MTS TestSuite™

TW Essential User Guide

100-298-095 B

be certain.

MTS Trademarks

Proprietary Software

Software Verification and Validation

MTS software is developed using established quality practices in accordance with the requirements detailed in the ISO 9001 standards. Because MTS-authored software is delivered in binary format, it is not user accessible. This software will not change over time. Many releases are written to be backwards compatible, creating another form of verification. The status and validity of MTS' operating software is also checked during system verification and routine calibration of MTS hardware. These controlled calibration processes compare the final test results after statistical analysis against the predicted response of the calibration standards. With these established methods, MTS assures its customers that MTS products meet MTS' exacting quality standards when initially installed and will continue to perform as intended over time.

Manual Part Number	Publication Date	Release
100-298-095 B	August 2015	TestSuite TWS 4.1 or later
100-298-095 A	October 2014	TestSuite TWS 3.0

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Technical Support

How to Get Technical Support

Start with your manuals

The manuals supplied by MTS provide most of the information you need to use and maintain your equipment. If your equipment includes software, look for online help and README files that contain additional product information.

Technical support methods

MTS provides a full range of support services after your system is installed. If you have any questions about a system or product, contact Technical Support in one of the following ways.

Type of Support	Details
Web site	www.mts.com > Contact Us > In the **Subject** field, choose **To escalate a problem**; **Problem Submittal Form**
E-mail	Worldwide: tech.support@mts.com Europe: techsupport.europe@mts.com
Telephone	Worldwide: 1 800 328 2255 - toll free in U.S.; +1 952 937 4000 - outside U.S. Europe: +800 81002 222, International toll free in Europe

Outside the U.S.

For technical support outside the United States, contact your local sales and service office. For a list of worldwide sales and service locations and contact information, use the Global MTS link at the MTS web site:

www.mts.com > **About MTS Systems** > **Global Presence** > **Choose a Region**

Before You Contact MTS

MTS can help you more efficiently if you have the following information available when you contact us for support.

Know your site number and system number

The site number contains your company number and identifies your equipment type (such as material testing or simulation). The number is typically written on a label on your equipment before the system leaves MTS. If you do not know your MTS site number, contact your sales engineer.

Example site number: 571167

When you have more than one MTS system, the system job number identifies your system. You can find your job number in your order paperwork.

Example system number: US1.42460

Know information from prior technical assistance

If you have contacted MTS about this problem before, we can recall your file based on the:

- MTS case number
- Name of the person who helped you

Identify the problem

Describe the problem and know the answers to the following questions:

- How long and how often has the problem occurred?
- Can you reproduce the problem?
- Were any hardware or software changes made to the system before the problem started?
- What are the equipment model numbers?
- What is the controller model (if applicable)?
- What is the system configuration?

Know relevant computer information

For a computer problem, have the following information available:

- Manufacturer's name and model number
- Operating software type and service patch information
- Amount of system memory
- Amount of free space on the hard drive where the application resides
- Current status of hard-drive fragmentation
- Connection status to a corporate network

Know relevant software information

For software application problems, have the following information available:

- The software application's name, version number, build number, and (if available) software patch number. This information can typically be found in the About selection in the Help menu.
- The names of other applications on your computer, such as:
    - Anti-virus software
    - Screen savers
    - Keyboard enhancers
    - Print spoolers
    - Messaging applications

If You Contact MTS by Phone

A Call Center agent registers your call before connecting you with a technical support specialist. The agent asks you for your:

- Site number
- Email address
- Name
- Company name
- Company address
- Phone number where you can be reached

If your issue has a case number, please provide that number. A new issue will be assigned a unique case number.

Identify system type

To enable the Call Center agent to connect you with the most qualified technical support specialist available, identify your system as one of the following types:

- Electrodynamic material test system
- Electromechanical material test system
- Hydromechanical material test system
- Vehicle test system
- Vehicle component test system
- Aero test system

Be prepared to troubleshoot

Prepare to perform troubleshooting while on the phone:

- Call from a telephone close to the system so that you can implement suggestions made over the phone.
- Have the original operating and application software media available.
- If you are not familiar with all aspects of the equipment operation, have an experienced user nearby to assist you.

Write down relevant information

In case Technical Support must call you:

- Verify the case number.
- Record the name of the person who helped you.
- Write down any specific instructions.

After you call

MTS logs and tracks all calls to ensure that you receive assistance for your problem or request. If you have questions about the status of your problem or have additional information to report, please contact Technical Support again and provide your original case number.

Problem Submittal Form

Use the Problem Submittal Form to communicate problems with your software, hardware, manuals, or service that are not resolved to your satisfaction through the technical support process. The form includes check boxes that allow you to indicate the urgency of your problem and your expectation of an acceptable response time. We guarantee a timely response—your feedback is important to us.

You can access the Problem Submittal Form at www.mts.com > Contact Us (upper-right corner) > In the **Subject** field, choose **To escalate a problem; Problem Submittal Form**

Preface

Before You Begin

Safety first!

Before you use your MTS product or system, read and understand the safety information provided with your system. Improper installation, operation, or maintenance can result in hazardous conditions that can cause severe personal injury or death, or damage to your equipment and specimen. Again, read and understand the safety information provided with your system before you continue. It is very important that you remain aware of hazards that apply to your system.

Other MTS manuals

In addition to this manual, you may receive additional manuals in paper or electronic form.

You may also receive an MTS System Documentation CD. It contains an electronic copy of the manuals that pertain to your test system.

Controller and application software manuals are typically included on the software CD distribution disc (s).

Documentation Conventions

The following paragraphs describe some of the conventions that are used in your MTS manuals.

Hazard conventions

Hazard notices may be embedded in this manual. These notices contain safety information that is specific to the activity to be performed. Hazard notices immediately precede the step or procedure that may lead to an associated hazard. Read all hazard notices carefully and follow all directions and recommendations. Three different levels of hazard notices may appear in your manuals. Following are examples of all three levels. (for general safety information, see the safety information provided with your system.)

Danger: Danger notices indicate the presence of a hazard with a high level of risk which, if ignored, will result in death, severe personal injury, or substantial property damage.

Warning: Warning notices indicate the presence of a hazard with a medium level of risk which, if ignored, can result in death, severe personal injury, or substantial property damage.

Caution: Caution notices indicate the presence of a hazard with a low level of risk which, if ignored, could cause moderate or minor personal injury or equipment damage, or could endanger test integrity.

Other special text conventions

Important:

Important notices provide information about your system that is essential to its proper function. While not safety-related, if the important information is ignored, test results may not be reliable, or your system may not operate properly.

Note:

Notes provide additional information about operating your system or highlight easily overlooked information.

Recommended:

Recommended notes provide a suggested way to accomplish a task based on what MTS has found to be most effective.

Tip:

Tips provide helpful information or a hint about how to most efficiently accomplish a task.

Access:

Access provides the route you should follow to a referenced item in the software.

Example: Examples show specific scenarios relating to your product and appear with a shaded background.

Special terms

The first occurrence of special terms is shown in italics.

Illustrations

Illustrations appear in this manual to clarify text. They are examples only and do not necessarily represent your actual system configuration, test application, or software.

Electronic manual conventions

This manual is available as an electronic document in the Portable Document File (PDF) format. It can be viewed on any computer that has Adobe Acrobat Reader installed.

Hypertext links

The electronic document has many hypertext links displayed in a blue font. All blue words in the body text, along with all contents entries and index page numbers, are hypertext links. When you click a hypertext link, the application jumps to the corresponding topic.

Introduction

Introduction to TW Essential

MTS TW Essential (TWS) provides streamlined test design and execution for electromechanical and static hydraulic testing. TWS is for systems equipped with MTS Insight and MTS Insight+ controllers.

TWE and other MTS Software Applications

TWS functionality is based on the robust feature set of MTS TestSuite TW Elite (TWE). TWS look-and-feel is based on the streamlined test flow of MTS TestWorks 4 (TW4).

Template Compatibility

All templates created with TWE can run on TWS. All standard templates provided by MTS, except Multicycle and Multihead templates, are editable in TWS. In addition, the legacy converter can convert TW4 method/samples into template/tests that can be edited in TWS.

Using TWE for Advanced Features

Using some advanced features in TWS, such as creating arrays and enabling remote start, requires the extended editing capabilities of TWE. For more information, contact an MTS Service Engineer.

Overview of MTS TestSuite File Structure

The MTS TestSuite file hierarchy comprises several individual components. Each component stores a specific set of information. For example, a test contains a set of activities (known as the procedure), and the test run contains the data acquisition information and variable values that were obtained when the test was run. Furthermore, each component has a relationship with at least one other component.

In the following table, each of the main components of the MTS TestSuite structure are described. In the following sections of this document, the data stored in these components and the relationships between these components is described in detail.

Components of the MTS TestSuite Architecture

Name	Icon	Description
Project		A project is a collection of tests and test templates. By creating separate projects, you can organize similar tests and test templates and various project-level settings, such as the language or unit types used.
Test		A test is the core component of MTS TestSuite software. The test contains the test definition along with any test runs, analysis definitions, or analysis runs that have been created.
Test Template		Test templates eliminate the need to re-create existing information and provide an easy way to run standard tests.
Test run		A test run contains all data that was gathered during a single run of the test.

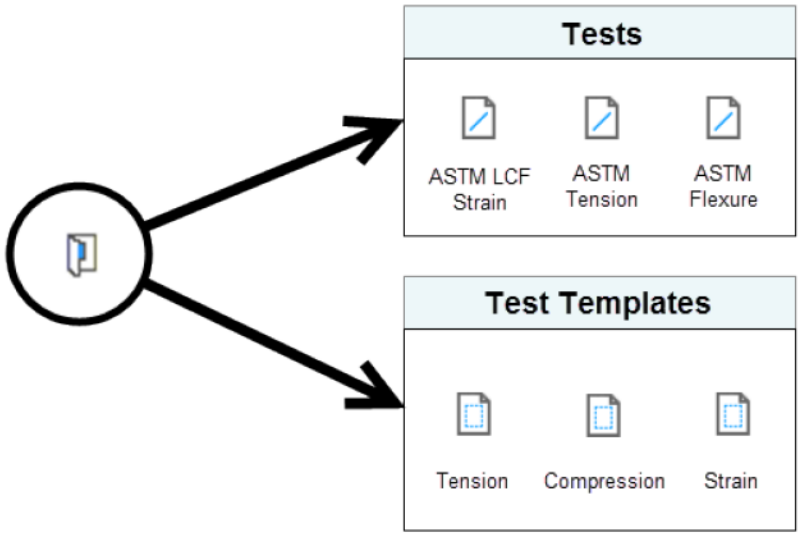
Project

A project is the highest level component in the MTS TestSuite file heirarchy. A project contains the following:

- A collection of tests
- A collection of test templates
- Project settings, such as the name and location of directories in which the tests, test templates, report templates, external files, and data exports are stored

To view or edit your available projects and their associated settings, click **Preferences**, select **Configuration**, and then select the **Project** tab. The other tabs on the Configuration window contains settings that are application-wide and persist regardless of which project you have selected.

A Project Contains Project Settings, Tests, and Test Templates

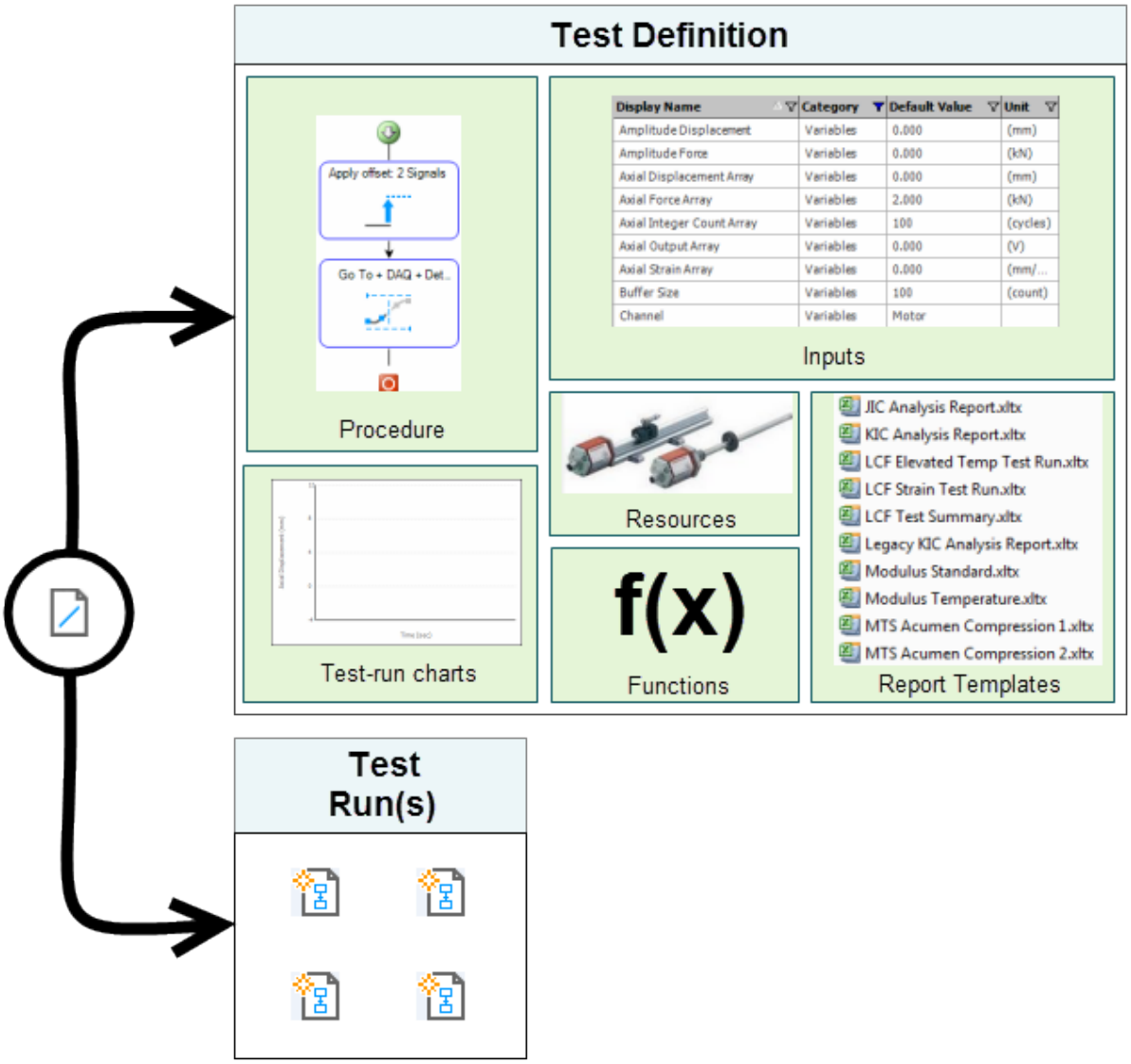


Test

A Test is stored within a project folder, and contains the following components:

- **Test Definition**—Contains the main components of the test, such as the procedure, variables, resources, functions, and report templates.
- **Test Run(s)**—Contains information that was gathered during the test run, such as variable values.

A Test Contains the Test Definition and Test Run(s)



Test Definition

The test definition is stored within the test. The test definition contains the following main components that define the test:

- **Procedure**—A collection of test activities that are performed step-by-step when the test is run.
- **Resources**—A collection of test resources mapped to the controller resources that will be used during the test.
- **Variables/ Inputs / Calculations**—Containers that can hold values that may change during the test run, such as time or axial displacement. Variables facilitate data manipulation and communication between different components of the test.

Note: Variables may also be referred to as **Inputs**, **Input Variables**, or **Calculations**.

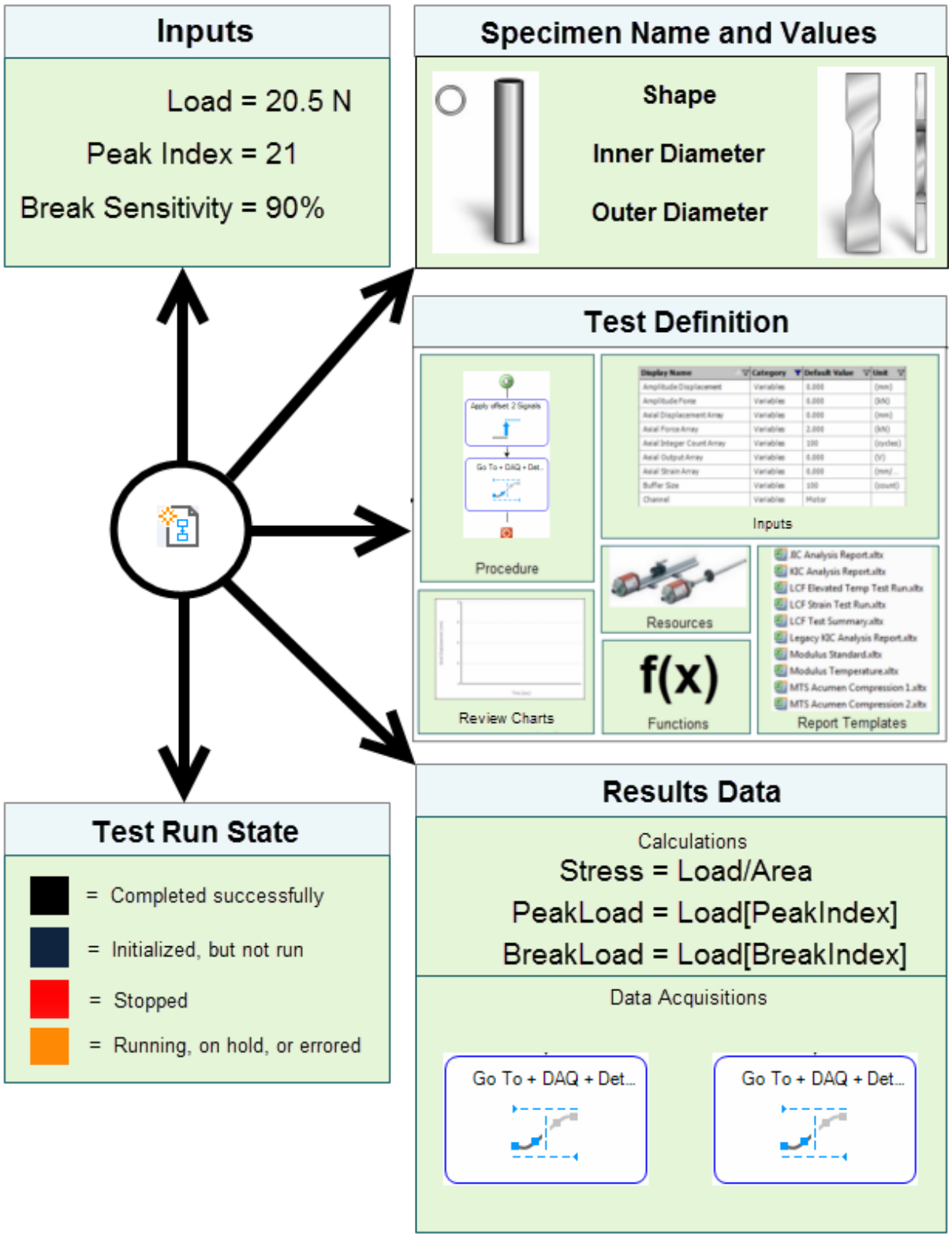
- **Test-Run chart**—A customizable user interface that shows data when the test is run.
- **Report templates**—A collection of Microsoft Excel Template files that define the layout of generated reports.
- **Functions**—A sequence of instructions that receives arguments and produces a result.

Test Run

A test run is the record of a test performed on a single, selected specimen. Test runs are stored in the test and include:

- A copy of the test definition, including the procedure, at the time the test run is created.
- A copy of the name of the selected specimen and its values at the time the test run is created.
- Input values during the test run.
- The state of the test run.
- Results data, such as data acquisitions and calculations.

Components of a Test Run

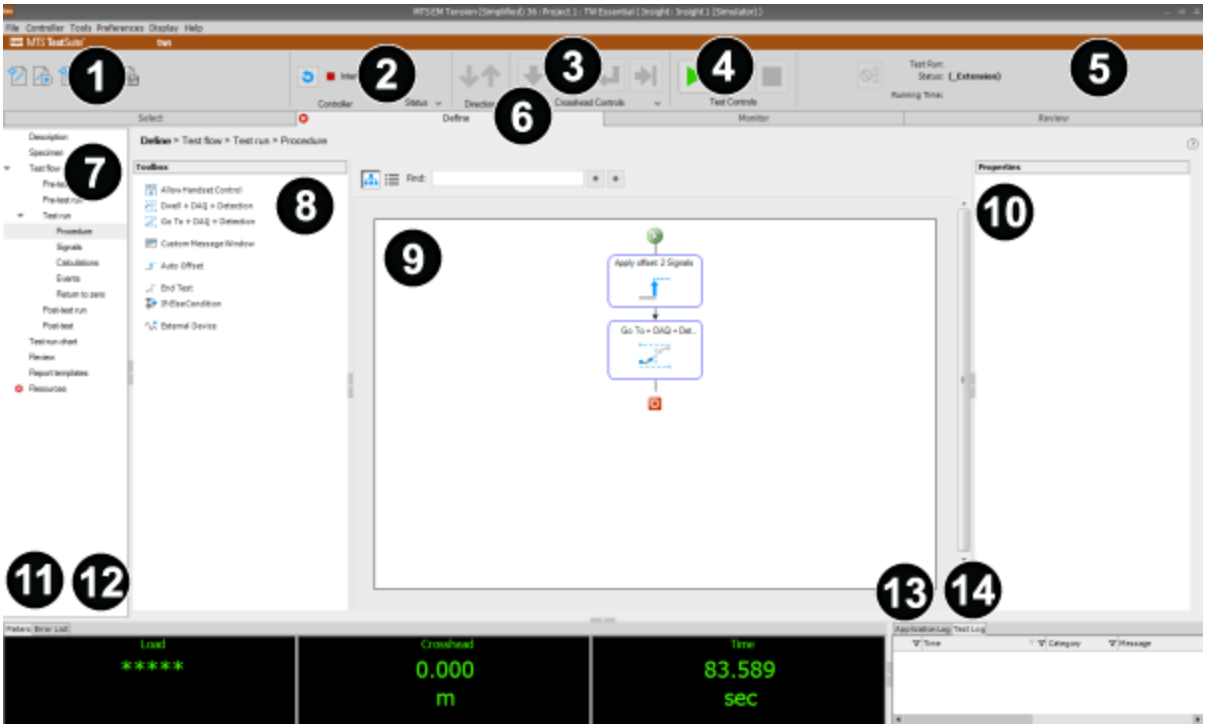


TW Essential Application Main Window

The main window of the TWS application is divided into the following sections:

- The top section contains menus, toolbar, a system control panel, and test status information.
- The left section contains the test definition tree, which is used to guide your progress as you define your test.
- The middle section contains panels for project editing.
- The bottom section contains the **Meters**, **Error List**, **Application Log**, and **Test Log** panels.

TWS Main Window



Main Window Options

Item	Name	Description
1	Menu bar and Quick Access panel	Provides menus and quick-access icons that allow you to perform tasks such as opening tests, opening tests from templates, and saving tests as templates.
2	**Controller** panel	Allows you to reset or override interlocks, display fault status, view the direction in which the crosshead is moving, position the crosshead, and start, stop, and hold the test. If your system has a clutch, the clutch indicator also appears (not shown). **Note:** When control is provided by a handset, the controls will be locked and overlaid by the Handset Exclusive Control icon: 
3	**Crosshead Controls** panel	Allows you to manually position the crosshead. You can configure the **Jog** and **Return** buttons at **Preferences** menu > **Configuration** > **Control Panel** tab. For more information, see "Crosshead Controls Panel" on page 270.
4	**Test Controls** panel	The **Test Controls** buttons allow you to start, pause, and stop the test.
5	**Test Run Status** panel	Shows test run information, including name, status, and running time. This panel includes a control that allows you to terminate the current test run.
6	Tabs	The various tabs and subtabs across the display are used to select a test, define a test, monitor a test, and review test results.
7	Test definition tree	On the **Define** tab, the test definition tree has two navigation modes. The two navigation modes are **Advanced Mode** and **Basic Mode**. **Advanced Mode** shows the entire test definition tree. **Basic Mode** shows a simplified version of the test definition tree.
8	**Toolbox** panel	The **Toolbox** panel appears when you click the **Define** > **Test flow** > **Test run** > **Procedure** tab. The **Toolbox** panel contains a list of all activities available when designing tests.
9	Work area	This is the work area in which you perform most of the tasks associated with the test definition tree. When designing a test, you can view the workflow in a Flowchart View or an Outline View by selecting the appropriate toggle button next to the search box. To quickly locate an activity in a large procedure, you can use the search box.

Item	Name	Description
		To create a test, drag test activities from the **Toolbox** panel to the work area.
10	**Properties** panel	Allows you to define or change the information, characteristics, and appearance of the selected procedure activities and runtime display components. For example, you can use the **Properties** panel to change the amplitude of a **Go To + DAQ + Detection** command test activity in a procedure.
11	**Meters**	The **Meters** show the current numeric value of the selected signal. The two default meters show load and crosshead extension. Right-click the meter to add or remove a meter, reset, change units or decimal places, and to configure the properties.
12	**Error List**	The **Error List** shows error and warning messages that describe both critical and non-critical conditions in the test definition. The **Error List** is dynamic and changes according to the part of the application you are using.
13	**Application Log**	The **Application Log** shows status information about application events in recent history, such as logging into the application, exiting the application, and interlock conditions present prior to opening tests. The application stamps each message with the type of message, generation date, and time. You can type notes into the **Message Log**, clear the log, and export the contents to a Microsoft Excel file. Messages persist from one session to another.
14	**Test Log**	The **Test Log** shows status information about test events for the current test, such as changes in program states. The application stamps each message with the type of message, generation date, and time. You can type notes into the **Test Log**, clear the log, and export the contents to a Microsoft Excel file. Messages persist from one session to another.

General Conventions

Entry-Type Toggle Button

The entry-type toggle button appears throughout the application. The following table indicates the icon and its meaning. Click the button to toggle between two entry options.

Variable Toggle Buttons

Icon	Description
	Input selection; click to select or enter a input.
	Item selection; click to select an item from the drop-down menu.
	Direct numeric entry; click to enter a numeric value.

Copy and Paste

You can copy tabular data, charts, chart data, functions, and variables to the clipboard.

Charts

To copy a chart or test control, right-click the item and click **Copy** or **Copy Image to the Clipboard**.

Chart data

To copy data values from a chart instead of the chart image, right-click the chart and click **Copy Values to the Clipboard**.

Tabular data

To copy an entire log or table, right-click the item, click **Select All**, and press **Ctrl+c** to copy or **Ctrl+x** to cut, or right-click again and click **Copy**.

To copy a single row of a log or table, double-click the row to select it, right-click at the same location, and select **Copy**.

You can select a block of rows two ways:

- Click in the column at the first row of the block, hold the mouse button down, roll over the block of rows, and release the mouse button. Right-click and select **Copy**.
- Select the first row of the block, press and hold the **Shift** key, and select the last row of the block. Right-click and select **Copy**.

To select multiple rows that are not in a block, hold the **Ctrl** key and click in the column at each row you want to select. Right-click and select **Copy**.

Naming Conventions

Unique names

In general, the components you create and name in the application, such as report templates and tests, must have unique names within the test. Components that are stored within the test, such as variables, and procedure activities must also have unique names.

For items that require unique names, the naming restrictions and the scope of the uniqueness vary by type of item.

Variable identifier name

The following guidelines apply for naming a variable identifier:

- It must be unique within the test.
- It is not case-sensitive.
- It can contain both alphabetical and numeric characters.
- It can contain a hyphen (-) or underscore (_).
- It cannot contain spaces, apostrophes, quotes, or other special characters.
- It cannot start with a number.

Variable display name

The following guidelines apply for naming a variable display name:

- It must be unique within the test but can match its variable identifier.
- It is case-sensitive.
- It can contain both alphabetical and numeric characters.
- It can contain special characters, including spaces, apostrophes, quotes, or other special characters.
- It can start with a number or special character.

Project name

The following guidelines apply for naming a project:

- The name must be unique among existing projects.
- It is case-sensitive.
- It must follow Microsoft file naming conventions and restrictions.
- It can contain both alphabetical and numeric characters.
- It can contain spaces, commas, '-', '_', '.', '!', '@', '#', '$', '%', '^', '(', ')', '=', '+', ';', '~', '{', and '}'.
- It cannot contain apostrophes, quotes, '&', '*', '[', ']', ':', '|', '<', '>', '\', '/', and '?'.
- It can start with a number or special character.

Test name

The following guidelines apply for naming a test:

- It is case-sensitive.
- It can contain both alphabetical and numeric characters.
- It can contain spaces, '-', '_', '.', '!', '@', '#', '%', '^', '(', ')', '=', '+', and ';'.
- It cannot contain apostrophes, quotes, '$', '*', '[', ']', ':', '|', '<', '>', '\', '/', and '?'.
- It can start with a number or special character.

Procedure activity name

The following guidelines apply for naming a procedure activity:

- It must be unique within a given path, such as a parallel path, conditional (If-Then) path, or main procedure path.
- It is case-sensitive.
- It can contain both alphabetical and numeric characters.
- It can contain special characters, including spaces, apostrophes, quotes, or other special characters.
- It can start with a number or special character.

Runtime display activity name

The following guidelines apply for naming a runtime display activity:

- It is case-sensitive.
- It can contain both alphabetical and numeric characters.
- It can contain spaces, apostrophes, quotes, or other special characters.
- It can start with a number or special character.

Report template name

The following guidelines apply for naming a report template:

- It is case-sensitive.
- It must follow Microsoft file naming conventions and restrictions.
- It can contain both alphabetical and numeric characters.
- It can contain spaces, apostrophes, '-', '_', '.', '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '{', '}', '=', '+', ';', and '~'.
- It cannot contain quotes, '[', ']', ':', '|', '<', '>', '\', '/', and '?'.
- It can start with a number or special character.

Tables

Sort Columns

To sort table columns:

1. To sort the table by the order of information in a column, click the column header cell.

   A down-arrow indicates descending order. An up-arrow indicates ascending order.

2. To reverse the sort order for information in a column, click the column header cell again.

Filter Data

Use the column filter to select the data you want to see and hide the rest. A filter changes only the display. The data does not change.

1. To apply a filter, click the filter icon in the column header to open the **Filter** menu.

   The filter menu lists all of the unique values in that column. The menu also provides options for showing values that are **blanks** or **not blanks**.

   The menu also provides a Custom filter. You can specify logical operators such as "All", "Custom", and "Read-Only" in the Operator column. "Matches Regular Expression" uses the .NET standard expression syntax.

2. Select a specific value to show all table entries that have the value or that meet the criteria.

   For example, if the table has a Units column and you want to filter the current view of the table to only show those rows that have Units of mm, select **mm** in the **Filter** menu for the Units column.

If you select a custom filter operator, select an operator and an operand. For example, you select the Value column and Custom filter. Select the operator **Does not equal** and operand **0.000**. The result is that the table shows the rows that have non-zero values. You can increase the criteria by clicking **Add a condition** and selecting another operator and operand. You can increase the filter further by applying filters to additional columns.

3. To display all table values, select the **All** command on the **Filter** menu.

Docking and Undocking Panels

Undocking and Docking Panels Overview

You can open and arrange multiple panels at the same time in your workspace. Moving panels around the screen is referred to as undocking or floating. Undocked panels are active, and you can perform actions in them. For example, you can undock a view and then modify the display properties.

Docking Symbols

Symbol	Description
Up Arrow	Highlights the top of the work area and docks the panel as a separate work area in the highlighted area.
Right Arrow	Highlights the right side of the work area and docks the panel as a separate work area in the highlighted area.
Left Arrow	Highlights the left side of the work area and docks the panel as a separate work area in the highlighted area.
Down Arrow	Highlights the bottom of the work area and docks the panel as a separate work area in the highlighted area.
Work Area	Highlights the work area and docks the panel as a separate tab in the highlighted work area.

If you have multiple monitors, the panel docks on the device to which the mouse is pointing.

Undocking a Panel

To undock a panel:

1. Click on the tab or title bar of the panel you want to undock.
2. Drag the view to where you want it.

   Arrows appear on the screen that indicate that you have undocked a panel. You do not have to use the arrow buttons to undock a panel.

Undocking a Panel

3. Resize the panel as necessary.
4. Repeat the previous steps to undock other panels as necessary.

Docking a Panel

To dock a panel:

1. Drag any side or corner of the panel.

   Docking symbols appear in the middle and on each side of the work area that indicate that you have undocked a panel.

Docking Symbols

2. When your mouse pointer reaches a docking symbol, a region of the work area becomes highlighted. To dock the panel in that region, release your mouse.

   Note: There can be multiple work areas, each with its own set of docking symbols.

   If you have multiple monitors, the panel docks on the device to which the mouse is pointing.
3. Repeat the previous steps to dock other panels as necessary.

Manage MTS TestSuite Files

MTS TestSuite Folders and Files Management Overview

Access

Preferences menu > **Configuration** > **Project** tab (default location)

In most cases, the management of TestSuite files is done using the MTS TestSuite application. Some files, such as templates and reports, can be managed using the Windows operating system.

Caution: Using the Windows OS to manipulate the files in a test folder can corrupt the test and its associated files.

Corrupting the test can make the test unusable and/or result in the loss of test-run data.

Do not manipulate any of the files in the test folder. Any changes to the test should be performed using the TestSuite application.

MTS TestSuite folders

MTS TestSuite Folder Management

File type	Folder Extension	From TestSuite Application	From Windows Explorer
Legacy Templates **Note:** Once a legacy template is converted, it has a .TSTemplate extension and can be managed like other templates.	.Test* (v2.0 or earlier) .Project (v1.6 or earlier)	New test from template, save as	No file manipulation allowed except delete
Project Directory Files	.Project	Import legacy projects (v1.6 or earlier), new, open, delete	No file manipulation allowed except delete
Test	.Test	New, open, save as, delete	No file manipulation allowed except delete

*

When using Windows XP to view TestSuite folders, the folder names are determined by the TestSuite application's naming convention and are appended with the folder extension. When using Windows 7 or Windows Vista to view TestSuite folders, the folder names are the user-defined display names and the file extensions do not appear.

MTS TestSuite files

MTS TestSuite File Management

File type	File Extension	From TestSuite Application	From Windows Explorer
User Templates  **Note:** Template files are not fully self-contained; they contain references to other files (such as, Report Templates).	.TSTemplate	Create new test from, save as	Copy, move, paste, delete, rename
MTS Templates  **Note:** Template files are not fully self-contained; they contain references to other files (such as, Report Templates).	.TSTemplate	Create new test from, save as	Contact MTS
Exported Tests	.tsproj	Import/export	Copy, move, paste, delete, rename
Exported Test Runs	.tsproj	Import/export	Copy, move, paste, delete, rename
Exported Unit Sets	.tsunitset	Import/export	Copy, move, paste, delete, rename
Report Templates	.xltx	Add, edit, remove from test	Copy, move, paste, delete, rename
Reports	.xlsx	Generate reports, open, print, delete, rename	Copy, move, paste, delete, rename
External Files (create test from)	.xml	Create test from the .xml file	Copy, move, paste, delete, rename
Data Export (exported raw data)	.txt .csv	Export	Copy, move, paste, delete, rename

Move Test Files to a New Directory

Use the following procedure to move a large number of tests from one location to another. The following procedure is not recommended for day-to-day testing but may be used to move a large number of test to a new (empty) location (such as a network drive).

Important: You can use the Windows OS to move test files only if the destination directory is empty. To move a test into a directory that already contains tests, use the **Save As** feature in the TestSuite application.

1. Use the Windows OS to create a new, empty directory where you want to store the test files.
2. If this new location is where you want to store and run all tests, set the default Test Directory setting to this new location.
    A. On the **Preferences** menu, click **Configuration**, and click the **Project** tab.
    B. Set the default Test Directory setting to the new directory that you just created.
3. Use Windows OS to move the tests to the new directory.
    A. Make sure that the new directory is empty.
    B. Move the tests to the new directory.

    Note: When using the Windows XP OS, each test appears as a Test folder, such as TST1.Test, that is numbered in the order that it was created. The user-defined test names are only visible when the test directory is viewed through the TestSuite application. When using Windows 7 or Windows Vista, the user-defined test names appear in the OS.
4. To move additional tests into the directory, you must use the **Save As** function in the TestSuite application.

    Important: Each Test folder contains a number of files required to run the test. Do not manipulate any of the files in the test folder.

Diagnostics

Diagnostic Files Overview

The MTS TestSuite applications include a feature to create a diagnostics file that you can send to MTS Technical Support for analysis. The diagnostic file is useful for troubleshooting and correcting problems. You can include the following in a diagnostic file:

- Diagnostic logs
- Station configuration
- Report templates
- Specimens (not applicable to the TWE or TWX applications)
- Test runs

If required, you can configure the content of the diagnostics file in collaboration with MTS Technical Support.

Note:
Contact MTS Technical Support to assist you with transmitting the diagnostics file.

Create a Diagnostic File

A test must be open to create a diagnostic file. You cannot create one while a test is loaded.

1. In the **Tools** menu, click **Create Diagnostic File**.
2. Click **Browse** in the Create Diagnostic Package window.
3. Click **File > Save As**, select an existing diagnostic (.tsdiag) file to overwrite or enter a new diagnostic file name, and click **Save**.
4. Expand the test hierarchy in the Create Diagnostic Package window and select the check boxes for the information you want to include in the file.

   If necessary, contact MTS for assistance with selecting information.
5. Click **Save**.

   A message notifies you of successful file creation and its location, or if errors are detected.
6. Click **OK**.

Application Log, Test Log and Error List

Application Log Overview

Access

C:\MTS TestSuite\GlobalLog\App.log

The information in the Application Log is automatically stored in a log (.log) file. Each new entry in the log is automatically written to the file when the entry occurs. The default file location is C:\MTS TestSuite\GlobalLog. The default file name is App.log.

When you start a session, the application looks in the default location for the default log file. If the file is there, the application opens the file and loads the contents into the Application Log. If the default file is not there (it is renamed or moved or cleared), the application creates and opens a new (empty) default log file.

The default log file is perpetual. As long as it is not renamed, moved, or cleared, it continues to accumulate the Application Log information.

Logging Levels

The Application Log warnings and errors pertain to loading or running the test. The following information can be displayed:

- **Errors**—These messages show information about critical conditions that prevent a test run or stop a test in progress.
- **Warnings**—These messages show information about conditions that may require attention, but do not prevent a test run or stop a test in progress.

- **Information**—These messages typically verify the start, result, or completion of a user-initiated action, or inform the user of the start, result, or completion of a system-initiated action. They can also be notes added by users.

The application records each message with a date and time stamp. Messages are also persisted from one session to another.

Application Log information

The Application Log provides the following information for each message:

- **Severity**—Shows a corresponding icon for the message type.
- **Time**—Includes the date and time the message was logged.
- **Category**—Shows the source of the message: **AppLog** indicates that the message was generated by an event related to the application. **TestRunLog** indicates that the message was generated by an event related to the test run.
- **Message**—Shows a description of the logged event.

Log menu options

The Application Log includes the following right-click menu options:

- **Select All Rows**—Selects all rows in the log. You can copy or export the selected rows.
- **Copy**—Copies the selected rows. Paste in the desired application.
- **Clear View**—Hides the contents of the log (automatically selects the **New Messages** log source) during your current session. Only new messages are subsequently displayed. All messages are restored when you select **Current Log** as the log source.
- **Export Log**—Exports your log file in Microsoft Excel format.
- **Add Note**—Adds a user-entered message to the log. A separate window is used to type your note. The note is added to the **AppLog** category.
- **Open Existing Log File**—Navigate to the log location, such as MTS TestSuite > GlobalLog, to open a log in a new window.

Log sources

The Application Log shows information from the following sources that appear on the **Log Source** list:

- **Current Log**—Shows the messages for all test runs for the current session.
- **New Messages**—Hides the existing messages and shows newly generated messages.
- **From Existing File**—Opens a Windows Explorer window so that you can show a log (.log) file from a previous test session.

Sorting log information

To sort the contents of the log based on a particular column, click the column header. To reverse the sort order, click the column header again. To return the log to the standard sort order, click the **Time** column header.

Filtering log information

To filter the contents of the log, click the filter icon in the column heading. A menu of available filter options is shown. After you filter a column, the filter icon turns blue.

When you click a filter option, the Application Log filters out all the messages (rows) except those for the selected option.

Test Log Overview

Access

C:\MTS TestSuite\GlobalLog\FaultError.log

The information in the Test Log is automatically stored in a log (.log) file. Each new entry in the log is automatically written to the file when the entry occurs. The default file location is C:\MTS TestSuite\GlobalLog. The default file name is FaultError.log.

When you start a session, the application looks in the default location for the default log file. If the file is there, the application opens the file and loads the contents into the Test Log. If the default file is not there (it is renamed or moved or cleared), the application creates and opens a new (empty) default log file.

The default log file is perpetual. As long as it is not renamed, moved, or cleared, it continues to accumulate the Test Log information.

Logging Levels

The Test Log warnings and errors pertain to loading or running the test. The following information can be displayed:

- **Errors**—These messages show information about critical conditions that prevent a test run or stop a test in progress.
- **Warnings**—These messages show information about conditions that may require attention, but do not prevent a test run or stop a test in progress.
- **Information**—These messages typically verify the start, result, or completion of a user-initiated action, or inform the user of the start, result, or completion of a system-initiated action. They can also be notes added by users.

The application records each message with a date and time stamp. Messages are also persisted from one session to another.

Test Log information

The Test Log provides the following information for each message:

- **Severity**—Shows a corresponding icon for the message type.
- **Time**—Includes the date and time the message was logged.
- **Category**—Shows the source of the message. **TestRunLog** indicates that the message was generated by an event related to the test run.
- **Message**—Shows a description of the logged event.

Log menu options

The Test Log includes the following right-click menu options:

- **Select All Rows**—Selects all rows in the log. You can copy or export the selected rows.
- **Copy**—Copies the selected rows. Paste in the desired application.
- **Clear View**—Hides the contents of the log (automatically selects the **New Messages** log source) during your current session. Only new messages are subsequently displayed. All messages are restored when you select **Current Log** as the log source.
- **Export Log**—Exports your log file in Microsoft Excel format.
- **Add Note**—Adds a user-entered message to the log. A separate window is used to type your note. The note is added to the **TestRunLog** category.
- **Open Existing Log File**—Navigate to the log location, such as **MTS TestSuite > GlobalLog**, to open a log in a new window.

Log sources

The Test Log shows information from the following sources that appear on the **Log Source** list:

- **Current Log**—Shows the messages for all test runs for the current session.
- **New Messages**—Hides the existing messages and shows newly generated messages.
- **From Existing File**—Opens a Windows Explorer window so that you can show a log (.log) file from a previous test session.

Sorting log information

To sort the contents of the log based on a particular column, click the column header. To reverse the sort order, click the column header again. To return the log to the standard sort order, click the **Time** column header.

Filtering log information

To filter the contents of the log, click the filter icon in the column heading. A menu of available filter options is shown. After you filter a column, the filter icon turns blue.

When you click a filter option, the Test Log filters out all the messages (rows) except those for the selected option.

Error List Overview

The **Error List** tab lists warnings and errors pertain to defining and configuring the test.

- **Errors**—Shows information about critical conditions that do not allow you to load or run a test.
- **Warnings**—Shows information about conditions that may require attention, but do not prevent you from loading or running a test.

The contents of the Error list are dynamic and change according to the part of the application you are using.

The application does not store the contents of the Error list in a file. Each message is removed when the corresponding condition is corrected.

Error List information

The **Error List** tab provides the following information for each error:

- **Message type icon**—Shows the corresponding icon for the message type.
- **Sequence number**—Shows the numbered order in which the application detects the error or warning.
- **Description**—Contains the text that describes the error or warning.

Error Location

In addition to the Error list, the application also marks the problem conditions with error and warning icons on or near the offending selection or setting in the user interface. If you double-click an item in the list, the program control point moves to (or near to) the location of the error.

Error Identification

The **Error** list identifies error conditions for the current context or object selected. For example, you can use the **Error** list and the Test Definition object to view a summary of the errors in a test.

If you double-click an item in the **Error** list, the view changes to show the location of the error. Error and warning icons are placed on or near the property, selection, or setting where the error exists so that you can quickly locate the source of the error.

Meters

Meters Overview

The **Meters** tab contains meters that show various data retrieved from signals on your system. On most systems, the Crosshead/Axial Force and Load/Axial Displacement meters are available when you first open the **Meters** tab. However, you can add other meters that show a variety of information such as interlock status, power status, or even the current time and date. After you add a meter, you can configure the meter type, unit type shown, number of decimal places shown, sensitivity (of Peak/Valley meters only), and font settings.

Adding a Meter

1. Click the **Meters** tab.
2. Right-click any existing meter.
3. Hover over **Add Meter**.
4. Select the meter you want to add.

Note: This list contains the most commonly-used meters. To add other meters, click **More** to open the **Meter Configuration** window. Then, move each desired meter over to the list of **Selected** meters on the right side on the window. When you are finished, click **OK**.

Resetting a Meter

You can reset the values recorded in Peak, Valley, Peak/Valley, Running Maximum/Minimum, and Mean/Amplitude meters. Keep in mind that if the crosshead or actuator is not at the zero position, resetting certain meters may not always return the displayed value to zero. For example, if the axial displacement of the crosshead or actuator is resting at 10 mm and you reset a maximum axial displacement meter, the meter will continue to show 10 mm. This occurs because immediately after you reset the meter, the axial displacement signal remains at 10 mm, which is considered the maximum. In this scenario, if you returned the axial displacement to 0 before you reset the meter, the maximum axial displacement meter would return to 0 after you reset the meter.

1. Click the **Meters** tab.
2. Right-click the Peak, Valley, Peak/Valley, Running Maximum/Minimum, or Mean/Amplitude meter that you want to reset.
3. Select **Reset** to reset the meter you selected or select **Reset All** to reset all meters that can be reset on the **Meters** tab.

Removing a Meter

1. Click the **Meters** tab.
2. Right-click the meter you want to remove.
3. Select **Remove**.

Configuring Meters

After you add a meter to the **Meters** tab, you can configure the meter type, units shown, number of decimal places shown, sensitivity, and font used in the meter.

Changing the Meter Type

1. Click the **Meters** tab.
2. Right-click an existing meter.
3. Hover over **Meter Type**.
4. Select the desired meter type. For more information about the available meter types, see “Meter Types” on page 42.

Changing the Units Displayed on a Meter

1. Click the **Meters** tab.
2. Right-click an existing meter.
3. Hover over **Unit**.
4. Select the unit type you want the meter to use.

Changing the Decimal Places Displayed on a Meter

1. Click the **Meters** tab.
2. Right-click an existing meter.
3. Hover over **Decimal Places**.
4. Select the number of decimal places you want displayed on the meter.

Changing the Color and Font of Meters

1. Click the **Meters** tab.
2. Right-click an existing meter.
3. Select **Properties**. The **Meter Configuration** window appears.
4. Adjust the **Label Font Size**, **Value Font Size**, **Foreground Color**, and **Background Color** as desired. When you modify these settings, they are applied to all meters available on the **Meters** tab.

Changing the Sensitivity of a Meter

The sensitivity setting defines how much the signal must change before a level is considered a peak or valley. The sensitivity setting can be used to keep the activity from seeing signal noise as a new peak or valley. When adjusting the sensitivity, keep in mind that setting the sensitivity too low may cause signal noise to be recognized as peaks and valleys, and setting the sensitivity too high may cause low amplitude signals to be missed.

1. Click the **Meters** tab.
2. Right-click the Peak, Valley, or Peak/Valley meter you want to adjust.
3. Hover over **Sensitivity**.
4. Select the desired sensitivity.

Meter Types

The meter type determines what type of information is shown on the meter. When you first launch the application, the Crosshead/Axial Force and Load/Axial Displacement meters appear on the **Meters** tab. By default, the meter type of those meters is set to Timed. This meter type shows the real-time axial displacement of the system's crosshead or actuator. However, if you change the meter type to Running Maximum/Minimum, the meter shows the minimum and maximum axial displacement experienced by the crosshead or actuator.

Available Meter Types

Item	Description
Timed	Displays signal values at timed intervals.
Minimum	Displays the running minimum value that was observed since the meter was added or reset.
Maximum	Displays the running maximum value that was observed since the meter was

Item	Description
	added or reset.
Peak	Displays the peak value for the most recent cycle monitored.
Valley	Displays the valley value for the most recent cycle monitored.
Peak/Valley	Displays both the peak and valley values for the most recent cycle monitored.
Running Maximum/Minimum	Displays both the running minimum and maximum values that were observed since the meter was added or reset.
Mean/Amplitude	Displays the midpoint value and the difference between the peak and valley values for the most recent cycle monitored.

