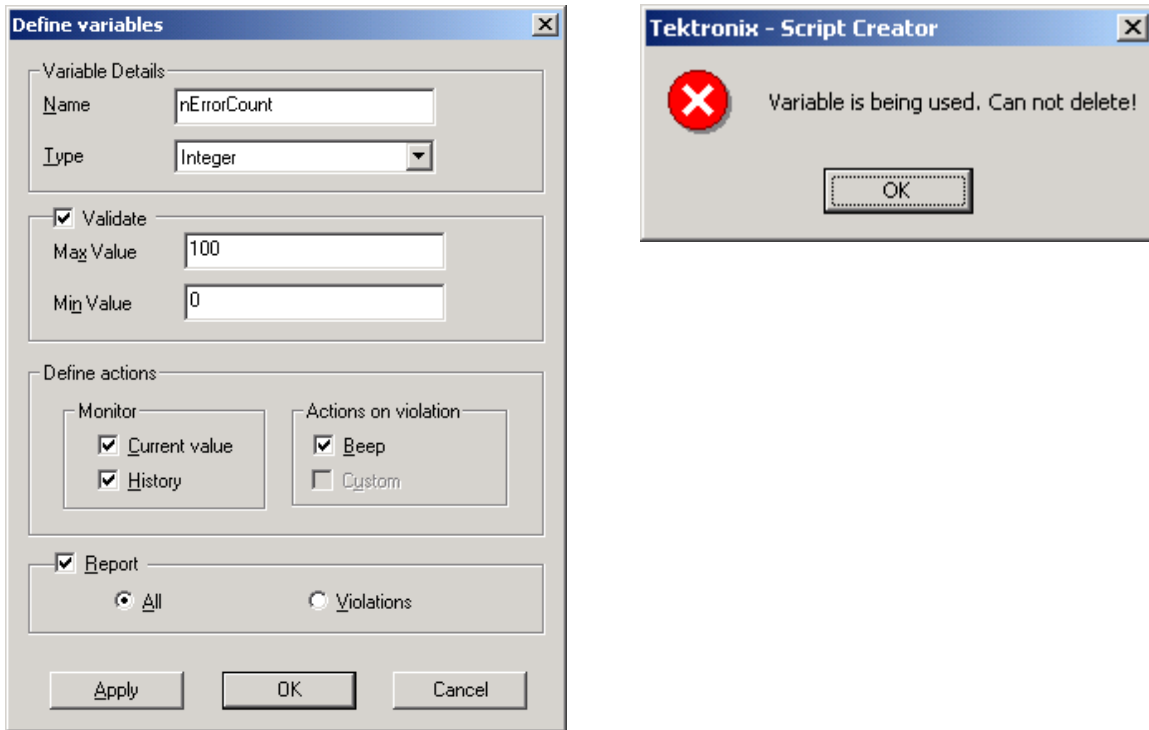


Figure 2-10: Variables Definition dialog and error when attempting to delete a variable in use

NOTE:

- o *All irretrievable actions, such as deletion of variables, etc., are preceded by a confirmation message box in which the user may elect to cancel the operation initiated.*
 - o *Variable names should contain only alphanumeric characters and underscore and cannot contain space characters. The first character of the variable should be alphabetic. The maximum number of characters allowed in the variable name is 20.*
-

Define Variables Dialog Menu

When creating or modifying variables for use in a script, the user may decide how they prefer to display and/or log variables depending on how they are used in a script. The Define Variables dialog menu shown below provides four primary variable logging options. These options are:

- Display current variable value in a variable value list.
- Display the variable in a separate “history” window that permits viewing changes in the variable value.
- Log variable value changes in the script results files (.TXT and .CSV files).
- Log variable values that exceed user specified limits (violations).

Any of the above variable display and logging options described above may be selected in the Define Variables dialog box, as shown in Figure 2-11.

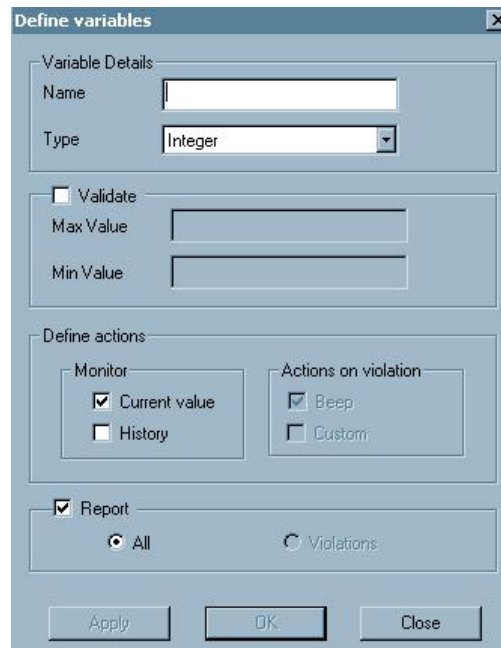


Figure 2-11: Define Variables dialog box

NOTE: Once a variable is used in a script, its definitions may be modified but the variable cannot be deleted. To delete a variable currently used in a script, delete all script lines using the variable in question or assign another variable of the same type using the variable assignment drop down dialog boxes in the Script Creator script table.

OTS Script Creator Control Statements

The OTS Script Creator uses a number of basic control statements that can be inserted into scripts created by the Script Creator tool. To access these commands, click on the right mouse button while the mouse is positioned over any line number. Keep in mind that the command will be inserted at the script line immediately above the selected line number.

ASSIGN

This control statement is used to insert assignment statements inside a script.

Syntax:

ASSIGN(expression to be evaluated)

Example:

ASSIGN (x = a + b) where x, a and b are integers.

ASSIGN (str = "SONET") where str is a string variable.

NOTE: The ASSIGN function may use any variable type and is particularly useful where the same value or string is used repeatedly in a script. Avoid a common mistake when using the script ASSIGN control statement in not using the == operator instead of the assignment = operator.

Comments

User supplied script comments may be inserted anywhere within a script. Right click on a script line and select the *Insert Comment* option. Insert the desired comment text string at the designated location. You may also edit or delete comments automatically provided with a new script template.

FINDSTR

The FINDSTR control statement does not appear in the control statement list selected by the user when right clicking on a script line. FINDSTR is a special string parsing command feature for use in evaluating string data, for example, string data returned from an OTS.

NOTE: *This command is not available through the right mouse button menu. It must be typed in by the user in the line where the user wants it.*

The FINDSTR function will search for a string subset in another string and accepts two parameters:

1. String in which to search
2. String to search for

Syntax:

```
IF (FINDSTR(<String1> , <String2>));
```

The FINDSTR function will search for *String2* in *String1*

NOTE: *<String2> may be enclosed in quotes.*

Arguments:

```
String1
    String in which to search
String2
    String to search for
```

Return Value:

Returns TRUE if string2 is found, else returns FALSE

Examples:

```
FINDSTR(strVar1,"abc")
FINDSTR(strVar1, strvar2)
FINDSTR(strVar1,"str\"1\"23")
```

Usage:

1. IF (FINDSTR(strVar1, "SDH") == TRUE)
 - LOG (Found)

NOTE: *See Boolean equivalencies described above that may be used instead of TRUE*

2. Assignment statement
 - bVar1 = FINDSTR(strVar1,"SONET")
3. Consider a string variable strVar1. Let the contents of the variable search string be the text in **bold** below

Description: **signal** is not available

In Script Creator, the FINDSTR function should appear as:

FINDSTR(strVar1,": signal")

4. Consider a string variable strVar2 where a portion of the string contains quotes:

StrVar2 = Description: "SONET" signal is not available

NOTE: *The word SONET is enclosed in quotes*

Assume we want to search for: "SONET" signal

In Script Creator the FINDSTR function should be entered as follows:

FINDSTR (strVar2,":\"SONET\"signal")

When entered into a script IF statement dialog box, the above evaluation statement will be formatted in the selected script line as:

**IF (FINDSTR(strVar2,":\"SONET\"signal"))
BEGIN
END**

Explanation:

If the search string in strVar2 contains a quote (") then it must be replaced with (\") and not entered as (") .

This is to facilitate differentiation of a string containing quotation marks embedded, against quotation marks indicating the beginning and ending of a string. This approach is consistent with C and C++ programming conventions.

FOR

This control statement is used to perform certain actions repeatedly until some user specified condition occurs. Please see the IF control statement description below for an explanation of logical operators that may be used in a FOR statement.

Syntax:

```
FOR(index = 0, index <5, index = index + 1)
BEGIN
    Statements to be repeated as long as index < 5.
END
```

NOTE: *index = user defined variable (any variable type may be used) or certain OTS SCPI commands.*

Example:

```
FOR(threshold = -100, threshold <= 100, threshold = threshold + 10)
BEGIN
    SENSE:INPUT:THREShold
    SENSE:TEST:STATe RUN
    SENSE:MEASure:PATH:CUMUlative? LSS_ES
    SENSE:MEASure:PATH:CUMUlative? PAYL_ER
    SENSE:TEST:STATe END
    WAIT(1000)
END
```

NOTE: *The above example increments the OTS Receiver Threshold Offset control in 10 unit steps starting at -100 and exits the loop when the threshold control value reaches or exceeds +100. A measurement routine included in the FOR loop records OTS measurement parameters during each iteration of the FOR loop.*

GOTO

This control statement is used to jump or skip to a specified script command line. The GOTO command uses user specified line labels for reference and may be used for limited looping functions.