Licenses

License Utility Overview

Access

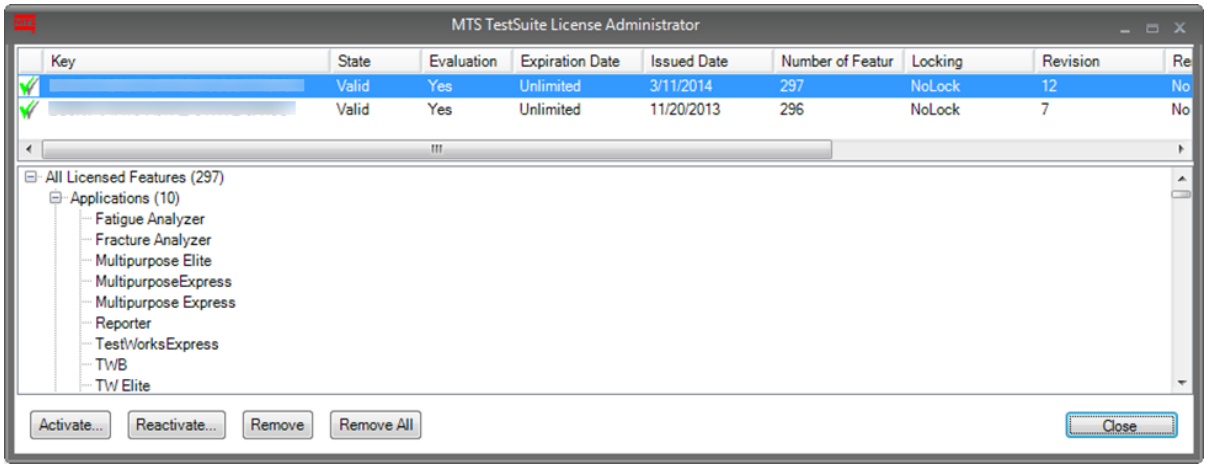
Programs > **MTS TestSuite** > **License Administrator**

During installation, the Activation Wizard prompts you to activate the license for your MTS TestSuite software. The process for obtaining the license file depends on whether the PC was connected to the Internet. After installation, you can add and manage licenses at any time with the MTS TestSuite License Administrator utility.

With the MTS TestSuite License Administrator utility, you can activate and manage your licenses and obtain information about currently installed licenses. You can view the Applications, Templates, Privileges, Activities, and other special features for which you are licensed.

Tip:
Always save your TestSuite license(s) into a plain text file using Windows Notepad. If you save a license using other text editors (such as WordPad, Notepad++, or Microsoft Word), characters in the license may be altered slightly or additional characters may even be appended to the end of the license. If this occurs, the license may not be recognized when you paste it into the License Administrator utility.

Open License Administrator Utility

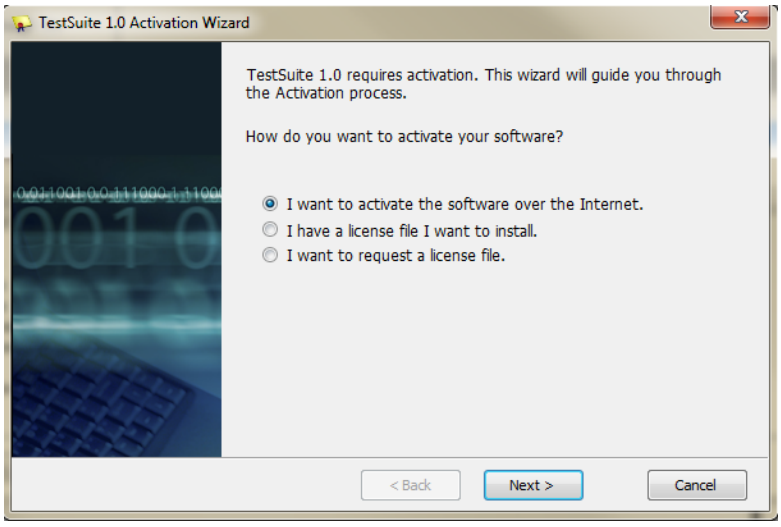


Activate a License with an Internet Connection

If you have an activation key and the PC on which you are installing a license has an Internet connection, perform the following steps to obtain, install, and activate a license.

1. Click **Programs** > **MTS TestSuite** > **License Administrator**.
2. Click **Activate**. The TestSuite Activation wizard is launched.

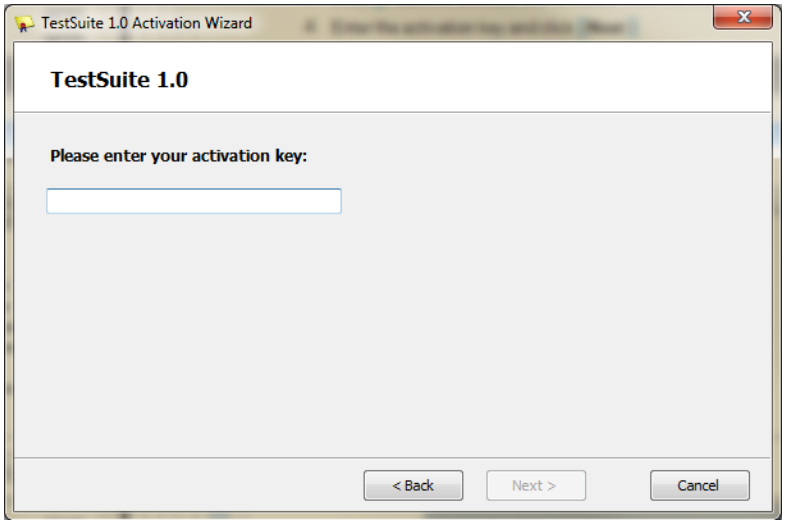
TestSuite Activation Wizard



3. Click **I want to activate the software over the Internet** panel and click **Next**. The window to enter your activation key is displayed.

4. Enter the activation key and click **Next**.

Activation Wizard Activation Key



A message indicates the license is being obtained from the server, and a progress bar is displayed. The Activation Wizard installs and activates the license.

5. When activation is complete, click **Finish** to exit the Activation Wizard.
6. Click **Close** to exit the MTS TestSuite License Administrator.

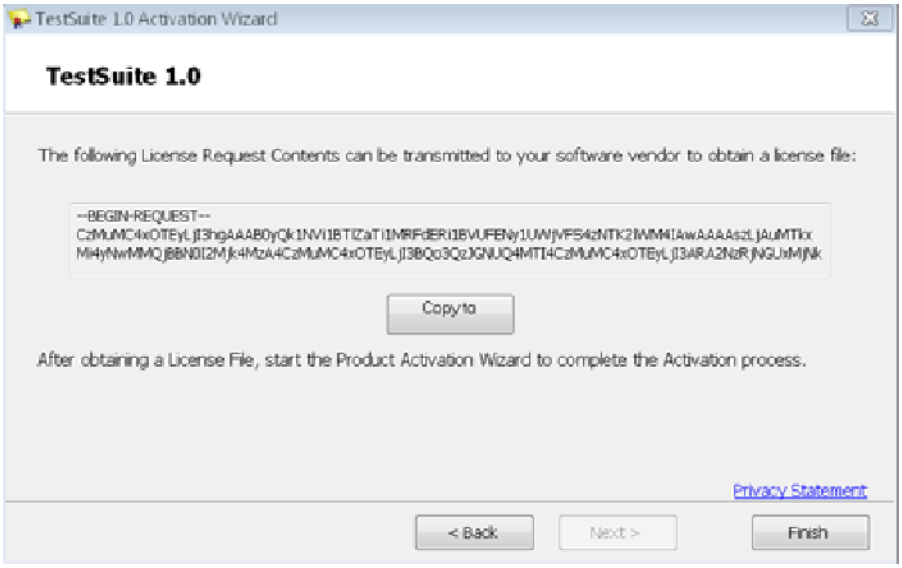
Request and Activate a License without an Internet Connection

If the PC on which you are activating a license does not have an Internet connection, follow these steps to request and activate a license file.

1. Click **Programs** > **MTS TestSuite** > **License Administrator**.
2. Click **Activate**.
3. Click **I want to request a license file** and click **Next**.

4. Enter the activation key and click **Next**.

Complete License Activation Window



5. To copy the License Request Contents, which is a PC-identifying string, click **Copy to**.
6. Paste the copied License Request into a text editor such as Notepad, and save the Request File to a USB drive.
7. Click **Finish** to exit the Activation Wizard.
8. Plug the USB drive into a PC that has Internet access. Go to the MTS Licensing Web site:

   www.mts.com/testsuite/licensing/
9. Open the Request File on the USB drive. Select and copy the contents.
10. Paste the content into the text box in the Web page and click **Download License File**. Save the license file to the USB drive.
11. Bring the USB drive back to the PC without an Internet connection onto which you are installing software.
12. Open the Activation wizard again. Select the **I have a license file I want to install** option and click **Next**.
13. Browse to the location of the license file. Select the license bin file and click **Open**.
14. Click **Next**. The Activation Wizard installs and activates the license.
15. Click **Finish** to exit the Activation Wizard.
16. Click **Close** to exit the MTS TestSuite License Administrator.

Remove a License

To remove a license, use the MTS TestSuite License Administrator utility.

1. Click **Programs** > **MTS TestSuite** > **License Administrator**.
2. In the upper part of the window, select the license you want to remove and click **Remove**. To remove all licenses, click **Remove All**.
3. You are prompted to confirm removing the license. Click **OK**.
4. Click **Close**.

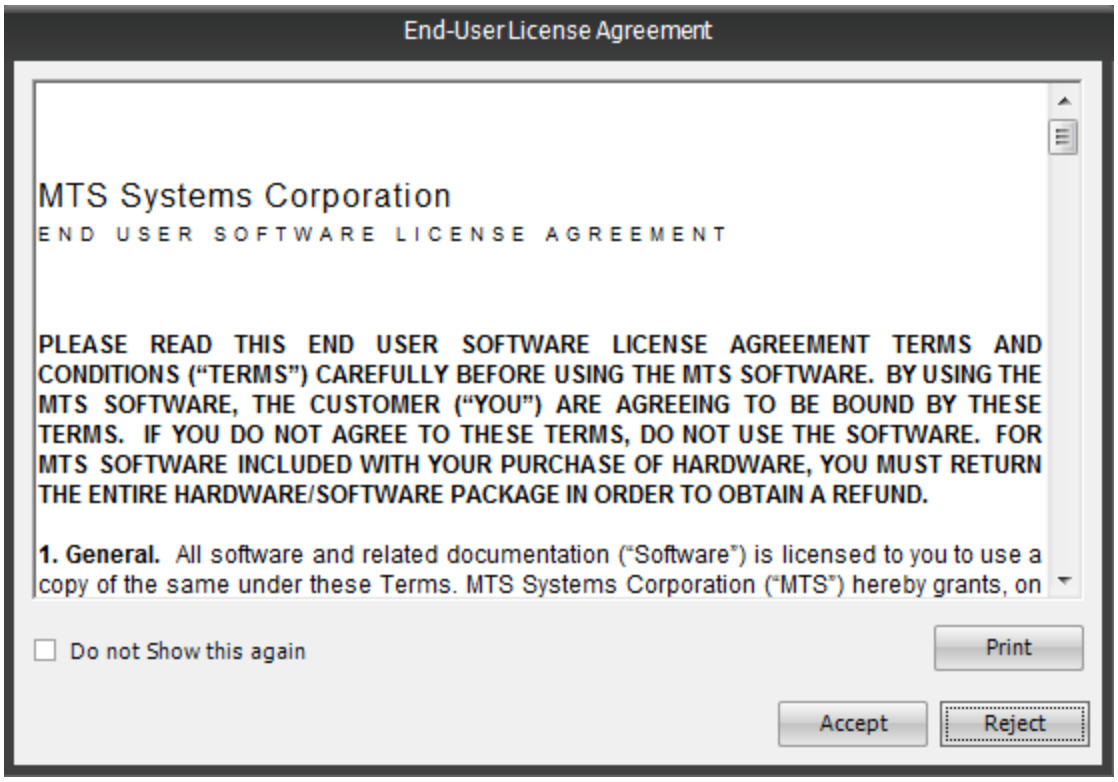
End-User License Agreement (EULA)

Access

Help > **End-User License Agreement**

When you first install and launch an MTS TestSuite application, the EULA window is shown. You can scroll to view the terms of the agreement, or print the license agreement.

End-User License Agreement Window



To Accept or Reject the EULA

- To accept the terms of the agreement, click **Accept**. The EULA window closes and the application is available for use.
- To reject the EULA, click **Reject**. The EULA window and MTS TestSuite application close. You cannot access the application until you accept the EULA.
- If you do not want to view the EULA each time the MTS application is launched, click the **Do**

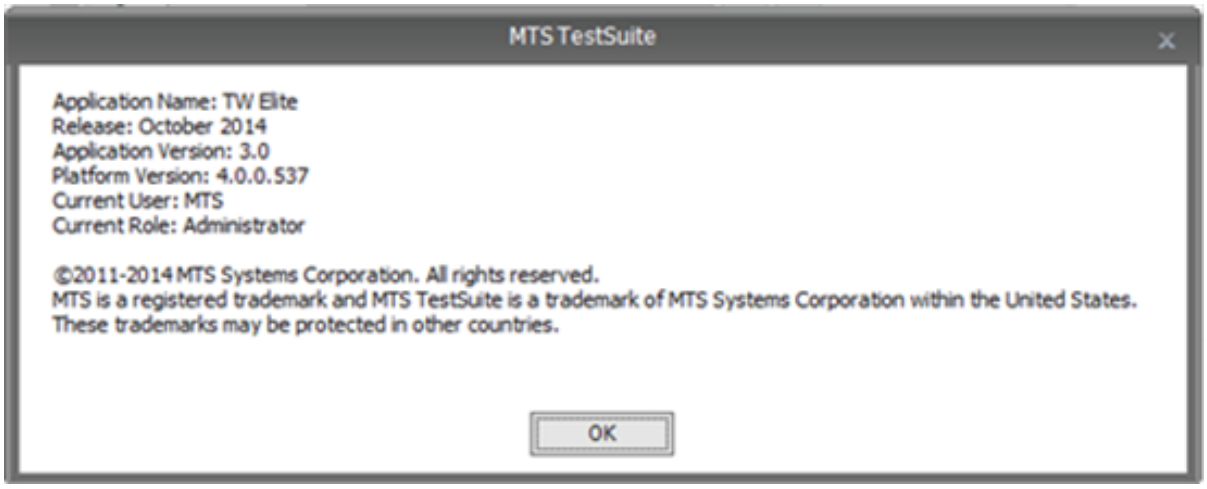
not show this again check box. You can access the EULA at any time from the **Help** menu.

Version Information Overview

The MTS TestSuite window provides the following information about the installed application:

- Name of the installed application
- Release date
- Application version number
- Platform version number
- Current user and role
- Copyright information

Version Information Window

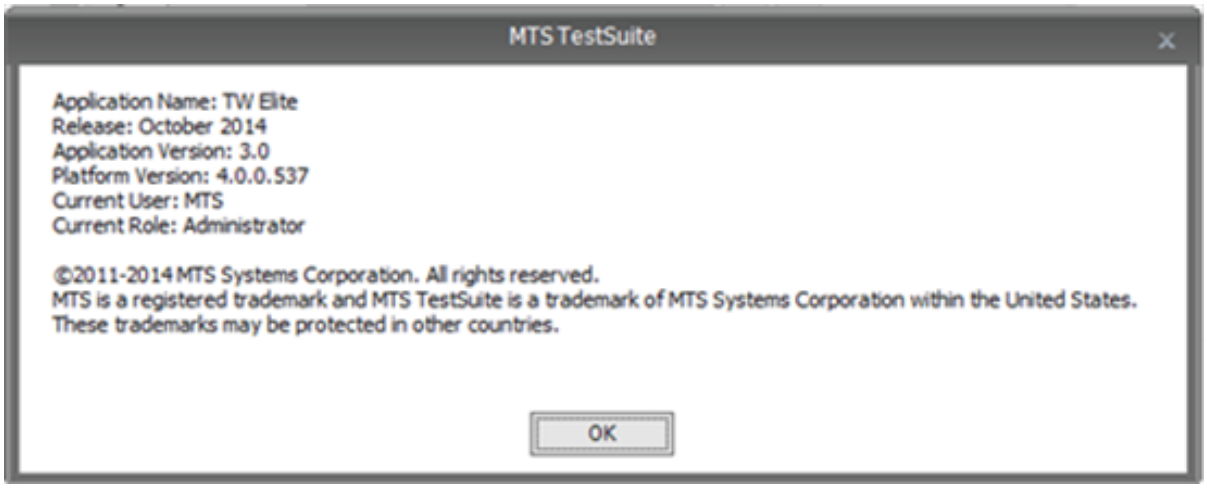


Version Information Overview

The MTS TestSuite window provides the following information about the installed application:

- Name of the installed application
- Release date
- Application version number
- Platform version number
- Current user and role
- Copyright information

Version Information Window



User and Role Management

Managing Users

User Management Overview

If your organization needs to manage multiple users of the MTS TestSuite application, you can manage users with or without their Windows user accounts. The default User Management option is none (the **No User Management** option), and users are not required to log in to the MTS TestSuite applications. Windows User Management automatically logs in users based on their Windows log in. By default, the user who installs the MTS TestSuite application is automatically added as an MTS TestSuite Administrator. Adding the default Administrator user provides the ability to switch to User Management if desired. Only a user with Administrator role or User Management privileges can add users and assign roles.

Note:
If your organization does not set up Local or Windows User Management, all users who launch an MTS TestSuite application have Administrator privileges.

An Administrator grants users privileges through the assignment of a specific role. Each privilege controls access to performing certain tasks within the application. A user may be assigned one or more roles; however, a user must select one of the roles when the application starts if **Local User Management** is the selected User Management option.

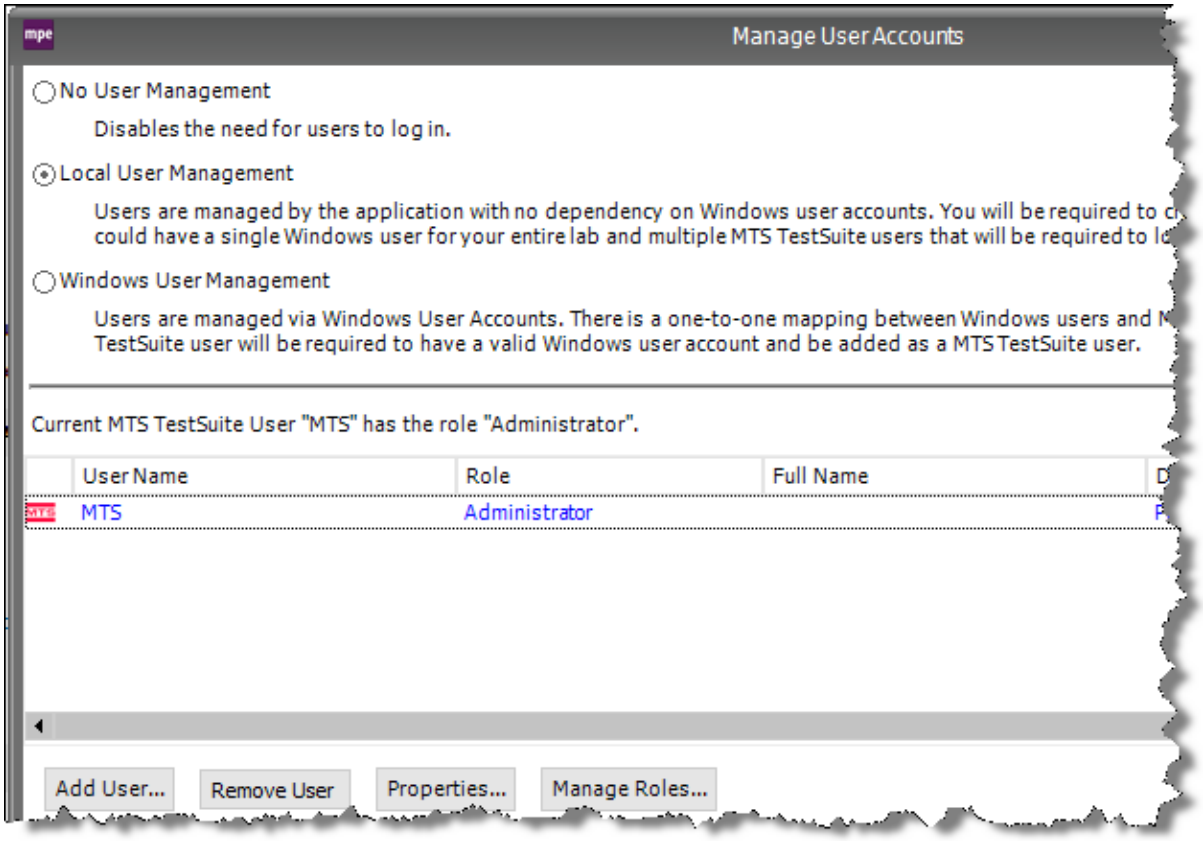
Manage User Accounts Window

Access

Preferences > **User Management** > **Manage User Accounts**

Use the Manage User Account window to add users and manage roles.

Manage User Accounts Window



The Primary MTS TestSuite Administrator is displayed in blue.

Manage User Accounts Window Description

Section	Description
User Management options	• **No User Management**—(Default) All users can access the application without having to log in if during installation, the **Anyone who uses this computer (all users)** option was selected. All users are automatically assigned an Administrator role. If the **Only for me** option was selected during installation, only the user who installed the application can launch it. • **Local User Management**—Users are managed by the MTS TestSuite application independently of Microsoft Windows user accounts. You must add MTS TestSuite users. • **Windows User Management**—Users are managed by their Windows User accounts. There is a one-to-one mapping between Windows and MTS TestSuite users. You must add MTS TestSuite users.
User accounts list	Displays all of the currently defined MTS TestSuite users.
Add User	Opens the Create new MTS TestSuite user window so you can add a new user and

Section	Description
button	assign the user a Role.
Remove User button	Removes the user selected in the Manage User Accounts window.
Properties button	Opens the Change User Properties window. You can change the name, password, and assigned roles.
Manage Roles button	Opens the Manage Roles window. You can view the privileges associated with the default roles, work with custom roles, and view all users currently assigned to a role.

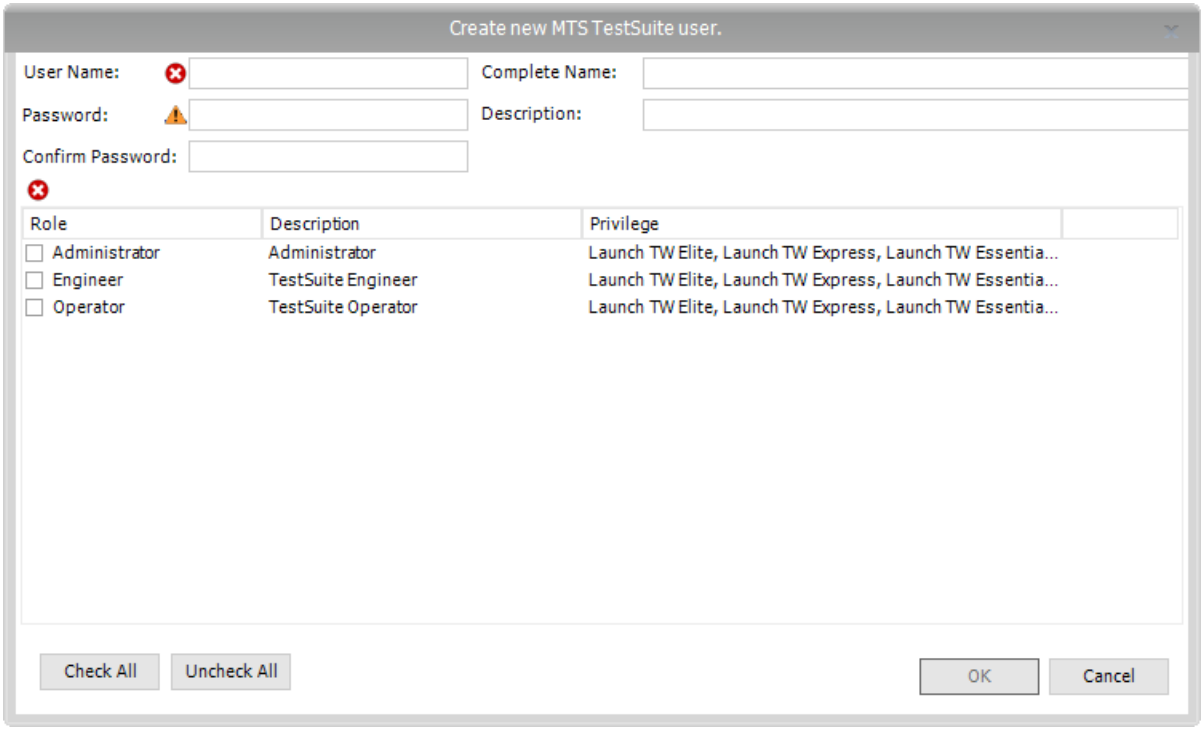
Add a Local User and Assign a Role

Note: To perform these steps, you must be assigned the Administrator role or be assigned a custom role with User Management privileges.

To add a local user and assign a role:

1. Click **Preferences > User Management > Manage User Accounts**. The Manage User Accounts window appears.
2. Click the **Local User Management** option.
3. Click **Add User**. The Create new MTS TestSuite user window appears.

Create New MTS TestSuite User Window



4. Enter the **User Name** for the user.

5. Enter a **Password** for the user and enter the password again in the **Confirm Password** box.
6. (Optional) Enter a **Complete Name** and **Description** of the user.
7. Select one or more **Roles** for the user.
8. Click **OK**. Repeat this procedure for each user you need to add.

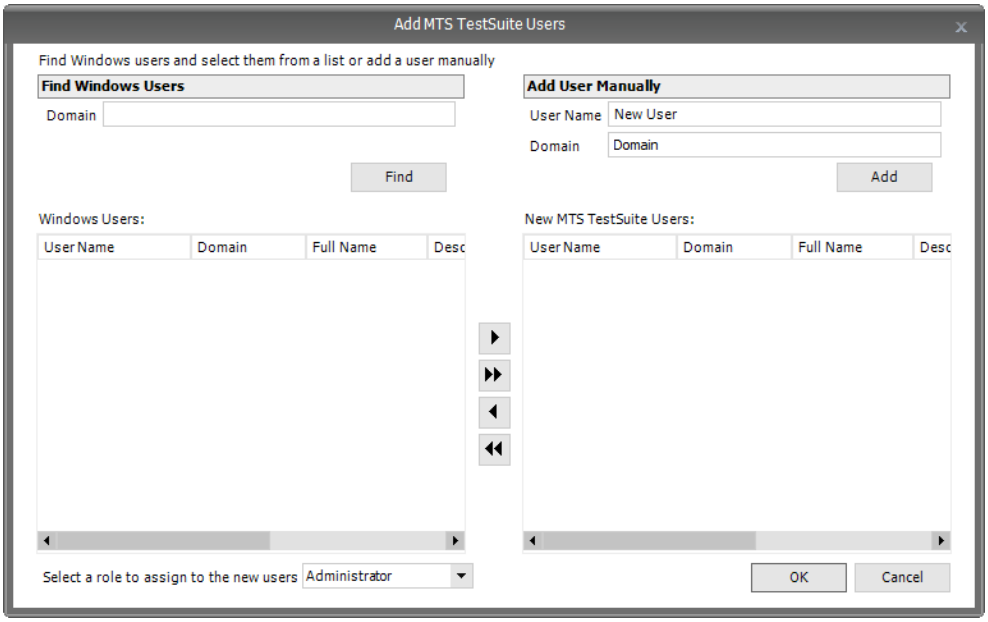
Add a Windows User and Assign Roles

Note: To perform this task, you must be assigned the Administrator role or be assigned a custom role with User Management privileges.

To add a Windows user and assign roles:

1. Click **Preferences > User Management > Manage User Accounts**. The Manage User Accounts window appears.
2. Click the **Windows User Management** option.
3. Click **Add User**. The Add MTS TestSuite Users window appears.

Add MTS TestSuite Users Window



4. Select the role you want to assign to the new users.
5. To add a user:
   - To find Windows users you want to add, enter the network **Domain** name, and then in the Find Windows Users panel, click **Find**. When the search results return, select the user or users you want to add. Click the arrows to move the selected users from the Windows Users panel to the New MTS TestSuite Users panel.
   - If you know the user name, enter the **User Name** and network **Domain Name**, and then click **Add** in the Add User Manually panel. Repeat for additional users.

6. Click **OK**.

Change a Role Assigned to a User

To change a role assigned to a user:

1. Click **Preferences > User Management > Manage User Accounts**. The Manage User Accounts window appears.
2. Select the user in the Manage User Accounts window and click **Properties**. The Change User Properties window appears.
3. Select the roles by selecting and clearing the corresponding check boxes.
4. Click **OK**.

Remove a User

To remove a user:

1. Click **Preferences > User Management > Manage User Accounts**. The Manage User Accounts window appears.
2. Select the user in the Manage User Accounts window.
3. Click **Remove User**. You are prompted to confirm removing the user.
4. Click **OK**.

Managing Roles

Note: To perform this task, you must be assigned the Administrator role or be assigned a custom role with User Management privileges.

The MTS TestSuite application provides several default (predefined) roles. You cannot edit or delete default roles; however, you can create custom roles that you can edit or delete.

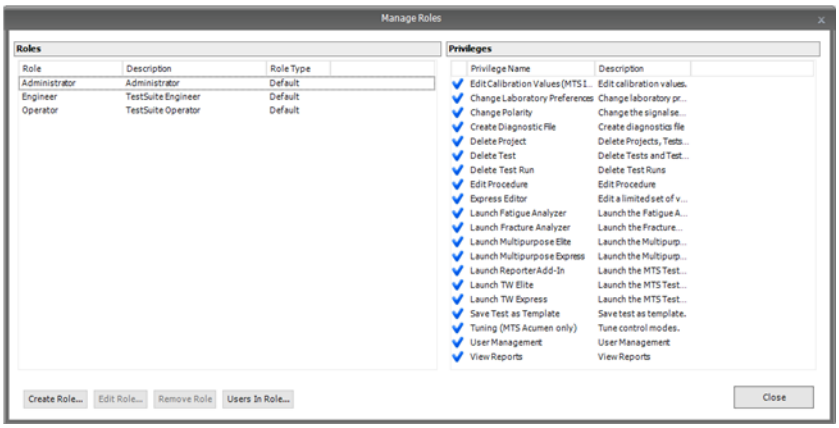
Manage Roles Window

Use the Manage Roles window to manage custom roles and view users assigned to roles.

Access

1. Click **Preferences > User Management > Manage User Accounts**.
2. Either the **Local** or **Windows User Management** option must be selected.
3. Click **Manage Roles**.

Manage Roles Window



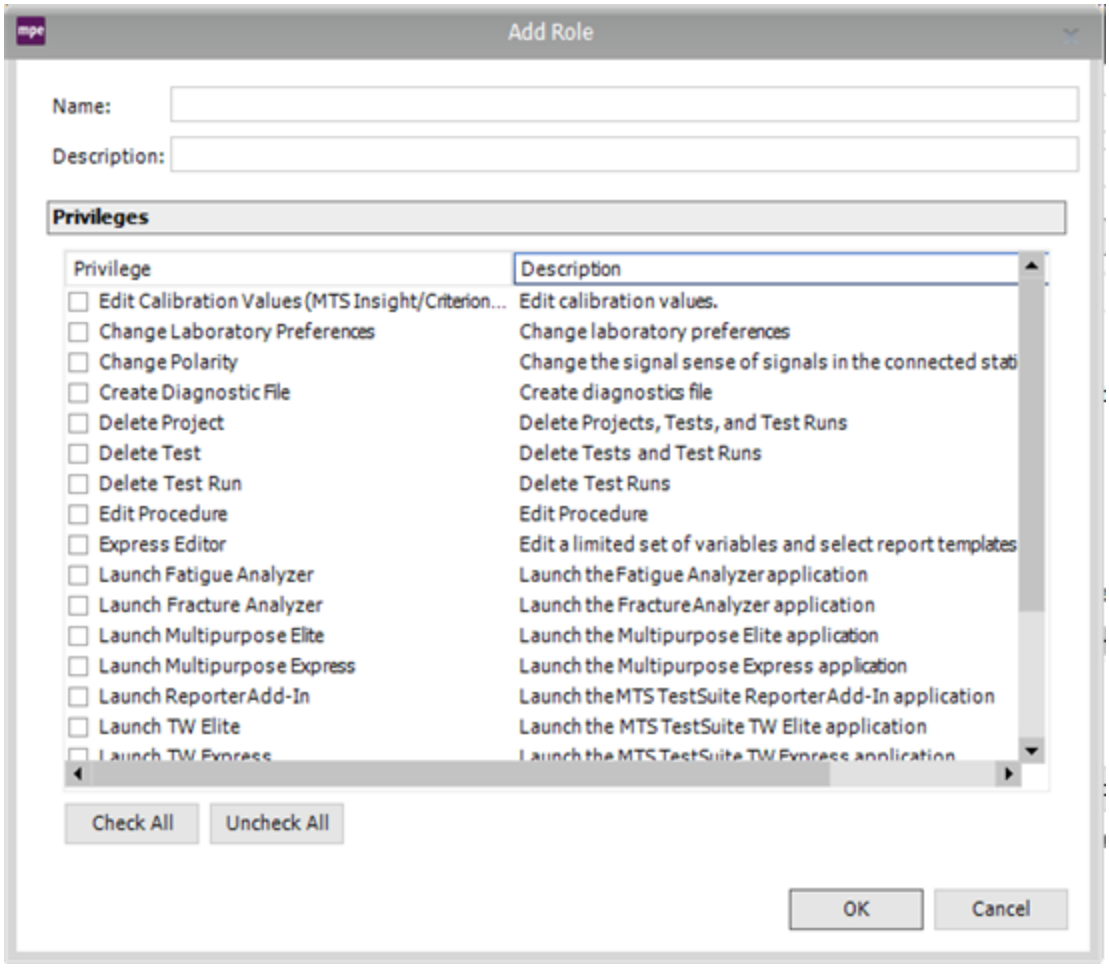
Create a Custom User Role

To create a custom user role:

1. Click **Preferences > User Management > Manage User Accounts**. The Manage User Accounts window appears.
2. Click **Manage Roles**. The Manage Roles window appears.

3. Click **Create Role**. The Add Role window appears.

Add Role Window



4. Enter a **Name** and **Description** for the new role.
5. Select the **Privileges** to assign to the role.
6. Click **OK**.

Default Roles

Use default roles to assign standardized user privileges. Default roles are predefined categories of user privileges. You cannot edit or delete default roles; however, you can create custom roles that you can edit or delete.

Note: The privileges that appear vary depending on the selected role. Only privileges for which you are licensed are shown.

Default User Roles

Default Role Name	Privileges
Administrator	At least one Windows user account must be assigned to the Administrator role. Only the Administrator can add users and assign them roles. The Administrator can also change their own role or assign the Administrator role to another user. The user who installs the application is automatically added as a user and an Administrator. The Administrator role has all privileges, including: • Edit Calibration Values (MTS Insight/Criterion only) • Change Laboratory Preferences • Change Polarity • Create Diagnostic File • Delete Project • Delete Test • Delete Test Run • Edit Procedure • Express Editor • Launch Fatigue Analyzer • Launch Fracture Analyzer • Launch Reporter Add-In • Launch TW Elite • Launch TW Essential • Launch TW Express • Launch Multipurpose Elite • Launch Multipurpose Express • Tuning (MTS Acumen only) • Save Test as Template • User Management • View Reports
Engineer	Includes most privileges, except for laboratory preferences, calibration, and user management. The Engineer role has the following privileges: • Change Polarity

Default Role Name	Privileges
	• Create Diagnostic File • Delete Project • Delete Test • Delete Test Run • Edit Procedure • Express Editor • Launch Fatigue Analyzer • Launch Fracture Analyzer • Launch Reporter Add-In • Launch TW Elite • Launch TW Essential • Launch TW Express • Launch Multipurpose Elite • Launch Multipurpose Express • Tuning (MTS Acumen only) • Save Test as Template • View Reports
Operator	Creates new projects or tests from templates and run those tests. The Operator role has the following privileges: • Create Diagnostic File • Launch TW Elite Launch TW Essential • Launch TW Express • Launch Multipurpose Elite • Launch Multipurpose Express • Tuning (MTS Acumen only) • View Reports  **Note:** Operators may start the system application but have limited capabilities using it. Operator privileges are the same throughout the MTS TestSuite software applications.

Edit a Custom User Role

To edit a custom user role:

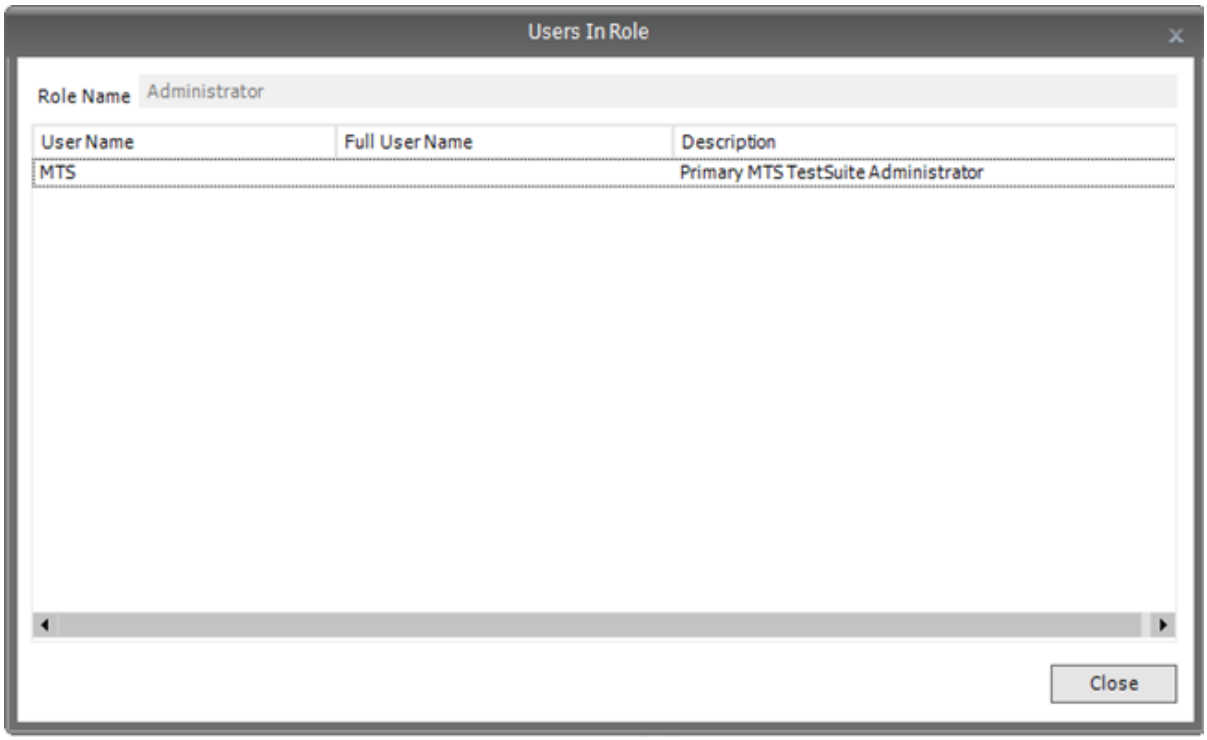
1. Click **Preferences** > **User Management** > **Manage User Accounts**. The Manage User Accounts window appears.
2. Click **Manage Roles**. The Manage Roles window appears.
3. To edit a user-defined role, select the role and click **Edit Role**. In the Edit Role window:
    A. If necessary, change the **Name** or **Description** for the role.
    B. Select the privileges to assign to the role.
    C. Click **OK**.

View Users Assigned to a Role

To view users assigned to a role:

1. Click **Preferences > User Management > Manage User Accounts**. The Manage User Accounts window opens.
2. Click **Manage Roles**. The Manage Roles window opens.
3. Select a role in the **Roles** list and click **Users in Role**. The Users in Role window shows a list of users assigned to that role.

User In Role Window



4. Click **Close**.

Remove a Custom User Role

Note: You cannot remove a role if a user is assigned to the role. Assign the user to another role and then remove the custom role.

1. Click **Preferences > User Management >** Manage User Accounts. The Manage User Accounts window appears.
2. Click **Manage Roles**. The Manage Roles window appears.
3. To remove a user-defined role, select the role and click **Remove Role**.
4. Click **Close**.

Privileges

Note: The privileges that appear vary depending on the selected role. Only privileges for which you are licensed are shown.

Privileges

Privilege Name	Description
Edit Calibration Values	(Applicable on MTS Insight/Criterion only.) Allows an MTS Field Service Engineer to work with TEDS Device files.
Change Laboratory Preferences	Allows you to configure default units and names within projects and tests. It enables: • **Preferences > Configuration > Default Names** • **Preferences > Configuration > Unit Set Manager**
Change Polarity	Allows you to change the signal sense of signals in the connected station.
Create Diagnostic File	Allows you to create a diagnostic file, which is used by MTS Technical Support for diagnostic purposes. It enables **Tools > Create Diagnostic File**.
Delete Project	Allows you to delete a project. It enables the ability to delete a test using **Configuration > Project**.
Delete Test	Allows you to delete a test. It enables the ability to delete a test using **File > Delete Test**.
Delete Test Run	Allows you to delete a test run.
Edit Procedure	Allows you to edit select report templates and a limited set of variable properties (Display Name, Default Value, Default Option, Unit, Pretest, Result, Editable Post-Test, Range, Formatting). The Formula Assistant feature (applicable on MTS Insight/Criterion only) is also available to apply calculation options to a variable.

Privilege Name	Description
Express Editor	Allows you to edit select report templates and a limited set of variable properties (Display Name, Express Editor Default Value, Default Option, Unit, Pretest, Result, Editable Post-Test, Range, Formatting). The Formula Assistant feature (applicable on MTS Insight/Criterion only) is also available to apply calculation options to a variable.
Launch Fatigue Analyzer	Launches the Fatigue Analyzer application.
Launch Fracture Analyzer	Launches the Fracture Analyzer application.
Launch Multipurpose Elite	Allows you to start the Multipurpose Elite application.
Launch Multipurpose Express	Allows you to start the Multipurpose Express application.
Launch Reporter Add-In	Allows you to start the Reporter Add-in to the Microsoft Excel application.
Launch TW Elite	Allows you to start the TW Elite application.
Launch TW Essential	Allows you to start the TW Essential application.
Launch TW Express	Allows you to start the TW Express application.
Save Test as Template	Allows you to save a test as a template.
Tuning (MTS Acumen Only)	(Applicable to MTS Acumen systems only) Allows you to tune control modes.
User Management	Allows you to view information related to users, roles, and privileges. It enables **Preferences > User Accounts**.
View Reports	Allows you to view reports generated by an activity in a test procedure. It enables the **View Report** option from the context menu.

Preferences and Default Settings

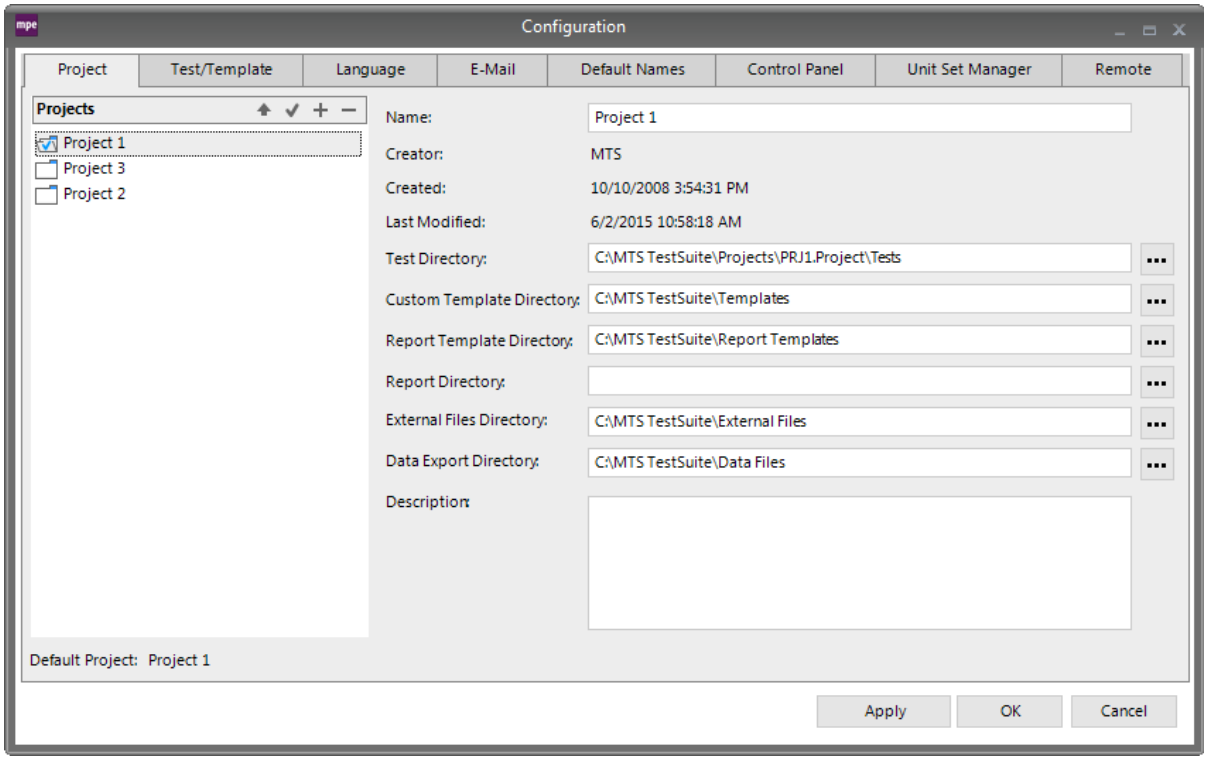
Configuration Window

Access

Preferences > **Configuration** > **Project** tab

Use the tabs in the Configuration window to select preferences and default settings. The window opens to the **Project** tab by default.

Configuration Window



Configuration Window Tabs

Tab	Description
Project	Modifies the settings file of the selected project. Project settings determine the organization and flow of data within a project. Project settings map data paths for the sub folders within projects.
Test/Template	Defines alternative ways to start the test (other than pressing the start button) using keyboard commands and digital inputs. For more information about using remote start, see "Using Remote Start" on page 67. Opens the current test when the application starts. Allows you to define the Default Log Type (Basic or Audit Trail).
Language	Selects a language for the MTS TestSuite application.

Tab	Description
Email	Sets up e-mail for the **Run Report** and **Send E-mail** activities. The **From** e-mail property sets the default for the **From** e-mail address property in the **Run Report** and **Send E-Mail** activities. The other settings are for specifying the e-mail server. Check with your e-mail administrator for the correct settings. You can send a test e-mail to verify the SMTP server settings are working properly.
Default Names	Selects default base names for new projects, tests, and test runs.
Control Panel	Configures the Ramp Rate for the Jog Buttons and the Ramp Rate for the **Return Crosshead to Zero** activity.
Unit Set Manager	Manages unit sets.

Using Remote Start

The settings for the **Digital Input** control support the Remote Test Start feature. The Remote Test Start feature allows the test to start from a digital signal in place of an operator pressing the run button.

Note: Remote Test Start is an advanced feature. It can only be enabled with MTS TW Elite, which has extended editing capabilities, or by an MTS Service Engineer.

If you have MTS TW Elite (TWE) and you want to enable the Remote Test Start feature for a template used with MTS TWS, perform the following:

1. Open the template in MTS TWE.
2. Ensure the **Digital Input** control is selected, and a digital input and transition type are selected in **Preferences** > **Configuration** > **Test/Template**.
3. On the **Define** tab, select **Test Definition** > **General Settings**.
4. Click **Edit**.
5. Select Enable Remote Test Start using Digital Input.
6. Save the template in MTS TW Elite.
7. Open the template in MTS TWS. The test can now be started remotely from the defined digital input signal.

Project Management

Working with Projects

Access

Preferences > **Configuration** > **Project** tab

System Default Project

When you install MTS TestSuite software, the MTS installer creates the *system default project* and is labeled Project 1. This project is the location where the installer places or updates example tests. The system default project cannot be deleted. However, you can change the properties of the system default project.

Add a project

To add a project, click the **+** button in the upper right side of the **Projects** list. The project is automatically created using the default naming convention specified on the **Default Names** tab.

Delete a project

To delete a project, select the project you want to delete and click the **–** icon in the upper right corner of the **Projects** list.

Alternately, you can right-click the project you want to delete and select **Remove**.

Important: When you delete a project, all the folders within that project's folder are deleted. If the Test Directory for the project is configured to store tests within the project's directory (default), those tests will also be deleted. For more information about the Test Directory, see "Test Directory" on page 69

Note: You cannot delete a project if a test within the project is open.

Note: You cannot delete the system default project (that is, Project 1).

Open a project

To open a project, double-click the project name, or right-click the project you want to open and select the **Open Project** menu option.

Note: If a test is open, you cannot open another project.

Edit a project

To edit a project, select the project in the **Projects** list and change the associated project settings as desired. Your changes are saved when you click the **Apply** or **OK** buttons in the Configuration window.

Set a project as the default project

You can change the default project, which is the project that automatically opens when the MTS TestSuite software starts. This selection is stored specific to your user credentials. If you do not have a default project set, the application uses the system default project as your initial default.

To set a project as the default project, right-click the project name you want to set as the default, and select **Set Default Project**.

Project Tab

Access

Preferences > **Configuration** > **Projects** tab

Use the **Project** tab to manage projects. With this tab, you can see all the projects, add and remove them, and see individual project settings.

As you select projects in the **Projects** list, the settings for the selected project are shown to the right. By default, the directory settings are global locations shared between all projects but you can change this setting for a particular project.

Note:
If you change any of the default directories for a project, an asterisk (*) appears next to the project.

Project Settings

Setting	Description
Name	Specifies the name of the project. This name is shown in the **Projects** list, as well as in the title bar of the MTS TestSuite application.
Creator	Displays the name of the person who created the project. This may be empty if user management s disabled. This field is read-only.
Created	Displays the date and time that the project was created. This field is read-only.
Last Modified	Displays the date and time that the project was last modified. This field is read-only.
Test Directory	Specifies the directory in which the project looks for and stores tests. By default, this directory exists under the project's directory, but can be set to any valid directory, including a network location. **Note:** If tests are stored in a project directory (default), they are deleted if the project is deleted. **Note:** This field is read-only if there is a test open when the project settings view is shown.
Custom Template Directory	Specifies the directory in which the MTS TestSuite application looks for existing custom templates and stores new custom templates. **Note:** MTS supplied templates are available from the MTS Templates list on the start page or from **MTS Templates** list on the **Select** tab.
Report Template Directory	Specifies the directory in which the MTS TestSuite application looks for and stores report templates.
Report Directory	Specifies the directory in which the MTS TestSuite application stores generated reports. By default, this field is blank and reports are stored in a test runs subdirectory of the project.
External Files	Specifies the directory in which the MTS TestSuite application looks for external files (such as XML files used to create tests).

Setting	Description
Directory	
Data Export Directory	Specifies the directory in which the MTS TestSuite application saves data export files created by the **Export Data** activity.
Description	(Optional) Displays a description entered about the project.

Audit Trail

Access

When activated, the Audit Trail feature allows you to check the integrity of your data files from a selection in the **File** menu. If the application detects any change, the files that do not match are listed in a message window. It also writes information about the controller's TEDS devices to the test log every time the test is run, including model and serial number, version and manufacturer number, and calibration date.

Note: The Audit Trail is an advanced feature. It can only be enabled with MTS TW Elite (TWE), which has extended editing capabilities, or by an MTS Service Engineer.

If you have MTS TWE and you want to enable the Audit Trail feature for a template used with MTS TWS, perform the following:

1. Open the template or test in MTS TWE.
2. Ensure the **Log Type** is **Audit Trail** in **Preferences** > **Configuration** > **Test/Template**.
3. Select **Test Definition** > **General Settings**.
4. Click **Edit**.
5. For **Log Type**, select **Audit Trail**.
6. Save the template in MTS TW Elite.
7. Open the template in MTS TWS. The **File** menu now includes a **Check Test Audit Trail** selection.

Set the Log Type to Audit Trail for a Test

Note: This setting only applies to the current test.

1. Click the **General Settings** tab.
2. Click **Edit**.
3. In the Log Type list, click **Audit Trail**. The Check Test Audit Trail option becomes available in the **File** menu.

Set the Default Log Type to Audit Trail

When the Default Log Type is set to **Audit Trail**, each new test that is created has its log type set to Audit Trail rather than Basic.

Note: This setting does not affect the current test.

1. On the **Preferences** menu, click **Configuration**.
2. Click the **Test** tab.
3. In the Default Log Type list, click **Audit Trail**.

Check the Test Audit Trail

1. Make sure that the Log Type for the test is set to **Audit Trail**.
2. Click **File** > **Check Test Audit Trail**. Files that do not match the audit trail are listed in a window. If verification is successful, a message states all files associated with the test have been verified.
3. Click **OK**.

Define a Keyboard Shortcut to Start the Test

You can define a keyboard shortcut to start tests from your keyboard. To do this:

1. On the **Preferences** menu, click **Configuration**.
2. Click the **Test** tab.
3. Select the **Function Key Shortcut** check box.
4. Click inside the field next to the **Function Key Shortcut** checkbox and define the shortcut by typing one of the following key combinations.

Note:
You cannot use a keyboard shortcut to resume a test.

Function keys

F1 - F10

Shift key combinations

- Shift Ins
- Shift Del
- Shift F1 - Shift F12

Control key combinations

- Ctrl Ins
- Ctrl Del
- Ctrl 1 - Ctrl 9

- Ctrl A - Ctrl Z
- Ctrl F1 - Ctrl F12
- Ctrl Shift 0 - Ctrl Shift 9
- Ctrl Shift A - Ctrl Shift Z
- Ctrl Shift F1 - Ctrl Shift F12

Alt key combinations

- Alt Backspace
- Alt Left Arrow
- Alt Up Arrow
- Alt Right Arrow
- Alt Down Arrow
- Alt 0 - Alt 9
- Alt F1 - Alt F12

E-Mail Overview

Access

Preferences > **Configuration** > **E-Mail** tab

You can set up e-mail activities to automatically notify interested parties of test progress. You can insert variables into the e-mail message and attach the message log. You can also send a report attached to an e-mail. The **Send E-Mail** and the **Run Report** activities require you to configure your SMTP server settings if you want to use the e-mail activities in your test workflow. The e-mail configuration uses the SMTP to relay e-mail messages for delivery.

Configuring SMTP Server for E-Mail Activity

Request the relevant SMTP server settings from your system administrator. To configure e-mail notification:

1. Click **Preferences** menu > **Configuration** option > **E-Mail** tab.
2. Enter the default e-mail address in the **From** address box.
3. Enter the name of your server in the **Server name** field.
4. Enter the TCP Port in the **SMTP port number** field. If your server supports secure connections, make sure the **Enable SSL** check box is selected.
5. Adjust the default value in the **Timeout** field if necessary.
6. If your server requires authentication to relay mail to non-local (external) users, select the **My server requires authentication** check box. The **Account Name** and **Password** fields are displayed.

A. Enter your Account Name.

B. Enter your Password.

7. Enter an e-mail address in the **Send test e-mail to address** field and click **Send**.
   - If successful, an e-mail sends the SMTP settings to the specified test e-mail address. A message window instructs you to check your Inbox. Click **OK**.
   - If the test e-mail is not successful, a message window instructs you to check the Application Log for details on the error message. Click **OK**. Correct the error and resend the test e-mail.

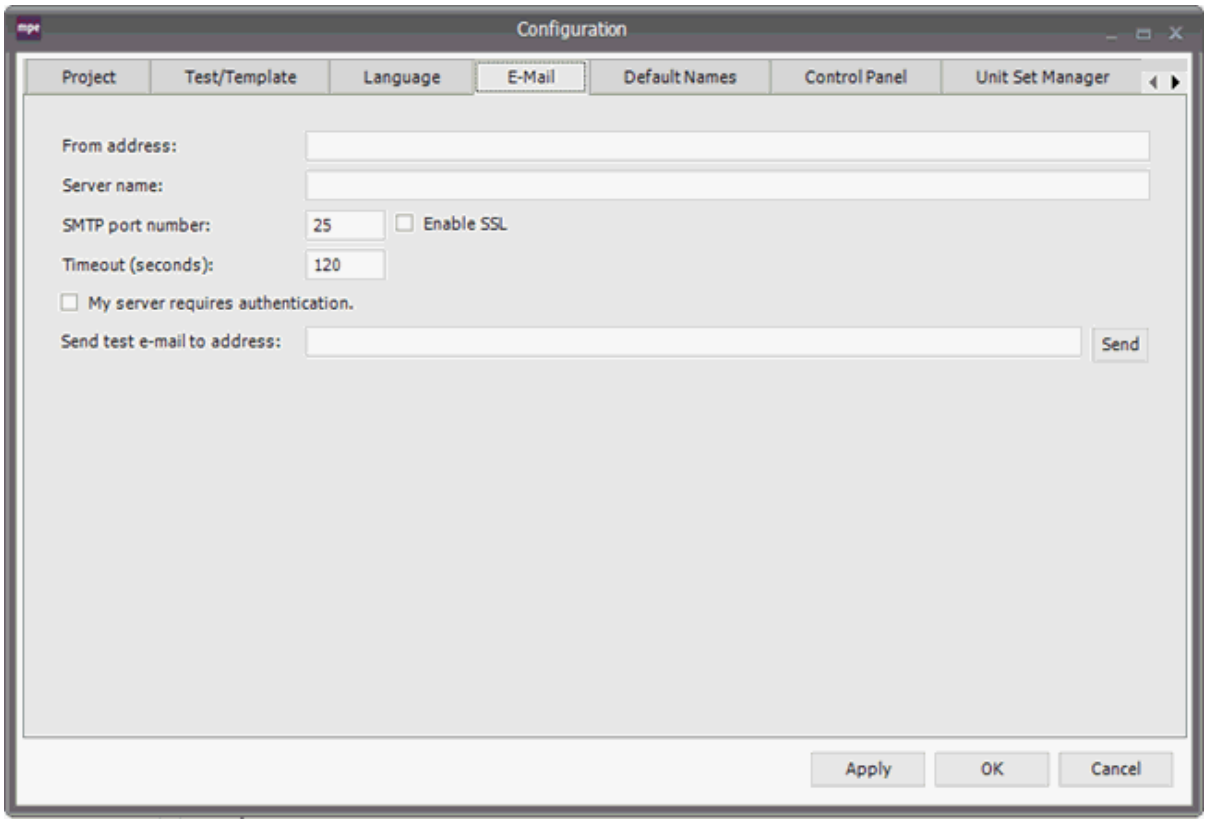
E-Mail Settings

Access

Preferences menu > **Configuration** > **E-Mail** tab

Use the **E-Mail** tab of the Configuration window to enter e-mail server settings.

E-Mail Tab—Configuration Window



The following table describes the fields for configuring E-Mail for test activities.

E-Mail Settings

Setting	Description
From address	The default address from which e-mail is sent. The **From** address entered in the E-Mail Configuration window automatically populates the **From** address in the Properties panels of the **Send E-Mail** and **Send Report to E-Mail** of the **Run Report** activities.
Server name	The name of the SMTP server for your organization.
SMTP port number	The SMTP (Simple Mail Transfer Protocol) port number is typically TCP Port 25. If the server is set up for SMTP, the port number is typically 465.
Enable SSL	If you server does not support SSL (Secure Sockets Layer), clear the **Enable SSL** check box. If your e-mail server supports secure connections, leave the **Enable SSL** check box selected. Default: Enabled
Timeout (seconds)	Adjust the default timeout value at the recommendation of your e-mail system administrator. Default: 120 seconds
My server requires authentication	Select the check box if your server requires authentication to relay mail to e-mail addresses that are external to your organization. Enter your **Account Name** and **Password**. Clear the check box if your server does not require authentication; that is, you are sending e-mail only internally. Default: Enabled
Send test e-mail to address	Enter the e-mail address to which you want to send a test e-mail. This is highly recommended to ensure you have correctly configured this option prior to using the **Send E-Mail** activity in a test. The body of the test message contains your SMTP server settings. The test e-mail address is cleared when you exit the Configuration window.

Control Panel Settings

Access

Preferences menu > **Configuration** > **Control Panel** tab

Use the **Control Panel** tab to configure the operation of the Jog and Return buttons on the Application control panel.

Note: When connected to an MTS FlexTest controller, the Return panel allows specifying the Ramp Rate for the Return to zero position. When connected to an MTS Insight controller, the Ramp Rate for the Return must be specified by an MTS Field Service Engineer.

Units Management

Unit Set Overview

Access

Preferences menu > **Configuration** > **Unit Set Manager** tab

Standard unit set

Use the **Unit Set Manager** tab to assign a unit set at the system level and at the project level. Unit sets are a collection of dimensions and associated units. For example, the unit set Si (mm-N) has a length dimension of millimeters (mm).

MTS TestSuite applications contain several built-in unit sets.These sets are read-only and cannot be modified.

- If a unit set is not defined as the system default, the application uses the **MTS TestSuite Default** unit set.
- Variables and charts can point to specific units or use the default unit set for the project.

Custom unit set

You can add custom unit sets that can be exported to or imported from other projects. You can select various units for the dimensions in your custom set. For more information about adding a custom unit set, see "Add a Custom Unit Set" on page 77.

Predefined Unit Sets

MTS TestSuite includes the following predefined unit sets:

Predefined Unit Sets

Unit Set	Description
cgs	Centimeters-Grams-Seconds. Defines a set of units that is based upon centimeters, grams, and seconds. Force is expressed as "dyn."
mks	Meters-Kilograms-Seconds. Defines a set of units that is based upon meters, kilograms, and seconds. Force is expressed as "N."
MTS 793	Millimeters-Kilograms-Seconds. Defines a set of units that is based upon millimeters, kilograms, and seconds. Force is expressed as "kN."
MTS TestWorks 4	Millimeters-Kilograms-Seconds. Defines a set of units that is based upon millimeters, kilograms, and seconds. Force is expressed as "kgf."
SI (mm-	Systeme International d'Unites. Defines a set of units that contains customary

Unit Set	Description
kN)	international (metric) units. It provides force-related units in “kN” and length-related units in “mm.”
SI (mm-N)	Systeme International d’Unites. Defines a set of units that contains customary international (metric) units. It provides force-related units in “N” and length-related units in “mm.”
US (in-kip)	Defines a set of units that contains customary U.S. engineering units. It provides force-related units in “kip” and length-related units in “in.”
US (in-lbf)	Defines a set of units that contains customary U.S. engineering units. It provides force-related units in “lbf” and length-related units in “in.”
MTS TestSuite	Meters-Kilograms-Seconds. Defines a set of units that is based upon meters, kilograms, and seconds. Force is expressed as “N.”

Unit Set Manager Overview

Access

Preferences menu > **Configuration** > **Unit Set Manager** tab

The Units Manager window enables you to assign unit sets, such as International System of Units (SI), United States Customary System of Units (US), or custom unit sets as your MTS TestSuite or project default. You can copy a predefined unit set and manage your custom unit sets using the following buttons:

- **Add**
- **Delete**
- **Edit**
- **Copy**
- **Export**
- **Import**

Default Unit Set Selections

To assign a unit set as the default for MTS TestSuite, in the **MTS TestSuite Default** box, click the list arrow and select a unit set from the dropdown list. Perform the same action to assign a unit set as the default for **Project Default for "Project 1"**.

Manage Unit Sets

To view the specific units of measurement defined for a unit set, click the name of a unit set in the Unit Sets panel; the units are then displayed in the Units column.

A project with an assigned unit set uses the unit set for all dimensions in the project procedures. A project without an assigned unit set uses the set defined as the system default.

To sort a list, click the column title.

Unit Set Manager Properties

Unit Set Manager

Item	Description
MTS TestSuite Default	Shows the current default unit set for the applications. Click the list arrow to assign another predefined or custom unit set.
Project Default for <ProjectName>	Shows the current assigned unit set for the project. Click the list arrow to assign another predefined or custom unit set.
Name	Shows the predefined unit sets and the custom unit sets.
Type	Shows the type of unit set: predefined or custom.
Dimension	Shows the type of dimension.
Unit	Shows the unit selections for that type of dimension. If the unit set is a custom unit set, click the list arrow to display selections for that type of dimension.
Add	Open the Add Custom Unit Set window where you can add a custom unit set. Enter a display name and description, and select between an International System of Units (SI) or a United States Customary System of Units (US).
Delete	Delete a custom unit set. You cannot delete a predefined system unit set.
Edit	Open the Edit Custom Unit Set window where you can change the display name or description. You cannot change the type of unit system or the individual unit settings.
Copy	Open the Copy Unit Set window to make a copy of the highlighted unit set. If you copy a predefined unit set, the copy becomes a custom unit set in which you can change the unit settings.
Export	Export a custom unit set to another project or analysis. Click the required custom unit set (.tsunitset) to export.  **Note:** If your project uses a custom unit set, it also exports with the project.
Import	Import a custom unit set from another project or analysis. Click the required custom unit set to import (.tsunitset).
Apply	Save any changes and keep the Unit Set Manager window open. This button is useful if you are making multiple custom changes.

Add a Custom Unit Set

To add a custom unit set:

1. Click **Preferences > Configuration > Unit Set Manager** to open the Unit Set Manager.
2. Click **Add** to open the Add Custom Unit Set window.
3. Enter a name for your unit set in the **Display Name** box.
4. Enter a description in the **Description** box.
5. (Optional) Click the drop-down button to select **US Customary Units** in the **Unit System** box as the predefined settings for your custom unit set. This optional step can minimize the number of individual changes required to customize your unit set. You are not required to select a unit system type.
6. Click **OK**.

Add a Custom Dimension

In the Unit Set Manager window, you can create a custom dimension. After creating a custom dimension, you must add at least one custom unit to the dimension. All custom dimensions that you create are visible in all unit sets.

You may want to create a custom dimension if your testing requires a dimension that does not already exist in one of the default unit sets. For example, you could create an Inverse Energy per Volume dimension and then add a default custom unit of 1/lbf*in².

To add a custom dimension:

1. Click **Preferences > Configuration > Unit Set Manager** to open the Unit Set Manager.
2. In the **Dimensions** section of the Unit Set Manager, click the **Add a new item** icon (**+**). The Add Custom Dimension window appears.
3. Right-click the dimension and select **Edit**. The Edit Dimension window appears.
4. Enter an **Internal Name** and **Display Name** for the new custom dimension.
5. Click the **Add a new item icon** (**+**) and enter an **Internal Name** and **Display Name** to create the a unit for the dimension. By default, this unit will be the default unit, but you can click the green check mark icon to set a different unit you add as the default unit. For detailed information about creating and editing custom units, see "Add a Custom Unit" on page 78.

Add a Custom Unit

In the Unit Set Manager window, you can create a custom unit for a dimension. When you add a new unit to a dimension, the unit can be used in various places throughout the application the same way other units are used. For example, you could configure a meter to display real-time data using the custom unit on the **Meters** tab. Additionally, you could configure a variable used in your test procedure to store data using the custom unit on the **Variables** tab.

It may be necessary to create a custom unit if your testing requires a unit that does not already exist in the available unit sets. For example, if you are measuring stress, there are several commonly-used units available (such as lbf/ft²). However, other units of stress (such as N/100cm²) are not available in any of the default unit sets. Therefore, you may want to add a N/100cm² unit to the stress dimension if it is required for your testing.

To add a custom unit:

1. Click **Preferences > Configuration > Unit Set Manager** to open the Unit Set Manager.
2. In the **Dimensions** section of the Unit Set Manager, select the dimension to which you want to add a custom unit.
3. Right-click the dimension and select **Edit**. The Edit Dimension window appears.
4. Click the **Add a new item** icon (**+**). The Add New Unit window appears.
5. Enter an **Internal Name** and **Display Name** for the new custom unit.
6. Click **OK** to close the Add New Unit window.
7. (Optional) Click the editable fields under the Scale and Offset columns to specify a scale and offset for the unit.
    - The scale determines how the unit scales with respect to other units in the dimension. To understand how scale works, consider time measurements. Seconds are treated as the base unit, so they have a scale of 1. Minutes and hours are scaled depending on how many seconds they contain, so their scales are 60 and 3,600 respectively.
    - The offset is a positive or negative value that is always applied to the unit. To understand how offset works, consider temperature measurements. K (Kelvin) is treated as the base unit, so it has a scale of 1 and an offset of 0. °C also has a scale of 1 because it scales linearly with K (that is, a 1 K change is the same as a 1°C degree change). However, because converting from K to °C requires subtracting 273.15,°C has an offset of -273.15.
8. Click **OK** to close the Edit Dimension window. The new unit is added to the dimension.

Remote Server Settings

Access

Preferences menu > **Configuration** > **Remote** tab

Use the **Remote** tab to enable and use remote extensions, if available. This tab is used to set up MTS TestSuite to communicate with MTS Echo.

Remote Panel Settings

Item	Description
Enable Remote Services	Enables the use of remotes services with MTS TestSuite.
Server URL	If desired, enter the URL of a custom remote server. By default, the built-in remote server will be used.
Hide Server Console	Select this option to hide the remote server console window.

Resources

Working with Resources

Resources Overview

Fastpath:
Define tab > **Resources** node

The **Resources** node in the test definition tree of the **Define** tab serves as a map between controller configuration resources and the test definition. The **Resources** node allows you to define a test independently, away from the controller, and then map the resources to the controller before the test is loaded. The application stores the resources with the test definition. You can re-map the resources to accommodate particular controller configurations.

Test resources can include:

- Actions
- Analog Outputs
- Channels
- Digital Inputs
- Digital Outputs
- Float Signals
- Integer Signals

Controller resources

You must map the test resources to controller resources before you can load and run the test procedure on that controller. Resources are stored with the test definition and validated against the available controller resources the next time the application connects to a controller.

Template resources

You can map controller resources and then create a test template that includes the controller resources at the time the template is created. You can then remap any resources as necessary to the controller the next time the application connects to the controller.

Imported resources

When resources are correctly mapped in a test, you can create other tests offline, and import the resources between tests. You can also manually add and map individual resources.

Simulated resources

If you are not able to connect to the physical controller, you can use the controller software in simulation mode and use a pre-configured controller configuration file. In this case, the simulation uses the controller configuration file as its source for controller resources.

Internal Name and Display Name

Internal Name

Generally, this is a name that is used internally to the application or by a calculation. These names do not usually contain spaces and have character restrictions. In some cases, internal names are assigned by the application and cannot change, and in other cases they are defined by the user and can change. Because other parts of the system use the internal name, some additional restrictions are in place to make sure that changing the internal name does not cause the program to fail. In some cases (such as with variables), the name is adjusted where it can be, but in others (like test resources) it will not allow the change to take place. These strings do not change when the language is changed.

Display Name

This is a display version of a string. It has some restrictions on allowed characters, but is much more open than the Internal Names. These strings can be changed by the user without changing any other part of the application. Some character restrictions apply, and some validation is done so two items at the same level do not have the same name. These strings are translated and will change based on language if it was saved in multiple languages (such as with templates).

Resource Details

Fastpath:
Define tab > **Resources** node

Resources Tab

Item	Description
Name	Shows the names of the resources that are used in the test procedure activities. The **Show Internal Names** check box toggles internal or display names in the Name column. Typically, display names are used instead of internal names. The Name column shows resources in a hierarchical view organization, allowing the expand-collapse of the individual resource entries. A context right-click menu is also available on the entries in the Name column that allows a variety of operations (add, delete, and rename) that are specific to each entry. Internal names are not typically edited and cannot be edited if the resource is in use somewhere in the test procedure. Display names can be edited whenever the test definition is editable. Test resources cannot be deleted if they are in use somewhere. Test resources can be created whenever the test definition is editable.
Dimension	The dimension of a test resource is only editable if no other test definition modules use the resource (such as activities and runtime components). If you are connected to a controller, do not change the dimension because that may invalidate the controller resource assignment and show an error symbol.
Polarity	Use this setting to change the polarity of a float signal. Changing the setting affects the **Effect of Increasing Extension** setting. Available settings are: • Normal

Item	Description
	• Inverted
Use Count	Track the use of the test resource by other parts of the test definition. If a test resource is in use, you cannot delete it.
Controller Resource	Assign names to the resources that are expected in the configuration when Test Manager is connected to the controller. The resource table maps the **Controller Name** to the test **Name**. The two names can be different. Only the **Controller Name** must match what is used in the configuration file. The Name column shows the names of the resources that are used in the test procedure activities. The **Show internal names** check box toggles internal or display names in the Name column. Typically, display names are used instead of internal names. The use of this column is different between online and offline modes.
Controller Resource (when connected to a controller)	Where a cell is editable in the column, a drop-down list shows the controller resources that are compatible with a given test resource. Resources are shown in the drop-down list that match the dimension. The edit line contains the current assignment, which may be in the list. If a displayed resource is not in the list of compatible resources, an error symbol is shown. You can select a resource from the list, or type the name into the edit box. The Controller Resource name contains a red error symbol if a conflict occurs between the resource in Test Manager and the resource in the controller configuration. Controller configuration resources can be uniquely identified by either their internal names or display names without knowing which is being specified. Therefore, you can type either the display name or internal name into the Controller Resource column. The box is updated to reflect the appropriate name, based on the **Show Internal Names** check box.

Item	Description
Controller Resource (when not connected to a controller)	The list is empty because no controller is connected from which to query the valid resource names. The edit box shows the internal name. If you type a different name, it is stored as the internal and display name. There is no validation error since there are no controller resources to validate against.
Capacity	Click **Check** to define Max/Min resource capacity for the signal and initiate a check against these settings when you attempt to run the test. For example, the Load Cell Capacity configuration item is used to define an optional minimum and optional maximum capacity for a load cell that is used with a specific template. A check is performed between the defined minimum/maximum capacity with the capacity of the load cell being used. If the load cell is inadequate, the test cannot start.
Effect of Increasing Extension	**Note:** This setting is only available with MTS TestSuite TW or TWS software running on an MTS Insight Controller. This setting defines the relationship between the crosshead/displacement channel and each float signal. • Select **Increases Value** when an increase in the crosshead or actuator extension will increase the value of the signal. • Select **Decreases Value** when an increase in the crosshead or actuator extension will decrease the value of the signal. • Select **Indefinite** for signals (such as time) where there is no direct relationship to the crosshead or actuator extension.

Resource Buttons

Resources Node Buttons

Item	Description
Add Resource	Add a resource to the table. Right-click the entries in the Name column for a menu that contains a variety of operations (add, delete, and rename) that are specific to each hierarchy entry. These operations are necessary if no controller or test resource information is available to import.
Import Resources > Import all unused controller resources	This function only works when connected to a controller. Test resources are created that correspond to all controller resources on the controller that is currently connected. If the table is empty, this function imports all the resources. If some resources are mapped already, these are not imported. If conflicts occur between existing and new test resources, the names are modified to be unique.
Import Resources > Import selected controller resources	This function only works when connected to a controller. A window opens where you select which resources to import. This function is useful if you have an existing test and must add a reference to a few resources that were not used previously.
Import Resources > Import resources from another test	Choose another test to import the test resources from.
Delete Unused	Delete controller resources that are listed in the resources table but are not used anywhere in the test definition. Typically, a user can add all possible controller resources, define the test, and delete the unused resources when the test definition is complete. This approach keeps the resource list to a minimum with only the required resources listed for the test, which makes the list easier to maintain if controller resources change in the future.
Use Controller Names	Resources currently shown in the Controller Resource column are copied to the test resource Name column. The resources in the test procedure are changed to reflect the items in the Name column. Use this feature if you want all resource names to match (between the test procedure and the controller configuration).

Import Test Resources

Note: This procedure applies only to the **Import Resources** button on the **Resources** node. You can only import test resources from the connected controller or another test in the same project.

When you import an entire project, the test resources can be included for each test in the project. Depending on the origin of the information, the imported resources may not map to the controller resources of the control system networked with your session or workstation.

To import test resources:

1. Open the desired test.
2. If you want to import resources from the test controller, select **Controller > Connect**.
3. Click the **Define** > **Resources** node of the test definition tree.
4. Click the **Import Resources** button on the work space to show the options menu.
5. To import resources from a connected controller, select one of the following options:
   - Click **Import the selected controller resources** to select which unused resources you want to add to the test and continue with step 6 below.
   - Click **Import All Unused Controller Resources** to import all the controller resources that are not currently in use by the test and continue with step 7 below.
6. If you selected **Import selected controller resources:**
   A. Expand the unused resources hierarchy and select the check boxes for the resources to import, or click **Select All** to select all unused resources.
   B. Select the **Show Internal Names** check box to show the internal names for the resources in the work area instead of their display names.
   C. Click **OK**.
   D. The table of test resources updates and a message window indicates that the import was successful. Click **OK**.
   E. If all the controller resources are already in use by the test, a corresponding message is shown. Click **OK**.
7. To import resources from another test:
   A. Click **Import resources from another test** to overwrite the test resources with the imported resources.
   B. In the Select Test for Resource Import window, select the required test, and then click **OK**.
   C. The table of test resources in the work area is updated and a message indicates that the import was successful. Click **OK**.

About Disabled Resources

To add versatility to templates, you can include variables, activities, and resources that you can disable or enable depending on the specific type of test you wish to perform. This allows one test procedure to contain resources for the various functions, while allowing the operator to “turn off” portions of the test procedure as needed. Typically, a combination of variables, activities, and resources might all need to be disabled in order to effectively turn off a portion of a template.

If a disabled resource is referenced by an enabled resource, variable, or activity, a validation error occurs for that reference, and the test will not run.

Disabled resources appear grayed out on the **Resources** node and in resource lists in other areas of the application.

Disable Resources

1. On the **Resources** node, locate the resources that you want to disable. If a validation error occurs, use the associated error icons to help locate resources that should be disabled.
2. Right-click the resource and select **Disable (resource)**.

NI M Series Multifunction DAQ

Note:
This feature only applies to MTS Criterion and MTS Insight systems.

Warning:
The information contained in this document should only be used by qualified personnel

Misunderstood, misread, or misapplied information used to set up and operate an MTS test system can expose personnel and equipment to severe hazards. This can result in damage to equipment (including test articles) and injury or death to personnel.

Before you use the information in this document, verify your qualifications with your system administrator or MTS.

MTS TestSuite supports the National Instruments (NI) M Series Multifunction Data Acquisition (DAQ) driver. It supports Analog Input (AI), Digital Input (DI), and Digital Output (DO) signals.

Analog Input (AI) signals can be configured to single-ended non-referenced, single-ended referenced, or differential if it is set up as a Voltage or Bridge device (Measurement Type). If it is setup as a Thermocouple, it has a number of thermocouple types instead. You cannot mix differential and single-ended signals on a test system. The virtual Transducer Electronic Data Sheet (TEDS) defines the scaling, dimension, and units of the AI signal. For more information about TEDS, see “TEDS Devices” on page 89.

The DI and DO signals are grouped into ports of 8 digital signals which are either all inputs or all outputs. DO signals support Toggle, Pulse, Set, and Clear.

AI and DI signals can be used in limit detectors.

For more information about National Instruments Data Acquisition products, see their website at National Instruments Data Acquisition (DAQ).

Note: Before purchasing NI M series hardware, contact MTS Technical Support to confirm your requirements.

Configuration

To configure the NI M Series Multifunction DAQ for your test, you must modify the TW Diag configuration file. Contact MTS technical support for this procedure.

Once configured, the devices will appear in your test and can be modified from the TW application **Resources** tab. For more information about the **Resources** tab, see "Resource Details" on page 83.

TEDS Devices

When the TW application is used with an MTS Criterion system, the application requires the storage of calibration information in a TEDS (Transducer Electronics Data Sheet) device or a virtual TEDS file. The TEDS assignments and other settings made in the TEDS Devices window are saved as controller resources and appear in each test that is run on that controller.

Note: The following TEDS procedures only apply to MTS Criterion and MTS Insight systems.

TEDS Devices Window

Note: This feature only applies to MTS Criterion and MTS Insight systems.

Access

Controller menu > TEDS Devices window

The TEDS Devices window shows the calibration source (virtual TEDS file or TEDS device) and provides access to the calibration settings for the signals that use a virtual TEDS file.

Overview

The TEDS Devices window shows all the signal resources and TEDS assignments associated with the controller and allows you to:

- Create a virtual TEDS file for a controller resource.*
- Update a virtual TEDS file.*
- Assign a virtual TEDS file to a controller resource.
- Configure a device verification for the selected device.
- Perform a device verification for the selected device.
- View the Device Verification History for the selected device.

* Requires the Edit Calibration Values privilege.

Note: Editing the information for a device that has a TEDS chip can only be performed by an MTS Field Service Engineer.

Display and Edit the Virtual TEDS Information for a Signal

Note: Editing the information for a device that has a TEDS chip is typically only performed by an MTS Field Service Engineer.

Note: This feature only applies to MTS Criterion and MTS Insight systems.

1. On the **Controller** menu, click **TEDS Devices**.
2. Select a signal (with a TEDS file assigned as a source) from the signal list.
3. Click **Details**. The TEDS Details window appears.
4. Edit the TEDS settings as required.

   Note: To create a backup copy of the TEDS file that you can revert to if needed, click **Save As** before making any changes.
5. Click **OK** to save your changes.

   Any changes will overwrite the settings in the TEDS file that is assigned to the signal.

Create a Virtual TEDS File

This feature only applies to MTS Criterion and MTS Insight systems.

Note: This feature only applies to MTS Criterion and MTS Insight systems.

1. On the **Controller** menu, click **TEDS Devices**.
2. Select a signal.
3. Click **New**.
4. If the selected signal is an encoder, continue to the next step. If the selected signal is _Load, _Strain1, or _Strain2, select the type of device in the TEDS Selector window:
   - Analog Input Device: Bridge-type sensor such as a load cell or strain gage.
   - High-Level Voltage-Output Sensor: Non-bridge sensor (such as a string pot) that provides a high-level voltage output.
5. The TEDS Details window appears with default settings.
6. If necessary, edit the settings and click **OK**.
7. In the Save As TEDS window, type a name for the TEDS file and click **Save**.
8. The new TEDS file is now listed as the source for the signal.
9. Click **OK**.
10. Click **Yes** to save the settings as a new file.

Assign a Virtual TEDS File to a Signal

Note: This feature only applies to MTS Criterion and MTS Insight systems.

1. On the **Controller** menu, click **TEDS Devices**.
2. Select a device.
    - If a device already has a file assigned to it, click **Remove** to remove the source assignment.
    - If a device does not have a file assigned to it, click **New**.
3. Click **Open**.
4. In the Open TEDS window, select a compatible TEDS file.

   The default directory for virtual TEDS files is: MTS TestSuite\Devices.
5. Click **Open**.
6. Perform a device verification. For more information about performing device verification, see "TEDS Device Verification Checks" on page 91.

Add TEDS Information to the Test Run Log

When the Log Type for the test is set to Audit Trail, each time a test is run, the application writes the TEDS model and serial number to the test run log.

Note: This feature only applies to MTS Criterion and MTS Insight systems.

Default Log Type Setting

On the **Preferences** menu, click **Configuration** and then click the **Test** tab. This setting applies to all tests.

Test Log Type Setting

Open a test, click the **Define** tab, click the **Settings** subtab, and set the Log Type to Audit Trail. This setting only applies to the current test.

TEDS Device Verification Checks

TEDS Device Verification Check procedures only apply to MTS Criterion and MTS Insight systems.

Perform a Device Verification Check

A device verification check compares device readings recorded during actual calibration to the readings recorded during the check to help identify problems associated with a device.

1. On the **Controller** menu, click **TEDS Devices**.
2. Select a signal from the signal list.
3. Click **Verify**.
4. Follow the on-screen prompts to verify the device.

View the Device Verification History

The Device Verification Check History window shows the history of all the device verification checks run against the selected signal (device).

1. On the Controller menu, click **TEDS Devices**.
2. Select a signal.
3. Click **History**.

Device Verification Settings

Device verification prompt

The device verification prompt is used to remind an operator to run a device verification before being running a test. To configure a device verification prompt:

1. On the **Controller** menu, click **TEDS Devices**.
2. Select a signal from the signal list.
3. In the TEDS Details panel, select the Device Verification Prompt for the device. For example, if set to Daily, each day, the operator will be prompted to run a device verification before being allowed to run tests.

Warning/Lock-Out Ranges

During a device verification, the application compares the new signal value obtained during the check to the value obtained when the device was calibrated. To configure a warning/lock-out range for a device:

1. On the **Controller** menu, click **TEDS Devices**.
2. Select a signal from the signal list.
3. Set the **Warning** and **Lock-Out** ranges.
    - If the difference between the new value(s) and the original value(s) is beyond the Warning Range, a warning message appears. The user must determine whether to proceed with testing or to resolve the problem with the device.
    - If the difference between the new signal value and the original signal value is beyond the Lock-Out Range, an error message appears. The application will not allow you to run a test until the problem has been resolved.

About the Devices Window

Access

The **Controller** menu > **Devices**

The Devices window is available in MTS TestSuite applications connected to a FlexTest controller. The Devices window allows an operator to review current sensor assignments and change them to compatible options as necessary.

Note: For systems running MP version 2.6 and earlier, and Series 793 version 5.7 and earlier, this functionality is available in the Station Manager application > Station Setup window > **Sensor** tab.

Clicking the **Details** button will show information about the selected device.

You can assign sensors resources:

- While the MTS TestSuite application is connected to the controller, and
- When changing between previously set-up devices.

Assign Sensor Calibration Files for New Hardware

To assign sensor resources:

1. Click the **Controller** menu > **Devices**.
2. Select the device that you want to change, and click **Assign**.
3. Select the new .scf file from the drop-down menu and click **Assign**.
4. Click **OK** to dismiss the Devices window after all changes are made.

External Devices

Access

Controller menu > **External Devices**

The External Devices window allows you to add and configure external devices as controller resources. After the resources are configured, they can be used in tests.

Important: Many of the configuration settings are device-dependent. For more information, see the documentation provided with the device.

Note: The External Devices procedures only apply to MTS Criterion and MTS Insight systems.

Set Up an External Device

Follow this checklist to configure an external device for use in the TWE, TWX, and TWS applications.

1. Select the Device Type you need to configure.
2. Complete the configuration settings for the device.
3. Add the External Device as a Resource in a test.
4. Map External Device commands to Controller Resources.
5. Add an External Device activity to the Test Procedure.

   Refer to the topics in this section for detailed reference information and step-by-step procedures.

Select an External Device Type

The Select Device Type window allows you to select the type of device that is connected to your COM channel or an Ethernet connection. The TW application automatically configures many of the configuration settings with default values; however, you may need to configure additional settings to install your external device.

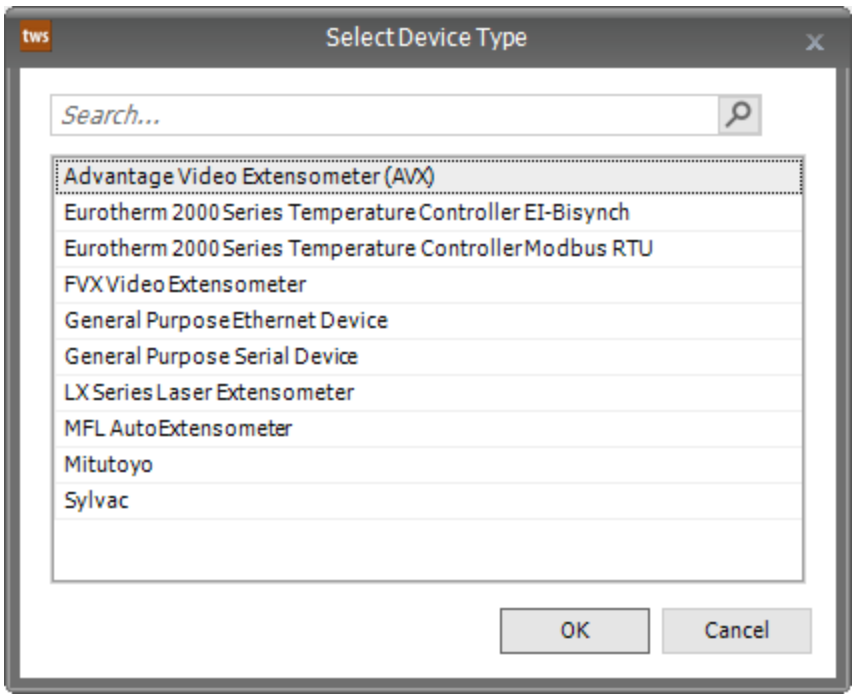
The TW application provides default configurations for several external devices. If your device is not listed, you can use either the General Purpose Ethernet Device or the General Purpose Serial Device option as a starting point.

Access

Controller menu > **External Devices**

1. In the Device title bar, click the plus (**+**) button.

Select Device Type Window



Select from the following device types:

- **Advantage Video Extensometer (AVX)**—Communicates with an MTS AVX extensometer.
- **Eurotherm 2000 Series Temperature Controller (EI-Bisynch or Modbus RTU protocols)**—Communicates with a 2000 series thermal controller made by Eurotherm. The device can read data or set the temperature setpoint.
- **FVX Video Extensometer**
- **General Purpose Ethernet Device**—Use to configure an external device that communicates over an Ethernet connection.
- **General Purpose Serial Device**—Use to construct a serial device configuration to communicate with any serial device. Select this option if you need to set up a device that is not shown in the list.
- **LX Series Laser Extensometer**—Communicates with an MTS LX Series laser extensometer. This device is designed to send commands such as turn the laser on or off to the laser extensometer.

- **MFL AutoExtensometer**
- **Mitutoyo**
- **Sylvac**

Create a Controller Resource for an External Device

Note: The following procedure applies to MTS Criterion and MTS Insight systems only.

1. Create the controller resource for the external device.
    A. From the **Controller** menu, click **External Devices**. The External Device window is displayed.
    B. In the External Devices window, click the plus (+) button. The Select Device Type window is displayed.
    C. In the Select Device Type window, click a device type from the list.

        Note: If the device you want to add is not listed, click the **General Purpose Ethernet Device Type** or **General Purpose Serial Device Type** as appropriate.

        Once added, the new external device appears in the **Device** list of the External Devices window.
2. From the **Devices** list, select the device that you want to configure.
3. Use the tabs in the External Devices window to configure all the settings required for communications with the external device.

    Note: After you create the External Device resource, you must use the **Resources** tab to add the device to your test and define the device commands.

External Device Configuration Settings

Many of these settings are device-specific. Refer to the device manual for information on device settings.

External Device Configuration Settings

Tab	Settings
Device Configuration (Serial Devices)	**Device Name**—The Controller Name that appears in the Resources list. **Serial Port** - Select the COM channel for the serial port.
Device Configuration (Ethernet Devices)	**Device Name**—The Controller Name that appears in the Resources list. **IP Address**—Enter the unique IP address set up on the device. **Port**—Enter a port number to identify the device. If multiple Ethernet devices are used, each device must have a unique port number assignment.
Device Configuration (Serial and Ethernet Devices)	**Read Termination String** (ASCII only)—Enter the string that appears at the end of each message from the external device. For more information, see "ASCII Control Codes for External Devices" on page 104. **Write Termination String** (ASCII only)—Enter the string that will appear at the end of each message sent to the external device. For more information, see "ASCII Control Codes for External Devices" on page 104. **Read Timeout**—Enter the time the TW application waits for a response after sending a read command to the external device before a timeout occurs. **Write Timeout**—Enter the time the TW application waits for a response after writing a command to the external device before a timeout occurs. **Error String**—Enter the error string returned by the device. **Protocol**—Select the protocol used to communicate with the device.  **Note:** The ASCII protocol does not support device addresses. • ASCII • **Eurotherm EI-Bisynch** - For use with Eurotherm 2000 Series Controllers that use the EI-Bisynch protocol. • **Modbus RTU** (remote terminal unit) - For use with devices that support this protocol. **Device Address** - Enter the device address. This setting is used by the TW application to identify the device. If multiple devices share the same COM port, set the **Device Address** to None and manually enter the device address in device-specific commands that you create in the **Command Settings** tab.
Port Settings (Serial Device only)	If necessary, adjust the default serial port settings for the external device: • Bits Per Second • Data Bits

Tab	Settings
	• Parity • Stop Bits • Flow Control • Restore Defaults - Click to restore default settings.
Command Settings	Use the **Command Settings** tab to add the commands that the TW application uses to communicate with the external device. **Command Name**—Enter the name of the command. For example, the LX Series Laser Extensometer uses the **Target Distance** command name for the **Tn** command. **Command**—Enter the command required by the device. For example, the LX Series Laser Extensometer uses a L0 command for "Laser ON" and R for "Read Data." **Note:** Enter **%s** (lower case) in the **Command** box if the output command will be determined by a variable defined in an External Device test activity. **Send Byte Data**— Select if the command is sent to the device as a byte. **Wait for Acknowledgement**—Select to have the TW application wait for a response after it sends information to the external device. **Supports Return Value**—Select if the external device returns an input value. **Important:** This check box must be selected if the input value will be written to a variable. **Value Type**—Select a value type, such as **Floating Point**. **Offset**—Select to enable an offset on the return value. In the field, enter the offset to be applied. **Regular Expression**—Enter the string that defines how to filter the string (return value) returned by the external device. **Note:** Click the ? button to display the Regular Expression Tool that allows you to check the expression's effect on various strings. For more information on how to use this utility, see "Regular Expression Tool" on page 98. **Multiplier**—Use the multiplier value to scale the return value sent by the external device into system units.
Signal Settings	**Can Be Streamed** check box—Select the check box if the device provides a signal stream that may be used in a test. Deselect the check box for devices that do not

Tab	Settings
	provide a signal stream. **Internal Name**—Enter the internal name of the controller resource. **Display Name**—Enter the display name (the controller name displayed in the Resources list). **Dimension**—Select the dimension for the external device signal. **Unit**—Select the unit for the dimension. **Query Command**—Select the command (defined in the **Command Settings** tab) that the TW application sends to the external device to start streaming data. **Time Interval**—Defines how often the TW applications sends the Query Command to the device to request information.
Device Verification	**Command**—Select the command (defined in the **Command Settings** tab) to send for verifying the device is communicating with your PC. **Send Command**—Click the **Send Command** button to verify the device is responsive. **Response**—Displays the response received from sending the command.

Add a Command to an External Device

To add a command to an external device:

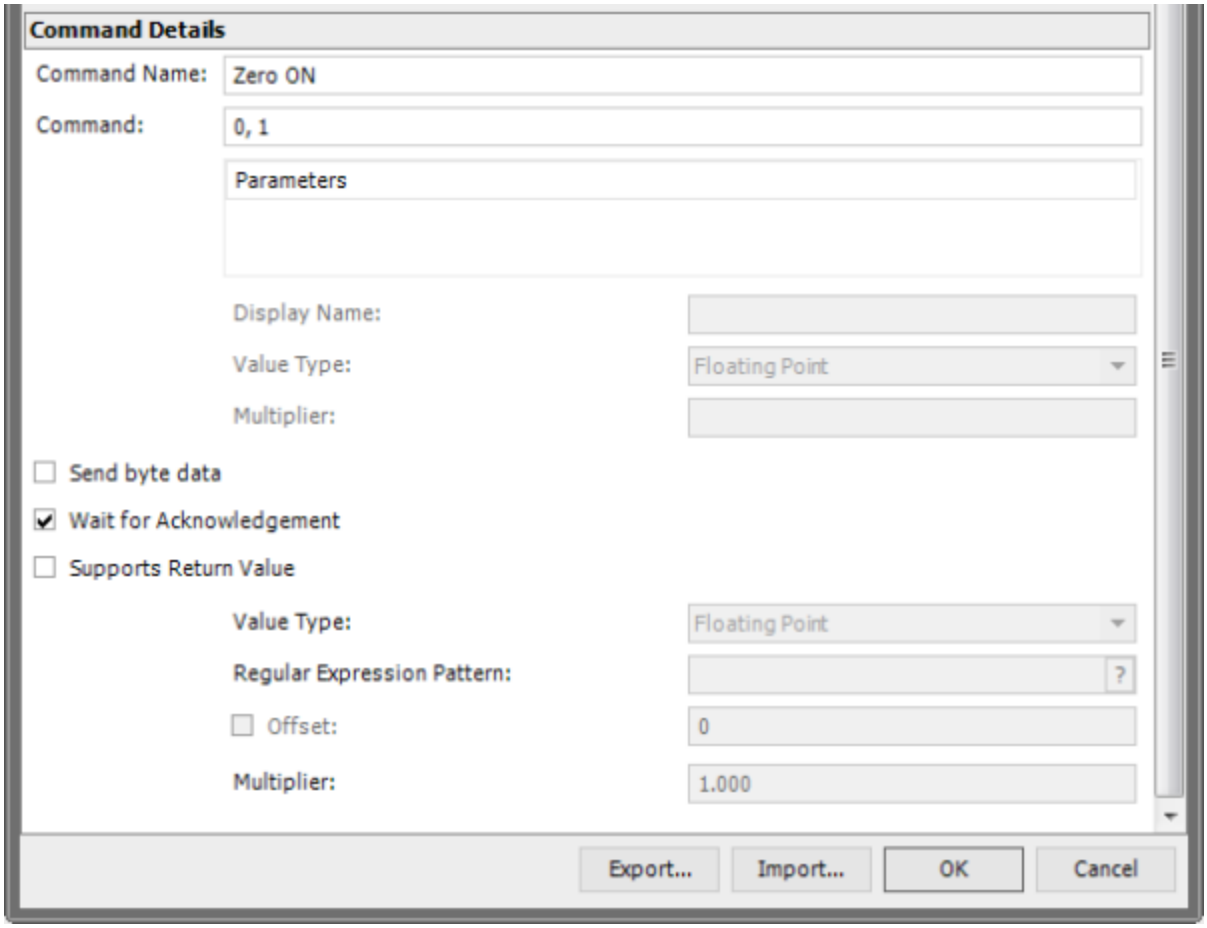
1. From the **Controller** menu, click **External Devices**. The External Device window is displayed.
2. In the **Device** list, select the device for which you want to add a command.
3. Click the **Command Settings** tab.
4. Click the plus (+) button.
5. In the Command Details panel, enter the **Command Name** and **Command**.
6. Make other selections as appropriate for the command and device. For more information, see "External Devices" on page 93.