Syntax:

GOTO (my label)

Example:

```
IF ((LOS != 0) || (LOF != 0))
  BEGIN
    LOG(Signal not present or degraded...)
    GOTO(my label)
  END
```

[Note: A script line Label may be typed and inserted in the script “Label” column in any line that does not contain a BEGIN, END or REM statement. To insert a line label, click the left mouse button on the desired script line Label column and type a short alphanumeric (no spaces) unique line identifier statement.]

NOTE: The GOTO “label” is a user supplied line definition statement associated with a particular script line and which the script will execute next when so directed using a GOTO control statement.

IF

This control statement is typically used to check for a specified condition and execute an action if the condition is true. If the specified condition is false, the specified action is ignored.

Syntax:

```
IF ((specified condition))
  BEGIN
    The statement(s) to be executed if the condition is true.
  END
```

Example:

```
IF ((LOS != 0) || (LOF != 0))
  BEGIN
    LOG(Signal not present or degraded...)
    GOTO(endtest)
  END
```

NOTE: The IF evaluation statement above programs the following action:

[IF the LOS variable value is NOT equal to 0 **OR** the LOF variable value is NOT equal to 0, then LOG the user supplied text and skip to the script execution line that contains the label “endtest”]

NOTE: LOGICAL OPERATORS that may be used in IF and FOR control statements:

The standard C programming logical operators, such as:

```
= =  EQUAL TO
!=   NOT EQUAL TO
<    LESS THAN
>    GREATER THAN
<=  LESS THAN OR EQUAL TO
>=  GREATER THAN OR EQUAL TO
&&  AND
||   OR
```

These operators may be used to evaluate variables. Also, note the following Boolean equivalencies that may be applied in FOR and IF control statements:

```
1 == TRUE
0 == TRUE
TRUE == 1
TRUE == 0
1 == FALSE
0 == FALSE
FALSE == 0
FALSE == 1
```

LOG

This control statement is used to insert text statements inside a script and which will be recorded in the script results file. Note that LOG statements may be invoked within IF statements and may be used to identify proper or improper OTS operation.

Syntax:

```
LOG(text string to be logged)
```

Example:

```
IF(LOS != 0.0)
  BEGIN
    LOG(Loss of Signal detected...)
    GOTO (endtest)
  END
```

POW

This function determines the value of a base expression taken to a specified power.

Syntax:

POW(base <>, exponent <>)

Arguments:

base

The base value of the expression.

exponent

The exponent value of the expression.

Return Value:

Numeric

Examples:

In the following example, a numeric expression equal to base exponent returns 1000.

POW(10,3)

Usage:

1. Assignment statement

nVar1 = POW(nVar1, nVar4);

2. IF Condition Check

IF (POW(nVar1,nVar4) > 10e-5)

LOG (Error)

REPEAT

This control statement is used to repeat a specified set of script commands a specified number of times. The script statements that are to be repeated are placed inside the BEGIN and END statements of the REPEAT block. The parameter to be supplied for the REPEAT control statement is an integer which specifies the number of times the REPEAT block statements are to be repeated.

Syntax:

```
REPEAT (5)
  BEGIN
    The script commands which are to be repeated
  END
```

Example:

```
REPEAT (5)
  BEGIN
    SENSE:TEST:STATe RUN
    WAIT( 600 )
    SENSE:MEASure:PATH:CUMUlative? LSS_ES
    SENSE:MEASure:PATH:CUMUlative? PAYL_ER
    SENSE:TEST:STATe END
    WAIT( 2000 )
    LOG( ***** )
  END
```

NOTE: *The above example makes 5 repeated OTS measurements (with a 600 millisecond pause between measurements) and stores the results in two user assigned integer variables (LSS_ES and PAYL_ER). The OTS test mode must be active when recording OTS signal measurements and requires turning the OTS test mode ON and OFF to clear any accumulated measurement values before the next measurement is made.*

RANGE

The RANGE command can be used to set a variable with different values (separated by commas) and use this variable in SCPI set commands.

Syntax:

```
RANGE(index = x, y, z)
BEGIN
    Action to be taken using RANGE variables
END
```

index = valid user defined variable (any variable type).

Example:

```
RANGE(signalStandard = SDH, SONET, BERT)
BEGIN
    WAIT(2000)
    SOURce:SIGNal:STANdard <Variable>
    SENSE:SIGNal:STANdard <Variable>
        WAIT (3000)
    SENSE:TEST:STATe RUN
    SENSE:MEASure:PATH:CUMUlative? LSS_ES
    SENSE:MEASure:PATH:CUMUlative? PAYL_ER
    SENSE:TEST:STATe END
END
```

NOTE: Choose the *signalStandard* variable in the script <Variable> column where the defined variable is to be used. This example changes the OTS Transmitter and Receiver Signal Standard format each of the three times the RANGE command loops (as defined by the RANGE command). This example might be used where a looped signal is transmitted over a transparent fiber optic channel in order to examine the performance of different signal formats.

NOTE: The above programming example loops 3 times with *index* = *x*, *index* = *y*, *index* = *z*. If *index* is an integer type, you could use: RANGE (*index* = 1, 20, 100, 100)

SQR

This function determines the square of the specified numeric expression.

Syntax:

SQR(< number >)

Arguments:

number

Specifies the numeric expression SQR () evaluates

Examples:

SQR(256)

SQR(nVar1)

Usage:

1. IF (SQR(nVar1) >= THRESHOLD_VALUE)
LOG (Critical Alarm)
2. Assignment statement:
nVar1 = SQR(nVar2) / (SQR(nMaxErrors – nErrorsFound)

SQRT

This function determines the square root of the specified numeric expression.

Syntax:

SQRT(< number >) Note: < number > cannot be negative

Arguments:

number

Specifies the numeric expression SQRT () evaluates

Examples:

SQRT (256)

SQRT (nVar1)

Usage:

1. IF (SQRT (nVar1) < MIN_VAR_VALUE)
LOG (OK)
2. Assignment statement:
nVar1 = SQRT (nVar2) + AVG_BER_VALUE

WAIT

This control statement is used to introduce WAIT times for a user-specified number of milliseconds between script execution steps. Script Creation tool requests a user supplied value (milliseconds) be entered when invoking this command

Syntax:

WAIT(1000)

Introduces a 1 second wait time

Script Debugger

To help identify faulty scripting, the *Script Creator* also comes equipped with a script debugger tool. This tool executes the script one step at a time, enabling you to follow along with the program and to see where and when specific scripting or command errors occur.

To Start the Script Debugger

To start the Script Debugger, perform the following steps:

1. Click the Script Debugger icon to start the tool.



2. The Debugger tool has started when the OTS Diagnostic Test dialog box is displayed, as shown in Figure 2-12.

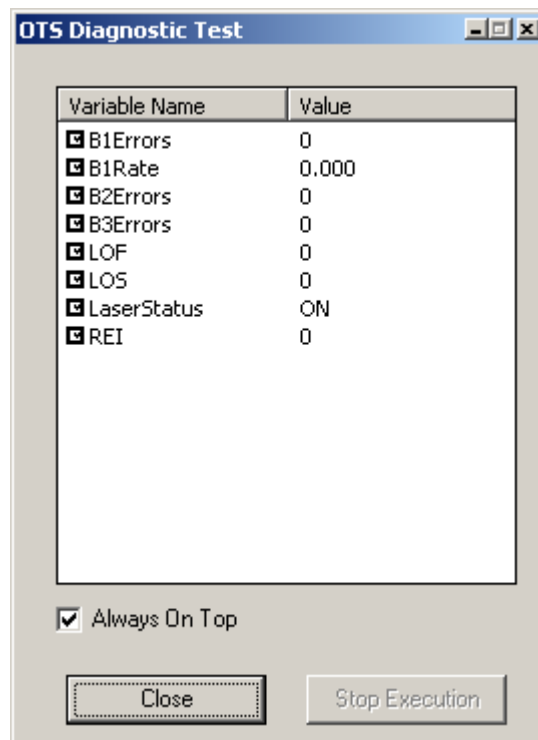
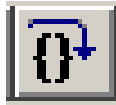


Figure 2-12: OTS Diagnostic Test dialog box

- Use the step icon to step the debugger through the script one line at a time. The yellow arrow beside the line number, as shown in Figure 2-13, indicates what step the debugger is currently performing.



NOTE: The script may also be stepped through by pressing F10.