Regular Expression Tool

To use the regular expression tool:

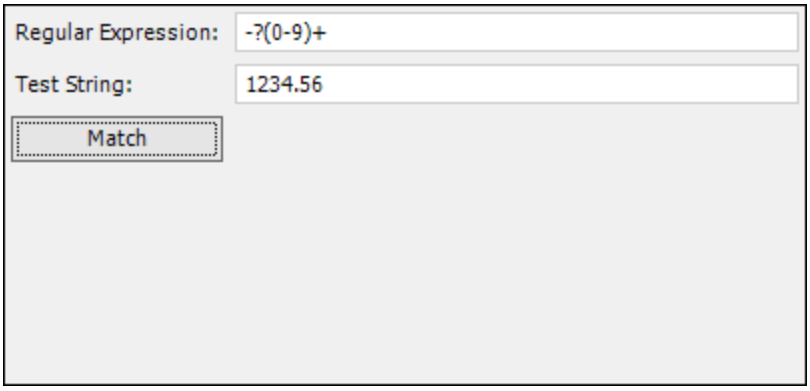
1. In the External Devices window, click the **Command Settings** tab.
2. Select a command that you want to edit. If necessary, click **Add** to add a new Read Data command.
3. In the Command Details panel, select the **Supports Return Value** check box. The **Regular Expression Pattern** field becomes available.

Command Details Window



4. Use the Regular Expression Tool to verify the expression.
    A. Click the Question Mark (**?**) button next to the **Regular Expression** text box to display a pop-up window.

Regular Expression Tool



    B. In the **Regular Expression** text box, enter the expression.

C. In the **Test String** text box, enter an example of the string returned by the external device.

D. Click **Match** and evaluate the filtered result that appears.

E. Change the expression as required until you achieve the desired result.

Device Address Considerations

The configuration for various devices affects the address and the command settings for these devices.

Multiple devices connected to a single COM port

Some device configurations, such as the MTS Model 409.83 Temperature Controller, include multiple devices that connect to a single COM port. The collection of devices is given a device name and the addition of device addresses to the commands allows the application to communicate with a specific device.

1. On the **Device Configuration** tab, set the **Device Address** setting to None.
2. On the **Command Settings** tab, type the device address in front of each command setting.

   For example, if the read-temperature command for a particular controller is PV and there are three temperature controllers whose device addresses are 1, 2, and 3, you could add three commands (1PV, 2PV, 3PV) and give the commands descriptive names such as Read Temperature Device 1, Read Temperature Device 2, and Read Temperature Device 3.
3. When the device is used in an External Device activity, you can select the command for a specific device.

Single devices connected to a COM port

For devices that are connected to a dedicated COM port:

1. On the **Device Configuration** tab, set the **Device Address** setting to the device's address number.
2. On the **Command Settings** tab, enter the desired commands.

   When the device is used in an External Device activity, you can select the command and the application automatically adds the device address to the command.

Add an External Device Resource to a Test

Use this procedure to create a test resource from an external device controller resource.

Method 1

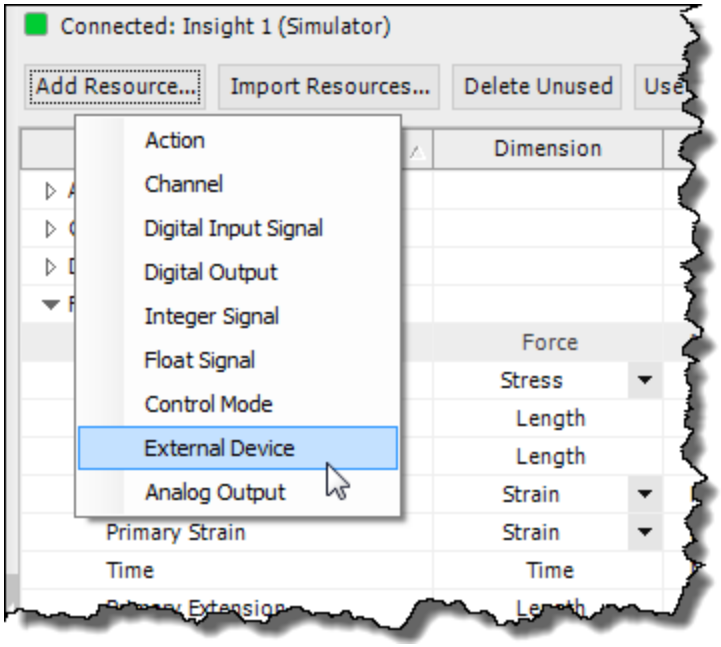
1. Click the **Test Definition** tab and then the **Resources**node.
2. Click **Import Resources** to add the external device to the resource list.

   All the commands associated with the device appear in the properties table.

Method 2

1. Add the external device to the **Resources** list for the test.
    A. Click the **Test Definition** tab and then the **Resources** node.
    B. Click **Add Resource** and choose **External Device**.

Choose External Device



2. In the Add External Device Resource window, define the test resource:

Add External Device Resource Window

3. Select the external device from the **Controller Resource** list.
4. Configure the controller Resource:
    - **Display Name** - Enter the display name for the controller resource.
    - **Internal Name** - Enter the internal name for the controller resource.
    - **Dimension** - Shows the dimension set in the External Devices window.

- **Show Controller Internal Names** - Select this check box if you want the internal name (defined above) displayed in the **Resources** list.

5. Click **OK**. The device now appears in the **Resources** list for the test.
6. To map the test resource commands to controller-resource commands associated with this device, you must add each external-device command by clicking the + button in the device properties table.

Map External Device Commands to Controller Resources

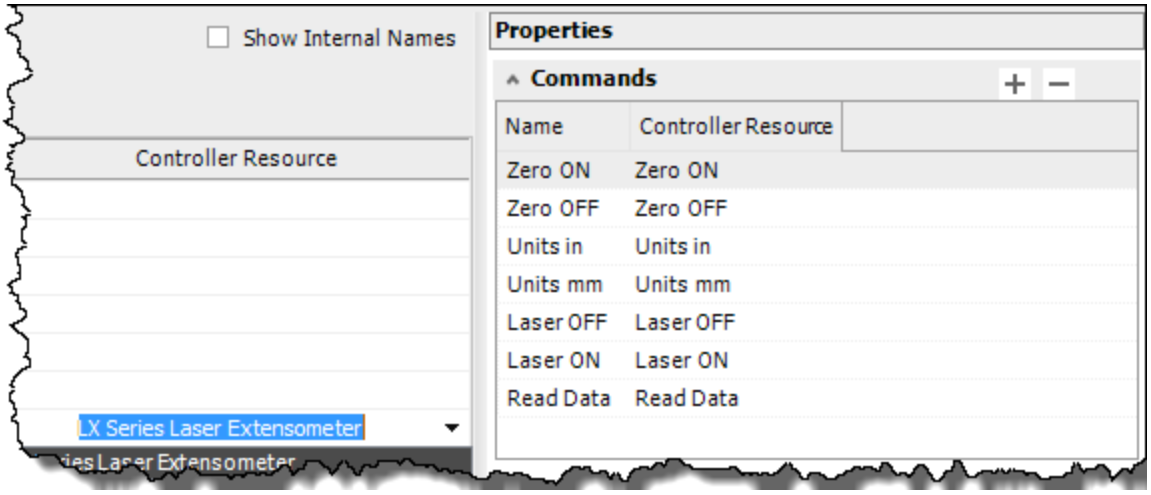
You can use the external device properties in the **Resources** node to map test resource commands to controller resource commands. This is helpful if you create a test offline and your test resource naming does not match the controller resource naming.

1. Click the **Test Definition** tab and then click the **Resources** node.
2. In the **Resources** list, select the external device that contains the command settings that you want to map.

   The properties table for the device has two columns:

   - The Name column lists all of the test resource commands.
   - The Controller Resource column shows the command settings for the selected device. Each controller resource includes a drop-down list that allow you to map it to a test resource.

Properties Window



Note: If you clicked **Import Resources** to add the external device to the resource list, all the commands associated with the device appear in the properties table.

Note: If you clicked **Add Resource** to add the external device to the resource list, you must add each command by clicking the + button in the device properties table.

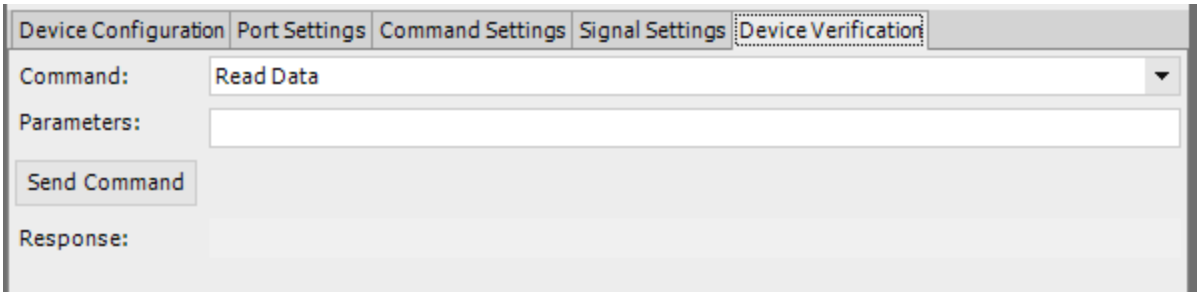
3. Use the controller resource lists to map test resources to controller resources.
    A. In the Properties table, select a test resource.
    B. Select a controller resource from the drop-down list.
4. Optional - to create a new test-resource command:
    A. On the properties table for the external device, click the plus (+) button.
    B. Select the new test resource (added to the bottom of the list).
    C. Select a controller resource from the drop-down list.

Device Verification for External Devices

The Device Verification tab allows you to send a command to an external device and observe the response. Use this feature to verify the operation of an external device or to send commands as part of a calibration procedure.

1. On the **Controller** menu, click **External Devices**.
2. On the **Devices** list, click the device that you want to verify.
3. In the External Devices window, click the **Device Verification** tab.

Device Verification Tab



4. From the Command list, select the command that you want to send to the device. The Command list contains all the commands defined in the **Command Settings** tab.
5. Click **Send Command**. The response returned from the external device is displayed in the **Response** field.

Export and Import External Device Files

You can export the configurations of your external devices for import into another TW workstation. The exported device file has the .tsExternalDevice file extension. Device files are exported to the MTS TestSuite > External Files directory.

Export an External Device File

1. Click **Controller > External Devices**. The External Devices window is displayed.
2. In the **Device** list, select the device you want to export.
3. Click **Export**. The Export the External Device File window is displayed.

4. Enter a **File name** and click **Save**.
5. Click **OK**.

Import an External Device File

1. Click **Controller > External Devices**. The External Devices window is displayed.
2. Click **Import**. The Select External Device File to Import window is displayed.
3. Select the file to import and click **Open**.
4. Click **OK**.

Remove an External Device Resource

Note: If the external device is in use as a controller resource within a test, removing the resource results in a test resource validation error.

1. Click **Controller > External Devices**. The External Devices window is displayed.
2. In the **Devices** list, select the device you want to remove and click the minus (**-**) icon. The device is removed from the list.

ASCII Control Codes for External Devices

The following are the control characters used by serial devices. The most commonly used are the ^M (carriage return – CR) and ^J (line feed – LF).

Control Characters Used By Serial Devices

Code	Meaning
^@	NULL
^A	SOH
^B	STX
^C	ETX
^D	EOI
^E	ENQ
^F	ACK
^G	BEL
^H	BS (Backspace)
^I	HI
^J	LF (Linefeed)
^K	VI

Code	Meaning
^L	FF
^M	CR (Carriage Return)
^N	SO
^Q	CS1
^R	DC2
^S	DC3
^T	DC4
^U	NAK
^Y	EM
^Z	STB
^[	ESC (Escape)
^	FS
^]	GS
^^	RS
^_	US

Analog Outputs

You can use analog outputs to control various analog devices including data recorders, external data subsystems, oscilloscopes, fans, or other devices. When you connect an analog device to the MTS Insight controller, the controller will send a specified voltage (between 0 and 10 V) to the device based upon signals from resources in your test, such as force, stress, or strain.

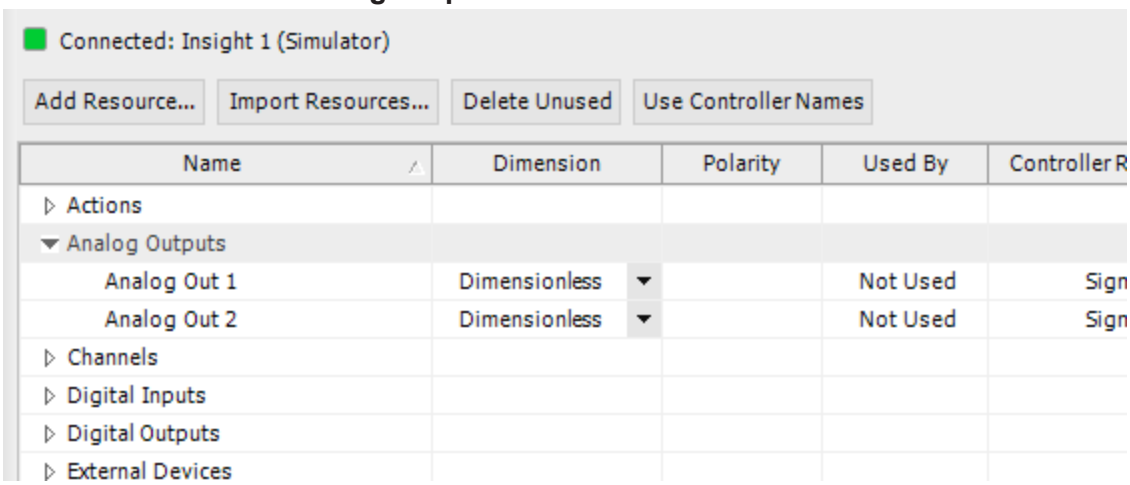
Using calculations to derive an analog signal

You can use calculations to derive the analog output voltage based on one or more resources in your test. For example, consider a scenario in which you are using two extensometers to test dog bone specimens. In this case, you might want the analog output to take the measurements of both extensometers into account when deriving the analog output signal. To do this, you would create a calculation that adds the extensometer signals together and then divides the result by two. This calculation causes the controller to produce an analog output signal based upon the *average* of the two extensometer signals.

Importing analog output resources to a test

MTS Insight controllers allow for up to two analog outputs. Before you can enable an analog output, you must import the analog output as a resource in your test. After you import the analog outputs, they appear on the **Resources** node of the **Test Definition** tab:

Analog Outputs on the Resources Node



Configuring Analog Outputs

After you add an analog output, you must select a **Dimension**, **Controller Resource**, and calculation for the analog output. Additionally, you will need to configure a TEDS file, which is used to translate signals from resources into voltages sent to the analog device.

1. On the **Resources** node of the **Test Definition** tab, select a **Dimension** (such as **Force**, **Stress**, or **Strain**). This dimension qualifies the values of the analog output when used by the analog calculation and the TEDS file.
2. If you modified the name of the controller resource, select the modified controller resource name in **Controller Resource**. By default, the controller resources used by the two analog outputs are set to **Analog Out 1** and **Analog Out 2**.
3. Create a calculation that will be used to derive the analog output signal:
    A. Select **Analog Output 1** and click the **…** button next to the **Calculation** field in the **Properties** panel. The Calculation Editor window appears.
    B. Enter a calculation for Analog Output 1. For example, if you want the analog output signal to rely on the stress signal, you would simply add the Stress signal. However, you could create a more complex calculation, if necessary. For example, if you create a calculation that adds the signals from two attached extensometers together and divides them by two, the calculation will produce the average of the two extensometer signals.
4. Click **Controller** > **TEDS Devices**. The TEDS Devices window appears.
5. Select the **Analog Out 1** device.
6. If the **Analog Out 1** device already has a **TEDS Source** listed, click **Details** to edit the existing TEDS file. Otherwise, click **New** to create a new TEDS file.
7. The TEDS Detail (Analog Output 1) window appears. For detailed information about configuring TEDS devices, see "TEDS Devices" on page 89.

When configuring the TEDS file for an analog output, the **Physical Measurand** must match the **Dimension** that you selected for **Analog Output 1** on the **Resources** subtab. When configuring the minimum and maximum measurand values on the TEDS Details (Analog Output 1) window, keep in mind that these values correspond to the **Minimum Electrical Value** and **Maximum Electrical Value**.

For example, consider a scenario in which the minimum force that will be applied during your test is 0 N, and the maximum force will be 100 N. In this case, you would configure the TEDS file as follows:

Configuring a TEDS File for an Analog Output

TEDS File Property	Value/Unit
Minimum Force/Weight	0 N
Maximum Force/Weight	100 N
Minimum Electrical Value	0 V
Maximum Electrical Value	10 V

As a result, the controller will produce 0 V when 0 N of force is applied. When the full 100 N of force is applied, the controller will produce a full 10 V. When only 50 N of force is applied, the controller will produce 5V. If the analog device connected to the controller is an oscilloscope, for example, the oscilloscope will show readings based upon these voltages.

Managing Tests

Tests

Tests Overview

Note: You must have the Administrator or Engineer privilege on a TWE system to create or modify a test procedure.

A test is a file that contains test run data (if test runs were performed), and includes a copy of the template. You can design your own test or create one from a template. Each test includes:

- A test definition
- Test runs

A test run is a single instance of running the test procedure. The data is stored in a test file with a copy of the test procedure or template.

Test definition

The following tabbed pages are available for creating or modifying tests.

- **Select**—Open new, existing, or recently run tests.
- **Monitor**—Observe Pre-test input values as the test progresses.
- **Define > Test flow > Pre-test/Pre-test run Inputs**—View the properties of existing variables and to create new variables.
- **Define > Test flow > Test run > Procedure**—Construct the sequence of activities (command, test control, and DAQ) that make up a test.

  Note: Variables are an important part of a test, because they represent test values or other information generated or accumulated during a test run. Variables are defined for and used with a particular test. Each new test run stores a copy of the variables as defined when the test run is created.
- **Define > Test flow > Report templates**—Create templates that are used to generate test reports. You can also import report templates previously exported from another test. Report templates are created in Microsoft Excel.
- **Define > Test flow > Report templates**—Configure, and manage the hardware and virtual resources that are available from the assigned test station. You can import resources previously exported from another test.

Error list window

The Error List window shows messages about possible problems with the configuration or procedure that is being constructed.

Test run reports

The application can create reports for each test run. The report content and appearance are specified by the templates you create with the MTS TestSuite Reporter Add-in for Microsoft Excel.

Test Procedure Overview

A test procedure includes the system actions you want the application to perform during a test. The test procedure is located on the **Define** > **Test flow** > **Test run** > **Procedure node**.

Each test has a single procedure that is stored with the test definition. The procedure is included when you create test templates. You can define the procedure while connected to hardware resources or while working offline with the resources stored with the test from an earlier connection.

Procedure design

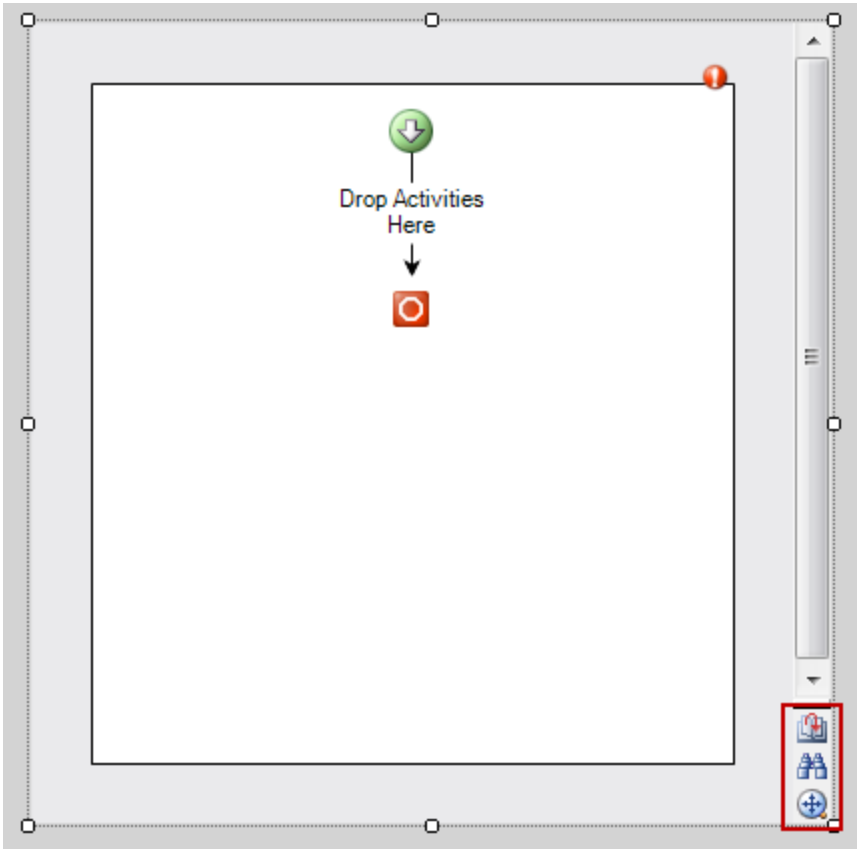
Whether you work with the Procedure workspace that shows the current test procedure as a flowchart or the Outline view that shows the procedure as a hierarchy in a tree view, the graphical display simplifies creating tests.

The Toolbox panel contains activity icons that you drag into the Test Editor and name to build a procedure. The Properties panel shows the properties that you set for the selected activity. You can hide or resize the panels to increase the amount of visible workspace.

Navigation

The flowchart view includes Pan and Zoom controls to help you move around complex text procedures. The Outline view provides the Find option for locating activities by name.

Test Procedure Pan and Zoom Controls



Procedure activities

Activities are the building blocks for test procedures. To add an activity to a procedure, click the activity in the toolbox and drag it to a location in the procedure marked with a plus sign. If it is the first activity being added, drag the icon to the **Drop Activities Here** area.

The types of procedure activities include:

- Operator messages
- Conditional if/else processing
- Command generation
- Data acquisition
- Auto offset
- External device control

Activity properties

Most activities require you to specify properties to fully define how the activity should perform. When you add an activity that has parameters you are required to specify, the activity in the procedure includes an error icon. The **Properties** tab identifies the parameters, and the Error list describes the problem. You must correct all errors in the test before you can initialize and run the test.

All activities have a **Display Name** that defaults to the activity name. You can enter a name to display for an activity. All activities provide an optional **Description** field to document the procedure design. You can also set the visibility of the activity in the Progress Table.

All activities have the **Enabled** check box selected by default. You can clear the **Enabled** check box to disable an activity.

Creating a Test

From the **File** menu, you can create a new test as follows:

- **New Test From Current Test**: Creates a new test from the test that is currently open. Any test runs from the test that is currently open will not be transferred to the new test.
- **New Test From Template**: Creates a container file called a test that is used to collect data and contains a copy of the template. Each time you run the template on a physical specimen, the application creates a test run. The test run is a data set that populates the test.

  Note: Only templates for which you are licensed appear in the **New Test From Template** list.
- **New Test From Existing Test**: Creates a new test that includes a copy of the test without test run data.
- **New Test From File**: Creates a new test from an XML file; typically used in automation.
- **New Test from TestWorks 4 Import File**: Creates a test from a legacy TestWorks 4 import file.

Saving Test Changes

The application saves test changes in two ways:

- User-prompted saves
- Automatic saves (background saves)

User-prompted saves

When you create a test, that is, while you are configuring a test on the **Test Definition** tab, only user prompted saves are performed. This allows you to make changes to your test design and then back out of those changes as long as you do not run the test. To save changes to a test before you run it, you must select **Save** from the **File** menu or choose to save changes when prompted upon closing the test or exiting the application.

Note: You cannot save parts of a test. When you click **Yes**, the entire test is saved.

Automatic saves

When you run a test, the application automatically saves the test without a user prompt as follows:

- At the beginning of the test run
- Periodically during the test run
- When the test run completes
- When the test run is interrupted (e.g., power loss to the test frame)
- When the test run is stopped (pressing the **Stop** button on the Test Controls panel)
- When the test is unloaded (pressing the arrow button to the right of the Test Controls panel)

Considerations for automatic saves

When you start a test run, the application automatically saves the current state of the test. If you want to modify a test without saving your changes, that is, experiment with the test, you should perform a **File > Save As** of your test, modify as desired, and then run the new test.

If you inadvertently run a modified test and want to back out of the changes you made to the test before you ran it (and the application automatically saved it), you must do so manually.

Saving a Test

To save a test:

1. To save a test without closing it, select **File > Save**, or click the **Save** icon.
2. To close the test, select **File > Close Test**.
3. You are prompted to save the test if you have made changes to the test.

Deleting a Test

Important: When you delete a test, all test definitions, test runs, test data, and reports are also deleted.

Using MTS TestSuite software

From the **File** menu, select **Delete Test**.

Using Windows Explorer

If you want to delete many tests at once, or if you need to remove tests that are corrupt and cannot be opened, use Microsoft Windows Explorer.

Test files have a .test extension, and reside in Project files, which have .project extension.

By default, Project files are located at: **C:\MTS TestSuite\Projects**.

Test Runs

Test Runs Overview

A test run is the record of a test performed on a single, selected specimen.

Test runs are stored in the test and include:

- A copy of the test definition, including the procedure, at the time the test run is created
- A copy of the name of the selected specimen and its values at the time the test run is created
- A copy of the variable definitions at the time the test run is created
- Variable values during the test run
- Runtime progress (state) information
- Results data

Note: Information that is not saved with the test run includes runtime display monitor information and the test resource mapping. The test run does not reflect changes made after the run concludes.

Test run definition

To create a test run, you must have a valid test definition that includes:

- One or more specimen definitions
- A test procedure
- Properly-mapped test station resources
- Any required variables
- Optional - Add functionality to generate reports for individual test runs or for all test runs.

Before you can create a test run, you must resolve any validation errors in the test definition. Click the test name in the test hierarchy to show a table with a summary of the validation errors for the test definition.

Test results

Reports

To generate reports for existing test run results, you must have the MTS TestSuite Reporter Add-In license.

Multiple instances of test runs

To avoid possible resource conflicts, do not open multiple instances of the same test run.

Test Run State Colors on the Review Tab

The color of a test run entry on the **Review** tab indicates its run state.

Test Run State Colors

Color	Description
Black	The test run completed successfully.
Dark Blue	The test run initialized successfully but has not run.
Red	The test run stopped.
Orange	The run is running, on hold, or an error occurred.

If a test run name is orange, data may not be available. If an error occurs while the test runs, no data is viewable. If an interruption occurs, some data may be available.

Test Run from an XML File

MTS TestSuite applications can read a user-supplied XML file that defines a number of test runs that can be run with minimal operator involvement.

The XML file will:

- Create a new test from the template defined in the XML file.

  The name of the test is defined in the XML file. Typically, each XML file includes a unique test name.
- Create a test run for each specimen defined in the XML file.
- Assign values to variables in the test run.
- Assign a name and geometry type to the specimen. The specimen name will also be used for the test-run name.

  Note: A sample XML file that can be used as a test-from-file template is located in the C:\MTS TestSuite\External Files directory.

Create a Test Run from an XML File

Note: Test runs imported from an XML file are always performed in the order defined by the XML file.

1. Start the MTS TestSuite application.
2. Optional - Use the Configuration settings to select a project to store the test.

3. On the **File** menu, click **New > Test from File**, or on the Select page, click **New Test** and then double-click the **New Test from File** icon.
4. In the file open window, select the XML file that defines the test runs that you want to run and click **Open**.

   The MTS TestSuite application reads the XML file and creates a list of test runs that appear in the **Review** tab. The test runs appear in the order defined in the XML file.
5. Click the **Review** tab to track the test run progress.

   In the **Review** tab, the text for each test run that has not been run appears in blue italic.
6. Click the **Run** button.

   The test run is executed and the test run results appear in the **Review** tab. As each test run is performed, the text changes from blue italic to regular (non-italic) black text to indicate that the test run is complete.
7. Click the **Run** button to run the next test run. Repeat until all the test runs are complete.

Pre-Allocating Multiple Test Runs

If you are testing multiple specimens, you can improve the speed and reliability of your testing by pre-allocating test runs for each specimen. When you pre-allocate test runs, you can specify unique variable values for each specimen prior to running tests.

Both test designers and test operators can pre-allocate specimens. The strategy you use to implement pre-allocation depends upon your lab's workflow. The following two examples illustrate scenarios in which either the test designer or operator would use the pre-allocate feature:

- When designing a test, the test designer knows that 10 specimens will be tested. The test designer also knows the properties of each specimen. So, the test designer pre-allocates 10 test runs and enters the variable values for each specimen. The test designer then changes the names of the test runs to Specimen 1, Specimen 2, Specimen 3, and so on for each specimen. Finally, the test designer marks each specimen with the corresponding number. When the operator receives the test and batch of specimens marked 1-10, he or she selects the appropriate specimen for each pre-allocated test run.
- The operator often receives batches of specimens that have varying properties. Because of this, the lab has decided that it is not efficient for the test designer to spend time designing a new test for each batch of specimens. So, when the operator receives a new batch of specimens, her or she pre-allocates test runs for each specimen based on the specimen properties that were recorded on a data sheet or USB flash drive that was sent with the batch of specimens. Additionally, because it is possible to open the **Review** tab while a test is in progress, the operator runs the first pre-allocated test run and then works on configuring the remaining test runs while the first test run is in progress. To pre-allocate multiple test runs, perform the following:

1. On the **Review** tab, click the **Actions** button.
2. Select **Preallocate**.
3. In the window that appears, enter the number of test runs you want to pre-allocate.

4. Click **OK**. The top-most table on the **Review** tab is populated with the pre-allocated test runs.

   Note: When you pre-allocate test runs, each test run uses the test's default variable values. When you select a test run, you can edit the variable values for each test run in the lower table of the **Review** tab.

   Note: Only variables with the **Preallocate** option selected (in the variable properties) are shown in the list of preallocated variables.

When you click the green Run button, the top-most test run in the list begins. When it completes, you can click the green Run button again to begin the second test run in the list, and so on until you reach the last test run at the bottom of the list. The names of test runs that have not yet been run are italicized. To run a specific test run as opposed to running tests in the list from top-to-bottom, right-click a test run and select **Run Test**.

Note: If the test definition is changed during test design (for example, variables, resources, and workflows are edited), any changes made will not be applied to existing pre-allocated test runs. In this case, the test designer must delete the existing pre-allocated test runs and pre-allocate new test runs.

Templates

Templates Overview

Access

C:\MTS TestSuite\Templates

Templates eliminate the requirement to re-create existing information and provide an easy way to run standard tests. Test templates can come from one of several sources:

- An existing test—You open a copy of an existing test and assign it a default name (the original test is not changed). The new test does not contain test runs or analysis runs from the source test.
- A test that is saved as a template—You can save a test as a test template **(File > Save As > Template**). With the exception of test and analysis runs, specimen definitions, and completed reports, the template contains all other test information.
- A template supplied by MTS—MTS offers a variety of templates designed to comply with test method standards (such as ASTM). MTS templates provide all the components you need to run a test, analyze the test data, and create reports of the results.

Test template content

A template can include all or part of the basic test definition information for one test:

- Procedure
- Test run charts or displays
- Variables
- Resources

Note: Although a template can include test resources, if the template is designed for a particular controller configuration, the resources may not map to (match) the test station resources in your system.

Template locations

Tests and templates exist on disk as folders with the .Test folder name extension. They are typically located at a subdirectory of C:\MTSTestSuite\Projects and C:\MTSTestSuite\Templates, respectively. Report templates are located in the C:\MTSTestSuite\Report Templates folder.

Note: Do not rename, move, or change the contents of the Projects or Templates folders outside of the MTS TestSuite applications. To rename an open test, use the Test Information window in any application.

About saving templates

When you save a template, MTS TestSuite compresses the template and assigns an extension of .MPTemplate for MP templates and .TWTemplate for TW templates.

Templates can be managed with the operating system.

Note: Template files are not fully self-contained; they contain references to other files (such as Report Templates).

Create a Template

To create a template:

1. Open the test that you want to save as a template.
2. Click **File > Save As > Template**.
3. In the Save As Template window, type a name for the new template.
4. Click **Save**.

   The current test is saved as a template. All variables and their default values are saved with the template. The template does not include the test runs or specimens from the source project.

Delete a Template

Access

C:\MTS TestSuite\Templates

You must use Microsoft Windows Explorer to delete templates. Delete it as you would any other file. Template files have either a .MPTemplate or a TWTemplate extension depending on which application the template is designed for.

Project

Projects Overview

A project is a collection of settings related to tests. When you open a test, it opens in the context of its parent project. Files associated with tests, such as external files and reports, are linked to tests with project settings.

You can use project settings to create logical groupings of tests that contain references to the same set of external files. You can also open an individual test in a different project, which allows you change the files referenced by the test without modifying the test itself (files are specified relative to the directories in the project settings).

Assign Project Names

Access

Preferences menu > **Configuration** > **Default Names** tab

Use the **Default Names** tab to specify default names for newly created:

- Projects
- Tests
- Test Runs

The names you specify are assigned to the items when you first create them. You can change the name at any time.

File Locations

Note: Do not rename, move, or change the contents of the Projects, Tests, Templates, or Report Templates folders.

Projects

Projects exist on disk as special file folders with a .Project file extension. Projects are typically located in the installation directory: C:\MTS TestSuite\Projects.

Tests

Tests exist on disk as special file folders with a .Test file extension. They are typically located in a directory under a project.

Templates

Test Templates exist on disk as special file folders with either a .MPTemplate or TWTemplate file extension. They are typically located in the installation directory: C:\MTS TestSuite\Templates.

Report Templates

Report Templates exist on disk as Microsoft Excel Templates (.xltx) files. They are typically located in the installation directory: C:\MTS TestSuite\Report Templates.

Import Projects

You can import projects stored in a file. The projects have a .tsproj file name extension. When you select a project to import, the entire project is imported. If the project you have selected is from an earlier version, the Conversion Wizard launches to convert it automatically to the version on disk.

Export and Import

Test Information Export Overview

When you export a test, MTS TestSuite combines the current test and its associated files into one compressed (and self-contained) file and assigns an extension of “tsproj”.

Exported tests can be managed with the operating system. Unexported tests must be managed within the MTS TestSuite application.

You can selectively export test and test run information. When you initiate an export, you can choose the item or items from the test hierarchy to export. With tests, for example, you can selectively export only the information that you want to import into new or existing tests.

Export options

You can use options from the **File** menu or right-click entries in the test hierarchy to export:

- A test
- One or more test runs
- Raw data from a test run

Raw test data export

You can export the raw data for a test run so that you can analyze it with external software applications. By default, raw data is exported to one or more tab-delimited text files that can be opened with most external analytical or statistical applications. Each **Data Acquisition** activity in the source test procedure produces a separate data export file.

Test Information Import Overview

You can import test and test run information from previously exported test and test run files. Export files have a .tsproj file name extension. When you import an entire test, any test that is currently open closes and a new test is created from the imported test file. When you import test runs, you can select the test runs you want to import into the open test.

Import options

You can import:

- A test
- Test runs
- Test resources

Test resources import

Click the Import Resources button on the **Define** > **Resources** node to import selected resources or all unused resources from the test station.

You can also import resources from another test. Depending on the origin of the information, the imported resources may not map to the station resources of the control system networked with your station. The application lists errors on the Error list and marks them in the work area when you select resources. You must correct all errors before you can load and run a test. When you use the **Import Resources** button, you can only import resources from the connected test station or another test. Three options become available:

- Import all Unused Controller Resources
- Import the selected controller resources
- Import resources from another test

Export a Test

To export a test:

1. Open the test that you want to export.
2. Select **File > Export > Test**.
3. In the Export Test window, click **Browse**.
4. In the Save As window:
    A. Locate the directory to which you wish to export the file.
    B. Enter a name.
    C. Click **Save**.
5. In the Export Test window:
    A. Select the desired specimen and test run content you wish to export with the test.
    B. Click **Save**.

Import a Test

To import a test:

1. Select **File** > **Import** > **Test**. The application automatically closes any open test and prompts you to save any changes.
2. In the Import Test window, click **Browse**.
3. In the browser window, locate and click the required test import (.tsproj) file, and then click **Open**.
4. Click **OK** to close the message window that indicates the test import was successful.

Exporting a Test Run

To export a test run:

1. Open the test with the test runs that you want to export.
2. Select **File** > **Export** > **Test Run**.
3. In the Export Test Run window, expand the export items hierarchy and select the check boxes for the test runs that you want to export.
4. Click **Browse** to open a browser window.
5. If you want to overwrite an existing test run export (.tsproj) file:
    A. Locate and click on the file name in the browser window.
    B. Click **Save**.
    C. Click **Yes** in the Save As window.
6. If you want to create a new test run export file, enter the new file name in the **File Name** box of the browser window, and then click **Save**.
7. Click **Save** in the Export Test Run window. The selected test runs from the source test are written to the test run export file.
8. Click **OK** in the confirmation window when the export is complete.

Import a Test Run

To import a test run:

1. Open the test.
2. Select **File > Import > Test Run** or right-click **Test Runs** for the required test, and then click **Import Test Run**. The test must be saved before you can import test runs. Click **Yes** to save the test.
3. In the Import Test Run window, click the **Browse** button.
4. In the browser window, locate and click on the required test run import (.tsproj) file, and then click **Open**.

    If the selected import file does not contain test runs, the Import Test Run window presents a message. If necessary, click the **Browse** button again and select a different import file.
5. In the Import Test Run window, select the target test from the **Test** list.
6. Expand the import items hierarchy and select the check boxes for the test runs you want to import and click **OK**. The selected test runs are added to the target test. They are marked as “Imported” in the **Test Runs** list.

Import Test Resources

The procedure applies only to the **Import Resources** button on the **Define** > **Resources** node.

You can only import test resources from the connected test station or another test in the same project. When you import an entire project, the test resources can be included for each test in the project. Depending on the origin of the information, the imported resources may not map to the station resources of the control system networked with your session or workstation.

1. Open the required test.
2. If you want to import resources from the test station, click **Controller > Connect**.
3. Click the **Define** tab, then click the **Resources** node.
4. Click the **Import Resources** button on the work area to show the **Options** menu.
5. To import resources from a connected station, select one of the following options: Click **Import all unused station resources** to import all the station resources that are not currently in use by the test. Click **Import selected station resources** to select which unused resources you want to add to the test.
6. If you clicked **Import selected station resources**:
    A. Expand the unused resources hierarchy and select the check boxes for the resources to import, or click **Select All** to select all unused resources.
    B. Select the **Display internal names** check box to show the internal names for the resources in the work area instead of their display names.
    C. Click **OK**.
    D. The table of test resources updates and a message window indicates that the import was successful. Click **OK**.
    E. If all the station resources are already in use by the test, a corresponding message is shown. Click **OK**.
7. To import resources from another test in the current project:
    A. Click **Import resources from another test** to overwrite the test resources with the imported resources.
    B. In the Import resources from another test window, select or browse to the source test, and then click **OK**.
    C. The table of test resources in the work area is updated and a message indicates that the import was successful. Click **OK**.

Defining a Test

Selecting Templates and Tests

About Templates

A template is a file that contains all of the MTS TestSuite TWS settings necessary to perform a specific test. You cannot edit templates directly. When you open a template, MTS TWS creates a test which contains an editable copy of the template.

If the template is user-defined, you can edit the copy of the template in the test and overwrite the original template by saving it with the same name.

MTS-defined templates are write-protected. If the template is MTS-defined, you can edit the copy of the template in the test, but you must save it with a unique name to create a new template.

About Tests and Test Runs

A test is a file that contains test run data (if test runs were performed), and includes a copy of the template. A test run is a single instance of running a template.

About Modifying Templates

You typically define tests to suit your specific needs by first locating a template that meets as many of your test requirements as possible. Then you open the template (which opens a copy of itself referred to as a test file), edit the test, and save the test as a template, using the same name as the original template.

Example: Suppose you want to modify template “EM Tension 2014”. To do this:

1. On the **Select** tab, navigate and open the “EM Tension 2014” template. When you do this, the application opens a test file, which is an editable copy of the template.
2. Using the functions in the **Define** tab, edit the test as desired.
3. Select **File** > **Save As** >**Template**, using “EM Tension 2014” as the file name.

This overwrites the existing template with the changes you made to the test file.

Defining Tests

Use the **Define** tab to edit templates to your specific test requirements. The controls on the **Define** tab allow you to:

- Enter custom test description and specimen information.
- Assign pre-test and pre-test run inputs and calculations.
- Select and position command and data acquisition activities in a test flow.
- Define limits to detect events during the test flow.
- Define Post-Test and Post-Test Run inputs and calculations.
- Configure the test run chart, results table, and test reports.
- View and optimize test and controller resources.

Note: Test designers typically use only a limited number of the controls on the **Define** tab to modify tests. Common modifications include changes to default values for test descriptions, specimen type and dimensions, test and break detection limits, and report specifications. Significant changes to the calculations or test flow of a standard template may compromise the industrial standard.

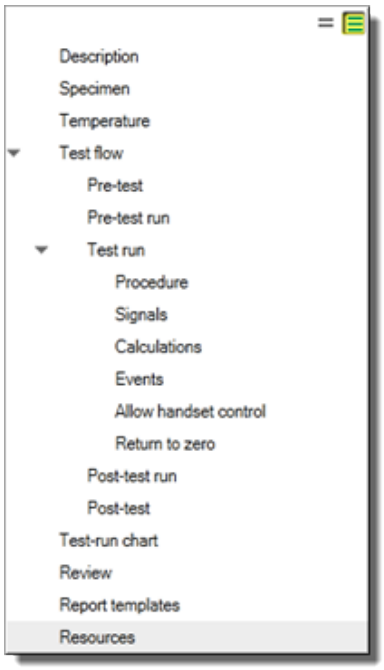
Tip: To acquire information from operators that may change between tests and test runs, test designers typically assign pre-test, pre-test run, post-test run, and post test inputs. Assigned inputs show default values at the appropriate time in the test for operators to verify and edit as required.

About the Test Flow

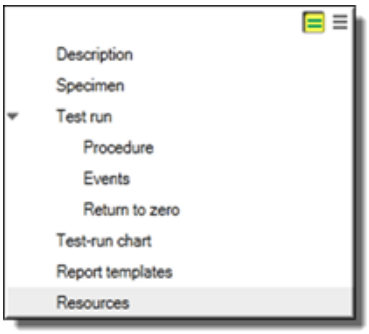
The test definition tree has two navigation modes. The two navigation modes are **Advanced Mode** and **Basic Mode**. **Advanced Mode** shows the entire test definition tree. **Basic Mode** shows a simplified version of the test definition tree. The active navigation mode is switched by the toggle buttons in the upper right-hand corner of the test definition tree pane.

Navigation mode toggle buttons

Advanced Mode



Basic Mode



The Test Flow is divided into Pre-Test, Pre-Test Run, Test Run, Post-Test Run, and Post-Test sections. You can assign inputs, add calculations, and configure actions for each section.

Pre-Test Section

The Pre-Test section is for activities you want the operator or application to perform before test runs begin, such as:

- Enter values for test inputs common to all test runs.
- Configure the load train (setting up fixtures, and so forth).
- Enter the test name, specimen geometry, lot number, batch name, and so forth.

The Pre-Test section runs only once per test, before the first test run.

Pre-Test Run Section

The Pre-Test Run section is for activities you want the operator or application to perform before each test run, such as:

- Enter values for nominal specimen dimensions.
- Calculate specimen area.

Test Run Section

The Test Run section is for activities you want the operator or application to perform during each test run, such as:

- Install the specimen.
- Observe the test as forces are applied to the specimen.
- Monitor limit detection levels.
- Respond to a message that indicates the frame will move to the return position.
- Remove the extensometer for specific types of extensometers and associated templates.

Post-Test Run Section

The Post-Test Run section is for activities you want the operator or application to perform after each test run, such as:

- Enter final values for specimen dimensions.
- Observe the test run information shown on the **Review** tab.
- Export data.
- Generate and print reports for the test run.

Post-Test Section

The Post-Test section is for activities you want the operator or application to perform when test runs are complete, such as:

- Generate and print reports for the test.
- Export data.

The Post-Test section runs only once per test, after all test runs are complete, or when the operator clicks the **Run Finish section** button on the **Review** tab.

Note: It is typical to design tests so that the Post-Test section runs when test runs are complete. However, running the Post-Test section does not end the test. If the operator continues to click the **Run** button after the Post-Test section is complete, new test runs will be added to the test.

About Post-Test Actions and Ending Tests

Post-Test actions will occur as a result of following:

- After a test run is complete and the **Review** page appears, the operator can click the **Run the Finish section** button. This is the manual way to run the post-test actions.
- The test designer can enter a value for the **Maximum Test Runs** variable. When the number of test runs equals this value, the test will run the Post-Test section. This is the automatic way to run the Post-Test section.

  Note: By default, this **Maximum Test Runs** variable value is set to 9999.

Entering a Test Description

Access

Define tab > **Description**

Enter a new, or modify the default, test description. The information you enter (and save) appears with the test names shown on the initial TWS application window and on the **Selection** tab.

Example: Suppose you want to create a test that uses English Units for rectangular specimens based on the ASTM A370-2009 template for a specific test sample. By default, the template uses metric units for round specimens, and the description is “This template conforms to ASTM A370-2009”. In this case, you may want to change the description of your test to “This test is for Sample ABC based on ASTM A370-2009 with English units.”

Tip: It is good practice to enter custom descriptions. Custom descriptions appear in the windows used for selecting tests, and help users scan the lists and select the proper test quickly.

Defining the Specimen

Access

Define tab > **Specimen**

Choose the specimen's geometry type and enter default parameters. The parameters you enter are automatically added to the **Test Flow** > **Pre-Test Run** > **Inputs** node, and are shown to the operator before each test run. This allows operators to verify and update specimen parameters before each test run.

Pre-Test Section

Assigning Pre-Test Inputs

Access

Define tab > **Test flow** > **Pre-test** > **Inputs**

Pre-test inputs are parameters you want the operator to enter or verify at the beginning of the test, before the first test run. Typical pre-test inputs are product identification, lot number, test date, and operator name.

To assign pre-test inputs:

1. Click + to show the Add Pre-Test Inputs window.
2. Use the arrow keys to move the desired inputs in the **Available** list to the **Selected** list. The inputs shown pertain to the test, not individual test runs.
3. Use the controls in the various tabs to define the input.

The default values of the pre-test inputs you select appear to the operator before the first test run in an Input Variables window. This allows the operator to verify or change the input values as required.

Example: Suppose you want the operator to enter the lot number (the number identifying the collection of specimens to be tested) before the test begins. To do this:

1. Click + and use the arrows to move the **Information 1** input from the **Available** list to the **Selected** list.
2. Click on **Information 1** in the **Selected** list.
3. On the **General** tab:
   - Optional: enter “LotNumber” for the **Identifier.**
   - Enter “Lot Number” for the **Display Name.**

When the test runs for the first time, the Input Variables window appears containing the **Lot Number** prompt.

Tip: To test changes that you make to pre-test inputs in a test that already has test runs, delete the test runs shown on the **Review** tab (select a test run, then right-click and select **Delete** > **All**). You can also create a new version of the test by selecting **File** > **Save As** or **File** > **Create Test From the Current Test**.

For information about **Pre-test details**, see "Input Property Details" on page 131.

Input Property Details

Input Property Details

Tab	Control	Description
General	**Identifier**	Uniquely identifies the input among all inputs in the template. Naming conventions: • Identifiers can contain only alphanumeric, hyphen (-) and underscore (_) characters. • Identifiers cannot begin with a number and must begin with an alphabetical character. • Identifiers cannot contain spaces, periods, or other special characters.
	Display name	Provides a more operator-friendly way of identifying the same input for use in displays and reports. The default display name is the same as the identifier. Naming conventions: • Display names must be unique; however, they can match their associated identifiers. • Display names can contain alphanumeric characters, spaces, and other special characters. • Display names cannot contain periods and must not begin with a number.
	Type	Specifies the type of input. **Number** (Default) holds numeric values.**Text** holds alphanumeric text.
	Default	Specifies a default value for the input to use until another value is provided for the input through either operator entry or as the result of a calculation. Enter a value that is consistent with the input type, such as a numeric or text value. **Note:** If you enter a string for a numeric input, the default value is forced to a numeric value of 0.000.

Tab	Control	Description
		If you enter a numeric value for a text input, the number is treated as text and not as a numeric value. If you enter a numeric value outside the input **Range**, an error message is generated.
	Dimension	Specifies the unit dimension. Dimension is used by the application to determine under which circumstances an input can be used. The dimension is used to filter out incompatible inputs from being incorrectly used within the MTS TestSuite application. Only inputs that have a dimension that matches the dimension of the signal can be used in an activity. For dimensionless values such as gain or other multipliers, **Dimension** and **Unit** are not required. **Note:** The **Dimension** and **Unit** properties are not applicable to the Text input type.
	Unit	Specifies a unit. The list only shows the set of units that are compatible with the **Dimension** of the input. This is the unit of measure that you want to assign to the input value.
	Calculation	**Note:** Pertains only to **Test Flow** nodes that include a **Calculations** selection. Shows the calculation that pertains to the selected input. Clicking the **Calculation** icon displays the Calculation Editor window, which allows you to edit or construct a new calculation for the selected input. For information about the Calculation Editor, see "Calculation Editor Overview" on page 165
	Enable	Enables or disables the input for use in the test. Clear **Enabled** if you do not want the input used in the test. Inputs that are not enabled appear grayed out in the inputs list and cannot be used in calculations of enabled inputs.
	Use previous test run value	**Note:** Available only for **Pre-test run Inputs** and **Post-test run Inputs**. Sets the value of the input to the value used in the previous test run. On the first test run, the default value is used.
	Editable post-Test	Determines post-test availability:

Tab	Control	Description
		• When selected, you can edit the input in post-test analysis. • When cleared, the input is read-only for post-test analysis. During post-test analysis, you cannot change the input value in tables or change the value by moving chart markers.
	Preallocate	**Note:** Available only for **Pre-test run Inputs** and **Post-test run Inputs**. Allows you to pre-define the value of the input for individual test runs on the **Review** tab. For more information, see "Pre-Allocating Multiple Test Runs" on page 116.
Format	**Type**	With **Fixed**, the number is shown in standard notation. For example, twenty million would be shown as 20000000.000 for a **Digit Type** of **Decimal with Digits** set to 3. With **Scientific**, the number is shown in E scientific notation. For example, twenty million is shown as 2.000E+007 for a **Digit Type** of **Decimal** with **Digits** set to 3. This notation is typically used for number values too large or small to be shown in standard decimal notation.
	Digit Type	With **Decimal**, specifies the number of digits to the right of the decimal symbol. For example, zero is shown as 0.000 with **Digits** set to 3. With **Significant**, specifies the number of digits that are significant. For example, zero is shown as 0.00 with **Digits** set to 3.
	Digits	Specifies how many digits to use for the **Significant Digit Type**.
Range	**Use Range**	Forces the value of the associated numeric input to be equal to or between the **Minimum** and **Maximum** limits. When selected, the application checks the value of the input during the test to determine if the value is within the specified range. If the value of the input is not within the range, the application adjusts the value to be within the range. If the value of the input does fall inside the specified range, the value remains unchanged.
	Minimum	Specifies the minimum value for the input. If **Inclusive** is used and the value of the input is less than the **Minimum** value, the application sets the input value equal to the **Minimum** value. If **Exclusive** is used and the value of the input is less than the **Minimum** value, the application sets the input value so that it equals the **Minimum**

Tab	Control	Description
		value plus the **Resolution** value.
	Maximum	Specifies the largest number the input value can be. If **Inclusive** is used, and the value of the input is greater than the **Maximum** value, the application sets the input value equal to the **Maximum** value. If **Exclusive** is used, and the value of the input is greater than the **Maximum** value, the application sets the input value so that it equals the Maximum value minus the **Resolution** value.
	Resolution	Sets the value of the input relative to the edge of the range. If the value of the input falls outside the specified (minimum and maximum) range, the value will be adjusted to fall within the range. The adjustment is relative to the closest edge of the range plus or minus the **Resolution** value. For example, if a value of -5 is entered and the range is **Minimum** 0 **Exclusive**, **Maximum** 10 **Exclusive**, and the **Resolution** is **1**, the value will be set to 1. However, if a value of 1.75 is entered, the value will not be changed because it falls within the specified range. If .75 is entered, the value will be set to 1 because .75 is outside of the **Minimum** 0 **Exclusive** limit.

Importing Pre-Test Inputs

Access

Define tab > **Test flow** > **Pre-test** > **Data import**

You may import data from an external file or a data base that defines inputs which apply to all test runs (Pre-Test Inputs).

Important: Ensure the external source of inputs is compatible with the MTS TWS format. For information about using MTS TWS with an external file or database, contact MTS Service.

Importing Inputs from a File

1. Select (to enable) **Data import**, and then select **File** as the source.
2. Select if you want the operator to be prompted for the file while the test is running, and if you want the imported Inputs to be shown to the operator.
3. Specify the **Folder path** and **File name** of the file.

Note: While assigning Pre-Test Inputs, you can specify the **Test ID** input as the name of the file from which you import pre-test inputs.

About Importing Inputs from a Database

The MTS TWS application can read test input data from a Laboratory Information Management System (LIMS).

The LIMS stores test procedure and test results data in a central database, accessible from a network directory shared by both systems.

The primary objective of automatically transferring input data from a LIMS/database is to eliminate human error and improve efficiency.

You can view and select test procedure files available in the LIMS with MTS TWS. When tests are complete, you can export the test data back to the LIMS.

Importing Inputs from a LIMS

1. Select (to enable) **Data** import, and then select **Database** as the source.
2. Select the **Data source** to specify the file format:
   - **dBASE** Files (*file*.dbf)
   - **Excel Files** (*file*.xls)
   - **MS Access Database** (multiple file extensions)
3. Add desired variables to the **Assign results** list using the + button.
4. Specify **query type** to specify query format.
5. Enter query.

Example: Suppose you want to import test input data from an MS Access Database using an SQL query. To configure MTS TWS to read Pre-Test Inputs from this data source:

1. For **data label**, select **MS Access Database**.
2. For **query type**, select **SQL**.
3. Enter query: “SELECT * FROM Tests WHERE Id=@varid”, where “SELECT”, “FROM” and “WHERE” are SQL keywords.

   This query selects a record from the “Tests” table that has an “Id“ that matches the value contained in the TestSuite TWS variable “varid”.

   The “@” in front of the variable name means that it takes the variable’s value. The “*” in “SELECT *” returns all columns for the record.

   The variables are populated in the order that they are listed by the columns that are returned. MTS TWS variables: “Id”, “Name”, and “Description” are the display names of the variables that will be populated with the columns returned from the query.

Adding Pre-Test Calculations

Access

Define tab > **Test flow** > **Pre-test** > **Calculations**

Pre-test calculations are calculations that pertain to global variables. Global variables are variables associated with the entire test, not individual test runs.

To add a pre-test calculation:

1. Click + to show the Add Pre-Test Calculations window.
2. Use the arrow keys to move the variable for which you wish to build a calculation from the **Available** list to the **Selected** list.
3. Click on a variable in the **Selected** list to show its properties.
4. Click the **Calculation** icon to show the Calculation Editor window.
5. Build the desired calculation from the available variables, signals, and functions on the associated tabs. For information about the Calculation Editor, see "Calculation Editor Overview" on page 165
6. Click OK.

Example: Suppose you want the **Test Name** variable, which is used in test reports, to be a concatenation of the lot number and the test date. To do this:

1. Click + and use the arrows to move the**Test Name** input from the **Available** list to the **Selected** list.
2. Click **Test Name** on the **Selected** list.
3. On the **General** tab, click the **Calculation** icon to show the Calculation Editor window.
4. Build the following equation: **LotNumber+TestDate**, ensuring that the **Errors** pane shows "Validation successful."

Note: See "Assigning Pre-Test Inputs" on page 130 for information about creating an input for lot number.

When you generate a test report, the **Test Name** is shown as a concatenation of the **Lot Number** and **Test Date** variables, such as "Lot A1234 8/10/2014 10:10:43 AM".

Pre-Test Run Section

Assigning Pre-Test Run Inputs

Access

Define tab > **Test flow** > **Pre-test run** > **Inputs**

Pre-test run Inputs are parameters you want the operator to enter or verify at the beginning of each test run. Typical Pre-Test Run Inputs are specimen dimensions and specimen identification.

To assign pre-test run inputs:

1. Click + to show the Add Pre-Test Inputs window.
2. Use the arrow keys to move the desired inputs in the **Available** list to the **Selected** list. The inputs displayed pertain to the test, not individual test runs.
3. Use the controls in the various tabs to define the input.

The default values of the pre-test run inputs you select are shown to the operator before each test run in an Input Variables window. This allows the operator to verify or change the input values as required.

Example: Suppose you want the operator to enter the identification number of each specimen before performing a test run. To do this:

1. Click + and use the arrows to move the **Information 2** input from the **Available** list to the **Selected** list.
2. Click **Information 2** on the **Selected** list.
3. On the **General** tab, enter:

- "EnterSpecimenIDNumber" for **Identifier.**
- "Enter Specimen ID Number" for **Display Name.**

At the beginning of each test run, the application shows the Input Variables window containing the **Enter Specimen ID Number** prompt.

For information about **Pre-test run details**, see"Input Property Details" on page 131.

Importing Pre-Test Run Inputs

Access

Define tab > **Test flow** > **Pre-test run** > **Data import**

You may import data from an external file or a data base that defines inputs which apply to individual test runs (Pre-Test Run Inputs).

Important: Ensure the external source of inputs is compatible with the MTS TWS format. For information about using MTS TWS with an external file or database, contact MTS Service.

Importing Inputs from a File

1. Select (to enable) **Data import**, and then select **File** as the source.
2. Select if you want the operator to be prompted for the file while the test is running, and if you want the imported Inputs to be shown to the operator.
3. Specify the **Folder path** and **File name** of the file.

Note: While assigning Pre-Test Inputs, you can specify the **Test ID** input as the name of the file from which you import pre-test inputs.

About Importing Inputs from a Database

The MTS TWS application can read test input data from a Laboratory Information Management System (LIMS).

The LIMS stores test procedure and test results data in a central database, accessible from a network directory shared by both systems.

The primary objective of automatically transferring input data from a LIMS/database is to eliminate human error and improve efficiency.

You can view and select test procedure files available in the LIMS with MTS TWS. When tests are complete, you can export the test data back to the LIMS.

Importing Inputs from a LIMS

1. Select (to enable) **Data** import, and then select **Database** as the source.
2. Select the **Data source** to specify the file format:
   - **dBASE** Files (*file*.dbf)
   - **Excel Files** (*file*.xls)
   - **MS Access Database** (multiple file extensions)
3. Add desired variables to the **Assign results** list using the + button.
4. Specify **query type** to specify query format.
5. Enter query.

Example: Suppose you want to import test run input data from an MS Access Database using an SQL query. To configure MTS TWS to read Pre-Test Run Inputs from this data source:

1. For **data label**, select **MS Access Database**.
2. For **query type**, select **SQL**.
3. Enter query: “SELECT * FROM Tests WHERE Id=@varid”, where “SELECT”, “FROM” and “WHERE” are SQL keywords.

   This query selects a record from the “Tests” table that has an “Id“ that matches the value contained in the TestSuite TWS variable “varid”.

   The “@” in front of the variable name means that it takes the variable’s value. The “*” in “SELECT *” returns all columns for the record.

   The variables are populated in the order that they are listed by the columns that are returned. MTS TWS variables: “Id”, “Name”, and “Description” are the display names of the variables that will be populated with the columns returned from the query.

Adding Pre-Test Run Calculations

Access

Define tab > **Test flow** > **Pre-test run** > **Calculations**

Pre-test run calculations are calculations the application performs that pertain to test run variables.

To add a pre-test run calculation:

1. Click + to show the Add Pre-Test Run Calculations window.
2. Use the arrow keys to move the variable for which you wish to build a calculation from the **Available** list to the **Selected** list.
3. Click on a variable in the **Selected** list to show its properties.
4. Click the **Calculation** icon to display the Calculation Editor window.
5. Build the desired calculation from the available variables, signals, and functions on the associated tabs. For information about the Calculation Editor, see "Calculation Editor Overview" on page 165
6. Click OK.

Test Run Section

Working with the Procedure Editor

Access

Define tab > **Test flow** > **Test run** > **Procedure**

A test procedure includes the system actions you want the application to perform during a test. It is made up of:

- Test activities displayed on the Test Editor showing a graphical representation of the test, and
- Inputs, calculations, and actions you configure in the various **Test Flow** nodes.

You can define the procedure while connected to hardware resources or while working offline with the resources stored with the test from an earlier connection.

Toolbox

The **Toolbox** panel contains icons for test activities that correspond to commands, data acquisition, and so on, which are the building blocks for test procedures.

To add an activity to a procedure, click the activity in the **Toolbox** and drag it to a location in the procedure marked with a green plus sign. If it is the first activity being added, drag the icon to the **Drop Activities Here** area.

The types of procedure activities include:

- Operator messages
- Conditional if/else processing

- Command generation
- Data acquisition
- Auto offset
- External device control
- Allow handset control

Note: For detailed information about test activities, see "Test Activities" on page 241.

Test Editor

The Test Editor shows the workflow as a sequence of connected activities, and provides a work area to edit tests.You can drag test activities from the **Toolbox** to the workflow.

The **Outline** button shows a hierarchical view of the test in the center panel when selected.

Test activities execute from top-to-bottom as they appear on the Test Editor. Other functions you configure for the **Test Flow** may run in parallel to the test activities. For example, **Limit Detection** runs in parallel to the command activities in the Test Editor.

The **Flowchart** view includes **Pan and Zoom** controls to help you move around complex text procedures. The **Outline** view provides the **Find** option for locating activities by name.

Properties Panel

The **Properties** panel allows you to define or change the information, characteristics, and appearance of the selected procedure activity. For example, you can use the **Properties** panel to change the **Rate** property in the **GoTo** panel of a **GoTo+DAQ+Detection** test activity.

To define the properties of an activity, select the activity icon on the Test Editor, and then select the Properties panel. You can choose to define properties as fixed values or variables.

Error indicators assist property definition. When you add an activity, red error icons appear next to the properties fields when information is missing or is incorrect. The **Properties** panel identifies the parameters, and the **Error List** located at the bottom of the display describes the problem. You must correct all errors in the test before you can initialize and run the test.

All activities have a **Display Name** that defaults to the activity name plus a few key parameters that define the activity. You can enter a unique name for an activity. All activities provide an optional **Description** field to document the procedure design. All activities also have the **Enabled** check box selected by default. You can clear the **Enabled** check box to disable an activity.

Test Design Process

The process of test design is iterative. It is good practice to implement your changes iteratively and run the test after each iteration.

Note: For information about setting up chart parameters for viewing test runs, see "Using Charts" on page 257.

Typical Test Design Flow

Step	Activity	Description
1	Start TWS and select a test or template to customize.	Start the MTS TestSuite TWS application and select a test or template that is as close as possible to the test you want the operator to perform.
2	Modify the test.	Modify the test by enabling disabled activities within the template, changing file options, adding resources, and editing the workflow.
3	Select and customize a report template.	Create or modify the existing test report template by adding test variables, changing the format, and so on.  **Note:** Editing report templates requires the optional MTS Reporter Add-In for Microsoft Excel
4	Run the customized test to validate its design.	You typically validate test design by running the test from the operator's perspective. You may choose to enhance your test with special operator information.
5	Save your test as a template.	When the test runs as desired, you typically save the test as a template. When the operator opens the template, they will actually be opening a test version of the template.
6	Deliver the template to the operator.	When your new template meets your requirements, deliver it to the operator.

Selecting Signals for Data Acquisition

Access

Define tab > **Test flow** > **Test run** > **Signals**

The system acquires data from the signals listed for the data acquisition components of all **Go To + DAQ+ Detection** and **Dwell + DAQ + Detection** activities in the test.

To add signals:

1. Click  to display the Add Signals window.
2. Use the arrow keys to move the signals for which you wish to acquire data from the **Available** list to the **Selected** list.
3. Click OK.

Editing Test Run Calculations

Access

Define tab > **Test flow** > **Test run** > **Calculations**

To edit a test run calculation:

1. Click on a variable in the **Test run calculations** list to show its properties.
2. Click the **Calculation** icon to show the Calculation Editor window.
3. Build the desired calculation from the available variables, signals, and functions on the associated tabs. For information about the Calculation Editor, see "Calculation Editor Overview" on page 165
4. Click OK.

Working with Limit Detection

Access

Define tab > **Test flow** > **Test run** > **Events** > **Limit Detection**

Use this feature to configure limit detection according to your test specifications.

Note: The limit detection you configure in this window is independent of the break detection controls in the command activities (**Go To + DAQ + Detection** and **Dwell + DAQ + Detection**) in **Test Flow** > **Test Run** > **Procedure**.

About Limit Detection

Limit detectors monitor signals and variables and compare their values against defined upper and lower limits. Limit detection is active only while command activities (such as **Go To + DAQ + Detection** or **Dwell + DAQ + Detection**) are active.

When a detector triggers, the test stops performing its activities (as shown in the procedure editor) and skips to the next enabled task. A message indicating the limit triggered is also written to the test log.

Note: In most tests, the next enabled task the system performs after a detector triggers is typically **Return to Zero**, (**Test Flow** > **Test Run** > **Return to Zero**).

Controller Limits

Independent of the limit detection feature, the MTS Insight controller continually monitors the frame capacity, the force on the load cell, and the physical limit switches on the frame. The controller issues an interlock if the frame capacity is exceeded, if the monitored force exceeds 120% of the load cell rating, or if the limit switches trip.

Limit Detection Property Details

Control	Description
Signals and variables	Select which signals and variables you want the limit detector to monitor. **Primary Extension** and **Load** variables are typically selected by default. Click + to open the Add Signals and Variables window.
Upper limit/Lower limit	Select to enable the type of limit detection you desire for the selected signal or variable. Toggle the variable/numeric value icons to select the associated variable or enter the desired signal value.

Limit Detection Examples

You can configure limit detection in a variety of ways. Review the following examples to find the method that best suits your needs.

Limit Detector Example 1: Suppose you want to set the value of the load limit to 250 N for a tensile test without using a variable, and without the limit being visible to the operator. To do this:

1. Access **Define** > **Test flow** > **Test run** > **Events** > **Limit detection**
2. In the **Signals and variables** panel, click + to show the Add Signals and Variables window.
3. Locate **Load Signal** in the **Available** list, and use the arrows to move it to the **Selected** list.
4. In the **Limits** panel, select to enable **Upper limit**.
5. Toggle the variable/numeric value icon to display #.
6. Enter **250** as the numeric value, and select **N** as the unit.

When the operator runs the test, the load limit triggers when the load signal reaches 250 N.

Limit Detector Example 2: Suppose you want to set the value of the load limit to 250 N for a tensile test using a variable, but without the limit being visible to the operator. To do this:

1. Access **Define** > **Test flow** > **Pre-test run** > **Inputs**
2. In the **Pre-test run inputs** panel, click + to show the Add Pre-Test Run Inputs window.
3. Locate and click the **Load Limit High** input in the **Available** list, but do not use the arrows to move it to the **Selected** list (this prevents the variable from being shown to the operator during the test).
4. In the **General** panel, change the **Default** to **250**, and **Unit** to **N**.
5. Click **OK**.

When the operator runs the test, the load limit triggers when the load signal reaches 250 N.

Limit Detector Example 3: Suppose you want the operator to have the ability to set the value of the load limit at the beginning of each test run in a tensile test. To do this:

1. Access **Define** > **Test flow** > **Pre-test run** > **Inputs**
2. In the **Pre-test run inputs** panel, click + to display the Add Pre-Test Run Inputs window.
3. Locate the **Load Limit High** input in the **Available** list, and use the arrows to move it to the **Selected** list.
4. Click **OK**.

When the operator runs the test, the **Load Limit High** input value is shown at the beginning of reach test run in a Edit Variable Values window. The operator can change the limit value and unit selection as desired.

Configuring Extensometer Removal

Access

Define tab > **Test flow** > **Test run** > **Events** > **Extensometer**

Use this feature for tests that use extensometers as their primary source of extension data for the initial part of the test, and then use the displacement sensor for the remainder of the test. The removal point is usually set as the full travel of the extensometer. For instance, a 1 inch-50% extensometer can open 0.5 inches, therefore, the removal point for this extensometer is set at 0.5 inches.

Extensometer Property Details

Control	Description
What type of extensometer will you use for the test?	Select the type of extensometer you are using for the test, ensuring the image matches the device you are using. If you select an extensometer that can be removed during the test, the window expands to show additional options. If you select an extensometer that cannot be removed during the test, additional options are not shown. **Note:** Extensometers that cannot be removed are typically non-contacting extensometers, and extensometers that measure downward deflection in compression testing.
Will you remove the extensometer before the specimen breaks?	Select **Yes** if you want the operator to remove the extensometer during the test. Select **No** if you do not want the operator to remove the extensometer during the test. Selecting **No** removes additional options in the window.
Which signal is the extensometer monitoring?	Click + to open the Add Signals window.

Control	Description
At what point will the extensometer be removed?	This is typically set as the full travel of the extensometer. You may enter a value and unit of measurement directly, or select a variable that specifies the same. Click x to toggle between numeric and variable selections.
Do you want the test to stop and hold when the extensometer reaches the removal point?	**Note:** The system shows a message informing the operator to remove the extensometer when the removal point is reached, regardless of whether you choose to stop and hold the test or not. Select **Yes** if you want the test to stop and hold when the extensometer reaches the removal point and the operator removes the extensometer. This will require the operator to click the **Run** button to resume the test. Select **No** if you want the operator to remove the extensometer while the test is running, after the extensometer reaches the removal point. **Note:** Removing the extensometer while the test is running is typically performed only on slow moving tests, where the operator has adequate time to remove the extensometer.

Configuring Return to Zero

Access

Define tab > **Test flow** > **Test run** > **Return to zero**

Select **Return extension to the zero position** to cause the crosshead or actuator to return to the zero position at the end of each test run. This performs the same action as manually clicking the **Return Crosshead** control on the Control Panel.

Select **Show confirmation before performing return to zero action** to display a prompt to the operator before the action begins.

When selected, this action occurs even when the test status is stopped.

Note: The return rate is determined by a hardware setting that only an MTS Field Service representative can access. If the rate requires adjustment, contact MTS.

Post-Test Run Section

Assigning Post-Test Run Inputs

Access

Define tab > **Test flow** > **Post-test run** > **Inputs**

Post-test run Inputs are parameters you want the operator to enter or verify at the end of each test run. Typical post-test run Inputs are commentary about specimen condition, and reason or type of failure (for instance, specimen, adhesive, or base material).

To assign post-test run inputs:

1. Click + to show the Add Post-Test Run Inputs window.
2. Use the arrow keys to move the desired inputs in the **Available** list to the **Selected** list. The inputs shown pertain to the test, not individual test runs.
3. Use the controls in the various tabs to define the input.

The default values of the post-test run inputs you select are shown to the operator after each test run in an **Input Variables** window. This allows the operator to verify or change the input values as required.

Example: Suppose you want the operator to enter the reason the test run ended from a predefined list. You expect that in most cases, the reason the test run will end is because a limit is reached rather than the specimen failing. To do this:

1. Click + and use the arrows to move the**Test Run End Reason** input from the **Available** list to the **Selected** list.
2. Click **Test Run End Reason** in the **Available List**.
3. On the **General** tab, change the **Default** selection from **Break Detected** to **Limit Detected**.

After each test run, the **Input Variables** window appears containing the **Test Run End Reason** prompt, with a default value of **Limit Detected**. Based on the circumstances, the operator may also choose **Test Stopped**, **Procedure Finished**, **Break Detected**, **Hardware Status Changed**, or **Digital Input State Changed**.

For information about **Post-test run details**, see "Input Property Details" on page 131.

Adding Post-Test Run Calculations

Access

Define tab > **Test flow** > **Post-test run** > **Calculations**

Post-test run calculations are calculations that apply to test run variables.

To add a post-test calculation:

1. Click + to show the Add Post-Test Calculations window.
2. Use the arrow keys to move the variable for which you wish to build a calculation from the **Available** list to the **Selected** list.
3. Click on a variable in the **Selected** list to display its properties.
4. Click the **Calculation** icon to show the Calculation Editor window.

5. Build the desired calculation from the available variables, signals, and functions on the associated tabs. For information about the Calculation Editor, see "Calculation Editor Overview" on page 165
6. Click OK.

Generating Post-Test Run Reports

Access

Define tab > **Test flow** > **Post-test run** > **Report**

Use this feature to automatically generate a report after each individual test run is finished.

Report Property Details

Control	Description
Run report	Select to run the report after the test run or test.
Name	Selects a report template to use for the report. If desired, you can explicitly override the default template set in the **Report Templates** tab. Parentheses indicate the **Default Test Report** template or **Default Test Run Report** template. **Note:** If you do not select a report template, a warning is shown in the **Error** list.
Open report	Select to open the report after performing the test run or test.

Control	Description
Print report	Select to print the report after performing the test run or test.
Send report to e-mail	Allows you to send a copy of the report to one or more e-mail addresses. Separate multiple addresses with semi-colons. The e-mail is sent right after the report is generated and added to the e-mail as an attachment. This feature requires valid syntax for To e-mail and From e-mail addresses. syntax. Selecting the Send report to e-mail check box expands the panel and enables the corresponding controls: **To**: (Required) Sends an e-mail message to the specified e-mail address or variable. **Cc**: Required) Sends an e-mail message to the specified e-mail address or variable. **From**: Shows the default address from which the e-mail is sent or a variable. The default From address is configured in **Preferences > Configuration > E-Mail.** **Subject**: Populates the Subject line of the e-mail. The default is “MTS TestSuite report”. **Message**: Shows the message in the body of the e-mail. You can use the default variables, insert custom or other variables, or enter text directly in the **Message** text box. The **Insert Variable** button opens the Variable Selection window. Selected variables appear in the **Message** box.

About the Report Layout

The report layout is based on the selected report template, which you select in the **Report Template** node. By default, the report format is a Microsoft Excel file. You can also select other report format options, such as PDF. For more information, see “Working with Report Templates” on page 155.

Sending a Report to an E-Mail Address

You can choose to send the report to one or more e-mail addresses.To send the report attached to an e-mail, you must configure the e-mail server settings. Click **Preferences** menu > **Configuration** option > **E-Mail** tab. If you are unsure of your mail server settings, contact your network administrator. For more information, see “E-Mail Settings” on page 73

Exporting Post-Test Run Data

Access

Define tab > **Test flow** > **Post-test run** > **Data export**

You may export data from individual test runs (Post-Test Run) to an external file or a database.

Important: Ensure the external source of inputs is compatible with the MTS TWS format. For information about using MTS TWS with an external file or database, contact MTS Service.

Exporting Data to a File

1. Select (to enable) **Export data**.
2. For **Export data to**, select **File**.
3. For **Select data to export**, click the + button to show the Variables Selection window.
4. Select the variables with which you wish to export data.
5. Use the arrow keys to move the selected variables from the **Available** list to the **Selected** list.
6. Click **OK** to dismiss the Variables Selection window.

   Note: You can select a variable when selecting the **Unit Set** and **File Name** in the next step. To do this, click x , and select the desired variable.
7. Select the desired unit set, specify or navigate to the desired export folder, and enter a file name.

About Exporting Data to a Database

The MTS TWS application can read test procedures from and write test data to a.Laboratory Information Management System (LIMS).

The LIMS stores test procedure and test results data in a central database, accessible from a network directory shared by both systems.

The primary objective of automatically transferring test data to a LIMS/database is to eliminate human error and improve efficiency.

You can view and select test procedure files available in the LIMS with MTS TWS. When tests are complete, you can export the test data back to the LIMS.

Exporting Data to a LIMS

1. Select (to enable) **Export data**, and then select **Database** as the source.
2. Select the **data source** to specify the file format:
   - **dBASE** Files (*file*.dbf)
   - **Excel Files** (*file*.xls)
   - **MS Access Database** (multiple file extensions)
3. Add desired variables to the **Assign results** list using the + button, and select the desired **Unit set**.
4. Specify **query type** to specify query format.
5. Enter query.

Example: Suppose you want to export test run data to an MS Access Database using an SQL query. To configure MTS TWS to export Post-Test Run data to this data source:

1. For **data source**, select **MS Access Database**.
2. For **query type**, select **SQL**.
3. Enter query: "INSERT INTO Results(Id,PeakLoadResult) VALUES(@Id,@PeakLoad)"", where "INSERT INTO" and "VALUES" are SQL keywords.

   In this example, the query inserts a new record into the table named "Results".

   The column names of the table are "Id" and "PeakLoadResult". These columns in the new record are filled-in from the data in the variables with the internal names of "Id" and "PeakLoad". The "@" in front of the variable name means that it takes the variable's value.

   The typical use of this export activity is to store test results back into the LIMS/database under the Id read from the LIMS/database.

Post-Test Section

Assigning Post-Test Inputs

Access

Define tab > **Test flow** > **Post-test** > **Inputs**

Post-test inputs are parameters you want the operator to enter or verify at the end of the test, after all test runs are finished.

To assign post-test inputs:

1. Click + to display the Add Post-Test Inputs window.
2. Use the arrow keys to move the desired inputs in the **Available** list to the **Selected** list.
3. Use the controls in the various tabs to define the input.

The default values of the post-test inputs you select are shown to the operator when the test is finished in the Input Variables window. This allows the operator to verify or change the input values as required.

For information about **Post-test details**, see "Input Property Details" on page 131.

Adding Post-Test Calculations

Access

Define tab > **Test flow** > **Post-test** > **Calculations**

Enter a new or modify the default test description. The Post-Test Calculations are calculations that pertain to global variables. Global variables are variables associated with the entire test, not individual test runs.

To add a post-test calculation:

1. Click + to show the Add Post-Test Calculations window.
2. Use the arrow keys to move the variable for which you wish to build a calculation from the **Available** list to the **Selected** list.
3. Click on a variable in the **Selected** list to display its properties.
4. Click the **Calculation** icon to show the Calculation Editor window.
5. Build the desired calculation from the available variables, signals, and functions on the associated tabs. For information about the Calculation Editor, see "Calculation Editor Overview" on page 165
6. Click OK.

Generating Post-Test Reports

Access

Define tab > **Test flow** > **Post-test** > **Report**

Use this feature to automatically generate a report when the test is finished. The report includes data for all of the test runs within the test.

Report Property Details

Control	Description
Run report	Select to run the report after the test run or test.
Name	Selects a report template to use for the report. If desired, you can explicitly override the default template set in the **Report Templates** tab. Parentheses indicate the **Default Test Report** template or **Default Test Run Report** template. **Note:** If you do not select a report template, a warning is shown in the **Error** list.
Open report	Select to open the report after performing the test run or test.

Control	Description
Print report	Select to print the report after performing the test run or test.
Send report to e-mail	Allows you to send a copy of the report to one or more e-mail addresses. Separate multiple addresses with semi-colons. The e-mail is sent right after the report is generated and added to the e-mail as an attachment. This feature requires valid syntax for To e-mail and From e-mail addresses. syntax. Selecting the Send report to e-mail check box expands the panel and enables the corresponding controls: **To**: (Required) Sends an e-mail message to the specified e-mail address or variable. **Cc**: Required) Sends an e-mail message to the specified e-mail address or variable. **From**: Shows the default address from which the e-mail is sent or a variable. The default From address is configured in **Preferences > Configuration > E-Mail.** **Subject**: Populates the Subject line of the e-mail. The default is “MTS TestSuite report”. **Message**: Shows the message in the body of the e-mail. You can use the default variables, insert custom or other variables, or enter text directly in the **Message** text box. The **Insert Variable** button opens the Variable Selection window. Selected variables appear in the **Message** box.

About the Report Layout

The report layout is based on the selected report template, which you select in the Report Template node. By default, the report format is a Microsoft Excel file. You can also select other report format options, such as PDF. For more information, see “Working with Report Templates” on page 155.

Sending a Report to an E-Mail Address

You can choose to send the report to one or more e-mail addresses.To send the report attached to an e-mail, you must configure the e-mail server settings. Click **Preferences** menu > **Configuration** option > **E-Mail** tab. If you are unsure of your mail server settings, contact your network administrator. For more information, see “E-Mail Settings” on page 73

Exporting Post-Test Data

Access

Define tab > **Test flow** > **Post-test** > **Data export**

You may export data from the entire test (Post-Test) to an external file or a database.

Important: Ensure the external source of inputs is compatible with the MTS TWS format. For information about using MTS TWS with an external file or database, contact MTS Service.

Exporting Data to a File

1. Select (to enable) **Export data**.
2. For **Export data to**, select **File**.
3. For **Select data to export**, click the + button to show the Variables Selection window.
4. Select the variables with which you wish to export data.
5. Use the arrow keys to move the selected variables from the **Available** list to the **Selected** list.
6. Click **OK** to dismiss the Variables Selection window.

   Note: You can select a variable when selecting the **Unit Set** and **File Name** in the next step. To do this, click x , and select the desired variable.
7. Select the desired unit set, specify or navigate to the desired export folder, and enter a file name.

About Exporting Data to a Database

The MTS TWS application can read test procedures from and write test data to a.Laboratory Information Management System (LIMS).

The LIMS stores test procedure and test results data in a central database, accessible from a network directory shared by both systems.

The primary objective of automatically transferring test data to a LIMS/database is to eliminate human error and improve efficiency.

You can view and select test procedure files available in the LIMS with MTS TWS. When tests are complete, you can export the test data back to the LIMS.

Exporting Data to a LIMS

1. Select (to enable) **Export data**, and then select **Database** as the source.
2. Select the **data source** to specify the file format:
   - **dBASE** Files (*file*.dbf)
   - **Excel Files** (*file*.xls)
   - **MS Access Database** (multiple file extensions)
3. Add desired variables to the **Assign results** list using the + button, and select the desired **Unit set**.
4. Specify **query type** to specify query format.
5. Enter query.

Example: Suppose you want to export test data to an MS Access Database using an SQL query. To configure MTS TWS to export Post-Test data to this data source:

1. For **data source**, select **MS Access Database**.
2. For **query type**, select **SQL**.
3. Enter query: “INSERT INTO Results(Id,PeakLoadResult) VALUES(@Id,@PeakLoad)””, where “INSERT INTO” and “VALUES” are SQL keywords.

   In this example, the query inserts a new record into the table named “Results”.

   The column names of the table are “Id” and “PeakLoadResult”. These columns in the new record are filled-in from the data in the variables with the internal names of “Id” and “PeakLoad”. The “@” in front of the variable name means that it takes the variable’s value.

   The typical use of this export activity is to store test results back into the LIMS/database under the Id read from the LIMS/database.

Configuring the Results Table

Access

Define tab > **Review** > **Results Table**

The results table on the **Review** tab allows you to compare one test run to another. The results table is automatically shown to the operator when a test run is finished. The results table shows parameter values for each test run in rows, and labels for each parameter in column headings.

Use the + and − buttons to edit the parameters shown on the results table.

Note: Some parameters include the **Editable post-test** attribute. If this is selected (checked), the parameter appears in two locations on the **Review** tab. It appears in a non-editable form in the results table, and in an editable form in the variables table. The variable table appears when you select the **Review** tab multi-panel view.

For information about **Results details**, see“Input Property Details” on page 131.

For information about the results table, see“Results Table” on page 291.

Working with Report Templates

Access

Define tab > **Report templates**

To set the default report template:

1. Locate the list in the **Report Templates** node that pertains to the part of the test for which want to generate a report. For instance, if you want to generate a report for individual test runs, locate the **Test Runs** list.

2. Select the row that contains the report template that you want to make the default, or add new report templates using the **Add** button.
3. Click the **Make Default** button. The **Default** column shows **Yes**.

To set the default report template properties:

Note: To create a report in the PDF format, select the **Microsoft Excel Workbook** format option and in Microsoft Excel, use the Save As PDF feature.

1. Select the desired format from the list in the **Report Format** column. The default format is **Microsoft Excel Workbook**You may also select **Comma-Separated Values** or **Tab Delimited Text**.
2. Select the desired write options from the list in the **Write Options** column. The default is **New**. You may also select **Overwrite** or **Append to File**.
3. Select the desired destination for the generated report in the **Report Location** column. You may enter a file path directly or enter a variable that defines the file path.

Working with Resources

Define tab > **Resources**

The **Resources** node serves as a map between controller configuration resources and the test definition. It allows you to resolve mapping errors between the controller and the test. It also allows you to define tests away from the controller, that is, without a controller connection. When you define a test this way, you can map the resources to the controller when you connect to the controller.

For detailed information about the **Resources** node, see “Resource Details” on page 83.

Working with Variables

Variable Basics

Variables Overview

A variable is a container that is used to store data values. A variable provides the ability to store and manipulate data in applications. Variables can represent numeric values or text strings. The MTS TestSuite applications use variables in test activities, data acquisition activities, analyzer applications, and reports.

Note: Variables may also be referred to as **Inputs**, **Input Variables**, or **Calculations**.

A simple variable can store a single text string, Boolean, or numeric value. An array variable stores a series of text, numeric, or Boolean values. Various activities such as calculations, operator inputs, and data acquisition can be used to set the value stored by the variable.

Variables defined in the test are stored with the test. The definition of a given variable, including its name and type, remains constant throughout the test run, even though its value may change. Each test run saves a "snapshot" of the test procedure and variable definitions at the time the test run is created.

Typical Uses for Variables

After you create a variable, you can use the variable in activities in the test procedure, test reports, and calculations.

Note: MTS supplied templates include pre-configured data acquisition activities and variables that save data for use in various charts and tables that appear in the test, report templates, and post-test analysis.

Typical Uses for Variables

Item	Description
Test Activities	Several test activity property settings can be entered as a static value or a variable. You can configure variables for an operator to enter. Other settings are defined by variables so that they can be changed during the test and in post-test analysis scenarios. Some variables only represent one value in a given test.
Charts	You can use variables to plot chart data in various places in both the **Review** and **Test-run chart** > **Chart** nodes of the test definition tree.
Test Reports	Most information that you want to include in a report must be saved in a variable: • If you are creating your own tests and want to include test data in reports, you must create data acquisition (DAQ) activities to collect the data. • Reports can also include other single-variable values that are not associated with data acquisition.
Data Analysis	When the Availability property of variables used in calculations are configured as Editable Post-Test, you can change these values in post-test analysis to correct mistakes in operator setup or to create what-if scenarios.

Item	Description
	Data acquisition activities can map signals to variables to store the test data that is displayed in various post-test charts and tables.

Variable Types Overview

The MTS TestSuite applications provide the ability to create the following types of variables:

- Numeric
- Text
- Calculated

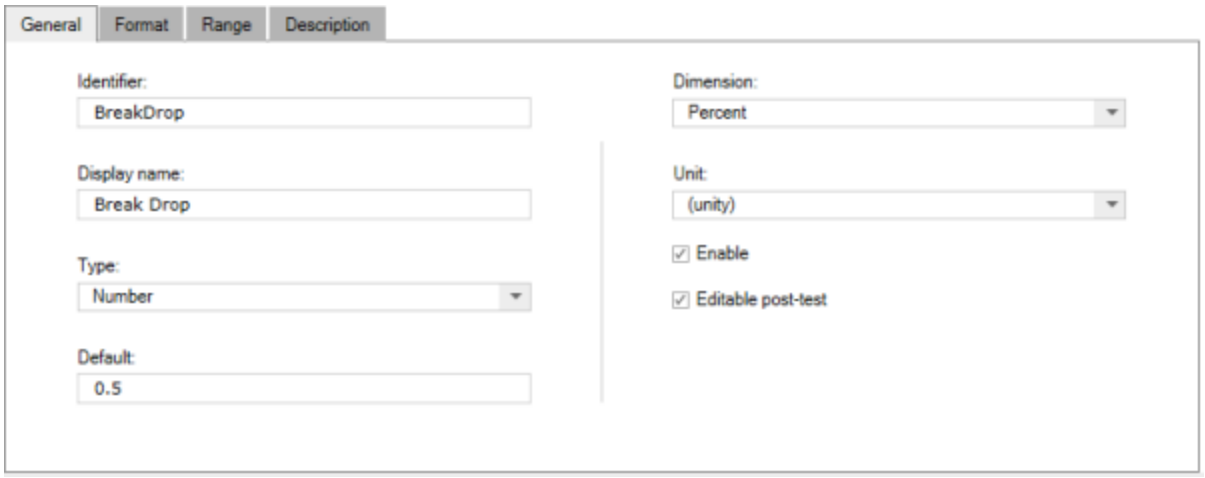
The desired use of the variable dictates the variable type (or combination of types) defined in the property settings for a variable. Other property settings define variable naming, default values, and so on.

Some variables are fairly straightforward in their use, such as a numeric variable for an End Point level defined by an operator at the beginning of a test run. Other variables are more complex and use a combination of variable types; for example, numeric variable values that are the result of a calculation.

Numeric Variables Overview

Numeric variables contain a single numeric value. To assign a numeric variable, select the **Number** option under the **Type** drop-down on the **Pre-test**, **Pre-test run**, **Post-test run**, or **Post-test** node of the test definition tree.

Numeric Variable Properties

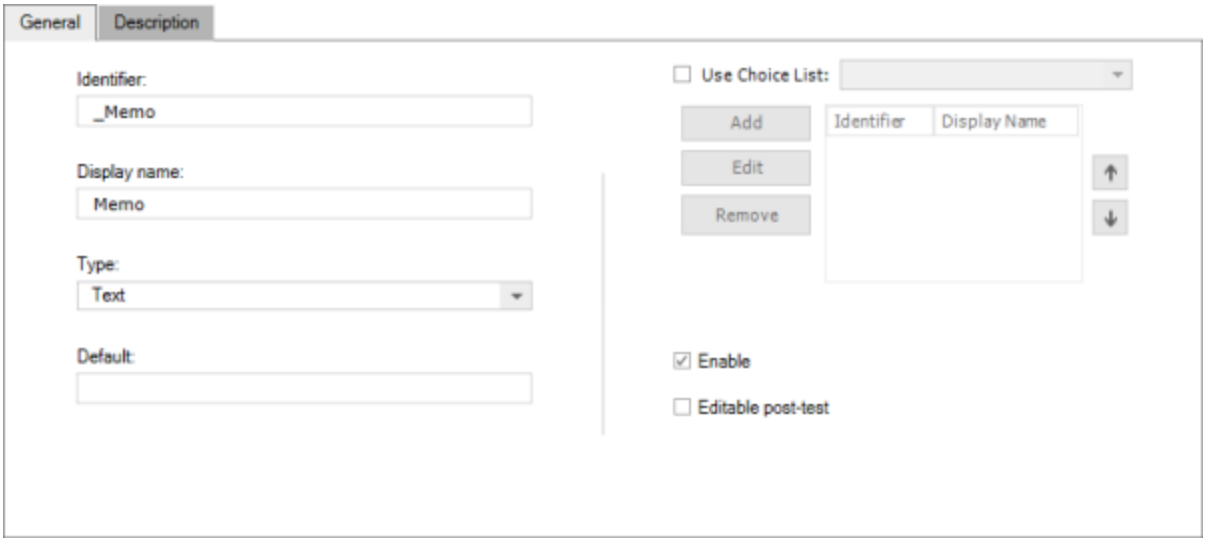


Text Variables Overview

Text variables are used to hold text, such as a test run name, operator name, or Test ID. A text variable can contain alphanumeric and space characters.

To specify that a variable hold a string, select **Text** as the **Type**. In addition to using text variables to store text values, you can also include text variables in calculations that use text functions to parse and manipulate the text values.

Text Variable Properties



Array Variables Overview

An array variable is a variable that holds a list. An array is a collection of data values, and each data value in an array has an associated index. Array indexes begin with zero. For example, an array with an identifier of “a” and a size of ten integers contains the following elements:

a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[7], a[8], a[9]

In the preceding example, a[4] refers to the fifth element of the array a.

An array variable in MTS TestSuite holds multiple values of the same data type. The available types of array variables include:

- Array of Boolean
- Array of Numbers
- Array of Text

The elements of an array can be individually referenced or assigned; that is, each element of an array is a variable of its own, and can be modified independently of other elements in the array.

Note: While you can view any existing Array of Numbers variables that may exist in an existing test or template, you cannot create new Array of Numbers variables.

Calculated Variables Overview

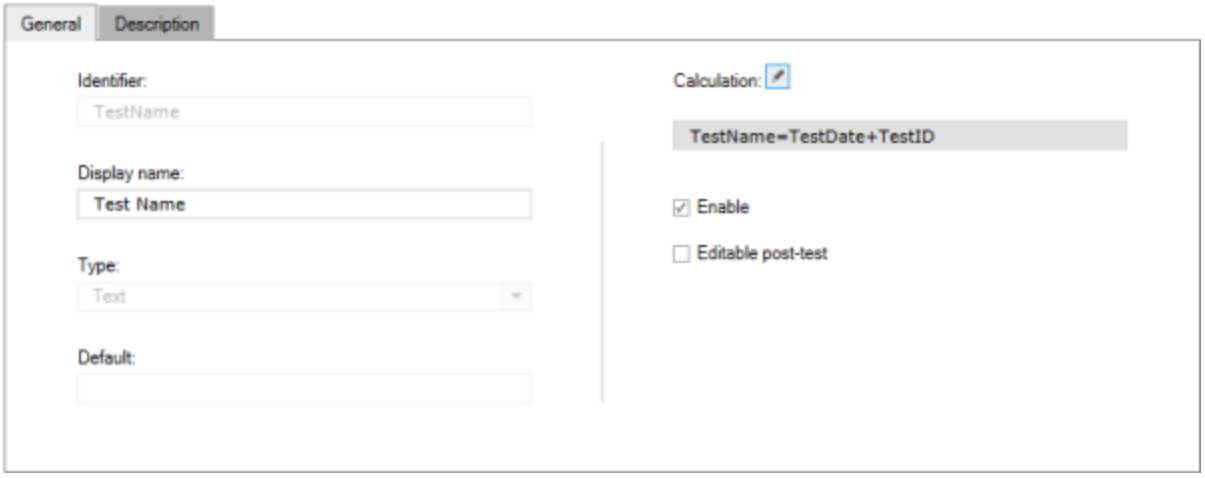
Most variable types allow you to define calculations that determine the value of the variable. The calculation can include operations, functions, and references to other variables. The result of the calculation must generate the same type of data that is stored in the variable.

Example

For example, you want to create several calculations that include velocity, which itself is a calculation. You can create a calculation that is used to populate the value of a variable, label it “velocity”, define its equation, and use it wherever an equation requires a value for velocity.

In the figure below, a **Test Name** variable uses the calculation **TestName=TestDate+TestID**:

Calculated Variable Properties



Viewing Variable Properties

To view the properties of a variable:

1. Select the **Test Definition** tab.
2. In the test definition tree, expand the **Test flow** node.
3. On either the **Pre-test**, **Pre-test run**, **Post-test run**, and **Post-test** nodes of the test definition tree, you can view the available Inputs (variables) and Calculations available for the respective section of the test.

   Note: Only global (common) variables can be viewed in the **Pre-test** and **Post-test** nodes.

Choice Lists Overview

Access

Properties panel > **Choice List** panel

You can set multiple values for a variable or text input in a choice list and use them in many activities. When you create a variable or text input with a choice list, you must assign a default value from your list of values. Choice list values are alphanumeric character strings.

Example 1

Use choice lists in If-Else conditions and While loops. For example, in an **If-Else** activity, the condition checks for the value of the variable to be equal to one of the choice list values. If that value matches, the test procedure follows the “if” path.

Example 2

You can create a wave shape variable with values of triangle, sine, and square as its values as another example.

Global choice lists

Use a global choice list if you want to assign the same choice list to multiple variables throughout a test. This keeps consistency and predictability in your test. A global choice list is defined from the **Tools** menu.

Local choice lists

Use a local choice list if you are using one instance of the variable that only contains those values. A local choice list has no name and is assigned only to one variable. A local choice list is defined directly in the properties panel of a text variable.

Considerations when deleting choice lists

If you delete a variable that uses a global choice list, only the variable is deleted. The global choice list remains and is available for use with other variables. If you delete a variable or text input that uses a local choice list, you also delete the choice list.

Defining a Global Choice List

Prerequisite: Before you can access the Define Global Choice List option, you must open or create a New Test.