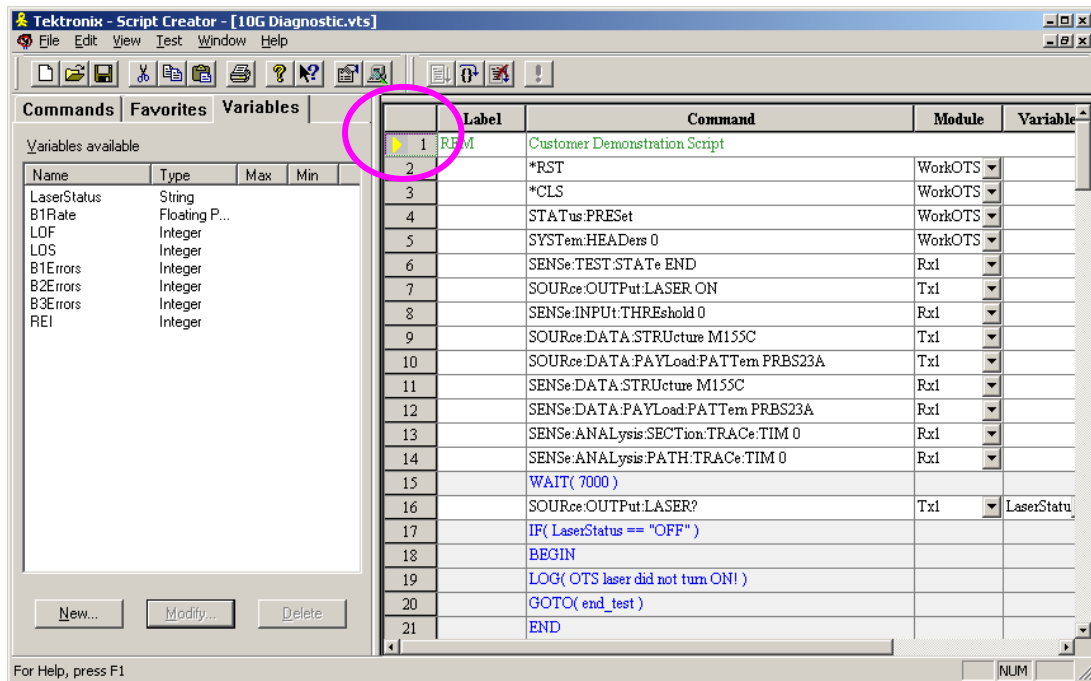
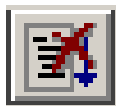


Figure 2-13: Yellow arrow indicates location of Debugger

- To stop the Debugger tool, click the Stop icon.



OTS Script Interpreter

The OTS Script Interpreter checks your written test scripts for errors and then executes the script. The Script Interpreter outputs an ASCII text results file that records the measurements taken during the test run. Note that Script Interpreter may be used in both attended and unattended modes with the primary difference being the unattended mode of operation does not require any user acknowledgement or intervention.

The features of the Script Interpreter tool include:

- o Execution of a script created by the OTS Script Creator tool.
- o Modification of OTS module and server information is accommodated in the Script Interpreter application before executing a script. Any changes made in the resources used as part of executing a script from the Script Interpreter are temporary and exist for the duration of the test only.
- o Displays the run time (current and/or historical) values of user defined variables, which are configured for monitoring.
- o Produces two types of reports (ASCII text and .CSV formats)
- o Generates an ASCII text report file that records scripted events in chronological order. This report also records any user specified LOG statements.
- o A summary .CSV (Comma Separated Values) formatted report file which displays the values of variables ordered by variable names.

If a user specified report file name is defined in the Script Creator Properties menu, the report files will include this user-supplied file name concatenated with the date and time of script execution. Scripts are automatically stored in a script results folder on the root hard-drive, which is created as part of the OTS *Toolkit*[™] installation.

NOTE: *Installation of OTS Toolkit[™] software will specify a default script report file storage path. However, if the user specifies a path as part of the script report file name in the script Properties menu, the user specified script report file storage path will be used.*

To Run a Script using Script Interpreter

When you first launch the Script Interpreter, the main screen opens, as shown in Figure 2-14. The primary purpose of this tool is to verify that a new script is operational. The Script Interpreter may also be used to immediately execute a script as opposed to scheduling one or more scripts for future execution using the Test Scheduler application. Operation of the Script Interpreter is very simple.

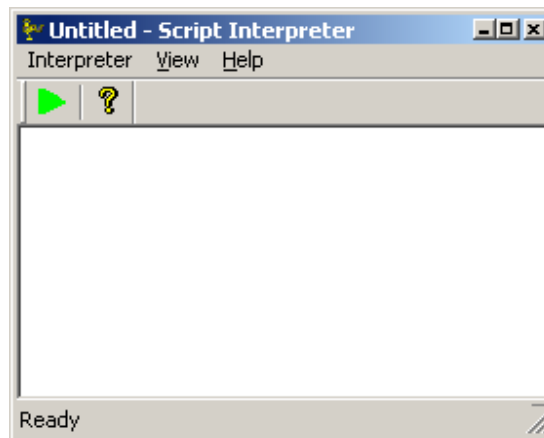


Figure 2-14: Script Interpreter main menu

To verify and execute a script, click the green arrow. You will be prompted to select a script file and to select the script options, as shown in Figure 2-15. The script options dialog box lists the hardware resources being used in the script.

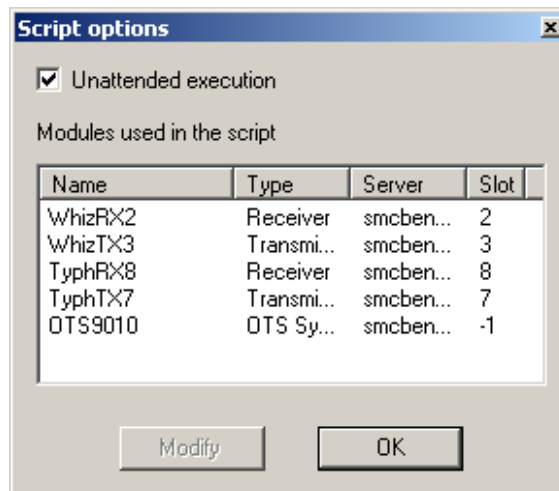


Figure 2-15: Script options dialog box

The Unattended execution checkbox is for use in conjunction with the Test Scheduler tool. If this box is checked (default setting), there is no human interaction required between the Script Interpreter script verification and the Test Scheduler implementation of the script.

An unexpected power failure or system reboot will halt any test currently in progress. There is no automatic restart. But a power failure will not prevent any future scheduled test applications from starting provided the scheduled test is capable of being executed.

NOTE: *Users are cautioned that executing any OTS script where OTS lasers are activated may constitute a human safety hazard. Users are advised that where remote OTS instruments are controlled through the use of the OTS Test Scheduler, all reasonable precautions should be taken to ensure inadvertent exposure to harmful laser energy is avoided.*

Once tests have been scheduled, the Test Scheduler application may be closed.

Deleting Scheduled Tests

Any scheduled tests may be deleted from the Test Scheduler. To delete a test, highlight the test and select Delete Test from the Scheduler menu.

Adding a Test to the Schedule

To add a test script to the schedule, click on the Clock button on the toolbar. The configure test dialog box is displayed, as shown in Figure 2-17.

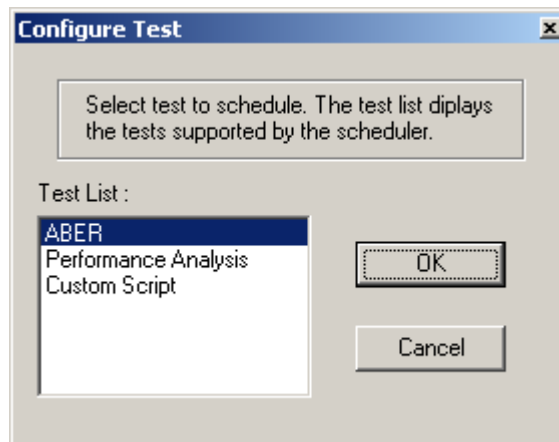


Figure 2-17: Configure test dialog box

ABER is a specific functional test provided by Digital Lightwave with the OTS *Toolkit*[™] feature set. These tests will be discussed in more detail later in this section.

If custom script is selected, the next dialog box is a standard open file dialog box from which you can browse to the desired file. Once the script is selected, the script options dialog box is displayed. The next dialog box is the Scheduling Parameters dialog box as shown in Figure 2-18.

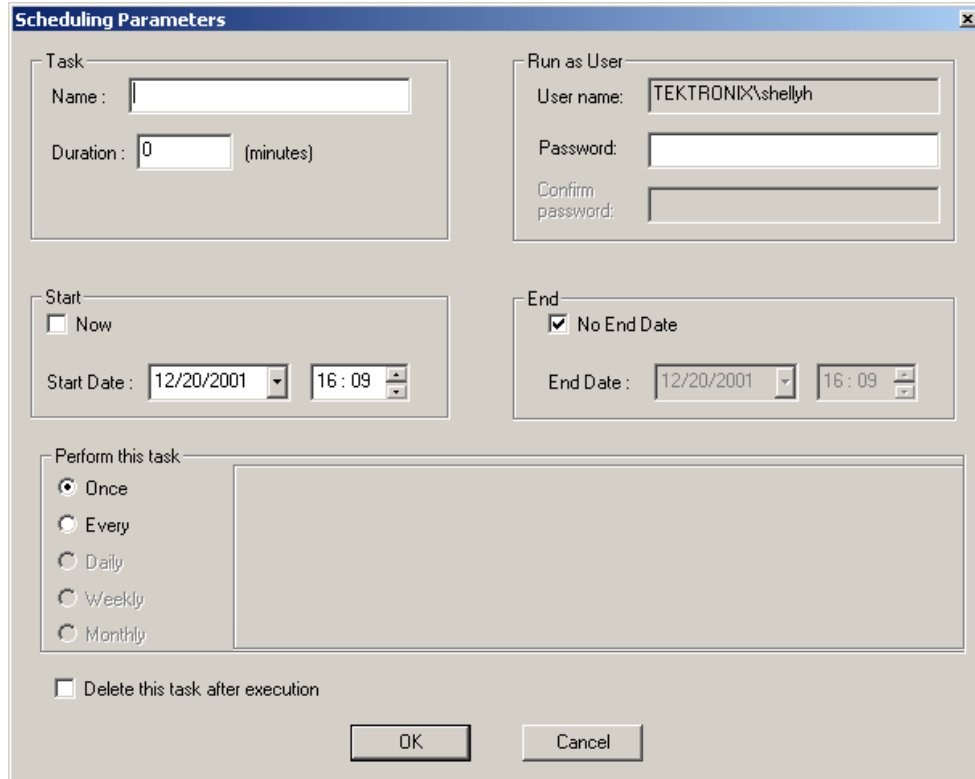


Figure 2-18: Scheduling Parameters dialog box

Viewing and Clearing the Test Log

The Log menu controls the test log. To view the Log in Notepad, select View from the Log menu. The Clear selection will delete all log entries. Each time a test is added to the schedule, it is added to the log. Log entries are also added when a test starts and finishes.