After you create a global choice list, it is available in the **Use Choice List** property of any text variable.

1. From the **Tools** menu, choose the **Define Global Choice List** option. The Define Global Choice List window opens.
2. Click **Add** in the top row of buttons to define a new choice list. A new choice list shows under **Display Name** with a default name of “Choice List n”.
3. Enter a name for the choice list. You can use spaces in the name for readability.
4. Click **Add** in the Choice List Items panel. The New Choice List Entry window opens.
    A. Enter a name for the choice list entry in the **Identifier** field. Do not enter spaces or periods in the name, or start with a number.
    B. Enter a display name in the **Display Name** field.
    C. Click **OK**. To add more values to your variable choice list, repeat these steps.
5. When you are done adding choice list items, click **OK**.

Editing a Global Choice List

You can edit the name of a global choice list, or edit the items within a global choice list.

1. From the **Tools** menu, choose the **Define Global Choice List** option. The Define Global Choice List window opens.
2. Click a display name or a choice list item to select it.
3. Click **Edit**. If you are editing a choice list item, the Edit Choice List Entry window opens. In this window you can edit the identifier and display name.
4. Click **OK**.

Removing a Global Choice List

To remove a global choice list:

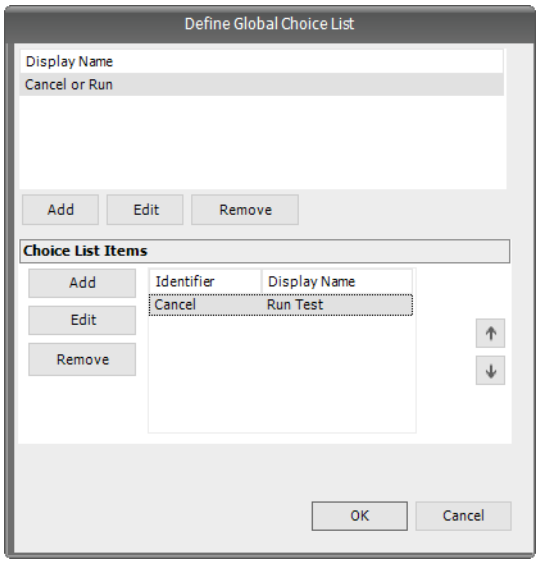
1. From the **Tools** menu, choose the **Define Global Choice List** option. The Define Global Choice List window opens.
2. Select the choice list to remove and click **Remove**.
3. Click **OK**.

Define Global Choice List Window

Access

Tools menu > **Define Global Choice List**

Define Global Choice List Window



Define Global Choice List Window

Item	Description
Display Name	Displays the Name of the global choice list. You can select the name and click Edit or Remove. Click the up or down arrow in the Display Name heading row to sort the available lists alphabetically.
Add	Adds a global choice list.
Edit	Edits the selected global choice list.
Remove	Removes the selected global choice list.
Choice List Items	
Add	Adds a choice list entry to the selected global list.
Edit	Opens the Edit Choice List Entry window to change the identifier or display name for the list entry.
Remove	Removes the selected list entry from the selected global list.

Adding a Local Choice List

Follow these steps to add a local choice list to a text variable. Unlike a global choice list, a local choice list is not available to other text variables in a test.

1. Navigate to the **Test Definition** tab > **Variables** tab.
2. Click the text variable to which you want to define a local choice list.
3. In the Choice List panel, select the **Use Choice List** checkbox.
    A. Click **Add**. The New Choice List Entry window opens.
    B. Enter a name for the choice list entry in the **Identifier** field. Do not enter spaces or periods in the name, or start with a number.
    C. Enter a display name in the **Display Name** field.
    D. Click **OK**. To add more values to your variable choice list, repeat these steps.
4. (Required) In the **Default Value** list, select the default value from your list.

Advanced Variable Information

Variables Calculations

Calculation Editor Overview

Access

Use the Calculation Editor to customize calculations for variables. To access the Calculation Editor, select either the **Pre-test**, **Pre-test run**, **Post-test run**, or **Post-test** nodes of the definition tree in the **Define tab**. Then, click the  button next to the **Calculation** field to open the Calculation Editor and create a new calculation.

Calculation panel

Use the Calculation panel of the Calculation Editor to build a variable calculation. You can edit text directly in the panel, or you can use the Variables and Functions panels to insert defined variable and functions. Note the following on the Calculation panel:

- Variables can use reference signals, other variables, and calculation parameters.
- Names are not case-sensitive.
- Variables and signal labels have identifiers and display names. Use the identifier name when creating a calculation that uses the signal.

When you are satisfied with the calculation, click **OK** to add the calculation to the variable selected in the Variables Editor window.

Note: If adjacent variable names are not separated by a space or function in the Calculation panel, they are interpreted as a single variable name. This is likely to result in a "...variable..., was not found" error message.

Errors panel

The Errors panel of the Calculation Editor continually evaluates the calculation as you build it.

Note: Calculations are validated only on syntax and references to variables and functions, not expected results.

Variables panel

The Variables panel of the Calculation Editor lists all variables defined for the project. Double-click a variable to add the variable to the calculation. To sort the variables, click a column heading.

Signals/Channels panel

The Signals/Channels panel of the Calculation Editor lists all the signals and channels defined in the project. Double-click a signal or channel to add the signal or channel to the calculation. The signal or channel that you added appears at the location of your cursor in the Calculation panel .

To sort the signals or channels, click a column heading.

Functions panel

The Functions panel of the Calculation Editor contains a list of defined program functions and operators. Double-click a function to add the function to the calculation.

Click the modify choice lists icon to select specific categories of functions. To sort the functions, click a column heading.

Square brackets [ ] indicate you can add an optional parameter. Remove the square brackets regardless if you add an optional parameter. A calculation error occurs if they are not removed.

Parentheses ( ) in the syntax means that you must add a parameter.

For example, the function below requires two variable parameters that contain the stress and strain data. The start and end indexes for loading and unloading modulus calculations are optional because they are in square brackets.

CycleModulus(Stress,Strain[, startIndexLoading,endIndexLoading, startIndexUnloading, endIndexUnloading])

Note: The variable parameters between the parentheses are not the identities of the variables. They indicate the type of variable that must be placed in the syntax.

Operators and Precedence

The following table shows all the recognized operators organized by group and listed in order of precedence. Within a group, all operators have the same precedence.

Note: Some programming languages use a semicolon as a list separator instead of a comma. If you are using one of those languages, the last operator of the table would be replaced with a semicolon.

Operator Precedence

Precedence	Operator	Function	Direction
1	[ ]	Array Index	Left-to-Right
	( )	Function Call	
	.	Variable-Specific Information	
2	!	Logical NOT	Right-to-Left
	+	Unary Plus	Left-to-Right
	-	Unary Minus	
	~	Ones Complement	
3	*	Multiply	Left-to-Right
	/	Divide	
	%	Modulus	

Precedence	Operator	Function	Direction
4	+	Addition	Left-to-Right
	–	Subtraction	
5	<<	Bitwise Shift Left	Left-to-Right
	>>	Bitwise Shift Right	
6	<	Less Than	Left-to-Right
	>	Greater Than	
	<=	Less than or Equal	
	>=	Greater Than or Equal	
7	==	Equal to	Left-to-Right
	!=	Logical Not	
8	&	Bitwise AND	Left-to-Right
9	^	Bitwise XOR	Left-to-Right
10	|	Bitwise OR	Left-to-Right
11	&&	Logical AND	Left-to-Right
12	||	Logical OR	Left-to-Right
13	=	Assignment	Right-to-Left
14	,	Comma - List Separator	Left-to-Right

Variable-Specific Information

You can reference variables with delimited notation. A dot separates the variable name from the specifier: <variable_name>.<specifier>

For example, the display name of a variable named PeakLoad is <PeakLoad>.<display name>

Specifier Properties

Item	Description
Display Name	Shows the display name of the variable.
Identifier	Shows the internal name of the variable.
Units	Shows the units of the variable.
Size	Shows the array size of the variable. If the variable is not an array, the value is 1.

Functions in Variable Calculations

Functions and Arguments in a Calculation

The Calculation Editor contains functions that can be inserted into the Calculation panel from the Functions panel. When an insertion occurs, the function is represented by an identifier followed immediately by a parenthetical representation of the expected argument or arguments. If multiple arguments are required, their representatives are separated within the parenthesis using delimiters, such as commas or semicolons, depending on the local language. The representative arguments are placeholders for an expected type of data.

Argument types

The types of arguments that can be passed to a function are:

- Number
- String
- Array
- Signal or test input

In all cases, the argument type that is shown in the Calculation window must be replaced by a meaningful value. The meaningful value can be an actual value, a variable that contains a value that is of the correct type for the argument, or another function that produces a result that is of the correct type for the argument.

For example, if the function cos(number) is added to the Calculation panel, the message, “The variable, number, was not found” is shown in the Error panel. The argument “number” must be replaced with a value, a variable of the correct type, or a function that evaluates to the correct type. In this case the number 30, the variable CycleCount, or the function cos(sin(30)) validates the calculation. The Error panel message reads, “Equation is OK” when one of these replacements is used.

Argument syntax

When replacing a representative number or string argument type with an actual number value or a string value, you must use the correct syntax for the value.

- A string value must be enclosed between double-quotation marks. For example: the function ToLower(string), can be replaced by ToLower(“This is a short sentence.”), but not by ToLower(This is a short sentence.).
- A numeric value is typed without quotation marks. For example: the function cos(number) can be replaced by cos(30), but not by cos(“30”).

Arguments can also be replaced with variables of the correct type.

Unique functions

There are unique functions available in the Calculation Editor that do not accept arguments.

- The function e() represents the natural logarithmic base, e.
- The function Pi() represents the mathematical constant, pi.
- The function SystemRate() represents the system data rate.

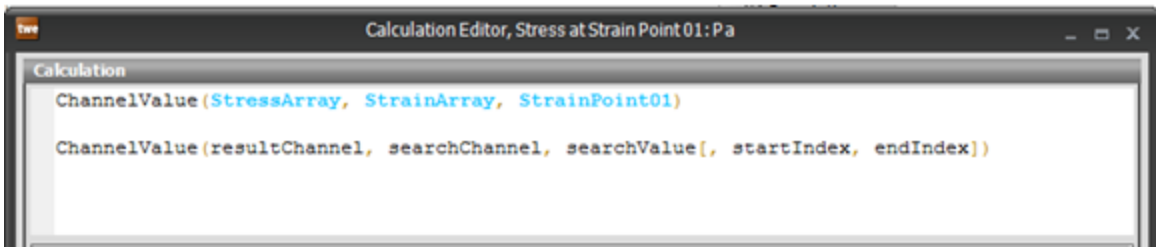
Understanding Functions Used in Calculations

Each argument shown in the Calculation window must be replaced by a meaningful value. The meaningful value can be an actual value, a variable that contains a value that is of the correct type for the argument, or another function that produces a result that is of the correct type for the argument.

To help understand the arguments in a pre-existing variable whose function is already configured with variables, put the cursor at the end of the expression and press Return. Then, use the Functions tab to reinsert the function. When you do this, the same function will be added to the calculation window. However, instead of displaying the actual values or the variable values the calculation is using, you can see the base arguments that the function expects to receive. By doing this, you can better understand which values are being fed into any specific function that is used in a calculation.

In the example below, the Channel Value function was reinserted at the bottom to provide descriptions of the arguments. This makes it apparent that the function searches the StrainArray (searchChannel) for the value of StrainPoint01 (search value) and returns the stress value (resultChannel) at a strain value = StrainPoint01.

Expanded ArrayValue Function



Modulus Calculation Functions

Modulus is calculated from the slope of the Least Squares Fit calculation of the Stress and Strain arrays between the starting and ending indexes.

LeastSquaresFit

LeastSquaresFit(StressArray, StrainArray, Slope1, Slope2

The slope of the least squares fit of the array that contains Y-axis data and X-axis data between the start and end indexes. This is typically used when calculating modulus in TestSuite TW applications.

Loading Modulus

LoadingModulus(Stress, Strain[, startIndex, endIndex])

The loading modulus is the modulus of the portion of the curve where the load on the specimen increases. This function calculates the loading modulus using the Stress and Strain arrays between the startIndex and endIndex.

If the startIndex is set to -1, the optimum start index is the valley offset by 5 points. If variables represent this field in the function and the original value is -1, the variable contains the calculated index at the completion of the function.

If the endIndex is set to -1, you can calculate the optimum endpoint:

1. Locate a nominal end index at 25% of the stress range and within the linear modulus range.
2. Calculate a nominal modulus at the nominal end index.
3. Expand the nominal end index away from the start index until the new modulus value differs from the nominal modulus by more than 2%.
4. The endIndex is set before the newly calculated modulus exceeds the nominal modulus.

If variables represent this field in the function and the original value is -1, the variable contains the calculated index at the completion of the function.

UnLoading Modulus

UnLoadingModulus(Stress, Strain[, startIndex, endIndex])

The unloading modulus is the modulus of the portion of the curve where the load on the specimen decreases. This function calculates the unloading modulus using the Stress and Strain arrays between the startIndex and endIndex.

If the startIndex is set to -1, the optimum starting index is the peak offset by 5 points. If variables represent this field in the function and the original value is -1, the variable contains the calculated index at completion of the function.

If the endIndex is set to -1, you can calculate the optimum endpoint:

1. Locate a nominal end index at 25% of the stress range and within the linear modulus range.
2. Calculate a nominal modulus at the nominal end index.
3. Expand the nominal end index away from the start index until the new modulus value differs from the nominal modulus by more than 2%.
4. The endIndex is set before the newly calculated modulus exceeds the nominal modulus.

If variables represent this field in the function and the original value is -1, the variable contains the calculated index at the completion of the function.

Cycle Modulus

CycleModulus(Stress, Strain[, startIndexLoading, endIndexLoading, startIndexUnloading, endIndexUnloading])

CycleModulus returns the average of LoadingModulus and UnloadingModulus.

First Cycle Modulus

FirstCycleModulus(Stress, Strain[, startIndex, endIndex])

The FirstCycleModulus function returns the modulus of the data in the region prior to the first peak or valley because the first cycle is typically a partial cycle. The modulus is either a loading or unloading modulus, based on an increase or decrease in the specimen load during the first cycle.

Fatigue Life Calculation Function

Fatigue Life

FatigueLife(YAxis, XAxis, startIndex, endIndex, percentageDrop)

The FatigueLife function returns the index of the point where the material under test fails. The failure point is after endIndex where the Y-Axis value falls below (percentageDrop) the Least Squares Fit line of X-Axis and Y-Axis data between startIndex and endIndex. This function provides the drop line for the Failure Cycle Chart.

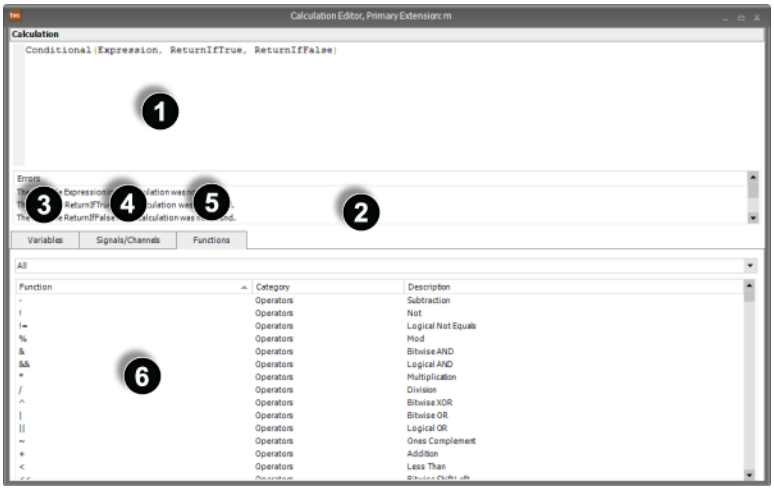
Calculated Variable Functions

Calculation Editor

Access

1. Select the Define tab.
2. Depending on the section of the test in which you want to edit a calculation, select either the **Pre-test**, **Pre-test run**, **Post-test run**, or **Post-test** nodes of the test definition tree.
3. Select the **Calculations** node.
4. Select a calculation from the list.
5. Click  next to the **Calculation** field.

Calculation Editor Window



Parts of the Calculation Editor Window

Number	Item	Description
1	Calculation panel	Use the Calculation panel to build a variable calculation. You can edit text directly in the panel, or you can use the Variables and Functions panels to insert defined variable and functions. Note the following on the Calculation panel: • Variables can use reference signals, other variables, and calculation parameters. • Names are not case-sensitive. • Variables and signal labels have identifiers and display names. Use the identifier name when creating a calculation that uses a signal. **Note:** If adjacent variable names are not separated by a space or function in the Calculation panel, they are interpreted as a single variable name. This is likely to result in a "...variable..., was not found" error message.
2	Errors panel	The Errors panel continually evaluates the calculation as you build it. **Note:** Calculations are validated only on syntax and references to variables and functions, not expected results.
3	Variables tab	The Variables panel lists all variables defined for the project. Double-click a variable or click a variable and click **Insert** to add the variable to the calculation. The cursor location in the Calculation panel shows the added variable. To sort the displayed variables, click a column heading to sort the variables by the values in that column. Click the same column heading again to reverse the sort order.

Number	Item	Description
4	Signal/Channels tab	Opens the **Signals/Channels** tab.
5	Function tab	Opens the Functions panel.
6	Functions panel	The Functions panel contains a list of defined program functions and operators. Double-click a function or click a function and then click **Insert** to add the function to the calculation. Click the **Functions** list to select specific categories of functions. To sort the displayed functions, click a column heading to sort the functions by the values in that column. Click the same column heading again to reverse the sort order. Square brackets [ ] indicate you can add an optional parameter. Remove the square brackets regardless if you add an optional parameter. A calculation error occurs if they are not removed. Parentheses ( ) in the syntax means that you must add a parameter. For example, the following function requires two variable parameters that contain the stress and strain data. The start and end indexes for loading and unloading modulus calculations are optional because they are in square brackets: **CycleModulus(Stress,Strain[, startIndexLoading,endIndexLoading, startIndexUnloading, endIndexUnloading])** **Note:** The variable parameters between the parentheses are not the identities of the variables. They indicate the type of variable that must be placed in the syntax.

Calculation Unit Conversion Issues

Important: To avoid unit-conversion issues, create a variable to hold any constant value (that has a dimension and units associated with it) used in a calculation. You can assign any units that you want to the variable and the application converts them to the base units when the calculation is performed.

Base units

Although MTS TestSuite applications allow the user to specify variable units, all calculations convert variable values and constants to a constant set of base units. This can be problematic if you add a constant to an equation and assume that it will use the same user-assigned units as the other variables in the calculation.

Note: When using the Calculation Editor, you can move the mouse over any variable in the calculation to display a tool tip that specifies the base units used when that calculation is performed.

Example

For example, the following calculation would yield incorrect results because the test designer has entered a constant that they assumed would match the user-assigned units.

y = x +(1)

where: the variable (x) has user-assigned units of feet and the constant (1) is assumed to also have the same units of feet.

If the value of x=3 feet the assumption is that:

y = 3 ft. + 1 ft. = 4 ft. where: y is a variable with a dimension = “length” and units = feet

But, because the constant (1) is converted to the application’s base units for length (meters in this example), the equation returns the following:

y = 0.9144 + 1 = 1.9144 so that: the displayed value of y = 6.281 ft. (not 4 ft.)

Notation for Referencing Variables

You can reference variables with delimited notation. A dot separates the variable name from the specifier:

<variable_name>.<specifier>

For example, the display name of a variable named PeakLoad is:

<PeakLoad>.<display name>

Specifier Properties

Item	Description
Display Name	Shows the display name of the variable.
Identifier	Shows the internal name of the variable.
Units	Shows the units of the variable.
Size	Shows the array size of the variable. If the variable is not an array, the value is 1.

Adding Pre-Test Calculations

Access

Define tab > **Test flow** > **Pre-test** > **Calculations**

Pre-test calculations are calculations that pertain to Common (global) variables. Common variables are variables associated with the entire test, not individual test runs.

A common use for pre-test calculations is concatenating strings to name the file, the test, and the test run.

Example: Suppose you want the **Test Name** variable, which is used as the file name and shown in test reports, to be a concatenation of the lot number and the test date. To do this:

1. Create a new test.

   Click the **New Test from Template** button. Select the Simplified Tension test.

2. Add two variables to accept lot number and test date data.
   A. In the Explorer window, select **Variables**.
   B. On the Variables tab, click the **+** sign to add a variable.
   C. In the New Variable window, enter **LotNumber**. Click **OK**.
   D. In the Properties window:
      a. Change **Display Name** to **Enter lot number**.
      b. Change **Type** to **Text**.
   E. In the **Variables** tab, click the **+** sign to add a variable.
   F. In the New Variable window, enter **TestDate**. Click **OK**.
   G. In the Properties window:
      a. Change Display Name to **Enter test date**.
      b. Change **Type** to **Text**.

3. Make the **Name of the Test** variable calculated.
   A. In the Explorer window, click **Variables**.
   B. On the **Variables** tab, click **Name of the Test**.
   C. In the Properties window, click to select **Is Calculated**.
   D. Click the ellipsis (**...**) in the **Calculation** tab.
   E. In the Calculation Editor, select **Enter lot number** from the list, enter **+"_"+**, and then select **Enter date** from the list. The resulting calculation is LotNumber+"_ "+TestDate. Click **OK**.
   F. In the Properties window **Availability** tab, click to select **During-Test** and **Result**.

4. Enable the **Input Variables** and **Calculate Variables** activities in the Procedure Flow and add the appropriate variables to them.
   A. Navigate to the Procedure Table by selecting the **Define** and **Procedure** tabs.
   B. Click the **Input Variables** icon in the **Set Up** group of the Procedure Flow.
   C. In the Properties window, click **Enable**.
   D. On the **Variable List** tab, click **+** to add a variable.
   E. In the Variables Selection window, move **Enter lot number** and **Enter test date** from **Available** to **Selected**. Click **OK**.
   F. Click the **Calculate Variables** activity in the Set Up group of the Procedure Flow.

G. In the Properties window, click **Enable**.

H. On the **Variable List** tab, click **+** to add a variable.

I. In the Variables Selection window, move **Name of the Test** from **Available** to **Selected**. Click **OK**.

5. Run the test. Notice that the file name changes and that the concatenated Test Name shows up in the **Review** tab.

Important: The calculation for this example will occur only the first time you run the program. This is because the **Input Variables** and **Calculation Variables** activities are in the Set Up Group. To perform calculations for each test run, use the Run group for the activities instead.

Tip: The input variables used to make up the test name could be selected from choice lists.

Calculation Functions Overview

The functions available in the Calculation Editor are tools for calculating the value of a variable. The functions support computations using arithmetic, logic, and project variables.

If a function has mandatory parameters, a corresponding value for the parameters must be supplied. The value of a parameter is referred to as an argument. The number of parameters of a function, along with the parameter names form the signature of a function. The function signature describes the parameters and parameter types with which to make a legal call (that is, properly use) to the function. The signature contains the name of the function, its parameters and their type, and the return value.

Functions can operate directly on an operand are referred to as operators. Functions may require one or more arguments be passed to them, or may not accept arguments.

Operators

When an operator is added to a calculation, it acts directly on the operands to its left and right by assigning a value, performing a comparison, or performing a mathematical calculation.

However, some operators like the Not (!), Ones Complement (~), and Subtraction when used as a Negate (-), operate on only one argument to the right.

Functions requiring arguments

In general syntax, functions that require arguments are written as a function name followed immediately by the required arguments enclosed in parenthesis. For example:

cos(number)

where “cos” is the function identifier and “(number)” is the required argument.

Functions with no arguments

There are unique functions available in the Calculation Editor that do not accept arguments:

- Function e() represents the natural logarithmic base, e.
- Function Pi() represents the mathematical constant, pi.
- Function SystemRate() represents the system data rate.

Inserting Functions with the Calculation Editor

The Calculation Editor contains functions that you can insert into the Calculation panel from the Functions panel. When you insert a function, the function is represented by an identifier followed immediately by a parenthetical representation of the expected argument or arguments. If multiple arguments are required, their representatives are separated within the parenthesis using delimiters, such as commas or semicolons, depending on the local language. The representative arguments are placeholders for an expected type of data.

Argument types

The types of arguments that can be passed to a function are:

- Number
- String
- Array
- Signal or test input

In all cases, the argument type that is shown in the Calculation window must be replaced by a meaningful value. The meaningful value can be an actual value, a variable that contains a value that is of the correct type for the argument, or another function that produces a result that is of the correct type for the argument.

For example, if the function cos(number) is added to the Calculation panel, the message, “The variable, number, was not found” is shown in the Error panel. The argument “number” must be replaced with a value, a variable of the correct type, or a function that evaluates to the correct type. In this case, the number 30, the variable CycleCount, or the function cos(sin(30)) validates the calculation. The Error panel message reads, “Equation is OK” when one of these replacements is used.

Argument syntax

When replacing a representative number or string argument type with an actual number value or a string value, you must use the correct syntax for the value.

- A string value must be enclosed between double-quotes. For example: the function ToLower (string), can be replaced by ToLower(“This is a short sentence.”), but not by ToLower(This is a short sentence.).
- A numeric value is typed without quotes. For example: the function cos(number) can be replaced by cos(30), but not by cos(“30”).

Arguments can also be replaced with variables of the correct type.

Function Categories Overview

MTS TestSuite functions available in the Calculation Editor are organized into the following categories:

- Array
- Controller
- Cyclic
- Date and Time
- Directory
- Fatigue and Fracture
- Geometry
- Index
- Math
- Operators
- Peel-Tear
- Sensor (applicable to TEDS Devices in TWE only)
- String
- Test Definition (custom user-defined functions)

The most commonly used functions are described in this reference.

Array Functions

This section provides reference information for functions used on arrays.

Compliance

The Compliance function is used to minimize the effects of frame deflections under high force.

Returns

The amount of the extension that needs to be compensated.

Syntax

Compliance(ForceArray, ComplianceCoefficients)

Parameters

ForceArray - The force array data.

ComplianceCoefficients - The coefficient array generated during the compliance Test Run.

Unit Class

Extension

Example

ExtensionArray - Compliance(ForceArray, ComplianceCoefficients)

CurveFitValue

Returns
The requested coefficient.

Syntax
CurveFitValue(Order, Coefficient, yArray, aArray, StartIndex, EndIndex)

Parameters
Order—The order of the polynomial fit to calculate using the yArray and xArray data.

Coefficient—The coefficient to return.

yArray—The Y array data.

xArray—The X array data.

StartIndex—The optional start index of the arrays.

EndIndex—The optional end index of the arrays.

Unit Class
Dependent upon equation order, coefficient number, and channels specified.

Example
CurveFitValue(2, 0, yArray, aArray, StartIndex, EndIndex)

This example returns the zero coefficient from the second order polynomial fit on the X and Y arrays between the indexes specified by StartIndex and EndIndex.

Polynomial
The Polynomial function returns the Y value based on an X value and the coefficients generated with the PolynomialFit function.

Returns
The Y value associated with the X value.

Syntax
Polynomial(X,Coefficients)

Unit Class
The units associated with the Y data used to calculated the coefficients.

Example
Polynomial(X, coefficients)

This example returns the Y value calculated at the value X.

PolynomialFit
The PolynomialFit function fits a polynomial curve of the specified order to the data arrays.

Returns
The coefficients as an array. The number of array values returned is one plus the order specified.

Syntax
PolynomialFit(yArray, xArray, Order[, StartIndex, EndIndex, rSquared])

Parameters
yArray—The Y array data being analyzed.

xArray—The X array data being analyzed.

Order—The order of the polynomial fit.

StartIndex—The optional start index of the region to be analyzed.

EndIndex—The optional end index of the region to be analyzed.

rSquared—An optional variable that is used to pass the coefficient of determination (R2)

Unit Class
Dependent on the coefficient.

Example
PolynomialFit(Force, Extension, 1, Slope1, Slope2)

This example returns the coefficients for a straight line through the regions specified by Slope1 and Slope2.

Controller Functions
This section provides reference information for controller functions.

Note:
The TimePattern depends upon your Windows settings in the **Format** tab for regional areas.

GetTransitionTime

Returns
Returns the time from the specified channel and transition time type.

Syntax
GetTransitionTime([timePattern])

Parameters
timePattern

SetTransitionTime

Returns
Sets the time from the specified channel and transition time type.

Syntax
SetTransitionTime([timePattern])

Parameters
timePattern

Signal

Returns
Current value from a signal.

Syntax
Signal(SignalName)

Parameters
SignalName

SignalFullScale

Returns
Full scale value from a signal.

Syntax
SignalFullScale(SignalName[, MinimumOrMaximumOption)

Parameters
SignalName

MinimumOrMaximumOption—One of the optional arguments: Minimum; minimum; Maximum; maximum; Upper; upper; Lower; or lower

SystemRate

Returns
The maximum data rate for the controller.

Syntax
SystemRate()

Trace

Returns
Gets the percentage complete of the current segment for a channel.

Syntax
Trace(ChannelName)

Parameters
ChannelName

Cyclic Functions

This section provides reference information for cyclic functions.

AnalysisRun

Returns

Accesses the variable across analysis runs.

Syntax

AnalysisRun[Index].variable

Block

Returns

Accesses data across blocks.

Syntax

Block[blockName, Index].variable

Parameters

blockName

Cycle

Returns

Accesses the variable in cycles.

Syntax

Cycle[Index].variable

TestRun

Returns

Accesses the variable across test runs.

Syntax

TestRun[Index].variable

Date and Time Functions

This section provides reference information for date and time functions.

The DatePattern and TimePattern depend upon your Windows settings in the Format tab for your region.

TestCreationDate

Returns

Returns the creation date of the test.

Syntax
TestCreationDate([datePattern])

Parameters
datePattern—Formats the date according to the specified pattern. For example, “d” is a short date pattern (MM/dd/yyyy); “D” is a long date pattern (dddd, MMMM dd, yyyy).

Detailed date and time pattern format specifier information is available from the MSDN resources Web site.

Examples
TestCreationDate([“d”**])**

Returns a test creation date in a short date format, assuming a United States English culture:

4/15/2012

TestCreationDate([“D”**])**

Returns a test creation date in a long date format, assuming a United States English culture:

Saturday, March 31, 2012

TestCreationTime

Returns
Returns the creation time of the test.

Syntax
TestCreationTime([timePattern])

Parameters
timePattern—Formats the time according to the specified pattern. Set the time pattern to the exact format desired. For example, to obtain the pattern h-mm-ss for hours-minutes-seconds, the function should specify "hh-mm-ss".

Detailed date and time pattern format specifier information is available from the MSDN resources Web site.

Example
TestCreationTime([“hh:mm:ss tt”**])**

Displays the time as 10:04:01 PM.

Unit Class
String

TestModificationDate

Note:
This function replaces the SampleModificationDate function from the TW4 application.

Returns
Returns the last modification date of the test run.

Syntax
TestModificationDate([datePattern])

Parameters
datePattern—Formats the date according to the specified pattern. For example, “d” is a short date pattern (MM/dd/yyyy); “D” is a long date pattern (dddd, MMMM dd, yyyy).

Detailed date and time pattern format specifier information is available from the MSDN resources Web site.

Examples
TestModificationDate([“d”**])**

Returns a test modification date in a short date format, assuming a United States English culture (us en):

4/15/2012

TestModificationDate([“D”**])**

Returns a test creation date in a long date format, assuming a United States English culture:

Saturday, March 31, 2012

Unit Class
Integer

TestModificationTime

Note:
This function replaces the SampleModificationTime function from the TW4 application.

Returns
Returns the last modification time of the test run.

Syntax
TestModificationTime([timePattern])

Parameters
timePattern—Formats the time according to the specified pattern. Set the time pattern to the exact format desired. For example, to obtain the pattern h-mm-ss for hours-minutes-seconds, the function should specify "hh-mm-ss".

Detailed date and time pattern format specifier information is available from the MSDN resources Web site.

Example
TestModificationTime([“hh:mm:ss tt”**])**

Displays the time as 10:04:01 PM.

Unit Class
String

TestRunCreationDate

Note:
This function replaces the SampleCreationDate function from the TW4 application.

Returns
Returns the creation date of the test run.

Syntax
TestRunCreationDate([datePattern])

Parameters
datePattern—Formats the date according to the specified pattern. For example, “d” is a short date pattern (MM/dd/yyyy); “D” is a long date pattern (dddd, MMMM dd, yyyy).

Detailed date and time pattern format specifier information is available from the MSDN resources Web site.

Examples
TestRunCreationDate([“d”**])**

Returns a test run creation date in a short date format, assuming a United States English culture (us en):

4/15/2012

TestRunCreationDate([“D”**])**

Returns a test run creation date in a long date format, assuming a United States English culture:

Saturday, March 31, 2012

Unit Class
String

TestRunCreationTime

Note:
This function replaces the SampleCreationTime function from the TW4 application.

Returns
Returns the creation time of the test run.

Syntax
TestRunCreationTime([timePattern])

Parameters
timePattern—Formats the time according to the specified pattern. Set the time pattern to the exact format desired. For example, to obtain the pattern h-mm-ss for hours-minutes-seconds, the function should specify "hh-mm-ss".

Detailed date and time pattern format specifier information is available from the MSDN resources Web site.

Example
TestRunCreationTime([“hh:mm:ss tt”**])**

Displays the time as 10:04:01 PM.

Unit Class
String

Directory Functions
This section provides reference information for directory functions that return information about the default project, test, test run, external files, and data export directories that are configured in Preferences > Configuration > Project.

CurrentProjectDirectory

Returns
Returns the current project directory.

Syntax
CurrentProjectDirectory()

Unit Class
String

CurrentTestDirectory

Returns
Returns the current test directory.

Syntax
CurrentTestDirectory()

Unit Class
String

CurrentTestRunDirectory

Returns
Returns the current test run directory. The directory is set in Preferences > Configuration > Project.

Syntax
CurrentTestRunDirectory()

Unit Class
String

DataExportDirectory

Returns
Returns the directory for exported data. The directory is set in Preferences > Configuration > Project.

Syntax
DataExportDirectory()

Unit Class
String

DefaultReportDirectory

Returns
Returns the default reports directory. The directory is set in Preferences > Configuration > Project.

Syntax
DefaultReportDirectory()

Unit Class
String

ExternalFilesDirectory

Returns
Returns the directory for external files. The directory is set in Preferences > Configuration > Project.

Syntax
ExternalFilesDirectory()

Unit Class
String

TestDirectory

Returns
Returns the directory for the test. The directory is set in Preferences > Configuration > Project.

Syntax
TestDirectory()

Unit Class
String

Fatigue and Fracture Functions

This section provides reference information about the functions used for Fatigue and Fracture applications.

CalcInelStrain

Calculated Inelastic Strain.

Returns

The CalcInelStrain function returns the value:

Strain - Stress/Modulus

The calculation provides the value of the Strain minus the Stress divided by the modulus. If Strain and Stress are arrays, the result is an array. If Strain and Stress are single numbers, the result is a single number.

Syntax

CalcInelStrain(Stress, Strain, Modulus)

Parameters

Stress

Strain

Modulus

HysteresisArea

Returns

The HysteresisArea function returns the area under the curve defined by the Stress and Strain array data.

Syntax

HysteresisArea(StressArray, StrainArray)

Parameters

StressArray

StrainArray

MeasInelasticStrainMax

Measured Inelastic Strain Maximum. The measured inelastic strain range is specific to materials fatigue testing, especially Low Cycle Fatigue (LCF). The application calculates the mean stress from the stress and strain data that represents one cycle. The two points where the mean intersects the curve determines the strain values. The MeasInelStrainMax function returns the maximum of these two strain values. The MeasInelStrainMin function returns the minimum of these two strain values.

Returns

Returns the maximum of the two strain values as described above.

Syntax
MeasInelStrainMax(stressVar, strainVar)

Parameters
stressVar

strainVar

MeasInelasticStrainMin
Measured Inelastic Strain Minimum. The measured inelastic strain range is specific to materials fatigue testing, especially Low Cycle Fatigue (LCF). The application calculates the mean stress from the stress and strain data that represents one cycle. The two points where the mean intersects the curve determines the strain values. The MeasInelStrainMax function returns the maximum of these two strain values. The MeasInelStrainMin function returns the minimum of these two strain values.

Syntax
MeasInelStrainMin(stressVar, strainVar)

Returns
The MeasInelStrainMin function returns the minimum of the two strain values.

Parameters
stressVar

strainVar

StrainA
StrainA(StrainMax, StrainMin)

The StrainA function returns the value:

((StrainMax - StrainMin)/(StrainMax + StrainMin))

StrainR
StrainR(StrainMax, StrainMin)

The StrainR function returns the value:

StrainMin/StrainMax

StressA
StressA(StressMax, StressMin)

StressR
StressR(StressMax, StressMin)

The StressR function returns the value:

StressMin/StressMax

Index Functions

This section contains reference information about index functions.

Understanding Index Functions

Index functions are used to find array index values. All array variables collected by a data acquisition activity share the same index values. Once you create a variable that stores an index value, you can use that variable in an Array Value function to return a value at that index. For example, to find the strain value at peak load, you would use the following variable calculations:

PeakLoadStrainValue and Peak Calculations

Identifier	Display Name	Calculation	Unit
PeakLoadStrainValue	Peak Load StrainValue	ArrayValueAtIndex(StrainArray, Peak)	(in/in)
Peak	Peak Index	PeakIndex(_LoadArray)	(count)

Note:
The “Peak” variable stores the index value for peak load. The “PeakLoadStrainValue” variable uses the ArrayValueAtIndex function to calculate and store the strain value at the peak-load index.

Determining Analysis Region

The “min load”, “max load”, and “% strain point arguments” are used to determine the bounding analysis region for the algorithm. On a successful completion of this function, the index point will be contained within the bounds of the analysis region. The analysis region is used to focus on important regions of data and skip anomalous regions. If neither the starting or ending regions are located, all of the data in the channels will be analyzed.

Starting Point of Region

To determine the starting point of the analysis region, the algorithm will select the first point where the “min load” value is located in the y-channel. If the “min load” is not found, the algorithm will attempt to locate the point where “% strain point” is located in the x-channel. If this point cannot be determined, the first data point in the channel is used.

Ending Point of Region

To determine the ending point of the analysis region, the algorithm will select the first point where the “max load” value is located in the y-channel data. The search will either begin at the first data point or the point where the “% strain point” was located in the x-channel. If the “max load” is not located, the end point of the region is assigned to the point associated with the peak value on the y-channel. If all else fails to locate the end point, the index of the last point of data in the channel is selected.

BreakIndexByDropFromPeak

The BreakIndexByDropFromPeak function is used to calculate the break point using a specified drop from the peak in the array data.

Returns

This function returns the index point in the array where the break point was detected.

Syntax
BreakIndexByDropFromPeak(channel, dropValue, [startIndex], [endIndex])

Parameters

BreakIndexByDropFromPeak Parameters

Parameter	Description
Channel	The array data being analyzed for the break point.
DropValue	The percentage drop from the peak that signifies a break occurred.
StartIndex	The optional starting point in the array. The starting point in the array is used if this parameter is not provided.
EndIndex	The optional ending point in the array. The last point in the array is used if this parameter is not provided.

Unit Class
Integer

Example
A test contains the input BreakDrop that is equal to 80%.

BreakIndexByDropFromPeak(_Load, BreakDrop)

This formula:

1. Searches the _Load channel.
2. Looks for the _Load channel to drop 80% from its peak value. 80% is the value of the input BreakDrop.

If PeakLoad = 100lbs, then the calculation returns the number of the data point when the load has dropped to 20lbs.

If PeakLoad = 80lbs, then the calculation returns the number of the data point when the load has dropped to 16lbs.

BreakIndexByDropPerExt
The BreakIndexByDropPerExt function is used to calculate the break point using a specified drop from the peak in the array data that occurs over a predefined change in the extension data.

Returns
This function returns the index point in the array where the break point was detected.

Syntax
BreakIndexByDropPerExt(dropChannel, dropValue, extensionChannel, extensionValue, [startIndex], [endIndex])

Parameters

Chart Descriptions

Parameter	Description
DropChannel	The array data being analyzed for the break point.
DropValue	The percentage drop from the peak that signifies a break occurred.
ExtensionChannel	The extension data being analyzed for the change in extension.
ExtensionValue	The change in extension value required for the break to be determined.
StartIndex	The optional starting point in the array. The starting point in the array is used if this parameter is not provided.
EndIndex	The optional ending point in the array. The last point in the array is used if this parameter is not provided.

Unit Class
Integer

Example
A test contains the input BreakDrop that is equal to 80% and the input. It also contains the input BreakElongation that is equal to 0.1in.

BreakIndexByDropPerExt (_Load, BreakDrop, PrimaryExt,BreakElongation)

This formula:

1. Searches the _Load channel and the PrimaryExtension channel.
2. Looks for the _Load channel to drop 80% from its peak value, and for the PrimaryExt channel to change by 0.1 in.

If PeakLoad = 100lbs, then the calculation returns the number of the data point when the load has dropped to 20lbs and the PrimaryExt channel has changed by 0.1 in.

If PeakLoad = 80lbs, then the calculation returns the number of the data point when the load has dropped to 16lbs and the PrimaryExt channel has changed by 0.1 in.

ArrayIndex
The ArrayIndex function is used to locate the index of the data in the array that is closest in value to the value being searched.

Returns
This function returns the index point in the array of the closest value.

Syntax
ArrayIndex (searchArray, searchValue,[startIndex], [endIndex])

Parameters
SearchArray—The array data being analyzed.

SearchValue—The value being search for in the array.

StartIndex—The optional starting point in the array. The starting point in the array is used if this parameter is not provided.

EndIndex—The optional ending point in the array. The last point in the array is used if this parameter is not provided.

Unit Class
Integer

Example
A test contains the input StrainPoint1 that is equal to 10%.

ArrayIndex (Strain, StrainPoint1)

This formula:

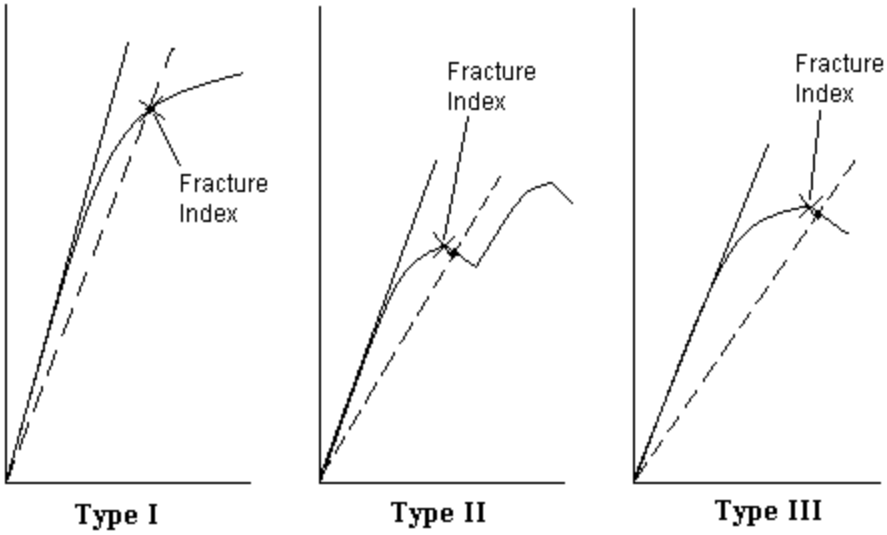
1. Searches the Strain channel.
2. Locates the data point whose value is closest to 10% strain.

If data point #100 has a strain value of 9.8% and data point #101 has a strain value of 10.1%, the calculation returns the value #101 because this data point is closest to the desired value.

FractureIndex
The FractureIndex function finds the fracture index. A line is drawn at a percentage (the fracture percent) of the slope of the X-Y curve.

Returns
The fracture index is where this line intersects the curve (Type I) or where it intersects a peak if it comes before this point (Type II or Type III).

Fracture Index Examples



Syntax

FractureIndex(fracture percent, slack value, Load Channel, Primary Extension channel, slope 1 index, slope 2 index)

Parameters

Fracture percent—Determines the slope at which the fracture line is drawn. This slope is a percentage of the slope of the X-Y curve.

Slack value—The offset used for calculating slack extension. If slack compensation is not used then this argument should be zero (0).

Load channel—The channel used for the Y-axis of the curve. This should almost always be the Load channel.

Primary Extension channel—(PrimaryExtension) The channel used for the X-axis of the curve. This should almost always be the PrimaryExtension channel.

Unit Class

Integer

Example

The following formula will calculate the fracture index using a 95% fracture percent and assumes that slack compensation is turned off.

FractureIndex(0.95, 0, _Load, PrimaryExt, Slope1, Slope2

LowerYieldIndex

This LowerYieldIndex function is used to calculate the first occurrence of a zero slope after the upper yield point.

1. The search for the Lower Yield Point begins after the upper yield index point.
2. The force must drop from the upper yield point by the listed tolerance before the lower yield index can be identified.
3. From this point, the algorithm keeps track of the minimum load point, and stops searching when the load value increases by the same amount as the drop from the upper yield index.
4. The Lower Yield index is reported as the minimum load point found during the search.

Returns
This function returns the index point in the array of the lower yield point.

Syntax
LowerYieldIndex(y-axis, tolerance, upperYieldIndex)

Parameters
y-axis—The force array data being analyzed for the zero slope.

Tolerance—The percentage drop in force from the upper yield index that must occur before a lower yield index can be identified.

UpperYieldIndex—The index representing the location of the upper yield index.

Unit Class
Integer

MaxSlopeStartIndex
The MaxSlopeStartIndex and MaxSlopeEndIndex functions are used to determine a region of data where the slope of the line is minimized based on the function arguments. The slope of the line is determined using a Least Squares Fit on the Y and X channel data.

Returns
The MaxSlopeStartIndex function returns the last point of the region. If a valid region cannot be located given the function arguments, a –1 will be returned as the index.

Syntax
MaxSlopeStartIndex (y-channel, x-channel, % segment length, % tolerance, min load, max load, %strain point)

Parameters
y-channel—This argument is the channel that represents the Y-axis for the Least Squares fit. The units for this channel are normally in Force units but it is not a requirement. The unit class of the min load and max load arguments must be the same as the unit class for this argument.

x-channel—This argument is the channel that represents the X-axis for the Least Squares fit. The units for this channel are normally in Strain units but it is not a requirement. The unit class of the % strain point argument must be the same as the unit class for this argument.

% segment length—This argument, which is a percentage, is multiplied by the value determined as the endpoint index of the analysis region. This number is not necessarily the total number of points in the range or in the specimen.

% tolerance—This argument is a comparison factor used to expand the region initially selected by the algorithm. The initial region identified as having the maximum slope is first determined. The region of data considered is expanded in small increments while the slope of the new line stays within the tolerance based on the initial region. A value of 100% will not allow the region to expand. A value of 90% will allow expansion while the slope of the data region is within 90% of the initially determined region.

min load—The argument is used to determine the start of the analysis region. The unit class of this argument must be the same one used with the y-channel.

max load—The argument is used to determine the start of the analysis region. The unit class of this argument must be the same one used with the y-channel.

% strain point—The argument is used to determine the start of the analysis region. The unit class of this argument must be the same one used with the x-channel.

Note: For more information about how the analysis region is determined, see "Determining Analysis Region" on page 191.

Unit Class

Integer

MaxSlopeEndIndex

The MaxSlopeStartIndex and MaxSlopeEndIndex functions are used to determine a region of data where the slope of the line is minimized based on the function arguments. The slope of the line is determined using a Least Squares Fit on the Y and X channel data.

Returns

The MaxSlopeEndIndex function returns the last point of the region. If a valid region cannot be located given the function arguments, a –1 will be returned as the index.

Syntax

MaxSlopeEndIndex (y-channel, x-channel, % segment length, % tolerance, min load, max load, %strain point)

Parameters

y-channel—This argument is the channel that represents the Y-axis for the Least Squares fit. The units for this channel are normally in Force units but it is not a requirement. The unit class of the min load and max load arguments must be the same as the unit class for this argument.

x-channel—This argument is the channel that represents the X-axis for the Least Squares fit. The units for this channel are normally in Strain units but it is not a requirement. The unit class of the % strain point argument must be the same as the unit class for this argument.

% segment length—This argument, which is a percentage, is multiplied by the value determined as the endpoint index of the analysis region. This number is not necessarily the total number of points in the range or in the specimen.

% tolerance—This argument is a comparison factor used to expand the region initially selected by the algorithm. The initial region identified as having the maximum slope is first determined. The region of data considered is expanded in small increments while the slope of the new line stays within the tolerance based on the initial region. A value of 100% will not allow the region to expand. A value of 90% will allow expansion while the slope of the data region is within 90% of the initially determined region.

min load—The argument is used to determine the start of the analysis region. The unit class of this argument must be the same one used with the y-channel.

max load—The argument is used to determine the start of the analysis region. The unit class of this argument must be the same one used with the y-channel.