Test Scheduler Status Display Messages

The OTS Test Scheduler main screen provides a status summary of all scheduled tests. The information provided on this summary screen includes:

- o Test Name - the user supplied test name used in the Test Scheduler application
- o Next Run - indicates any future scheduled run times (repeated execution) for a given test application
- o Status - the current test status of that particular scheduled test, i.e., Scheduled, Completed, or Running
- o Last Update - the time at which the status of the last test was changed
- o Exit Message - any error messages explaining why a scheduled test did not execute

OTS *Toolkit*[™] Applications

The custom OTS *Toolkit*[™] applications are those items found under the main OTS *Toolkit*[™] main menu button marked 'Toolkit Apps'. At present, these custom applications include:

- o ABER
- o Custom Script

NOTE: Custom Scripts may be executed from:

- o Script Creator application
- o Script Interpreter application
- o Test Scheduler utility

Digital Lightwave supplied test modules, e.g., the ABER test, may be executed from:

- o OTS *Toolkit*[™] main screen Test Apps menu
- o Test Scheduler utility

Custom Script

Custom Script provides an immediate access to start running the *Script Interpreter* application on any previously saved script.

ABER (Accelerated BER)

The ABER test is designed to automatically perform stress testing using the Accelerated Bit Error Rate (ABER) technique. The test automatically shifts the OTS receiver detection threshold level above and below a nominal point in order to simulate signal stress. The ABER test process eventually results in offset levels sufficiently removed from optimum so that data errors are seen in the detected signal. Digital signals that contain accumulated stress before having the OTS receiver threshold offset levels shifted will show error activity sooner than more robust signals since the process of shifting receiver threshold levels emulates locally generated stress.

While this measurement is designed to accurately estimate signal robustness, some knowledge is required to ensure a correct analysis of the test results are made. The Reference section of this manual provides more detailed information regarding the functionality of the ABER test.

Scheduling the ABER test

To schedule the ABER test from the *Test Scheduler*, perform the following steps:

1. Click the clock icon and then select ABER from the Configure Test dialog box.
2. From the server selection dialog box, as shown in Figure 2-19, select the name of the OTS system to be tested.

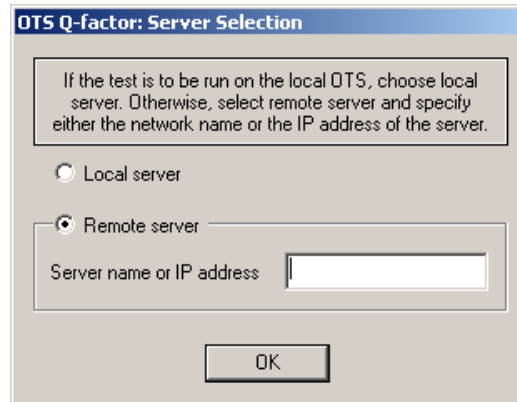


Figure 2-19: Server Selection dialog box

3. The Instrument setup dialog box is now displayed, as shown in Figure 2-20. Fill out the required information and click next to continue to the next page.

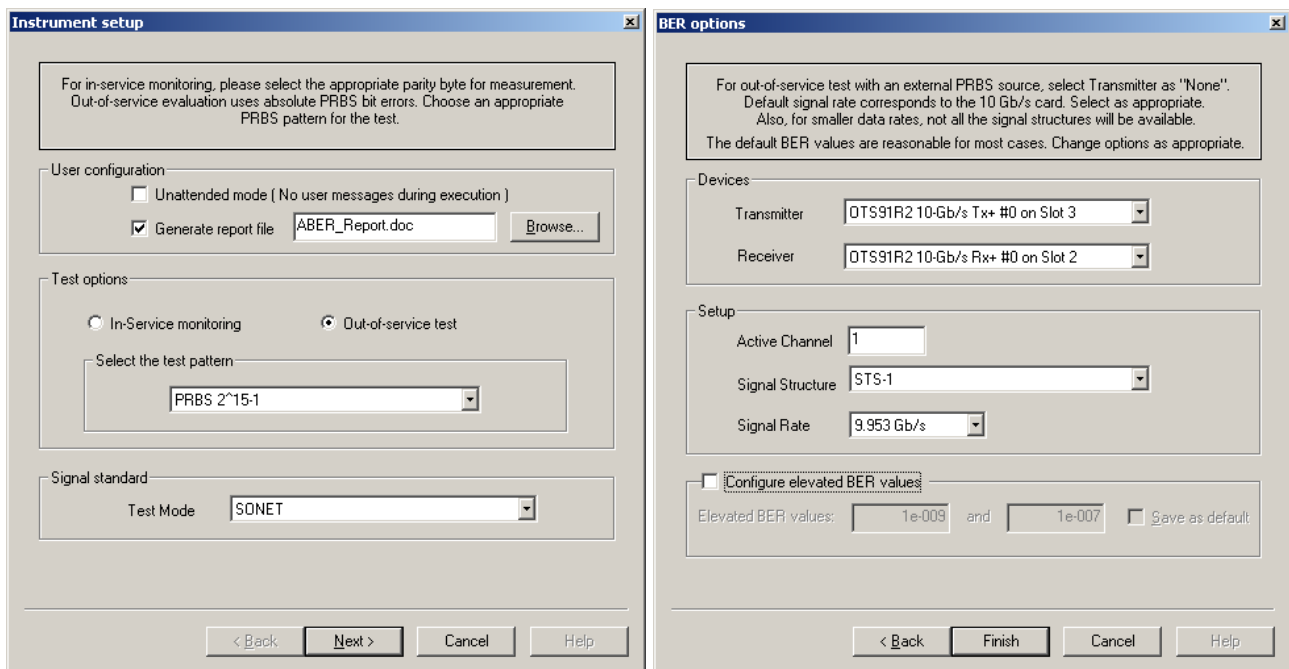


Figure 2-20: Instrument setup dialog box

4. The second page of instrument setup information is now displayed, as shown in Figure 2-20. Complete the form and click Finish.
5. The last step is to fill out the schedule parameters dialog box. The ABER test application will start as soon as the Finish button is clicked.

Reference

This section provides additional information to assist the user in learning and using the OTS *Toolkit*TM software utility.

General Scripting Notes

For best results, examine the sample scripts supplied and adapt the techniques used in them to customize and enhance test and measurement scripts of your own design. The following general script creation notes are offered to assist users in creating efficient OTS custom test scripts:

- ?? While a script template that contains several OTS modules may be used to create a script, you should modify the “modules used” properties in the Script Creator utility to delete any unused modules. This simplifies script creation in that Script Creator will automatically populate OTS server and “Modules Used” script fields with the first module listed which may not be the module used in the script.
- ?? When creating script variables, decide beforehand whether you prefer to have these variables displayed in the variable display window (history) or in the list of current variables, or both. These selections are made by clicking or un-clicking the variable Current or History check boxes.
- ?? Note that use of the ASSIGN function will automatically log the assigned variable value when and where the associated variable is used in a script. Aside from mathematical operations, this feature is useful for displaying script execution status in the script variable display window during script execution by using appropriate string variables.
- ?? Inserting a blank LOG statement will produce a blank line in the script results file printout.
- ?? Any variable currently used in a script line cannot be deleted while being used in a script. In order to delete a variable, any script line that uses this variable must be deleted or have the variable not selected (temporarily select another variable).
- ?? Note that most OTS SCPI command arguments may be either entered directly as values, i.e., NR1, NR2, etc., or by identifying a defined variable. Note that the variable types must match, i.e., integer, floating point or string.
- ?? Most OTS SCPI command arguments that require large positive or negative number arguments may be entered using either a decimal number or in a scientific notation format, i.e., NR3. For example, the value 1.0E-06 may also be entered as 0.000001, and vice-versa.
- ?? It is highly recommended that script creation contain descriptive “comments” entered wherever an explanation might help better explain script operation. To insert a comment on the line above, left click on a script line number (highlight), then right click and select “Insert Comment”.

- ?? Avoid infinite loops using the GOTO command since stopping a script under these circumstances may require use of the CTRL-ALT-DEL function to exit the script.
- ?? Most scripts created for 10G modules will work equally well when used with 2.5G (multi-rate modules) although there are a few commands that may require modification. To modify a script created for 10G modules for use with 2.5G modules, add the 2.5G modules in the resources list and use the drop down “Modules” script dialog windows to select the added modules. Save the script using the “Save As” function to identify the new script function. Carefully test the modified script to ensure all script control functions work properly.

Explanation of the server timeout feature

Prior to executing a script in the attended mode of operation, either from the Script Creator or Script Interpreter utility, the user is allowed to change the 30 second default "server timeout" value. This permits more efficient script execution over congested data networks or where OTS invalid commands may have been received by one or more remote OTS units addressed.

In the case where an invalid OTS command is received, e.g., where a SCPI command is sent to an unequipped slot, etc., the application may not be aware an error condition has been encountered unless the *ESR? (Error Status Register) query is sent and the OTS response evaluated. The timeout value change permitted is used to send an *ESR? query after a user-specified (or default) time interval to determine if a delayed response is due to an invalid command or a network communications problem. If the *ESR? query indicates the delay is due to an invalid command, the user will be allowed to acknowledge the error (attended mode) after which the script will continue to execute as before. If the *ESR? query does not result in a response from the remote OTS addressed, script execution will wait for a server response for 20 seconds and either terminate script execution (no response condition), display an invalid command error condition if the *ESR? query indicates this is the case, or continue with script execution if a response to the initial command was received before the timeout interval has expired.

In the unattended mode of operation, timeout operation remains the same except user acknowledgement of an invalid command condition is not required. Any invalid command response event detected in the unattended mode will be written to the results file, which includes logging the *ESR? error code and the command sent that resulted in this event.

Read-only script file attribute

Files copied to CD-R or CD-RW media may have their file attributes changed to Read-only. For example, if you modify and attempt to save a test script only to receive an indication the *Save* or *Save As* action did not complete, the read only attribute may be active. For this situation, try the following:

1. Open Windows Explorer or Windows File Manager and highlight (single click) the script .vts file in the file tree display.
2. Right click and select "Properties".
3. If you find the file "Read-only" attribute checked, uncheck this attribute. This should allow changes made in the script file to be saved after modification.

Script Compile function

You may separately compile a script without executing the script using the CTRL + F7 keys or from the Script Creator main screen Test menu. This feature is useful when you do not have immediate access to an OTS instrument. Please note that all script execution, either from Script Creator (Execute Test or CTRL + F5), or when using Script Interpreter, performs a script compile check prior to execution.

Copying and Pasting script lines

A very useful *copy + paste* feature is provided and is useful where the same OTS command or commands, and those script control functions not furnished with BEGIN/END blocks, may be copied to multiple locations elsewhere in a script. This feature allows single or multiple script lines to be copied into a buffer and inserted elsewhere in the same or in another script. See the OTS *Toolkit*TM help documentation for details on how this feature may be used.

Modifying scripts for use with multiple OTS instruments

Scripts created for a particular OTS hardware configuration, i.e., modules installed, server addresses, etc., are easily modified at run time in the Script Interpreter application or may be permanently changed in the Script Creator application. It is usually best to first test a script using the Script Creator application where changes in the *Resources Used* dialog may be made. After successful script execution using Script Creator, save the script using the *Save As* feature and provide a unique script filename that identifies the new configuration. This action will keep both the new and old script files and their associated hardware configurations available for future use.

Searching data strings using the FINDSTR function

You can set the script to search data strings using the FINDSTR function in Script Creator. With the exception of certain OTS system commands, such as *LRN? and *IDN?, most OTS SCPI query commands directed to OTS test modules where string data is returned have the string data enclosed in quotes. For example, the command, SENSE:DATA:STRUCTURE?, might return "M155C". In these cases and because the OTS returned string data is encased in quotes, an IF statement string evaluation cannot be used since the (") character is used as a delimiter by the IF evaluation process. The FINDSTR function should be used in those cases where OTS returned string data are encased in quotes.

See the OTS *Toolkit*TM help documentation and the sample script "System_Discovery.vts" for additional information. Please note that OTS responses in this case are consistent with IEEE-488.2 convention in that instrument command responses that cannot be sent back to an instrument as valid commands should be enclosed in quotes.

NOTE: *Where OTS string responses are not enclosed in quotes, use the standard IF(strvar = "ots response") evaluation. Where OTS string responses are encased in quotes, use the IF(FINDSTR(strvar, "ots response")) == TRUE evaluation approach.*

ABER Error Messages

During the ABER application test, a number of error messages can be triggered. These error messages are identified and briefly described below.

1. "Signal is absent or out of ABER test range boundaries. Please verify test connections and/or approximate signal quality. See ABER help documentation for additional test suggestions."

Context: When the routine encounters LOS, LOF, or LSS conditions during the test. The signal is too degraded for a meaningful execution of the test. This condition can happen at any stage of execution.

2. "Another test is already running on the OTS. Unable to execute ABER test."

Context: A test is already running on the target Rx module. Hence, the ABER routine returns without disturbing anything.

3. "Test conditions have crossed allowable boundaries. Unable to Complete the ABER test. Please retry with lower values of BER. See ABER help documentation for additional test suggestions."

Context: When the threshold offset value has reached the maximum allowable value on either side of the zero point.

4. "OTS laser has been locked out and hence can not be switched on. Test abandoned. Please unlock the laser and retry the test."

Context: Laser locked out with physical key.

5. "Serious error. No specific error description available."

Context: Any generic error, not specifically handled/known by the ABER routine.

Appendix A: Sample Script

This section provides a fully functioning sample script for reference. This sample script was built using the *OTS Script Creator* and verified functionally using the *OTS Script Interpreter*. The script is included to help you understand the mechanics of scripting for the OTS systems.

In addition, completed OTS scripts may be printed, or portions of a script copied and pasted into a document, all of which can assist users with understanding the test methodology used and serve as a training tool for effective OTS use.

The table below lists the entire script in a straight text table format.

Appendix B provides an explanation of the functionality of this script.

Label	Command	Module	Variable
	*RST	OTS9010	
	*CLS	OTS9010	
	STATus:PRESet	OTS9010	
	SYSTem:HEADers 0	OTS9010	
	SENSe:TEST:STATe END	Rx1	
	SOURce:OUTPut:LASER ON	Tx1	
	SENSe:INPUt:THREshold 0	Rx1	
	SOURce:DATA:STRUcture M155C	Tx1	
	SOURce:DATA:PAYLoad:PATTern PRBS23A	Tx1	
	SENSe:DATA:STRUcture M155C	Rx1	
	SENSe:DATA:PAYLoad:PATTern PRBS23A	Rx1	
	SENSe:ANALysis:SECTion:TRACe:TIM 0	Rx1	
	SENSe:ANALysis:PATH:TRACe:TIM 0	Rx1	
	WAIT(7000)		
	SOURce:OUTPut:LASER?	Tx1	
	LaserStatus		
	IF(LaserStatus == "OFF")		
	BEGIN		
	LOG(OTS laser did not turn ON!)		
	GOTO(end_test)		
	END		
	SENSe:TEST:STATe RUN	Rx1	
	WAIT(5000)		
	SENSe:MEASure:SECTion:CUMUlative? LOS_ES	Rx1	LOS
	SENSe:MEASure:SECTion:CUMUlative? LOF_ES	Rx1	LOF
	SENSe:MEASure:SECTion:CUMUlative? B1_C	Rx1	B1Errors
	SENSe:MEASure:LINE:CUMUlative? B2_C	Rx1	B2Errors
	SENSe:MEASure:PATH:CUMUlative? B3_C	Rx1	B3Errors
	IF(LOS == 0.0) && (LOF == 0.0) && (B1Errors == 0.0) && (B3Errors == 0.0)		
	BEGIN		
	LOG(No errors detected - continue testing...)		
	END		
	IF(LOS != 0.0) (LOF != 0.0)		

Appendix A: Sample Script

BEGIN		
GOTO(end_test)		
END		
LOG(Single error test - 5 B1 errors inserted...)		
LOG(-----)		
SENSE:TEST:STATe END	Rx1	
SOURCE:INSErt:ANOMaly:TYPE B1BIP	Tx1	
WAIT(1000)	Rx1	
SENSE:TEST:STATe RUN		
WAIT(2000)		
REPEAT(5)		
BEGIN		
SOURCE:INSErt:ANOMaly:MODE SINGLE	Tx1	
WAIT(2000)		
END		
SENSE:MEASure:SECTion:CUMULative? LOS_ES	Rx1	LOS
SENSE:MEASure:SECTion:CUMULative? LOF_ES	Rx1	LOF
SENSE:MEASure:SECTion:CUMULative? B1_C	Rx1	B1Errors
SENSE:MEASure:LINE:CUMULative? B2_C	Rx1	B2Errors
SENSE:MEASure:PATH:CUMULative? B3_C	Rx1	B3Errors
IF((LOS == 0.0) && (LOF == 0.0) && (B1Errors == 5.0) && (B3Errors == 0.0))		
BEGIN		
LOG(Single error test with 5 B1 errors inserted - PASSED)		
END		
IF((LOS != 0.0) (LOF != 0.0) (B1Errors != 5.0) (B2Errors != 0.0) (B3Errors != 0.0))		
BEGIN		
LOG(Single error test with 5 B1 errors inserted - FAILED)		
END		
LOG(Single error test... 5 B3 errors inserted...)		
LOG(-----)		
SENSE:TEST:STATe END	Rx1	
SOURCE:INSErt:ANOMaly:TYPE B3BIP	Tx1	
SENSE:TEST:STATe RUN	Rx1	
WAIT(2000)		
REPEAT(5)		
BEGIN		
SOURCE:INSErt:ANOMaly:MODE SINGLE	Tx1	
WAIT(2000)		
END		
SENSE:MEASure:SECTion:CUMULative? LOS_ES	Rx1	LOS
SENSE:MEASure:SECTion:CUMULative? LOF_ES	Rx1	LOF
SENSE:MEASure:SECTion:CUMULative? B1_C	Rx1	B1Errors
SENSE:MEASure:LINE:CUMULative? B2_C	Rx1	B2Errors
SENSE:MEASure:PATH:CUMULative? B3_C	Rx1	B3Errors
IF((LOS == 0.0) && (LOF == 0.0) && (B1Errors == 0.0) && (B3Errors == 5.0))		
BEGIN		
LOG(Single error test with 5 B3 errors inserted - PASSED)		