% strain point—The argument is used to determine the start of the analysis region. The unit class of this argument must be the same one used with the x-channel.

Note: For more information about how the analysis region is determined, see "Determining Analysis Region" on page 191.

Unit Class

Integer

MinSlopeStartIndex

The MinSlopeStartIndex and MinSlopeEndIndex functions are used to determine a region of data where the slope of the line is minimized based on the function arguments. The slope of the line is determined using a Least Squares Fit on the Y and X channel data.

Returns

The MinSlopeStartIndex function returns the first point of the region. If a valid region cannot be located given the function arguments, a –1 will be returned as the index.

Syntax

MinSlopeStartIndex (y-channel, x-channel, % segment length, % tolerance, min load, max load, %strain point)

Parameters

y-channel—This argument is the channel that represents the Y-axis for the Least Squares fit. The units for this channel are normally in Force units but it is not a requirement. The unit class of the min load and max load arguments must be the same as the unit class for this argument.

x-channel—This argument is the channel that represents the X-axis for the Least Squares fit. The units for this channel are normally in Strain units but it is not a requirement. The unit class of the % strain point argument must be the same as the unit class for this argument.

% segment length—This argument, which is a percentage, is multiplied by the value determined as the endpoint index of the analysis region. This number is not necessarily the total number of points in the range or in the specimen.

% tolerance—This argument is a comparison factor used to expand the region initially selected by the algorithm. The initial region identified as having the minimum slope is first determined. The region of data considered is expanded in small increments while the slope of the new line stays within the tolerance based on the initial region. A value of 100% will not allow the region to expand. A value of 90% will allow expansion while the slope of the data region is within 90% of the initially determined region.

min load—The argument is used to determine the start of the analysis region. The unit class of this argument must be the same one used with the y-channel.

max load—The argument is used to determine the start of the analysis region. The unit class of this argument must be the same one used with the y-channel.

% strain point—The argument is used to determine the start of the analysis region. The unit class of this argument must be the same one used with the x-channel.

Note: For more information about how the analysis region is determined, see "Determining Analysis Region" on page 191.

Unit Class
Integer

MinSlopeEndIndex
The MinSlopeStartIndex and MinSlopeEndIndex functions are used to determine a region of data where the slope of the line is minimized based on the function arguments. The slope of the line is determined using a Least Squares Fit on the Y and X channel data.

Returns
The MinSlopeEndIndex function returns the last point of the region. If a valid region cannot be located given the function arguments, a –1 will be returned as the index.

Syntax
MinSlopeEndIndex (y-channel, x-channel, % segment length, % tolerance, min load, max load, %strain point)

Parameters
y-channel - This argument is the channel that represents the Y-axis for the Least Squares fit. The units for this channel are normally in Force units but it is not a requirement. The unit class of the min load and max load arguments must be the same as the unit class for this argument.

x-channel – This argument is the channel that represents the X-axis for the Least Squares fit. The units for this channel are normally in Strain units but it is not a requirement. The unit class of the % strain point argument must be the same as the unit class for this argument.

% segment length – This argument, which is a percentage, is multiplied by the value determined as the endpoint index of the analysis region. This number is not necessarily the total number of points in the range or in the specimen.

% tolerance – This argument is a comparison factor used to expand the region initially selected by the algorithm. The initial region identified as having the minimum slope is first determined. The region of data considered is expanded in small increments while the slope of the new line stays within the tolerance based on the initial region. A value of 100% will not allow the region to expand. A value of 90% will allow expansion while the slope of the data region is within 90% of the initially determined region.

min load – The argument is used to determine the start of the analysis region. The unit class of this argument must be the same one used with the y-channel.

max load – The argument is used to determine the start of the analysis region. The unit class of this argument must be the same one used with the y-channel.

% strain point – The argument is used to determine the start of the analysis region. The unit class of this argument must be the same one used with the x-channel.

Note:
For more information about how the analysis region is determined, see "Determining Analysis Region" on page 191.

Unit Class
Integer

OffsetYieldIndex
The OffsetYieldIndex function is used to calculate the index of the data where the X-Y curve intersects with a line offset from the modulus line by a defined X-Axis offset.

Returns
The index of the first point past where the line would intersect with the X-Y curve.

Syntax
OffsetYieldIndex(y-axis, x-axis, slope1Index, slope2Index, offset[, endIndex])

Parameters
y-axis - The Y-Axis data.

x-axis - The X-Axis data.

Slope1Index - The start of the region of peak slope (modulus line).

Slope2Index - The end of the region of peak slope (modulus line).

Offset - The offset from the modulus line to use in the calculation.

EndIndex - An optional end index that is used to limit the search region. If this parameter is missing, the last data point in the array is used as the end.

Unit Class
Integer

PeakIndex
The PeakIndex function locates the index of the maximum value in the array.

Returns
The index of the array associated with the maximum value of the array.

Syntax
PeakIndex(channel[, startIndex, endIndex])

Parameters
Channel - The array data used in the function.

StartIndex - The optional starting index of the region to evaluate.

EndIndex - The optional ending index of the region to evaluate.

Unit Class
Integer

PeakSlopeIndex
The PeakSlopeIndex function locates the region of peak slope of the X-Y curve.

Returns
This function returns the start or end index of the region of the peak slope of the X-Y curve.

Syntax
PeakSlopeIndex(markerNumber, y-axis, x-axis, segmentLength[, startIndex, endIndex, tolerance])

Parameters
MarkerNumber - 1 returns the start index; 2 returns the end index.

y-axis - The Y-Axis array data.

x-axis - The X-Axis array data.

SegmentLength - The percentage of the peak Y-Axis used to determine the regions used in the slope calculations.

StartIndex - The optional start index of the region. If this parameter is missing, the start is associated with the first data point in the array.

EndIndex - The optional end index of the region. If this parameter is missing, the end is associated with the end of the array data.

Tolerance - The percentage of the slope used to optimize the maximum slope region. The region can be optimized by reducing the region and rechecking the slope. The optimization step will be terminated if the new slope is outside of the tolerance band of the original peak slope. If this parameter is missing, no additional optimization is used beyond locating the maximum region.

Unit Class
Integer

RuntimeOffsetYieldIndex
The RuntimeOffsetYieldIndex function determines when an Offset Yield is reached in a test.

Example: You can use this function to switch test speeds after the offset yield points have been reached. Basically, you start by using one GoTo activity with a slower speed until the offset yield has been reached. After the offset yield has been reached, you can use a second GoTo activity running at a faster speed.

Using the RuntimeOffsetYieldIndex function, you can calculate the offset yield during a test and determine when it has been reached. To do this, add a variable using this calculation to the first GoTo activity. Then, configure the limit detection for the first GoTo activity to monitor this variable and cause the first GoTo activity to move to the second GoTo activity when the offset yield point is reached.

Returns
The index of the data where the x-y curve intersects with a line offset from the modulus line by a defined x-axis offset when the yield point is reached. The function returns an invalid number before it locates the yield point.

Syntax
RuntimeOffsetYieldIndex(y-axis, x-axis, offset, numberOfPoints, threshold, slope, intercept [, startIndex, endIndex])

Parameters
y-axis - The y-axis array variable.

x-axis - The x-axis array variable.

offset - The variable that contains the offset yield goal in x-axis units. It should be set to a slightly higher value than the actual yield point. For example, if you want a 2% yield, set it to 3% to guarantee that you actually reached the desired yield point before moving on.

numberOfPoints - This variable holds the number of points used in the yield point calculations in counts.

threshold - This variable is used to set a minimum y-axis value that must be exceeded before looking for the yield point in y-axis units.

slope - This variable holds the slope of the modulus line.

intercept - This variable holds the intercept of the modulus line.

startIndex - The optional start index of the region. If this parameter is missing, the start is associated with the first data point in the array.

endIndex - The optional end index of the region. If this parameter is missing, the end is associated with the end of the array data.

Unit Class
Integer

TestRunNumber

Description
The TestRunNumber function returns the current test run number.

Returns
Returns a 1-based number representing the placement of the Test Run in the list of Test Runs.

Syntax
TestRunNumber()

Unit Class
Integer

ValleyIndex
The ValleyIndex function locates the index of the minimum value in the array.

Returns
The index of the array associated with the minimum value of the array.

Syntax
ValleyIndex(channel[, startIndex, endIndex])

Parameters
Channel - The array data used in the function.

StartIndex - The optional starting index of the region to evaluate.

EndIndex - The optional ending index of the region to evaluate.

Unit Class
Integer

Example
Given a force array whose minimum value is located at index 100 in the array, this function returns 100.

YieldIndexByZeroSlope
The YieldIndexByZeroSlope function determines the yield index by searching the curve in segments until the angle of the curve decreases to the specified angle.

Returns
The function returns the index associated with the Yield point.

Syntax
YieldIndexByZeroSlope(y-axis, x-axis, angle, segmentLength[, threshold, startIndex, endIndex)

Parameters
y-axis - The Y axis data array used in the calculation of the yield index.

x-axis - The X axis data array used in the calculation of the yield index.

Angle - The angle of the slope used to determine the location of the yield point.

SegmentLength - The number of points to use in each slope calculation.

Threshold - An optional percentage of the peak value of the y-axis data that must be exceeded before starting the search for the Yield Index. If this field is missing, 2% is used.

StartIndex - An optional starting index for the yield index search region. If this field is missing, the search starts at the beginning of the data.

EndIndex - An optional ending index for the yield index search region. If this field is missing, the search ends at the last point of the data.

Unit Class
Integer

Example
Y-Axis = Force array

X-Axis = Extension array

Angle = 0 rad

Segment Length = 10

Threshold = 2 N

Given these values, the algorithm starts looking for the first point in the Force array that exceeds 2 N.

The first Force point that exceeds 2 N becomes the start of the search region.

The slope of the Force and Extension data is calculated using 10 point segments.

The algorithm increments through the data until there is no more data to check or the slope decreases to an angle of 0 radians.

YpeEndIndexByIncreasingLoad
The YpeEndIndexByIncreasingLoad function determines the end of the Yield Point Elongation.

Returns
The function returns the index associated with the end of the Yield Point Elongation region.

Syntax
YpeEndIndexByIncreasingLoad(ForceArray, YPEStartIndex, Tolerance)

Parameters
ForceArray - The force array used to determine end of the Yield Point Elongation region.

YPEStartIndex - The index associated with the start of the Yield Point Elongation region.

Tolerance - The percentage difference in force at the YPE start index used to detect the YPE end index.

Unit Class
Integer

Example
YPEStartIndex = 100

Tolerance = 1 %

Force at YPE Start = 1000 N

Force Tolerance = 1000 N * 1% = 10 N

Assuming the peak was located at index 200, the algorithm starts at index 200 and go towards index 100. At each point, the force is compared to the force at YPE Start. When the Force of the currently compared points is within 10 N of the YPE Start value, the YPE end index is located.

Algorithm
This function uses the following algorithm:

1. Find the peak load index after the YPE Start index.
2. Starting at this point search backwards down the curve until the load value is within the specified tolerance of the load at the YPE Start index.
3. Report the index where this criterion is met as the YPE End index.

YpeEndIndexByTwoSlopes
The YpeEndIndexByTwoSlopes function located the end of the Yield Point Elongation region by finding the point of intersection of the maximum slope line and the zero slope line (both after the YPE Start index).

Returns
The function returns the index associated with the end of the Yield Point Elongation region.

Syntax
YpeEndIndexByTwoSlopes(ForceArray, DisplacementArray, YPEStartIndex, PeakTolerance, ZeroSlopeSegmentLength, MaxSlopeSegmentLength)

Parameters
ForceArray - The force array used to determine end of the Yield Point Elongation region.

DisplacementArray - The displacement array used to determine end of the Yield Point Elongation region.

YPEStartIndex - The index associated with the start of the Yield Point Elongation region.

PeakTolerance - The percentage drop from the of the peak force that must occur before starting the search for the zero slope region of the curve.

ZeroSlopeSegmentLength - The percentage of the total points used to determine the zero slope region.

MaxSlopeSegmentLength - The percentage of the total points used to determine the maximum slope region.

Unit Class

Integer

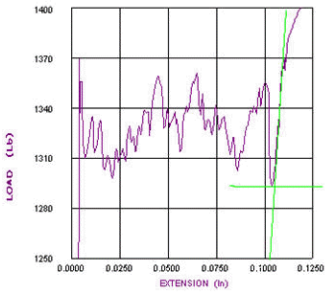
Algorithm

This function uses the following algorithm:

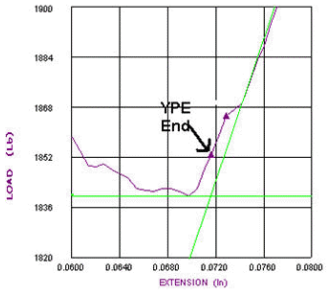
1. Starting at the peak load index, search the curve backwards until the load drops by the tolerance amount (tolerance is a percentage of peak load).
2. Using the zero slope segment length argument, search the curve backwards until the slope of the segment becomes less than zero.
3. Find the minimum load point of this segment, referred to as the zero slope index.
4. Starting at the zero slope index, use the maxSlopeSegLen argument to search up the curve for the segment with the highest slope value.
5. Draw a line for this segment and draw a horizontal line through the zero slope index.
6. Where these two lines intersect, calculate the value of the extension channel.
7. The YPE End index is the data point whose extension value is closest to this value.

Note:
This value can never be lower than the zero slope index.

Zero Slope Index



YPE End Index



YpeStartIndex

The YpeStartIndex function finds the start of the Yield Point Elongation region by locating the point after Yield where the slope of the curve exceeds the specified angle.

Returns

The function returns the index associated with the start of the Yield Point Elongation region.

Syntax

YpeStartIndex(ForceArray, DisplacementArray, Angle, SegmentLength, YieldIndex)

Parameters

ForceArray - The force array being analyzed for the Yield Point Elongation region.

DisplacementArray - The displacement array being analyzed for the Yield Point Elongation region.

Angle - The angle which must be exceeded.

SegmentLength - Percentage of points to use in the slope calculations.

YieldIndex - The index point into the array where the Yield Index was determined.

Unit Class

Integer

Example

Angle = 0

SegmentLength = 2%

YieldIndex = 85

Total points on the curve = 200

Number of points in segment = 0.02 * 200 = 4 points

1. Start with the segment from point 85 to point 89.
2. Increment points until slope is less than zero. For this example, assume that this occurs at points 100 to 104.
3. Increment points starting at 100 to 104 until the slope value is greater than zero. For our

example, assume this occurs at points 105 to 109.

4. Report the YPE Start Index as the minimum load point between the Yield index and the last point of the search (point 109).

Math Functions

This section provides reference information about the math functions.

abs (Absolute Value)

This function is used to calculate the absolute value of the number specified.

Important:
This function is only used for integer numbers.

Syntax

abs(number)

acos (Arc Cosine)

Syntax

acos(number)

This function is used to calculate the arc cosine of the number specified.

Returns

Angle with a cosine equal to the specified number.

asin (Arc Sine)

Syntax

asin(number)

This function is used to calculate the arc sine of the number specified.

Returns

Angle with a sine equal to the specified number.

atan (Arc Tangent)

Syntax

atan(number)

This function is used to calculate the arc tangent of the number specified.

Returns

Angle with a tangent equal to the specified number.

Atan2

Syntax:
atan2(y,x)

The angle with a tangent that is the quotient of the two specified numbers.

avg

Syntax:
avg(number1, number2,[numberN])

Returns the average of a series of numbers.

Ceiling
This function is used to round the specified number up to the nearest whole number.

Syntax
Ceiling(number)

The smallest integer greater than or equal to the specified number.

Examples
The following formula returns the value 6: ceiling(5.4)

The following formula returns the value 7: ceiling(6.9)

ArrayValue
The ArrayValue function returns the value of one array that corresponds with the location of a value found in another array.

Returns
The value from the array at the specified index or NaN (Not a Number) if not found.

Syntax
ArrayValue(resultArray, searchArray, searchValue[, startIndex, endIndex])

Parameters
ResultArray - The array where the result is extracted.

SearchArray - The array used in the search.

SearchValue - The value to search for in the search array.

StartIndex - The optional starting index for the search. If this parameter is not present, the search starts at the beginning of the array.

EndIndex - The optional ending index for the search. If this parameter is not present, the search ends at the last point in the array.

Unit Class
Same as result channel.

Example
ArrayValue(Extension, Load, PeakLoad)

This example returns the Extension value associated with the PeakLoad value located in the Load array.

cos (Cosine)

Syntax
cos(number)

This function is used to calculate the cosine of the number specified.

Cosine of the specified angle

cosh (Hyperbolic cosine)

Syntax
cosh(number)

Hyperbolic cosine of the specified angle.

e (Natural logarithmic base, e)

Syntax
e()

CurveArea
The CurveArea function calculates the area under the curve defined by the Y and X axis data.

Syntax
CurveArea(xArray, yArray, startIndex, endIndex)

Returns
The area under the curve.

Parameters
y-axis – The array holding the Y-Axis data.

x-axis – The array holding the X-Axis data.

StartIndex - The start index of the region to analyze.

EndIndex - The end index of the region to analyze.

Unit Class
X-Axis unit / Y-Axis unit

Example 1
CurveArea (_Load, SlackExt, 0 , SizeOfArray(_Load)-1)

This formula calculates the area under the entire Load versus Extension curve.

Example 2
CurveArea (_Load, SlackExt, 0, Peak)

This formula calculates the area under the Load versus Extension curve up to the peak load.

ElasticStrainValue
The ElasticStrainValue function is used to calculate the portion of strain in a specimen that is recoverable, or elastic. The point where this is determined is based on some percentage of the peak load.

Returns
The portion of the strain that is recoverable.

Syntax
ElasticStrainValue(ForceArray, StrainArray, StartIndex, EndIndex, Tolerance)

Parameter
ForceArray – The array containing force data.

StrainArray – The array containing strain data.

StartIndex – The start index of the modulus region.

EndIndex - The end index of the modulus region.

Tolerance – The percentage drop from peak force used to determine the transition point.

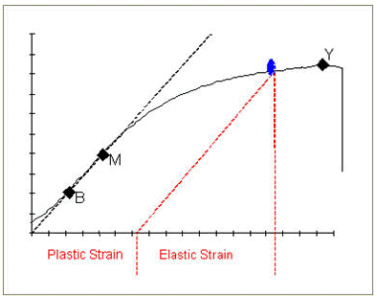
Unit Class
Strain

Example
Tolerance = 98% (shown at blue oval)

The point is found where the load value is 98% of the peak load. Then a line parallel to the modulus line is drawn from this point back to the x-axis. The Elastic strain is the difference in strain values as shown on the graph.

Note:
Unlike plastic strain, there is no flag to set whether the specified load point should be found after the peak or before. This is because the Elastic Strain value would be the same in either case.

Elastic Strain Value



EnergyValue

The EnergyValue function calculates the area under the curve defined by the Y and X axis data. If using the Force and Extension data, this function returns the amount of energy absorbed by the specimen during the test.

Returns

The area under the curve.

Syntax

EnergyValue(y-axis, x-axis, startIndex, endIndex)

Unit Class

X-Axis unit / Y-Axis unit

Example 1

EnergyValue (_Load, SlackExt, 0 , SizeOfArray(_Load)-1)

This formula calculates the area under the entire Load versus Extension curve (energy).

Example 2

EnergyValue (_Load, SlackExt, 0, Peak)

This formula calculates the area under the Load versus Extension curve (energy) up to the peak load.

Exp

This function is used to calculate the value of e (2.71828...) raised to the power specified by the exponent.

Syntax

exp(number)

Returns

A value “e” raised to the specified power.

FindNearestValue

Locate the closest value in an array to the search value.

Syntax
FindNearestValue(array, searchValue[, startIndex, endIndex])

FindNearestValueIndex
Locate the index of the closest value in an array to the search value.

Syntax
FindNearestValueIndex(array, searchValue[, startIndex, endIndex])

Floor

Syntax
floor(number)

Returns
The largest integer less than or equal to the specified number.

This function is used to round the specified number down to the nearest whole number.

Example 1
Floor(5.4)

This formula returns the value 5.

Example 2
Floor(6.9)

This formula returns the value 6.

IsInvalidNumber

Syntax
IsInvalidNumber(number)

Checks the validity of the supplied value.

Returns
This function returns 1 if the value being tested is invalid; otherwise, 0 (zero) is returned.

Example
IsInvalidNumber(PeakLoad)

Where PeakLoad is a non-array variable.

IsValidNumber
Returns “1” if the number is valid. Otherwise, it returns “0”.

Syntax
IsValidNumber(number)

Example
IsValidNumber(PeakLoad)

Where PeakLoad is a non-array variable.

LeastSquaresFit
The slope of the least squares fit of the array that contains Y-axis data and X-axis data between the start and end indexes.

Returns
The slope of the least squares fit line calculated over the specified region.

Syntax
LeastSquaresFit(yVariable, xVariable, startIndex, endIndex)

Parameters
yVariable – The Y-Axis data.

xVariable – The X-Axis data.

StartIndex – The starting index of the region.

EndIndex – The ending index of the region.

Units
Y-Axis unit / X-Axis Unit

Example
LeastSquaresFit(Force, Extension, Slope1, Slope2)

This formula returns the slope of the line between the two indexes of Slope1 and Slope2.

log(number)

Syntax
log(number)

The natural log of a specified number.

log10(number)
The base-10 logarithm of a specified number.

max

Syntax
max(number1, number2[, numberN])

Returns
The maximum of a series of numbers.

MaxDouble()

Syntax
MaxDouble()

Returns
Returns the maximum value for the type of number.

MaxLong()

Syntax
MaxLong()

Returns
Returns the maximum value for the type of number.

min
The minimum of a series of numbers.

Syntax
min(number1, number2[, numberN])

MinDouble()
Returns the minimum value for the type of number.

Syntax
MinDouble()

MinLong()
Returns the minimum value for the type of number.

Syntax
MinLong()

NaN()
Returns an invalid value for the type of number, which is Not a Number.

Syntax
NaN()

PI()
Value of Pi.

Syntax
PI()

PlasticStrainValue

The PlasticStrainValue is used to calculate the portion of strain in a specimen that is non-recoverable, or plastic. The point where this is determined is based on some percentage of the peak load.

Returns

The portion of the strain that is non-recoverable.

Syntax

PlasticStrainValue(ForceArray, StrainArray, StartIndex, EndIndex, Tolerance[, AfterPeakFlag])

Parameters

ForceArray – The array containing force data.

StrainArray – The array containing Strain data.

StartIndex – The start index of the modulus region.

EndIndex – The end index of the modulus region

Tolerance – The percentage of the peak force used to locate the transition point.

AfterPeakFlag – An optional flag to specify looking for transition point after the peak load.

Unit Class

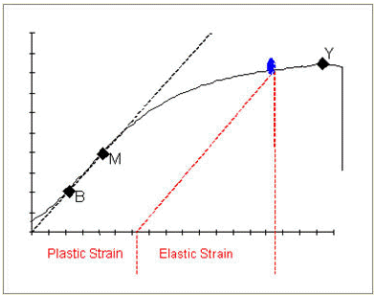
Strain

Example

Tolerance = 98% (shown at blue oval)

The point is found where the load value is 98% of the peak load. Then a line parallel to the modulus line is drawn from this point back to the x-axis. The Plastic strain is the difference in strain values as shown on the graph.

Plastic Strain Value



pow(base, exponent)

A number (base) is raised to an exponential power (exponent).

Returns

Returns the value of the base raised to the power specified by the exponent.

Syntax
pow(base, exponent)

Example 1
Pow(2, 3)

This formula returns the value 8 (2^3, or 2*2*2)

Example 2
Pow(5, 2)

This formula returns the value 25 (5^2, or 5*5)

rem(dividend, divisor)
This function returns the remainder from the division of two numbers. When using this function, the result is calculated differently than the modulus operator. This function uses the following ANSI/IEEE-compliant formula:

*Remainder = divided - (divisor * round(divided / divisor))*

Syntax
rem(dividend, divisor)

round(number)
Rounds a value to the nearest integer.

Syntax
round(number)

sign(number)
Value that indicates the sign of a number.

Returns
This function returns –1 is the number is negative, or +1 is the number is positive. If the number is exactly 0, the function returns 0.

Syntax
sign(number)

Example 1
sign(-500)

This formula returns the value -1.

Example 2
sign (20)

This formula returns the value +1

sin(number)

Returns
Sine of the specified angle.

Syntax
sin(number)

sinh(number)
Hyperbolic sine of the specified number.

Syntax
sinh(number)

sqrt(number)
The square root of a number.

Syntax
sqrt(number)

tan(number)
Tangent of the angle.

Syntax
tan(number)

tanh(number)
Hyperbolic tangent of the angle.

Syntax
tanh(number)

truncate
Rounds a value to the nearest integer towards zero.

Returns
The X-Intercept from the least squares fit line.

Syntax
truncate(number)

Parameter
number – The number to be truncated.

Unit Class
Same the number.

Example
truncate(4.5) = 4

XInterceptValue
The XInterceptValue function returns X-Intercept of the least squares fit line calculated over the specified region.

Returns
The X-Intercept from the least squares fit line.

Syntax
XInterceptValue (y-axis, x-axis, slope1Index, slope2Index)

Parameters
y-axis – Y-Axis data array.

x-axis – X-Axis data array.

Slope1Index – The start index of the region.

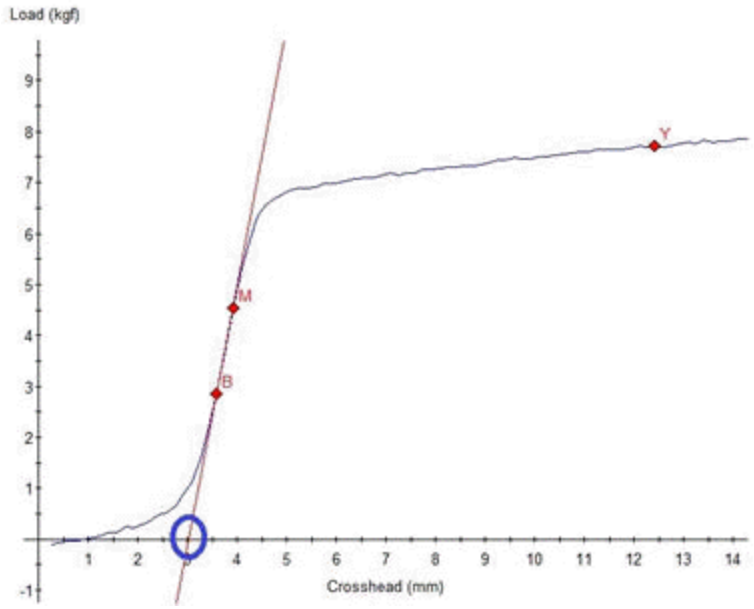
Slope2Index – The end index of the region.

Unit Class
Same as X-Axis data.

Example
With the following function, the circled value on the X-Axis is returned.

XInterceptValue (_Load, _Extension, Slope1, Slope2)

XInterceptValue



YInterceptValue

The YInterceptValue returns the point where the line determined by the index values crosses the Y-Axis.

Returns

The index in to the arrays where the Y-intercept occurs.

Syntax

YInterceptValue(y-axis, x-axis, slope1Index, slope2Index)

Parameters

y-axis – The Y-Axis data array being analyzed.

x-axis – The X-Axis data array being analyzed.

Slope1Index – The starting index of the region used to calculate the straight line.

Slope2Index - The ending index of the region used to calculate the straight line.

Unit Class

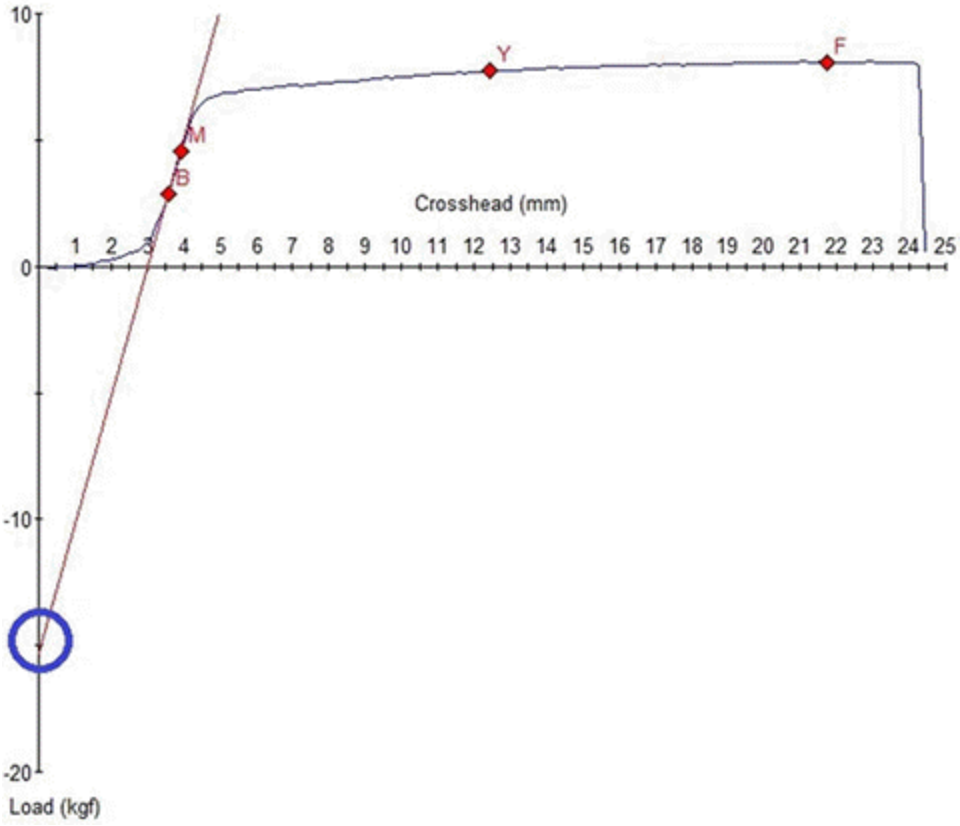
Same as Y-Axis data.

Example

The following function returns the Y-Intercept of the line defined by Slope1 and Slope2.

YInterceptValue(_Load, _Extension, Slope1, Slope2)

YInterceptValue



Operator Functions

This section describes Operator functions and precedence.

Operators and Precedence

The following table shows all the recognized operators organized by group and listed in order of precedence. Within a group, all operators have the same precedence.

Note: Some programming languages use a semicolon as a list separator instead of a comma. If you are using one of those languages, the last operator of the table would be replaced with a semicolon.

Operator Precedence

Precedence	Operator	Function	Direction
1	[ ]	Array Index	Left-to-Right
	( )	Function Call	
	.	Variable-Specific Information	
2	!	Logical NOT	Right-to-Left

Precedence	Operator	Function	Direction
	+	Unary Plus	Left-to-Right
	-	Unary Minus	
	~	Ones Complement	
3	*	Multiply	Left-to-Right
	/	Divide	
	%	Modulus	
4	+	Addition	Left-to-Right
	–	Subtraction	
5	<<	Bitwise Shift Left	Left-to-Right
	>>	Bitwise Shift Right	
6	<	Less Than	Left-to-Right
	>	Greater Than	
	<=	Less than or Equal	
	>=	Greater Than or Equal	
7	==	Equal to	Left-to-Right
	!=	Logical Not	
8	&	Bitwise AND	Left-to-Right
9	^	Bitwise XOR	Left-to-Right
10	|	Bitwise OR	Left-to-Right
11	&&	Logical AND	Left-to-Right
12	||	Logical OR	Left-to-Right
13	=	Assignment	Right-to-Left
14	,	Comma - List Separator	Left-to-Right

Choose

Returns

Returns a value based on the evaluation of the first argument.

Syntax

Choose(Expression, EvaluatedZero,EvaluatedOne[, EvaluatedN])

Parameters

Expression—The Expression must evaluate to an integer.

EvaluatedZero—If the Expression evaluates to a 0 (zero), the EvaluatedZero value is returned.

EvaluatedOne—If the expression evaluates to a 1 (one), the EvaluatedOne value is returned, and so on.

EvaluatedN —If the Expression is < 0 or > N, the EvaluatedN value is returned.

Note:
The Evaluated arguments must evaluate to be either all strings or all numbers

Example

Examples of the Choose function:

Choose(chosen, “Red”, “Blue”, “Green”)

Choose(chosen, 1.0, 2.0, 3.0, 4.0, 5.0)

Choose(chosen, Choice0, Choice1, Choice2, Choice3)

Peel-Tear Functions

This section describes Peel-Tear functions.

AverageMinPeaks

The AverageMinPeaks function calculates the average of the minimum peaks located in the data using the specified criterion.

Syntax

AverageMinPeaks(VarArray, peakCriterion, numberOfPeaks[, startIndex, endIndex, threshold])

Parameters

VarArray - The data array to be analyzed.

PeakCriterion - A percentage of the peak magnitude used to detect when peaks occur.

NumberOfPeaks - The number of minimum peaks to include in the calculations.

StartIndex - The optional index into the array used as the starting location for the calculation.

EndIndex - The optional index into the array used as the ending location for the calculation.

Threshold - The optional minimum value, in array units, which must be exceeded before the data is used in the calculation.

Unit Class

Same as array data.

Example

AverageMinPeaks(_Load, LoadMinPeakCriteria, 5, 0, SizeOfArray(_Load)-1,MinimumLoad)

This formula:

1. Searches the _Load channel.
2. Looks for drops in the value of _Load channel that correspond to the input LoadMinPeakCriteria.
3. Determines the average of the 5 minimum peaks.
4. Searches between the first data point and the last data point.
5. Calculates the result only if the _Load channel becomes greater than the value of the input MinimumLoad.

AveragePeaks

The AveragePeaks function calculates the average of the maximum peaks located in the data using the specified criterion.

Returns

The average of the maximum peaks.

Syntax

AveragePeaks(VarArray, peakCriterion[, startIndex, endIndex, numberOfPeaks, threshold])

Parameters

VarArray - The array data being analyzed.

PeakCriterion - The percentage of change from the peak that identifies the valley.

StartIndex - The optional starting index of the region to analyze.

EndIndex - The optional ending index of the region to analyze.

NumberOfValleys - The optional maximum number of valleys to identify.

Threshold - The optional threshold that has to be exceeded to start looking for peaks.

Unit Class

Same as the array data.

Example

AveragePeaks(_Load, LoadPeakCriteria, 0, SizeOfArray(_Load)-1)

This formula:

1. Searches the _Load channel.
2. Looks for drops in the value of _Load channel that correspond to the input LoadPeakCriteria.
3. Searches between the first data point and the last data point.

AverageValleys

The AverageValleys function returns the average of all the valleys determined using the peak criteria.

Returns
The average valley calculated.

Syntax
AverageValleys(VarArray, valleyCriterion[, startIndex, endIndex, numberOfValleys, threshold])

Parameters
VarArray - The array data being analyzed.

ValleyCriterion - The percentage of change from the valley that identifies the valley.

StartIndex - The optional starting index of the region to analyze.

EndIndex - The optional ending index of the region to analyze.

NumberOfValleys - The optional maximum number of valleys to identify.

Threshold - The optional threshold that has to be exceeded to start looking for valleys.

Unit Class
Same unit as the array.

Example
AverageValleys(_Load, LoadMinValleyCriteria)

This formula:

1. Searches the _Load channel.
2. Looks for increases in the value of _Load channel that correspond to the input LoadMinValleyCriteria.

AverageValue
The AverageValue function returns the average array value over a specified region.

Returns
The average array value.

Syntax
AverageValue(VarArray, startIndex, endIndex)

Parameters
VarArray - The variable being analyzed.

StartIndex - The start index of the region being analyzed.

EndIndex - The end index of the region being analyzed.

Unit Class
Same as the array data.

Example
AverageValue(_Load, Peak, SizeOfArray(_Load)-1)

This formula calculates the average value of the _Load channel between the Peak load point and the end of the test.

CombinedExtension
The CombinedExtension function returns the value of the extensometer or crosshead/actuator based on the value of the removal point.

Returns
Extensometer value before removal point, crosshead/actuator after removal point.

Syntax
CombinedExtension(extensometer, crosshead, removalPoint)

Parameters
Extensometer - The extensometer array.

Crosshead - The crosshead/actuator array.

RemovalPoint - The value, in extensometer units, used in the determination of what data to return from the function.

Unit Class
Length (Same as the extensometer, crosshead, and removal point)

Example
CombinedExtension(Strain, Extension, RemovalPoint)

This formula:

1. Reports the value of the Strain channel
2. Checks to see if the value of the Strain channel has exceeded the value of the RemovalPoint input.
3. Reports the value of the Extension channel when the Strain channel has exceeded the value of RemovalPoint.

CombinedStrain
The CombinedStrain function returns the strain value calculated from either the extensometer or the crosshead/actuator based on the value of the removal point.

Returns
The function returns the extensometer value divided by extGL prior to the removal points or crosshead divided by gripSeparation.

Syntax
CombinedStrain(extensometer, crosshead, removalPoint, gripSeparation, extGL)

Unit Class
Strain

Example
CombinedStrain(Strain, Extension, RemovalPoint, GageLength2, GageLength1)

This formula:

1. Reports the value of the Strain1 channel divided by the value of GageLength1.
2. Checks to see if the value of the Strain1 channel has exceeded the value of the RemovalPoint input.
3. Reports the value of the Extension channel divided by value of GageLength2 when the Strain1 channel has exceeded the value of RemovalPoint.

MedianPeak
The MedianPeak function determines the median of the peaks located in the specified region that meet the specified peak criterion.

Returns
The median peak value.

Syntax
MedianPeak(VarArray, peakCriterion[, startIndex, endIndex, threshold])

Parameters
VarArray - The array data being analyzed for peaks.

peakCriterion - The percentage change from the peak that signifies a peak.

startIndex - The optional start index of the region. The search starts at the first data point if this parameter is missing.

endIndex - The optional end index of the region. The search ends at the last data point if this parameter is missing.

Threshold - The optional threshold that must be exceeded before searching for peaks.

Unit Class
Same as the array data.

Example
AveragePeaks(_Load, LoadPeakCriteria, 0, SizeOfArray(_Load) - 1, MinimumLoad)

This formula:

1. Searches the _Load channel.
2. Looks for drops in the value of _Load channel that correspond to the input LoadPeakCriteria.
3. Determines the median of all of the peaks.

4. Calculates the result only if the _Load channel becomes greater than the value of the input MinimumLoad.

NumberOfPeaks

The NumberOfPeaks function determines the number of peaks in a region that meet the specified peak criterion.

Returns

The number of peaks found in the region.

Syntax

NumberOfPeaks(VarArray, peakCriterion[, startIndex, endIndex, threshold])

Parameters

VarArray - The array data being analyzed for peaks.

peakCriterion - The percentage change from the peak that signifies a peak.

startIndex - The optional start index of the region. The search starts at the first data point if this parameter is missing.

endIndex - The optional end index of the region. The search ends at the last data point if this parameter is missing.

Threshold - The optional threshold that must be exceed before searching for peaks.

Unit Class

Integer

Example

NumberOfPeaks(_Load, LoadPeakCriteria, 0, SizeOfArray(_Load) - 1 , MinimumLoad)

This formula:

1. Searches the _Load channel.
2. Looks for drops in the value of _Load channel that correspond to the input LoadPeakCriteria.
3. Searches between the first data point and the last data point.
4. Determines the total number of peaks found.
5. Calculates the result only if the _Load channel becomes greater than the value of the input MinimumLoad.

NextPeak

The NextPeak function searches the data array specified in **array** and returns the next peak's index, as defined by the peak criteria, after the data point specified by the start index. The values in square braces, [ ], are optional.

Syntax

NextPeak(array, peakCriteria, startIndex[, endIndex, threshold]))

Parameters

array—The array data being analyzed for the next peak.

peakCriteria—A percentage of the peak magnitude used to detect when a peak occurs.

startIndex—The start index of the array used in locating the peak. The first point in the array is used if this parameter is not provided.

endIndex—The optional end index of the array used in locating the peak. The last point in the array is used if this parameter is not provided.

threshold—The optional minimum value, in array units, which must be exceeded before the data is used in the calculation.

Dimension

Count

Example

NextPeak(_LoadArray, LoadPeakCriteria, Peak, SizeOfArray(_LoadArray)-1, MinimumLoad)

This formula:

1. Searches the _LoadArray data.
2. Looks for the next drop in the _LoadArray data's value that corresponds to the input LoadPeakCriteria.
3. Searches between the peak load's point and the last data point.
4. Finds the next peak and returns its array index.
5. Only calculates the result if the _LoadArray data becomes greater than the value of the input MinimumLoad.

NextValley

The NextValley function searches the data array specified in **array** and returns the next valley's index as defined by the valley criteria, after the data point specified by the start index. The values within square braces, [], are optional.

Syntax

NextValley(array, valleyCriteria, startIndex[, endIndex, threshold])

Parameters

array—The array data being analyzed for the next valley.

valleyCriteria—A percentage of the peak magnitude used to detect when a valley occurs.

startIndex—The start index of the array used in locating the valley. The first point in the array is used if this parameter is not provided.

endIndex—The optional end index of the array used in locating the valley. The last point in the array is used if this parameter is not provided.

threshold—The optional minimum value, in array units, which must be exceeded before the data is used in the calculation.

Dimension
Count

Example
NextValley(_LoadArray,LoadValleyCriteria,Peak,SizeOfArray(_LoadArray)-1,MinimumLoad)

This formula:

1. Searches the _LoadArray data.
2. Looks for the next increase in the _LoadArray data's value that corresponds to the input LoadValleyCriteria.
3. Searches between the peak load's point and the last data point.
4. Finds the next valley and returns its array index.
5. Only calculates the result if the _LoadArray data becomes greater than the value of the input MinimumLoad.

StDevValue
Calculates the standard deviation of a variable array between two index points.

Syntax
StDevValue(VarArray, startIndex, endIndex)

Parameters
VarArray - The array data being analyzed.

StartIndex - The start index of the region being analyzed.

EndIndex - The end index of the region being analyzed.

Unit Class
Same as array data specified.

Example
StDevValue(_Load, Slope1, Slope2)

This formula calculates the standard deviation between given by the variables Slope1 and Slope2.

TearIndex
Locates the index of the tear value in an array.

Syntax
TearIndex(VarArray, TearCriteria[, startIndex, endIndex, Threshold])

Parameters
VarArray - The array data being analyzed.

Tear Criteria

StartIndex - The start index of the region being analyzed.

EndIndex - The end index of the region being analyzed.

Threshold

Unit Class
Same as array data specified.

Sensor Functions
This section provides basic reference information about the available sensor functions.

CalibrationDate

Returns
Returns the last calibration date of the sensor attached to a signal.

Syntax
CalibrationDate(SignalName)

Parameter
Signal Name—The “SignalName” in signal functions is the internal name of the signal and must be quoted or passed in as a string variable. To determine the internal name of a signal, select the **Show Internal Names** check box in the **Resources** tab. The Name column shows the internal names.

Example
CalibrationDate(“_Load”)

Unit Class
String

CalibrationDueDate

Returns
Returns the calibration due date of the sensor attached to a signal.

Syntax
CalibrationDueDate(SignalName)

Parameter
Signal Name—The “SignalName” in signal functions is the internal name of the signal and must be quoted or passed in as a string variable. To determine the internal name of a signal, select the **Show Internal Names** check box in the **Resources** tab. The Name column shows the internal names.

Example
CalibrationDueDate(“_Load”)

Unit Class
String

ModelNumber

Returns
Returns the model number of the TEDS device attached to a signal.

Syntax
ModelNumber(SignalName)

Parameter
Signal Name—The “SignalName” in signal functions is the internal name of the signal and must be quoted or passed in as a string variable. To determine the internal name of a signal, select the **Show Internal Names** check box in the **Resources** tab. The Name column shows the internal names.

Example
ModelNumber(“_Load”)

Unit Class
String

SerialNumber

Returns
Returns the serial number of the TEDS device attached to a signal.

Syntax
SerialNumber(SignalName)

Parameter
Signal Name—The “SignalName” in signal functions is the internal name of the signal and must be quoted or passed in as a string variable. To determine the internal name of a signal, select the **Show Internal Names** check box in the **Resources** tab. The Name column shows the internal names.

Example
SerialNumber(“_Load”)

Unit Class
String

String Functions
This section provides basic reference information about the available string functions.

CompareStrings
Compares two strings, ignoring case.

Returns
The result is 0 if the strings are equal; negative if the first string is ordered before the second string, and positive if the second string is reordered before the first string.

Syntax
CompareStrings(string1, string2)

Parameters
string1-First string

string2-second string compared to string1.

Unit Class
String

FindSubString

Returns
Finds the index of the first occurrence of a substring within a string starting at or after a specified starting index. Results is the index where the substring begins, or negative if not found.

Syntax
FindSubstring(substring, string, startIndex)

Parameter
substring

string

startIndex

Unit Class
String

left

Returns
Extracts a substring from the left side of a string.

Syntax
left(string, length)

Parameters
string

length

Unit Class
String

mid

Returns
Extracts a substring from the middle of a string.

Syntax
mid(string, start[, length]

Parameters
string

start

length

Unit Class
String

NumberToString

Returns
Creates a string from the specified number with the optionally specified digits.

Syntax
NumberToString(number[, digits])

Parameters
number

digits

Unit Class
String

right

Returns
Extracts a substring from the right side of a string.

Syntax
Right(string, start)

Parameters
string

start

Unit Class
String

StringLength

Returns
The length of the specified string.

Syntax
StringLength(string)

Parameters
string

Unit Class
String

StringToInteger

Returns
Converts a string to an integer number.

Syntax
StringToIntger(string)

Parameters
string

Unit Class
String

StringToNumber

Returns
Converts a string to a floating-point number.

Syntax
StringToNumber(string)

Parameters
string

Unit Class
String

tolower

Returns
Converts a string to all lower case.

Syntax
tolower(string)

Parameters
string

Unit Class
String

toupper

Returns
Converts a string to all uppercase.

Syntax
toupper(string)

Parameters
string

Unit Class
String

TrimStringEnd

Returns
Removes whitespace (spaces, tabs, and new lines) from the end of a string.

Syntax
TrimStringEnd(string)

Parameters
string

Unit Class
String

TrimStringStart

Returns
Removes whitespace (spaces, tabs, and new lines) from the start of a string.

Syntax
TrimStringStart(string)

Parameters
string

Unit Class
String

varIdentifier.DisplayValue

This construct is implicit in the MTS TestSuite application. Use this syntax in a calculation to return a string representation of a string variable; including its name, value, and units.

Returns

Returns the string representation of the variable in display units.

Syntax

variableStringName.displayvalue

Replace variableStringName with the actual name of the string variable.

Example

Create a string variable in any of the TWE application EM templates and enter the following calculation:

StrnAtBreak.display+" "+StrnAtBreak.displayvalue+" "+StrnAtBreak.units

The result is similar to the following:

Strain at Break 0.236 mm/m

“StrainAtBreak” is the string variable, the display value of the variable is 0.236, and the units are mm/m.

Compare Tool

TestSuite Compare Tool Overview

Access

Tool menu > **Compare** > **Variable** or **Function**

Compare Variables or Functions

The Compare tool shows differences of variable properties or function properties between the currently opened test or template and other test definitions, test runs, analysis definitions, and analysis runs. You can identify design differences and build on test definitions when you compare definitions and runs.

You can compare:

- Test definitions
- Test runs with analysis runs (In the TW application, an analysis run includes any changes made in the **Review** tab.)
- Default templates with modified templates
- Specimen variables with other specimen variables

Merge or Add Changes

The Compare tool includes an **Add Changes** and **Remove Changes** buttons that copies functions or variable properties (with the same name) from another test definition, analysis definition, or analysis runs into the open test or analysis. You can also use **Add Changes** to add functions or variables from another test or analysis to the open test or analysis.

Compare a Variable or Function

To compare a variable or function:

1. In the **Tools** menu, click **Compare** and select **Variables** or **Functions**. The Select Groups to Compare Window opens.
2. Select the check box next to the name of the test, template, analysis definition, or analysis run you want to compare and click the arrow to move them to the box on the right. You can also click **Select an External Test** button (**...**) to browse for an external test file.

   Note: The **Count** column shows the number of variables or functions in each selected item.
3. Repeat until you have selected what you want to compare.
4. Click **OK**. The Comparison window opens to show all the variables or functions in the tests and indicates mismatches in red.
5. To view only the differences, select the **Show Only Differences** check box.
6. Click the plus icon to expand the window and view the properties for the variable or function. Once expanded, the variable property names appear in the first column followed by a column that shows the variable properties for the currently open test, and then columns for each item selected in step 3.

   Note: If a part of a calculation for a function does not match, you must examine the whole calculation; the Compare tool does not indicate which part of the calculation does not match.
7. To add the variable or function to your current test, click the **Add Change** button below the appropriate test name and click **Apply**. After you have added all your selections, click **OK** to close the window.