END		
IF(LOS != 0.0) (LOF != 0.0) (B1Errors != 0.0) (B2Errors != 0.0) (B3Errors != 5.0))		
BEGIN		
LOG(Single error test with 5 B3 errors inserted - FAILED)		
END		
LOG(LOS test - 128 mSec burst...)		
LOG(-----)		
SENSe:TEST:STATe END	Rx1	
SOURce:INSErt:DEFECt:MODE NONE	Tx1	
SOURce:INSErt:DEFECt:TIME 128	Tx1	
SOURce:INSErt:DEFECt:TYPE LOS	Tx1	
SENSe:TEST:STATe RUN	Rx1	
WAIT(2000)		
REPEAT(2)		
BEGIN		
SOURce:INSErt:DEFECt:MODE BURSt	Tx1	
WAIT(3000)		
END		
SENSe:MEASure:SECTion:CUMUlativE? LOS_ES	Rx1	LOS
IF(LOS != 2.0)		
BEGIN		
LOG(LOS insertion FAILED: expected LOS events is 2)		
LOG(-----)		
END		
IF(LOS == 2.0)		
BEGIN		
LOG(LOS insertion PASSED: 2 LOS events expected and received)		
LOG(-----)		
END		
LOG(Error rate test - approximate 1.0E-6 B1 BER generated)		
LOG(-----)		
SENSe:TEST:STATe END	Rx1	
SOURce:INSErt:ANOMaly:TYPE B1BIP	Tx1	
SOURce:INSErt:ANOMaly:RATE 1.0E-06	Tx1	
WAIT(1000)		
SENSe:TEST:STATe RUN	Rx1	
WAIT(1000)		
SOURce:INSErt:ANOMaly:MODE CONTInuous	Tx1	
WAIT(5000)		
SENSe:MEASure:SECTion:CUMUlativE? LOS_ES	Rx1	LOS
SENSe:MEASure:SECTion:CUMUlativE? LOF_ES	Rx1	LOF
SENSe:MEASure:SECTion:CUMUlativE? B1_ER	Rx1	B1Rate
SENSe:MEASure:LINE:CUMUlativE? REI_C	Rx1	REI
IF(B1Rate < 0.0000009) (B1Rate > 0.00009))		
BEGIN		
LOG(Error rate test FAILED: Expected 9.9E-07 to 1.01E-06 B1 BER)		
END		
IF(B1Rate >= 0.0000009) && (B1Rate <= 0.00009))		
BEGIN		

Appendix A: Sample Script

	LOG(Error rate test PASSED: B1 BER within expected range)		
	END		
	LOG(-----)		
	SOURce:INSErt:ANOMaly:MODE NONE	Tx1	
	SOURce:INSErt:DEFEct:MODE NONE	Tx1	
end_test	SENSE:TEST:STATe END	Rx1	
	SOURce:OUTPut:LASER OFF	Tx1	
	DURING_TEST		
	BEGIN		
	*RST		
	OTS9010		
	*CLS		
	OTS9010		
	STATus:PRESet		
	OTS9010		
	END		

NOTES:

- o The term “OTS9010” references the particular OTS server name associated with a networked OTS instrument, i.e., referenced to a network TCP/IP address.
- o Tx1 and Rx1 designations reference OTS Transmitter and Receiver 10G modules installed in specified OTS cage slots in the OTS instrument being controlled.

Appendix B: Sample Script Description

This section provides a detailed description of the sample script provided in *Appendix A*. This explanation was originally generated using the *OTS Script Interpreter* tool. Statements inside [brackets] indicate notations that have been added for clarification.

***** OTS Diagnostic Test Routine Script *****

```
<TESTNAME>OTS Diagnostic Script</TESTNAME>
<DATE>Wednesday, October 10, 2001</DATE>
<AUTHOR>Vani.K. (TEDI)</AUTHOR>
<DESCRIPTION>Digital Lightwave Factory Diagnostic Test
<REMARKS>Based on D. Matte's C++ OTS Routine</REMARKS>
```

Variable definitions:

LaserStatus (defined as **string** variable)

B1Rate (defined as **floating point** variable)

LOF (defined as **integer** variable)

LOS (defined as **integer** variable)

B1Errors (defined as **integer** variable)

B2Errors (defined as **integer** variable)

B3Errors (defined as **integer** variable)

REI (defined as **integer** variable)

OTS Hardware configuration definition:

```
<SLOTNUM>2</SLOTNUM> Rx1
```

```
<SLOTNUM>3</SLOTNUM> Tx1
```

Start of script execution...

```
*RST
```

```
STATus:PRESet
```

```
SYSTem:HEADers 0
```

```
SENSE:TEST:STATE END
```

Appendix B: Sample Script Description

```
SOURce:OUTPut:LASER ON
SENSE:INPUt:THREShold 0
SOURce:DATA:STRUcture M155C
SOURce:DATA:PAYLoad:PATtern PRBS23A
SENSE:DATA:STRUcture M155C
SENSE:DATA:PAYLoad:PATtern PRBS23A
SENSE:ANALYsis:SECTion:TRACe:TIM 0
SENSE:ANALYsis:PATH:TRACe:TIM 0
```

WAIT 7000

```
SOURce:OUTPut:LASER? [LaserStatus]
```

```
IF LaserStatus == "OFF"
```

BEGIN

```
LOG "OTS laser did not turn ON!"
```

END

[IF statement: If the string variable "LaserStatus" is "OFF", then display the message "OTS laser did not turn ON!"]

```
SENSE:TEST:STATE RUN
```

WAIT 5000

```
SENSE:MEASure:SECTion:CUMUlative? LOS_ES [LOS]
```

```
SENSE:MEASure:SECTion:CUMUlative? LOF_ES {LOF}
```

```
SENSE:MEASure:SECTion:CUMUlative? B1_C [B1Errors]
```

```
SENSE:MEASure:LINE:CUMUlative? B2_C [B2Errors]
```

```
SENSE:MEASure:PATH:CUMUlative? B3_C [B3Errors]
```

```
IF (LOS == 0.0) && (LOF == 0.0) && (B1Errors == 0.0) && (B3Errors == 0.0)
```

BEGIN

```
LOG "No errors detected - continue testing..."
```

END

[IF statement: If the LOS, LOF, B1Errors and B3Errors variables equal 0, then display the message "No errors detected - continue testing..."]

```
LOG "Single error test - 5 B1 errors inserted..."
```

```
SENSE:TEST:STATE END
```

```
SOURce:INSErt:ANOMaly:TYPE B1BIP
```

```
WAIT 1000
```

```
SENSe:TEST:STATE RUN
```

```
WAIT 2000
```

```
FOR 1,1,5
```

```
  BEGIN
```

```
    SOURce:INSErt:ANOMaly:MODE SINGLE
    WAIT 2000
```

```
  END
```

```
[FOR statement: Insert a B1 parity error (anomaly) 5 times in a loop, pausing 2 seconds between each anomaly insertion]
```

```
SENSe:MEASure:SECTIon:CUMUlate? LOS_ES [LOS]
```

```
SENSe:MEASure:SECTIon:CUMUlate? LOF_ES {LOF}
```

```
SENSe:MEASure:SECTIon:CUMUlate? B1_C [B1Errors]
```

```
SENSe:MEASure:LINE:CUMUlate? B2_C [B2Errors]
```

```
SENSe:MEASure:PATH:CUMUlate? B3_C [B3Errors]
```

```
  IF (LOS == 0.0) && (LOF == 0.0) && (B1Errors == 5.0) && (B3Errors == 0.0)
```

```
    BEGIN
```

```
      LOG "Single error test with 5 B1 errors inserted - PASSED"
```

```
    END
```

```
[IF statement: If B1Errors equals 5, and LOS, LOF and B3Errors are equal to 0, display PASS message]
```

```
  IF (LOS != 0.0) || (LOF != 0.0) || (B1Errors != 5.0) || (B3Errors != 0.0)
```

```
    BEGIN
```

```
      LOG "Single error test with 5 B1 errors inserted - FAILED"
```

```
    END
```

```
[IF statement: If LOS, LOS, and B3 Errors are NOT equal to 0, and if B1Errors is NOT equal to 5, then display FAIL message]
```

```
LOG "Single error test... 5 B3 errors inserted..."
```

```
SENSe:TEST:STATE END
```

```
SOURce:INSErt:ANOMaly:TYPE B3BIP
```

```
SENSe:TEST:STATE RUN
```

Appendix B: Sample Script Description

WAIT 2000

```
FOR 1,1,5
```

```
BEGIN
```

```
SOURce:INSErt:ANOMaly:MODE SINGLE
```

```
WAIT 2000
```

```
END
```

```
[FOR statement: Repeat of B1 error insertion routine for B3 errors]
```

```
SENSe:MEASure:SECTIon:CUMUlative? LOS_ES [LOS]
```

```
SENSe:MEASure:SECTIon:CUMUlative? LOF_ES [LOF]
```

```
SENSe:MEASure:SECTIon:CUMUlative? B1_C [B1Errors]
```

```
SENSe:MEASure:LINE:CUMUlative? B2_C [B2Errors]
```

```
SENSe:MEASure:PATH:CUMUlative? B3_C [B3Errors]
```

```
IF (LOS == 0.0) && (LOF == 0.0) && (B1Errors == 0.0) && (B3Errors == 5.0)
```

```
BEGIN
```

```
LOG "Single error test with 5 B3 errors inserted - PASSED"
```

```
END
```

```
IF (LOS != 0.0) || (LOF != 0.0) || (B1Errors != 0.0) || (B2Errors != 0.0) || (B3Errors != 5.0)
```

```
BEGIN
```

```
LOG "Single error test with 5 B3 errors inserted - FAILED"
```

```
END
```

```
LOG "LOS test - 128 mSec burst..."
```

```
SENSe:TEST:STATe END
```

```
SOURce:INSErt:DEFECT:MODE NONE
```

```
SOURce:INSErt:DEFECT:TIME 128
```

```
SOURce:INSErt:DEFECT:TYPE LOS
```

```
SENSe:TEST:STATe RUN
```

WAIT 2000

```
FOR 1,1,2
```

```
BEGIN
```

```
SOURce:INSErt:DEFECT:MODE BURSt
```

```

        WAIT 3000
    END
    [FOR statement:  Generate 2 LOS events - 128 mSec bursts]
SENSE:MEASure:SECTION:CUMUlation? LOS_ES  [LOS]

    IF LOS != 2.0

        BEGIN
            LOG "LOS insertion FAILED:  expected LOS events is 2"
        END

    IF LOS == 2.0

        BEGIN
            LOG "LOS insertion PASSED:  2 LOS events expected and received"
        END
    [IF statements:  The two previous IF statements verify proper OTS behavior in
    detecting the two LOS events generated]

    LOG "Error rate test - approximate 1.0E6 B1 BER generated"

    SENSE:TEST:STATE END

    SOURCE:INSErt:ANOMaly:TYPE B1BIP

    SOURCE:INSErt:ANOMaly:RATE 1.0E06
    [SOURCE command generates B1 parity errors at an approximate 1E-06 BER]

WAIT 1000

    SENSE:TEST:STATE RUN

WAIT 1000

    SOURCE:INSErt:ANOMaly:MODE CONTinuous

WAIT 5000

    SENSE:MEASure:SECTION:CUMUlation? LOS_ES  [LOS]
    SENSE:MEASure:SECTION:CUMUlation? LOF_ES  [LOF]
    SENSE:MEASure:SECTION:CUMUlation? B1_ER  [B1Rate]
    SENSE:MEASure:LINE:CUMUlation? REI_C  [REI]

    IF (B1Rate < 0.0000009) || (B1Rate > 0.00009)

        BEGIN
            LOG "Error rate test FAILED:  Expected 9.9E07 to 1.01E06 B1
BER"
        END

```

Appendix B: Sample Script Description

```
IF (B1Rate >= 0.0000009) && (B1Rate <= 0.00009)
```

```
BEGIN
```

```
LOG "Error rate test PASSED: B1 BER within expected range"
```

```
END
```

```
[IF statements: The previous two IF statements verify the B1 parity  
BER is within the expected range]
```

```
SOURce:INSERt:ANOMaly:MODE NONE
```

```
SOURce:INSERt:DEFECT:MODE NONE
```

```
SENSE:TEST:STATE END
```

```
SOURce:OUTPut:LASER OFF
```

```
.....  
End of script execution...
```

Script Results File

The following ASCII text results file is the un-edited file created when the above OTS diagnostic script was executed on an OTS9010 instrument equipped with 10 Gb/s optical modules.

```
Test Properties:
```

```
-----
```

```
File Name           = 10GDiag  
Test Name           = OTS Diagnostic Script  
Date Of Creation    = Wednesday, October 10, 2001  
Author Name         = Vani.K.  
Test Description    = Digital Lightwave Factory Diagnostic Test  
Remarks            = Based on C++ Test Routine
```

```
Modules used in this test are:
```

```
-----
```

```
Module Type         = Receiver  
Module Name         = Rx1  
Slot Number         = 2  
Server Name         =
```

```
Module Type         = Transmitter  
Module Name         = Tx1  
Slot Number         = 3  
Server Name         =
```

```
-----
```

```
Report Date & time = Thursday, October 18, 2001, 14:20:15
```

```

Variable Name      = LaserStatus
Value              = ON

Report Date & time = Thursday, October 18, 2001, 14:20:20
Variable Name      = LOS
Value              = 0

Report Date & time = Thursday, October 18, 2001, 14:20:20
Variable Name      = LOF
Value              = 0

Report Date & time = Thursday, October 18, 2001, 14:20:20
Variable Name      = B1Errors
Value              = 0

Report Date & time = Thursday, October 18, 2001, 14:20:20
Variable Name      = B2Errors
Value              = 0

Report Date & time = Thursday, October 18, 2001, 14:20:20
Variable Name      = B3Errors
Value              = 0

```

No errors detected - continue testing...
Single error test - 5 B1 errors inserted...

```

-----
Report Date & time = Thursday, October 18, 2001, 14:20:33
Variable Name      = LOS
Value              = 0

Report Date & time = Thursday, October 18, 2001, 14:20:33
Variable Name      = LOF
Value              = 0

Report Date & time = Thursday, October 18, 2001, 14:20:33
Variable Name      = B1Errors
Value              = 5

Report Date & time = Thursday, October 18, 2001, 14:20:33
Variable Name      = B2Errors
Value              = 0

Report Date & time = Thursday, October 18, 2001, 14:20:34
Variable Name      = B3Errors
Value              = 0

```

Single error test with 5 B1 errors inserted - PASSED
Single error test... 5 B3 errors inserted...

```

-----
Report Date & time = Thursday, October 18, 2001, 14:20:46
Variable Name      = LOS
Value              = 0

Report Date & time = Thursday, October 18, 2001, 14:20:46
Variable Name      = LOF
Value              = 0

Report Date & time = Thursday, October 18, 2001, 14:20:47

```

Appendix B: Sample Script Description

Variable Name = B1Errors
Value = 0

Report Date & time = Thursday, October 18, 2001, 14:20:47
Variable Name = B2Errors
Value = 0

Report Date & time = Thursday, October 18, 2001, 14:20:47
Variable Name = B3Errors

Value = 5

Single error test with 5 B3 errors inserted - PASSED
LOS test - 128 mSec burst...

Report Date & time = Thursday, October 18, 2001, 14:20:56
Variable Name = LOS
Value = 2

LOS insertion PASSED: 2 LOS events expected and received

Error rate test - approximate 1.0E-6 B1 BER generated

Report Date & time = Thursday, October 18, 2001, 14:21:03
Variable Name = LOS
Value = 0

Report Date & time = Thursday, October 18, 2001, 14:21:03
Variable Name = LOF
Value = 0

Report Date & time = Thursday, October 18, 2001, 14:21:03
Variable Name = B1Rate
Value = 5.615200e-006

Report Date & time = Thursday, October 18, 2001, 14:21:04
Variable Name = REI
Value = 0

Error rate test PASSED: B1 BER within expected range

End of Script Text File Report

Appendix C: Supplied Sample Scripts

This section identifies and briefly explains the function for each of the Digital Lightwave sample scripts included with the OTS *Toolkit*TM installation CDrom.

The Digital Lightwave supplied OTS *Toolkit*TM sample scripts attempt to demonstrate a wide range of custom test scripting techniques from which a user may develop their own custom test scripts. While these sample scripts contain a number of script display and logging approaches, by no means do they cover the full extent to which the OTS *Toolkit*TM scripts may be used to develop useful scripted test routines.

Our intent in providing them is to provide new user with the opportunity to study these sample scripts and thus gain a basic understanding of the scripting techniques used.

NOTE: *These scripts were designed for and tested on OTS91T2/R2 and OTS93T1/R1 modules installed in an OTS9010 Bench Top chassis. Scripts created for use with similar or upgraded OTS systems may require modification.*

Sample Scripts Listing

This table lists the file name and short description of each sample script provided. A more detailed explanation of each script is available in following sections of this Appendix.

Script File Name	Script Description
10G BER test.vts	Configures OTS, verifies signal quality, starts 1 hour timed test
10G BER PM.vts	Configures OTS, verifies signal continuity, runs 1 hour test
10G SONET Monitor.vts	Configures OTS for in-service (live traffic) SONET monitoring
10G Signal Check.vts	Configures OTS for a looped or monitored 10 second signal check
10GDiag_2_3.vts	Performs basic OTS functional test on 10G looped signal
10Gdiag_GNG.vts	10G OTS Diagnostic test with PASS/FAIL display during execution
MRDiag_7_8.vts	Performs basic OTS functional test on 2.5G looped signal
MRDiag_9_10.vts	Performs basic OTS functional test on 2.5G looped signal
10G_ABER.vts	Determines 10G signal robustness using threshold offset control
MR_ABER.vts	Determines 2.5G signal robustness using threshold offset control
System_Discover.vts	Discovers OTS current software and hardware configuration

In addition to the sample scripts, two script template files in which basic OTS hardware definitions may be defined are provided. These script templates include:

Script Template File Name	Template Description
Default.vtt	No OTS resources are defined in this template (blank)
OTS9010.vtt	Fully populated OTS9010 w/10G and 2.5G modules

Detailed Script Explanations

The following detailed script explanations are offered to assist users in understanding basic script operation and creation.

For best results, examine the sample scripts supplied and adapt the techniques used in them to customize and enhance test and measurement scripts of your own design. Keep in mind that OTS *Toolkit*[™] scripts can perform any test or series of tests as well, or better than, someone manually controlling an OTS instrument.

10G BER test.vts

This script conditions the OTS for a one hour “timed” test using typical SONET test configuration parameters. The test uses the OTS PRBS (payload) test signal as the test signal source and verifies test signal continuity by inserting and looking for PRBS bit errors. Assuming test signal continuity is verified, the script conditions the OTS for a 1 hour timed test and terminates immediately after the test starts. Test results are analyzed using the elaborate and flexible OTS results viewer utility. Listed below are the script commands that establish the OTS timed test mode.

```
SENSe:TEST:MODE TIMEd
SENSe:TEST:DESCRiption 1 hour BER soak test
SENSe:TEST:TIME 3600
SENSe:TEST:TIME?
ASSIGN( Test_Hours = TestTime/3600 )
SENSe:TEST:STATe RUN
```

Typical Usage:

- Use with the Test Scheduler to capture BER soak test history results.
- Use the RANGE function to change test parameters for different testing combinations.
- Modify the script to include multiple 1 hour or longer test “segments” in case of test interruption.

10G BER PM.vts

This script conditions the OTS for signal monitoring using typical SONET test configuration parameters. The test uses the OTS PRBS (payload) test signal as the test signal source and verifies test signal continuity by inserting and looking for PRBS bit errors. Assuming test signal continuity is verified, the script starts four consecutive 15-minute performance monitoring (PM) periods in which SONET payload and overhead error performance measurements are made at the end of each 15 minutes test period. Unlike the 10G BER test.vts script, this script may or may not employ the OTS results viewer since important performance measurement data is captured in the script results file. This script may be easily modified to extend both the number of 15-minute test intervals as well as the test duration of each test interval. The main measurement (REPEAT) loop is scripted as follows:

```

REPEAT( 4 )
BEGIN
SENSE:TEST:STATe RUN
LOG( ***** )
ASSIGN( Loop = Loop + 1 )
WAIT( 900000 )
SENSE:MEASure:PATH:CUMUlative? PAYL_C
SENSE:MEASure:PATH:CUMUlative? PAYL_ES
SENSE:MEASure:PATH:CUMUlative? PAYL_ER
SENSE:MEASure:PATH:CUMUlative? B3_C
SENSE:MEASure:PATH:CUMUlative? B3_ES
SENSE:MEASure:PATH:CUMUlative? AIS_ES
SENSE:MEASure:PATH:CUMUlative? REI_ES
SENSE:MEASure:PATH:CUMUlative? LSS_ES
SENSE:MEASure:PATH:CUMUlative? J1TIM_ES
SENSE:MEASure:LINE:CUMUlative? B2_C
SENSE:MEASure:LINE:CUMUlative? B2_ES
SENSE:MEASure:SECTion:CUMUlative? B1_C
SENSE:MEASure:SECTion:CUMUlative? B1_ES
SENSE:MEASure:SECTion:CUMUlative? LOS_ES
SENSE:MEASure:SECTion:CUMUlative? LOF_ES
SENSE:MEASure:SECTion:CUMUlative? OOF_ES
LOG( ***** )
SENSE:TEST:STATe END
WAIT( 2000 )
END

```

Typical Usage:

- Use with Test Scheduler to capture BER soak test history results.
- Use the RANGE function to change test parameters for different testing combinations.
- Modify the script to include additional measurement intervals and/or longer or shorter measurement periods.

10G SONET Monitor.vts

This script configures the OTS for live traffic monitoring. The script uses four 15 minute measurement loops and measures the following SONET overhead parameters:

```
SENSE:MEASure:LINE:CUMUlative? B2_C
SENSE:MEASure:LINE:CUMUlative? B2_ES
SENSE:MEASure:LINE:CUMUlative? AIS_ES
SENSE:MEASure:PATH:CUMUlative? B3_C
SENSE:MEASure:PATH:CUMUlative? B3_ES
SENSE:MEASure:PATH:CUMUlative? AIS_ES
SENSE:MEASure:SECTion:CUMUlative? B1_C
SENSE:MEASure:SECTion:CUMUlative? B1_ES
SENSE:MEASure:SECTion:CUMUlative? LOS_ES
SENSE:MEASure:SECTion:CUMUlative? OOF_ES
```

These measurements are intended to indicate if the Path, Line and Section performance is degraded.

Typical Usage:

- Live traffic monitoring
- Script may be modified to characterize multiplex loading, i.e., channel framing, etc.

10G_Signal_Check.vts

A simple, short duration (10 seconds), signal check script that measures degraded performance in a SONET overhead signal.

Typical Usage:

- Base for similar signal monitoring applications.

10GDiag_2_3.vts

This script performs basic OTS diagnostic checks on 10G modules installed in slots 2 and 3 (OTS9010). Script action involves inserting B1 and B3 overhead bit anomalies (bit errors), Loss-of-Signal conditions, and a continuous B1BIP parity byte BER at 1.0E-06 BER. Each inserted defect or anomaly is followed by a measurement that verifies the anomaly was indeed detected.

Typical Usage:

- Verify proper operation of 10G OTS modules.

10GDiag_GNG.vts

This script performs basic OTS diagnostic checks on 10G modules identical to the 10Gdiag_2_3.vts script except this script includes a GO/NOGO display using the ASSIGN feature. This allows the user to identify whether all tests conducted have met requirements by using numerical assignments as part of an IF evaluation. If an individual test passes, a number greater than 0 is assigned. If a test fails a 0 value is assigned to an associated integer variable. At the conclusion of several individual tests, the various test integer variables are added and evaluated to determine if all tests passed or if one or more tests have failed. Using the ASSIGN variable then displays an all tests PASS or FAIL indication.

```

ASSIGN( go_nogo = B1test + B3test + LOSstest + BERtest )
LOG
LOG( ***** )
IF( go_nogo == 54 )
BEGIN
ASSIGN( Results = "ALL TESTS PASSED" )
END
IF( go_nogo < 54 )
BEGIN
ASSIGN( Results = "TEST(S) FAILED" )
END
*RST

```

MRDiag_7_8.vts

Similar to 10GDiag script with 2.5G (multi-rate) modules installed in slots 7 and 8.

MRDiag_9_10.vts

Similar to MRDiag_7_8.vts script with minor variations in OTS configuration and script execution style. Expects OTS93T1/R1 modules installed in slots 9 and 10 (OTS9010 unit).

10G_ABER.vts

This script uses the powerful OTS receiver threshold offset control in conjunction with the script FOR loop function to determine approximate signal robustness. After initial test signal configuration and continuity verification is accomplished, the PRBS test signal bit error performance is monitored while the OTS receiver threshold offset is incremented, first in the positive direction, then in the negative direction. When 10 or more bit errors are detected, the threshold level and bit error counts are recorded and logged. The basic control and measurement loop functions are scripted using:

```
FOR( Threshold = 0, Threshold <= 200, Threshold = Threshold + 10 )
BEGIN
SENSE:TEST:STATe RUN
SENSE:INPUt:THREshold <Threshold>
SENSE:MEASure:PATH:CUMUlativE? PAYL_C
IF( Bit_Errors > 10 )
BEGIN
LOG
LOG( Error count exceeded --- signal robustness limit reached... )
GOTO (exitloop)
SENSE:INPUt:THREshold?
SENSE:MEASure:PATH:CUMUlativE? PAYL_C
END
```

In this case the <Threshold> variable is incremented in each loop by 10 units until the PAYL_C (PRBS bit error) count exceeds 10. At this point the script exits the FOR loop and records the threshold offset value and bit error counts. A repeat of this control and measurement function is repeated in the negative offset direction. The logged results indicate the approximate signal robustness in that signals that can tolerate greater positive and negative offsets before they begin to show error activity are more robust in nature.

Typical Usage:

- 10G and 2.5G signal quality checks.
- Used to compare and identify various signals for poor overall performance characteristics.

System_Discover.vts

This unique test script only requires knowledge of the OTS network name or IP address. The script does not disturb or reset the current OTS configuration. A number of system queries are used to determine OTS mainframe identification information as well as the type of any Digital Lightwave modules that may be installed.

Typical Usage:

- Used to identify modules within an OTS chassis
- Used to assist in updating the Resources Used dialog box
- Useful in verifying the modules loaded into the OTS system under test