Change or Add a Variable or Function During a Comparison

To change or add a variable or function during a comparison:

1. In the **Tools** menu, click **Compare** and select **Variables** or **Functions**.
2. Select the check box next to the name of the test, test definition, analysis definition, or analysis run you want to compare.
3. Repeat until you have selected what you want to compare.
4. Click **OK**. The next window shows all the variables or functions in the tests and indicates mismatches in red.
5. Click the plus icon to expand the window and view the properties for the variable or function that you want to merge into your current test. Once expanded, the variable property names

appear in the first column followed by a column that shows the variable properties for the currently open test, and then columns for each item selected in step 2.

6. To change or add variables or functions to the currently open test:
    A. Scroll down the list to locate the variable/function that you want to change.
    B. Determine the column that contains the variable properties or function that you want to change or add to the currently open test.
    C. Click **Add Changes**. The properties of the function or variable that you want to merge or add are copied to the **Update Test** column and are shown in blue.
7. Click **Apply** or **OK**.

Test Activities

Editing General Activity Information

Use the **General** panel to customize the **Display name** and the **Description** of the activity.

The **Display name** is the name that is shown on both the **Flowchart View** and **Outline View** of the test designer. If you leave this entry blank, the default name of the test activity will be used along with a summary of test activity parameters. For example, an **Auto offset** activity with two signals will read **Apply offset: 2 Signals** by default.

The **Description** is shown when you hover the cursor over the activity icon in the test procedure. If you are designing or modifying a complicated test, you may want to enter a detailed description of each new or modified activity. By doing this, it will be easier for you and others to understand the purpose of activities and tests in the future.

Example: Suppose you are creating or modifying a test that contains several **Go To + DAQ + Detection** activities. By default, the activity name in the test procedure is accurate, but you know that someone else at your organization will be making changes to your test in the future.

To make the test easier to understand for other test designers, you enter a detailed description in the **Description** field of each **Go To + DAQ + Detection** activity. This description might include notes about the what data is intended to be gathered while the activity executes, which ASTM standards must be satisfied by the activity, or reasons why certain break detection parameters were selected.

Allow Handset Control Activity Overview

The **Allow Handset Control Activity** displays a user-defined message and allows the operator to take control of the system with the handset while the test is running. If the handset becomes active while this activity is running, the test control panel is locked.

Note: This activity is not available on systems equipped with an MTS Series 793 controller (FlexTest).

Example: The **Allow Handset Control** Activity might be useful when specimen installation is part of a test procedure.

You can configure the activity to display a message window that displays instructions for how to insert the specimen. While this activity is running, the handset is allowed to take control of the system so the operator can safely position the crosshead or actuator while manipulating a specimen or fixturing.

To continue the test procedure, the operator must release handset control and dismiss the dialog displayed by this activity.

For more information about configuring the options available in the **Allow Handset Control** activity, see the following topics:

- "Editing Custom Messages" on page 250
- "Resizing Custom Message Windows" on page 251

- "Editing Custom Message Window Buttons" on page 251

Dwell+DAQ+Detection Activity Overview

Use the **Dwell + DAQ + Detection** activity to maintain a command for a specified duration at either the current level or a specified level in a specified control mode. By default, the activity dwells at the current level (when the **Dwell at current level** check box is selected).

For detailed information about configuring the properties of the **Dwell** panel, see "Configuring Dwell Properties" on page 243. For detailed information about configuring the data acquisition properties on the **DAQ** panel and the break detection properties on the **Break Detection** panel, see "Performing Data Acquisition" on page 248 and "Specifying Break Detection Parameters" on page 247.

Important: If you are commanding this activity using a load or strain control mode, you may need to tune the load or strain control mode before using this activity within a test.

Configuring Dwell Properties

Use the settings on the **Dwell** panel to specify the control mode, end level, and end condition of the **Dwell + DAQ + Detection** activity.

Dwell Properties

The following properties are available when configuring the options under the **Dwell** panel of the **Dwell + DAQ + Detection** activity.

Dwell Properties

Item	Description
Control mode	Select a control mode for the **Dwell + DAQ + Detection** activity.
Dwell at current value	Select this option to maintain the dwell at the current value as opposed to a specific end level. If you clear this option, you can enter a specific end level into the **End level** field below.
End level	Enter a value and unit of measurement or input for when the dwell should end. To toggle between entering an specific value and an input, click # or x. When configuring the dwell to maintain a command at a specified level instead of the current value, the rate at which the command ramps to the specific value is the maximum rate of the crosshead for the test frame. **Note:** This feature appears when you clear the **Dwell at Current Value** check box.
End condition	Indicate the condition in which the dwell will end. You can choose to either end the dwell after a certain period of time or when a signal reaches a specific limit. • **Time** • **Duration**—Enter the time and unit of measurement. To toggle

Item	Description

	between entering an specific value and an input, click # or x.
	• **Limit**
	  • **Signal**—Click ... to open the Select a Signal window. In this window, select the desired signal and click **OK**.
	  • **Comparison**—Select the comparison metric that will cause the dwell to end: **Becomes Greater Than**, **Becomes Less Than**, **Increases By**, **Decreases By**, or **Crosses**.
	  • **Value**: Enter the value and unit of measurement used in the calculation at which the dwell will maintain the command. To toggle between entering an specific value and an input, click # or x.

Example: Suppose you want to add a **Dwell + DAQ + Detection** activity that dwells at the current level for 5 seconds.

First, you would set the **Control mode** to **Crosshead**. Then, enable the **Dwell at current value** option. Under **End condition**, select **Time** and enter **5** seconds under **Duration**.

GoTo+DAQ+Detection Activity Overview

Use the **Go To + DAQ + Detection** activity to command a control channel to move the crosshead or actuator at a specified rate and direction.

For detailed information about configuring the properties of the **Go To** panel, see "Configuring Go To Options" on page 244. For detailed information about configuring the data acquisition properties on the **DAQ** panel and break detection properties on the **Break Detection** panel, see "Performing Data Acquisition" on page 248 and "Specifying Break Detection Parameters" on page 247.

Important: If you are commanding this activity using a load or strain control mode, you may need to tune the load or strain control mode before using this activity within a test.

Configuring Go To Options

Use the settings on the **Go To** panel to specify the control mode, direction, rate, and end condition of the **Go To + DAQ + Detection** activity.

Go To Properties

The following properties are available when configuring the options under the **Go To** panel of the **Go To + DAQ + Detection** activity.

Go To Properties

Item	Description
Control mode	Select the type of feedback to use in the control loop for the selected channel.
Use tuning parameters	If the control mode is set to **Load** or **Strain**, you must enter the tuning parameters for the control mode. The tuning parameters will be different for different type of specimen materials. **Note:** Tuning parameter settings are only available with MTS TestSuite TW Software running on an MTS Insight Controller.
Direction	Select the signal direction used in the control loop for the selected channel for this activity. **Increase**—Move the crosshead or actuator in a direction that will increase the control-mode feedback signal value. **Decrease**—Move the crosshead or actuator in a direction that will decrease the control-mode feedback signal value. **Auto**—The application will use the "Effect of Increasing Extension" resource settings to determine which direction the crosshead or actuator must travel to reach the specified end condition. **Note:** The **Auto** setting will not work with resources that have the "Effect of Increasing Extension" setting set to **Indefinite**.

Item	Description
Rate	Enter the rate at which the crosshead or actuator will move. To toggle between entering an specific value and an input, click # or x.
End condition	Select the **End condition** check box if you want to define limits that will end this activity. **Signal**—Select the signal to monitor for the ending condition. **Comparison**—Select an absolute or relative comparison, between the actual signal value and the value entered below, that will end the activity. **Note:** When the **Direction** setting is set to **Auto**, the Crosses comparison setting becomes available. When **Crosses** is selected, the activity ends when the actual signal equals the set value. **Value**—Enter the signal value used by the comparison setting. To toggle between entering an specific value and an input, click # or x.
Brake distance	Control-mode braking slows the command to help avoid overshoot. The brake distance value is subtracted from the ending condition value to determine where control-mode braking starts. To toggle between entering an specific value and an input, click # or x. For example, if the ending condition value is 100 N and the brake distance is set to 10 N, at 90 N, the command will slow down to 10% of the rate setting and at 99 N, it will slow down to 1% of the rate setting. **Note:** This option is not available when the control mode is crosshead or extension.

Example: Suppose you want to create a **Go To + DAQ + Detection** activity that moves in a direction that causes the signal feedback to become greater than 5 mm.

First, you need to set up the command signal. Under **Control mode**, you select **Crosshead**. Under **Direction**, you select **Increase**. Under **Rate**, you select the **Test Rate** input because you want the crosshead to move at the default test rate.

Next, you need to set up the end condition. To enable an end condition, you select the **End condition** check box. Then, under **Signal**, you click ..., select the **Crosshead** signal, and click **OK**. Since you want the crosshead to stop moving when it reaches 5 mm, you select **Becomes Greater Than** under **Comparison**. Finally, under **Value**, you enter **5 mm**.

When the activity executes during the test run, the crosshead will increase at the test rate and stop moving when the crosshead signal becomes greater than 5 mm.

Specifying Break Detection Parameters

Use the options on the **Break Detection** panel of the **Dwell + DAQ + Detection** and **Go To + DAQ + Detection** activities to monitor signals for the occurrence failure events (peak or valley values), and determine whether the failure events meet the criteria for specimen failure.

When a break is detected in a **Dwell + DAQ + Detection** activity or a **Go To + DAQ + Detection** activity, any subsequent activities in the **Procedure** node of the test definition tree will not be performed. Instead, the next node in the test definition tree (**Return to Zero**) is executed. When the **Review** tab is shown, the **Test Run End Reason** column will be **Break Detected**.

The following properties are available when configuring the options under the **Break Detection** panel of the **Dwell + DAQ + Detection** and **Go To + DAQ + Detection** activities.

Break Detection Properties

Item	Description
Enable	Enables the process. (Clear this box to disable the process.)
Display Name	Specifies the process name displayed on the Procedure or Group process window. If left blank, the software will automatically generate a name based on the selected signal(s).
Description	Enter an optional description to document the procedure design.
Progress Table Visibility	Indicates whether the activity will have a listing in the Progress Table. The Progress Table is a control that can be placed on the Test-Run Display. Options are: • **Fixed**—The activity will have a listing in the Progress Table that shows while the test is executing and persists after the test is complete. • **Transient**—The activity will have a listing in the Progress Table only while the program is executing. • **Never**—The activity will not have a listing in the Progress Table.
Completion	This control is only pertinent if more than one signal is listed in the Signals list or if the machine has multiple heads (an available option for some Criterion machines). Options are: • **All Breaks**—The activity completes when all signal breaks have been detected. • **Any Break**—The activity completes when any one signal break is detected.

Item	Description
Signals	Click + to open the Add Signals window. In this window, select the signal(s) you wish to monitor for a break.
Percent Change	Specify the percentage of the monitored signal's referenced value (Peak or Valley) that represents a specimen failure.
Threshold	Select the threshold that the signal must exceed before the activity begins monitoring for breaks.

Example: Suppose you want to detect a break in either your **Dwell + DAQ + Detection** or **Go To + DAQ + Detection** activity whenever the load signal changes by at least 5%.

In the **Percent change** box, you enter **5** and set the units to **%**. Since you do not want the break to be detected for changes under 20 N, you set the **Threshold** to **20 N**.

Performing Data Acquisition

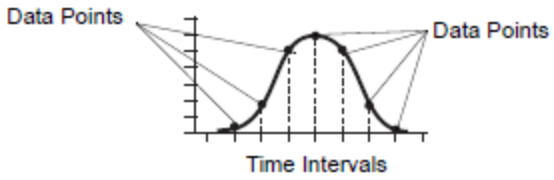
Use the **DAQ** panel to customize how data acquisition is performed in both the **Dwell + DAQ + Detection** and **Go To + DAQ + Detection** activities. You can customize how data is acquired and the sample rate that is used to acquire data.

Triggering Data Acquisition

You can trigger data acquisition to occur in one of two ways:

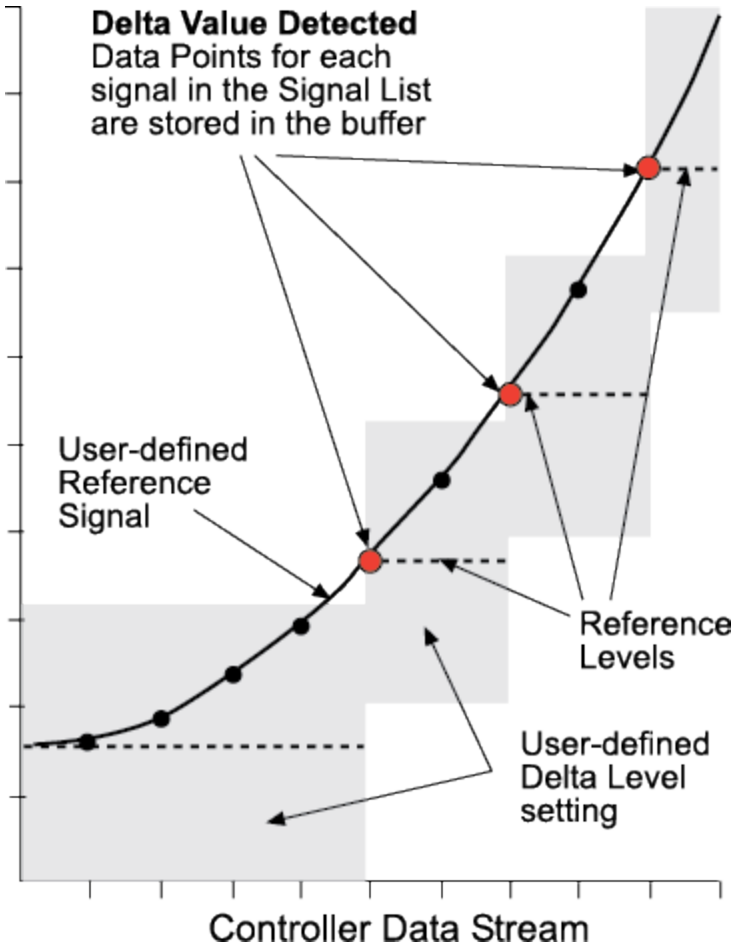
- **Timed** data acquisition records the values of selected signals at a frequency you define (sample rate) for a specified duration. This type of data acquisition is useful if you want to collect data on certain intervals regardless of the feedback received by signals.

Timed DAQ Triggering



- **Delta Level** data acquisition places the selected signal values in the buffer when the reference signal changes by a delta value that you define.

Delta Level Triggering Example



When a value outside of the delta value is detected, the trigger establishes a new reference level and monitors the reference signal for another delta value change from that reference level.

This type of trigger is useful when the reference signal sometimes changes quickly, and at other times changes very little. It stores more data when the signal is changing and less when it is not changing. If you were to perform timed data acquisition on a very long test run, you would acquire a large amount of data during the portion of the test run that is not important to you. However, when using delta level data acquisition, you only acquire data when signal values start to change by a delta level that you specify.

Configure Timed Data Acquisition

To configure timed data acquisition:

1. Select **Acquire data**. If you leave this option unselected, data will not be acquired during the **Dwell + DAQ + Detection** or **Go To + DAQ + Detection** activity.
2. In **How would you like to acquire data?**, select **Timed**.

3. In **Sample rate**, specify the frequency at which data is acquired. For example, with a sample rate of 1 Hz, the buffer acquires data points once each second. The sample rate must be a sub-multiple of the system rate:
    - If you are connected to a controller, the sample rate that you enter is automatically adjusted to be the closest multiple of the system rate.
    - If you are not connected to a controller, you can enter any value; however, when you do connect to a controller, the application checks the system rate and automatically adjusts the sample rate to the closest multiple of the system rate.

    To toggle between entering an specific value and an input, click # or x.

Configure Delta Level Data Acquisition

To configure delta level data acquisition:

1. Select **Acquire data**. If you leave this option disabled, data will not be acquired during the **Dwell + DAQ + Detection** or **Go To + DAQ + Detection** activity.
2. In **How would you like to acquire data?**, select **Delta Level**.
3. In **Signal**, select the signal that will be monitored for triggering the delta level data acquisition.
4. In **Delta value**, specify the amount of change that must occur in the **Signal** before data values are collected on the selected signals. To toggle between entering an specific value and an input, click # or x.

Custom Message Window Activity Overview

The **Custom Message Window** activity shows messages to an operator and records the response from an operator. This activity is used to alert the operator to perform certain tasks, such as insert or remove a specimen or remove an extensometer. It can also be used to notify the user that the test is about to begin or end.

For detailed information about editing the message using the **Message** panel, see “Editing Custom Messages” on page 250. For detailed information about creating custom buttons using the **Buttons** panel, see “Editing Custom Message Window Buttons” on page 251.

User Input

The Custom Message window captures the name of the button you use to acknowledge the message window and assigns it to a specified input. You can later use this input in subsequent activities, such as **If-Else Condition** activity.

Editing Custom Messages

To edit a message, click **Edit message** to show the Custom Message Editor window.

The message window is a basic HTML text editor. You can enter text, images, and hypertext links as well as specify the text size and color, font family, and background color of the message. Use the **Preview** button to preview the message appearance. You can import the entire contents of a valid HTML file as well as cut, copy, and paste content using the operating system clipboard.

Note: The editor does not support dynamic content or images stored on a web server. It also does not support scripting extensions, such as JavaScript. If you import or paste content with unsupported content, it generates error messages when you try to preview the message.

Variables in Messages

To show the value of test variables in a message, click **Insert Variable** located in the Custom Message Editor. Alternatively, you can reference the variable directly using the following syntax:

For a single-value, enter $var_<variable_name>$

Note: If you change the variable identifier name, you must manually change the name in the Custom Message Editor; otherwise, you will not be able to run the test due to a validation error.

Resizing Custom Message Windows

Use the **Window size** panel to customize the size of the custom message in pixels. You can either enter a specific **Height** and **Width** dimension, or you can click the up or down arrows next to the field to increase or decrease the window size.

Editing Custom Message Window Buttons

Use the **Buttons** panel to add, remove, and customize the buttons that are shown on the custom message window.

By default, the message window includes the **Close** button. Custom button options include **Yes**, **No**, **OK**, and **Cancel** in several combinations. You can also set up the Custom Message window to have no buttons with the **No Buttons** option.

Create one or more custom buttons with values that you specify either manually or in a predefined choice list.

The following properties are available when configuring the options on the **Buttons** panel:

Buttons Panel Properties

Item	Description
Buttons	Custom button options include **No Buttons, Yes**, **No**, **OK**, and **Cancel** in several combinations. You can also create one or more custom buttons with labels that you specify. Click **No Buttons** to control the Custom Message window with a parallel activity. When the controlling activity completes, the Custom Message window closes. Click **Customize** to add, modify, and delete custom buttons to the custom message window. When you select **Customize**, the area expands with more options to create custom buttons, which are described below.
Button Alignment	Place the buttons on the lower left, center, or right part of the message window.
Results	Select a input from the list to store the text value of the button that you click. Click the list

Item	Description
input	arrow to see the list of all project inputs defined to hold a string value. **Note:** If you choose a input with a choice list, any existing buttons are replaced with the buttons defined by the choice list.
Add	Add a new button to the message window.
Edit	Edit the selected button label. The Edit Button window has two boxes. Enter the return value for the button in the value field. The value can be a text string or the name of a string variable that you have previously defined. Enter the text for the button label in the text box.
Remove	Remove the selected button.
Remove All	Remove all custom buttons.
Add Separator	Add a separator line between groups of buttons. The buttons align vertically with a separator between each group.
Up Arrow	Move the selected entry toward the top of the list. The entry at the top of the list appears as the left-most button in the message window. Click **Preview** to see the current button arrangement.
Down Arrow	Move the selected entry toward the bottom of the list. The entry at the bottom of the list appears as the right-most button in the message window. Click **Preview** to see the current button arrangement.

Auto Offset Activity Overview

Use the **Auto Offset** activity to apply an automatic offset for a group of selected feedback signals.

For detailed information about configuring the options on the **Auto offset** panel, see "Applying or Removing an Automatic Offset" on page 252.

Feedback Offset

Feedback offset alters the feedback signal used by the controller to zero the conditioner output. External factors such as specimen size, test component forces, and cable length can affect calibrated sensor outputs. To compensate for these static external factors, you can add an offset to the feedback signal. Feedback offset alters the feedback signal used by the controller without shifting the conditioner zero reference. Feedback offset is included in control loop calculations.

Applying or Removing an Automatic Offset

Use the **Auto offset** panel to customize the signals to which the auto offset is applied. Additionally, you can specify how the application handles any errors that are encountered during the auto offset.

Apply New Offset or Remove Existing Offset

The options under **Do you want to apply a new offset or remove an existing offset** allow you to select one of the following behaviors for the **Auto offset** activity:

- **Apply offset to zero signals**—Select this option to automatically adjust the signal to zero based on the current signal feedback reading. For example, if you have heavy fixtures installed on your system, the feedback signal will report a load that is equivalent to the weight of the fixture. If you select this option, an offset will be automatically applied so that the load feedback becomes zero with the installed fixtures.
- **Remove offset from signals**—Instead of applying an auto offset, select this option to clear any offsets that were previously-applied.

Signals

Specifies the signals to which you want to apply the auto offset. Click + to open the Add Signals window. Click − to remove the selected signal.

Note: Applying an offset to an active control mode will generate an error. Depending on the Error Handling method you select, the process can stop or continue by skipping the control mode.

Error Handling

Specifies how the process will respond to an error:

- Continue test and log error
- Stop test and log error

Example: Suppose that you are running a test that requires a heavy grip fixture attached to the crosshead or actuator.

To compensate for the fixturing, you add an **Auto offset** activity to the test procedure prior to any **GoTo** or **Dwell** activities in the test. In the activity properties, you select **Apply offset to zero signals**, and then add the load signal to the **Signals** table. Since you do not want to stop the test if an error occurs, you select **Continue test and log error** under **Error handling**. By doing this, you ensure that the **GoTo** or **Dwell** activities start from a zeroed load signal.

End Test Activity Overview

Use the **End Test** activity to force the test run to end before completing all stages of the test run. When the test procedure reaches the **End Test** activity, workflow activities that follow and are parallel to the **End Test** activity do not execute, and the procedure immediately proceeds to the end of the test. A message is written to the test run log indicating that the test run was stopped due to the **End Test** activity.

If-Else Condition Activity Overview

The **If-Else Condition** activity creates two possible paths for a test procedure based on a conditional expression that evaluates to True or False. If the expression evaluates to True, the test procedure follows the True path. If the expression evaluates to False, the test procedure follows the False path.

The evaluated condition can be the result of a response from the operator, or it can be an evaluation of a specific test value or condition.

Note: The **Condition** must evaluate to True or False or else a validation error will appear. Use logical operators such as "==", rather than assignment operators such as "=". The Calculation Editor provides a list of available variables, operators, and functions.

The two possible paths for the procedure to follow are automatically created when you add the **If-Else Condition** activity to the test procedure. Each path can contain zero or more activities, including **If-Then Condition** and other activities.

For detailed information about editing if-else conditional expressions on the **If-Else Condition** panel, see "Specifying an If-Else Condition" on page 254.

Specifying an If-Else Condition

To specify the conditional expression that evaluates to True or False, click ... (next to the **Condition** field) to open the Calculation Editor. To add activities to either the **True** or **False** paths, simply drag and drop the activity into the desired path from the **Toolbox**, which is located on the left side of the Procedure Editor.To specify which path is followed when the expression evaluates is true, select the path in the **If true, follow** list.

External Device Activity Overview

The **External Device** activity is a program action used to execute one of the commands defined for an external device. These commands can be an output that controls the device or an input that reads data from a device.

For example:

- An output command can set a furnace to a specific temperature.
- An input command can read data from a laser extensometer.

Note:
The external device controlled by this activity must first be created and configured using the External Devices window (available from the **Controller** menu).

Set up an External Device

To set up an **External Device** activity:

1. Prerequisites—before you can configure the **External Device** activity, you must perform these steps:

A. Use the External Devices window (available from the **Controller** menu) to add and configure external devices and their commands.

B. Add the External Device resource to your test.

   On the **Resources** node of the test definition tree, click **Add Resource** and click **External Device**.

C. (Optional) Create the inputs to use with this activity.

   Note: For more information about setting up and configuring external devices, see "External Devices" on page 93.

2. Add the **External Device** activity to your test.
3. Define the properties for the **External Device** activity.

For detailed information about configuring the **External Device** activity, see "Configuring External Devices" on page 255.

Configuring External Devices

The following properties are available when configuring the options under the **External device** panel of the **External Device** activity.

External Device Properties

Item	Description
Device	Select the external device that you want this activity to control.
Command	Select the external device command that this activity will execute. All commands defined in the External Device window are listed.
Command Variable (optional)	Select the input that defines the command value that is written to the device. The list below only appears if the **Command** box for one or more commands in the External Devices window contains the following characters: {0}, {1}, and so on.
Result input (optional)	Select the input where value returned by the device is written. This list only appears if one or more command settings in the External Devices window has the **Supports Return Value** check box selected.
Error handling	Select an error handling option: • Continue test and log error • Stop test and log error

Using Charts

Navigating the Review and Test-run Charts

The following two charts are available on the test definition tree:

- **Test-run chart**—Edit the properties of this chart to configure the display that appears while a test run is in progress.
- **Review** > **Chart**—Edit the properties of this chart to configure the display that appears on the **Review** tab after a test run is finished.

The options available for each of these charts are grouped under four icons. Click the links below for more information about each of the panels available for the charts.

Axis Properties

The following panels are available when you click the  button:

- "Configuring the X Axis" on page 259
- "Configuring the Y Axis" on page 260

Test Run Properties

The following panels are available when you click the  button:

- "Plotting Previous Test Runs" on page 262
- "Editing Limit or Curve Fit Lines" on page 262

Note: Limit or Curve Fit Line functionality is only available on the **Review** > **Chart** node of the test definition tree.

Chart Navigation Properties

The following panels are available when you click the  button:

- "Zooming to Region of a Chart" on page 263

Note: Zoom to region functionality is only available on the **Review** > **Chart** node of the test definition tree.

- "Picking Points on a Chart" on page 264

Chart Color and Title Properties

The following panels are available when you click the  button:

- "Customizing the Chart Title" on page 265
- "Customizing Chart Colors" on page 266

Configuring the X Axis

Access

Define tab > **Test-run chart** node >  button > **X Axis** panel

Define tab > **Review** node > **Chart** node >  button > **X Axis** panel

Overview

Use the options on the **X Axis** panel to determine what is shown on the X axis of the chart. By using one of the several different X axis data measurements available in conjunction with one or more **Y Axis** data measurements, you can easily present the data you acquired from your test run(s) visually.

Example: suppose you want to show a basic load versus extension chart. First, you would select **Extension** under the **Data** list on the **X Axis** panel. Then, in the **Data** field under the **Y Axis** panel, you would click + to add a **Load** signal. If necessary, you could even view more data measurements in relation to the X axis by adding more Y axes to the chart, such as Stress, Strain, or Time.

For more information about configuring the **Y Axis** options, see "Configuring the Y Axis" on page 260.

X Axis Properties

The following properties are available when configuring the options under the **X Axis** panel of the **Test-run chart** and **Review** > **Chart** nodes in the test definition tree.

X Axis Properties

Item	Description
Data	Select the data measurement that will be used for the X axis of the chart. The options available here are based on the float signal resources that are available in your test. These resources are listed under the **Float Signals** drop-down on the **Resources** node of the test definition tree.
Units	Select the units that will be displayed for the selected X axis **Data** measurement. For information about configuring unit sets, see "Units Management" on page 75.
Minimum / Maximum	Enter the **Minimum** and **Maximum** display range values of the X axis. The values you enter here will determine the starting point on the left side of the X axis (minimum) and end point on the right side of the X axis (maximum). To toggle between entering an specific value and an input, click # or x.

Item	Description
	Tip: By entering a lower minimum value and a higher maximum value, you can effectively zoom out and see a larger amount of data on the chart. Similarly, by narrowing the range between the minimum and maximum, you can effectively zoom into a specific area of the chart.
Automatically Scale	Select this option to allow the chart to automatically expand so that it accommodates data outside the display range you specified in the **Minimum** and **Maximum** fields above. If you disable this option, the chart will not expand to show data that exists outside of the display range. Therefore, you may not see all of the data that was acquired during the test run.

Configuring the Y Axis

Access

Define tab > **Test-run chart** node >  button > **Y Axis** panel

Define tab > **Review** node > **Chart** node >  button > **Y Axis** panel

Overview

Use the options on the **Y Axis** panel to determine what is shown on the Y axis of the chart. Unlike the X axis, you can optionally add more than one data measurement to the Y axis of the chart. This can be beneficial if you want to compare multiple data measurements (such as **Load**, **Strain**, or **Stress**) in relationship to an X axis measurement, such as **Displacement** or **Time**. For more information about configuring the X Axis options, see "Configuring the X Axis" on page 259.

Y Axis Properties

The following properties are available when configuring the options under the **Y Axis** panel of the **Test-run chart** and **Review** > **Chart** nodes in the test definition tree.

Y Axis Properties

Item	Description
Data	Click + to open the Y Axis Data window. On this window, you can select one or more data measurements that will be used for the Y axis of the chart. To select a category of input variables based on the dimension (**Length**, **Stress**, **Force**, **Strain**, or **Time**), click the list under **Dimension**. The available Y axis data measurements are based on the float signal resources that are available in your test. These resources are listed under the **Float Signals** list on the **Resources** node of the test definition tree. **Note:** There must be at least one signal in the **Data** list. If there is only one signal, you will not be able to delete it. If you want to change the last signal, click + and select a different dimension or signals in the Y Axis Data window. When you are finished moving the desired data measurements from the **Available** section on the left side of the Y Axis Data window to the **Selected** section on the right side, click **OK**. The data measurements appear under Data on the **Y Axis** panel.
Units	Select the units that will be shown for the selected A axis **Data** measurement. For information about configuring unit sets, see "Units Management" on page 75.
Minimum / Maximum	Enter the **Minimum** and **Maximum** display range values of the selected Y axis data measurement. The values you enter here will determine the starting point on the bottom of the Y axis (minimum) and the end point at the top of the Y axis (maximum). To toggle between entering an specific value and an input, click # or x. **Tip:** By entering a lower minimum value and a higher maximum value, you can effectively zoom out and see a larger amount of data on the chart. Similarly, by narrowing the range between the minimum and maximum, you can effectively zoom into a specific area of the chart.
Automatically Scale	Select this option to allow the chart to automatically expand so that it accommodates data outside the display range you specified in the **Minimum** and **Maximum** fields above. If you disable this option, the chart will only show chart data that exists inside the **Minimum** and **Maximum** axis display range. It will not expand to fit data outside of this range, so you may not see all of the chart data.

Plotting Previous Test Runs

Access

Define tab > **Test-run chart** node >  button > **Previous test runs** panel

Define tab > **Review** node > **Chart** node >  button > **Previous test runs** panel

Overview

Use the options on the **Previous test runs** panel to plot one or more test runs that have already been completed.

When you plot previous test runs on the test-run chart, you can compare the real-time results of your current test run with the results from previous test runs. By plotting previous test runs on the **Review** tab chart, you can easily compare the results of multiple test runs on a single chart, which may help you analyze the results easier.

On the **Test-run chart** node, you can specify that the results of the first test run (select **First test run**) and up to 20 previous test runs (select **Last test run(s)** and select number) display on the same chart. To displace the multiple traces, modify the Y-Offset and X-Offset values.

Editing Limit or Curve Fit Lines

Access

Define tab > **Review** node > **Chart** node >  button > **Limit or curve fit lines** panel

Overview

Use the options on the **Limit or curve fit lines** panel to add or edit a limit or curve fit lines.

Limit or Curve Fit Lines Properties

The following properties are available when configuring the options under the **Limit or curve fit lines** panel of the **Review** > **Chart** nodes in the test definition tree.

Limit or Curve Fit Lines Properties

Item	Description
Limit type	Select one of the following limit types. • Horizontal • Vertical • Slope Intercept, m·x + b • General XY (Single Value or Array) • Indexed Values • Horizontal at Index • Vertical at Index
Y Variable	Select a single value variable or an array variable for which you want data plotted on the Y axis of the chart.
X Variable	Select a single value variable or an array variable for which you want data plotted on the X axis of the chart.
Legend label	Select this option to add a custom legend label to the chart. In the text box, enter the name of the legend label.

Zooming to Region of a Chart

Access

Define tab > **Review** node > **Chart** node >  button > **Zoom to region** panel

Overview

Use the options on the **Zoom to region** panel to configure regions of the chart that you can easily zoom into when analyzing the chart on the **Review** tab. In other words, this feature allows you to set up shortcuts to important places on your chart.

After you add a region, you can easily zoom to the region by right-clicking the chart on the **Review** tab, hovering over **Zoom to Region**, and selecting the region.

Add a New Region

To add a new region:

1. On the table in the **Zoom to region** panel, click  . The Zoom to Region window appears.
2. Enter a **Display name** for the new region. This display name will appear when you right-click the chart on the **Review** tab and hover over **Zoom to Region**.

3. In the **Points After** and **Points After** fields, enter the number of data points (between 1 and 5) before and after the **Indexes** that you want to zoom into when selecting the region. For example, if you enter 5 for both of these fields, the region will contain the index and 5 data points on each side of the index.
4. In the **Indexes** table, click +.
5. In the window that appears, use the arrow buttons to move the desired indexes from the **Available** column to the **Selected** column.
6. Click **OK** to close the Variables Selection window.
7. Click **OK** to close the Zoom to Region window.

Edit an Existing Region

To edit an existing region:

1. On the table in the **Zoom to region** panel, click . The Zoom to Region window appears.
2. Edit the **Points After**, **Points Before**, and **Indexes** settings as desired. For more information about these properties, see "Add a New Region" on page 263.
3. Click **OK** to close the Zoom to Region window.

Delete a Region

To delete a region:

1. On the table in the **Zoom to Region** panel, select the region you want to delete.
2. Click −.

Picking Points on a Chart

Access

Define tab > **Test-run chart** node >  button > **Point pick** panel

Define tab > **Review** node > **Chart** node >  button > **Point pick** panel

Overview

Use the options on the **Point pick** panel to customize point picking options. You can use **Point pick** to select traces in the scope and view the delta and slope of the line drawn through those two points.

Point pick Properties

The following properties are available when configuring the options under the **Point pick** panel on the **Test-run chart** and **Review** > **Chart** nodes in the test definition tree.

Point Pick Properties

Item	Description
Show highlight circle	Select to show a crosshair, circle, and data point value when selecting a data point in the chart.
Show delta and slope Labels	Select to show the values of two points, the difference between their values, and the slope of a line drawn through the two points.
Number of rows of values and slope	Enter the number of rows used to arrange the labels, delta, and slope. Click the up arrow or down arrow to increase or decrease the number of rows. The value range is 1 - 6.

Customizing the Chart Title

Access

Define tab > **Test-run chart** node >  button > **Chart title** panel

Define tab > **Review** node > **Chart** node >  button > **Chart title** panel

Overview

Use the options on the **Chart title** panel to customize the title that appears on the chart.

- The title that you set on the **Test-run chart** node is the title of the chart on the **Monitor** tab (shown while the test run is in progress).
- The title that you set on the **Review** node also appears as the chart title on the **Review** tab, which is shown after the test run is complete. Additionally, this is the title of the chart if you generate a report that contains the test run's chart.

Chart Title Properties

The following properties are available when configuring the options under the **Chart title** panel on the **Test-run chart** and **Review** > **Graph** nodes in the test definition tree.

Chart Title Properties

Item	Description
Title	Enter the title of the chart. For example, if the chart is a load versus extension chart, the title “Load versus Extension” may be appropriate. If the title field is gray and you cannot enter a title, de-select the **Autofill title** checkbox (see below).
Autofill title	Select this check box and the application will automatically populate the chart title based on the X and Y axes. For example, if you create a chart with Extension as the X axis and Stress as the Y axis, the chart title will automatically become **Stress versus Extension**. Clear this check box if you want to enter a custom chart title.
Font size	Enter a font size for the title of the chart.

Customizing Chart Colors

Access

Define tab > **Test-run chart** node >  button > **Chart color** panel

Define tab > **Review** node > **Chart** node >  button > **Chart color** panel

Overview

Use the options on the **Chart color** panel to customize the color of the chart title. For information about customizing the chart title, see “Customizing the Chart Title” on page 265.

Chart Color Properties

The following properties are available when configuring the options under the **Chart color** panel on the **Test-run chart** and **Review** > **Chart** nodes in the test definition tree.

Chart Color Properties

Item	Description
Foreground	Select the foreground color of the chart. This color is used to draw the chart title, legend, and border of the chart.
Background	Select the overall background color of the chart. **Tip:** When selecting a background color for the chart that appears on the **Review** tab, keep in mind that the background color you select here will also appear if you print reports that contain the chart. For example, if you select a very dark color, much more ink will be used to print the chart.

Working with Test Controls

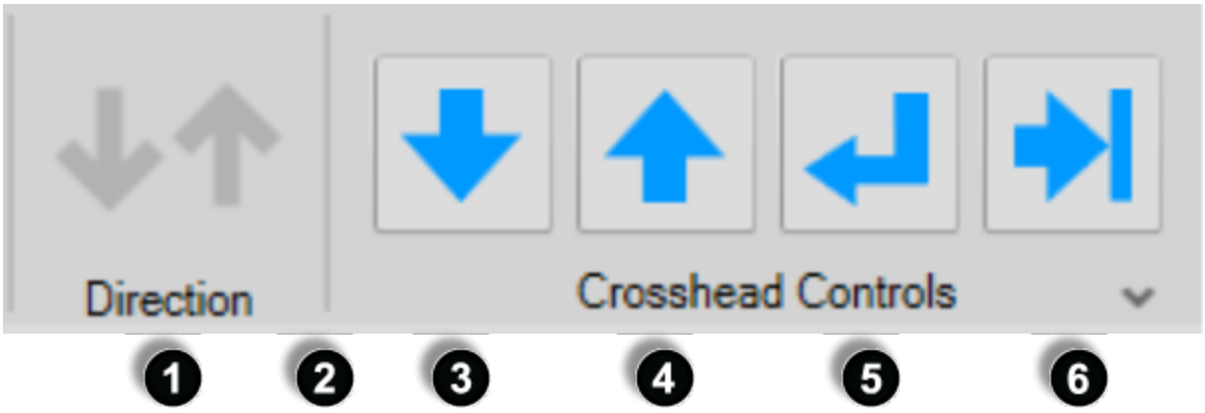
Crosshead Controls Panel for Electromechanical and Static Hydraulic Test Systems

The controls described in this section are only available with MTS Criterion and MTS Insight systems.

Crosshead Controls Panel

The Crosshead Controls provide buttons that allow you to move and position the crosshead: up and down (Jog) buttons, a Return button that moves the crosshead to the Crosshead Return position, and a Go To button that allows you to specify a position and then move the crosshead to that position.

Crosshead Controls Panel



Crosshead Controls

Number	Control	Description
1	Direction	Indicates the direction of crosshead movement.
2	Clutch indicator (not shown)	This only applies to electromechanical systems that have a clutch; servohydraulic and static hydraulic systems do not have a clutch. This indicator shows the clutch that your frame/machine is currently using. If the indicator points up, the machine is in high clutch. If the indicator points down, the machine is in low clutch. If your system has a clutch and the indicator is not visible, open MTS TestSuite Insight Diagnostics, and in the Select Controller pane, select an MTS Insight Renew frame, and select the checkbox to indicate a clutch is being used.
3	Down Jog button	Click to move the crosshead downward. To configure the **Jog Mode** and **Ramp Rate**, click the down arrow in the lower right corner of the **Crosshead Controls** area.

Number	Control	Description
4	Up Jog button	Click to move the crosshead upward. To configure the **Jog Mode** and **Ramp Rate**, click the down arrow in the lower right corner of the **Crosshead Controls** area.
5	Move the crosshead to the return position	Click to move the crosshead to the Return to Zero position.
6	Move the crosshead to a specified position	Click to specify a position and then move the crosshead to that position

Note: The crosshead can also be controlled with a handset. For more information on using a handset, see the product manual that came with your system. When control is provided by a handset, the crosshead controls will be locked and overlaid by the handset exclusive control icon:

Mechanical Crosshead Limits

There are mechanical crosshead limit switches located above and below the crosshead of your machine. The crosshead stops moving when it encounters a mechanical limit. Adjust these limit switches so that they will stop the crosshead before it damages the grips and other fixtures.

For more information

For detailed instructions on how to set the mechanical crosshead limit switches, see the Product Manual provided with your system.

Caution:
Unintended force can be applied to a mounted specimen, grips and fixtures, or other objects in the crosshead path.

Unintended force can injure anyone in its path and damage a mounted specimen, grips and fixtures, or other objects in the crosshead path.

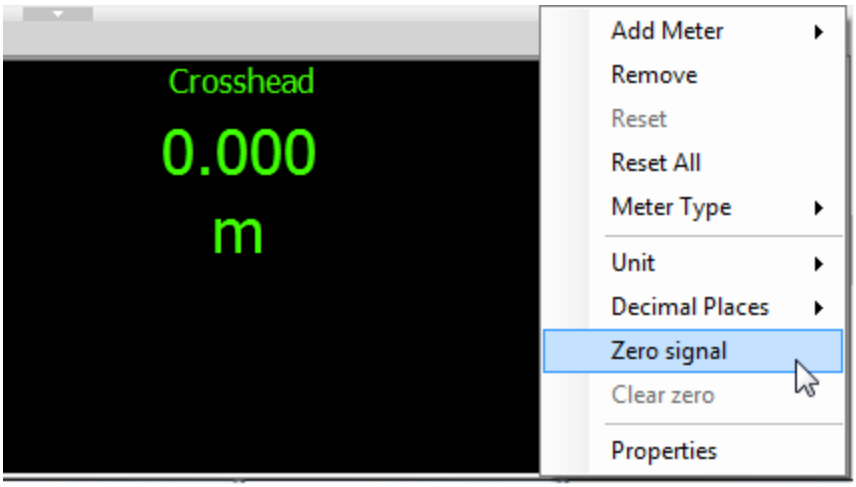
Before you move the crosshead, make sure that the mechanical crosshead limits are set to help prevent the crosshead from damaging grips and fixturing and make sure to clear the crosshead area to help prevent personal injury.

Set the Zero Crosshead Position

To display and configure the Crosshead meter:

1. Display the Crosshead meter if it is not already visible. To display the Crosshead meter, click the **Meter** tab on the bottom of the window. Right-click on a meter, select **Add Meter**, and click **Crosshead**.
2. Click the Jog buttons to move the crosshead to the position that you want to establish as zero.
3. Right-click on the Crosshead meter and on the pop-up window, click **Zero signal**. To remove this offset, click **Clear zero**.

Crosshead Meter Zero Signal



Move the Crosshead Up and Down

Caution:
Unintended force can be applied to a mounted specimen, grips and fixtures, or other objects in the crosshead path.

Unintended force can injure anyone in its path and damage a mounted specimen, grips and fixtures, or other objects in the crosshead path.

Before you move the crosshead, make sure that the mechanical crosshead limits are set to help prevent the crosshead from damaging grips and fixturing and make sure to clear the crosshead area to help prevent personal injury.

1. Display the Extension meter to provide a real-time indication of the crosshead position.
2. Use the mouse to click and hold the Up or Down button to move the crosshead up or down.

   Keep the mouse button depressed for as long as you want the crosshead to move.
3. When the crosshead reaches the desired position, release the mouse button to stop the crosshead.

Move the Crosshead to the Return Position

The crosshead return position is the position the crosshead moves to when you click the Move Crosshead to Return Position button. This position is typically set to a position that allows you to attach a specimen.

Note: Only an MTS FSE can adjust the ramp rate for the Crosshead Return to Zero with the MTS Insight/Criterion (electromechanical) controller. Contact your MTS service representative for assistance.

Caution:
Unintended force can be applied to a mounted specimen, grips and fixtures, or other objects in the crosshead path.

Unintended force can injure anyone in its path and damage a mounted specimen, grips and fixtures, or other objects in the crosshead path.

Before you move the crosshead, make sure that the mechanical crosshead limits are set to help prevent the crosshead from damaging grips and fixturing and make sure to clear the crosshead area to help prevent personal injury.

1. Set the mechanical crosshead limit switches to help prevent the crosshead from damaging the grips and fixtures.
2. Click the Move Crosshead to Return Position button to move the crosshead to the Return position.

Move the Crosshead to a Specified Position

Follow these steps to move the crosshead to a specific position. For example, if the crosshead meter reads 6 inches, and you specify 3 inches as the Go To position, the crosshead moves down until the meter reads 3 inches.

Caution:
Unintended force can be applied to a mounted specimen, grips and fixtures, or other objects in the crosshead path.

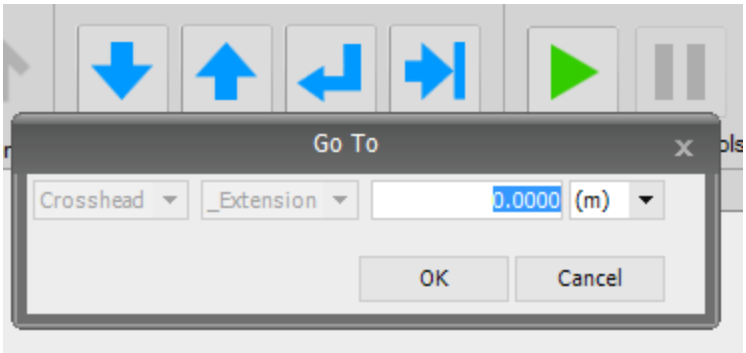
Unintended force can injure anyone in its path and damage a mounted specimen, grips and fixtures, or other objects in the crosshead path.

Before you move the crosshead, make sure that the mechanical crosshead limits are set to help prevent the crosshead from damaging grips and fixturing and make sure to clear the crosshead area to help prevent personal injury.

1. Set the mechanical crosshead limit switches to help prevent the crosshead from damaging the grips and fixtures.

2. Click the Move crosshead to the specified position button to display the Go To window.

Move Crosshead Go To Window



3. In the Go To window, select the appropriate units and type the desired crosshead position. Click **OK** to move the crosshead to the specified position.

   The Crosshead meter updates as the Crosshead moves to the specified position. The Direction arrow flashes and indicates the current direction the crosshead is moving, and the Move crosshead button flashes until the crosshead reaches its specified target.

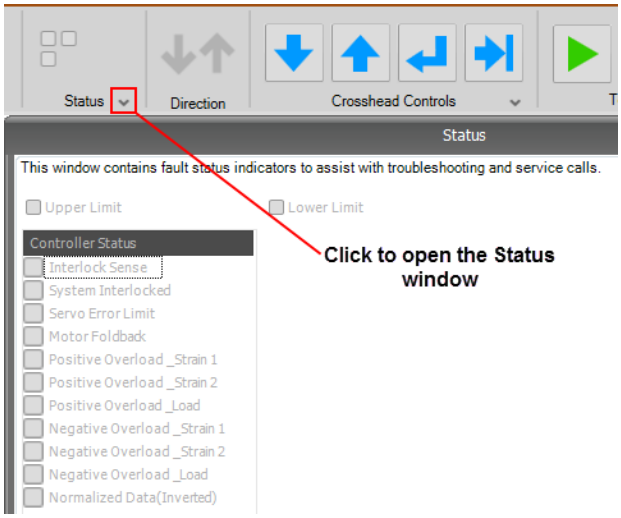
4. When the crosshead reaches its destination, a message indicates the specified target has been reached. Click **OK**.

Status Panel

Note: The following information only pertains to MTS Criterion and MTS Insight.

The Status portion of the control panel indicates the status of various fault indicators, such as Upper and Lower limits and the Enclosure Open switch. A red light indicates a fault indicator has been tripped and must be reset. Hover your cursor over the red light in the Status panel and a message will appear that states which fault indicator has been tripped. To open the Status window and see a list of all fault indicators, click the Open Window icon on the Status panel.

Fault Status Indicator Window



Reviewing and Analyzing Test Results

Review Tab Features

The **Review** tab shows the results of test runs and provides features that allow you to display and analyze data. The amount of data collected for each test run is determined by the **DAQ** (data acquisition) activities included in your test.

Note: Use the **Define** > **Review** node of the test definition tree to configure which parameters (inputs) appear on the Results table of the **Review** tab when test runs are complete. For more information about configuring these parameters, see "Configuring the Results Table" on page 155.

Views and layout features

The **Review** tab has a number of features that define how data appears in the **Review** tab:

- Set the number of display panels that appear in the **Review** tab.
- Add views (charts and tables) to display panels.
- Customize the appearance of charts and tables.
- Select the test run data that appears in charts and tables.
- Create and save displays (settings of the panels and associated views) that you can select.

Data analysis and report features

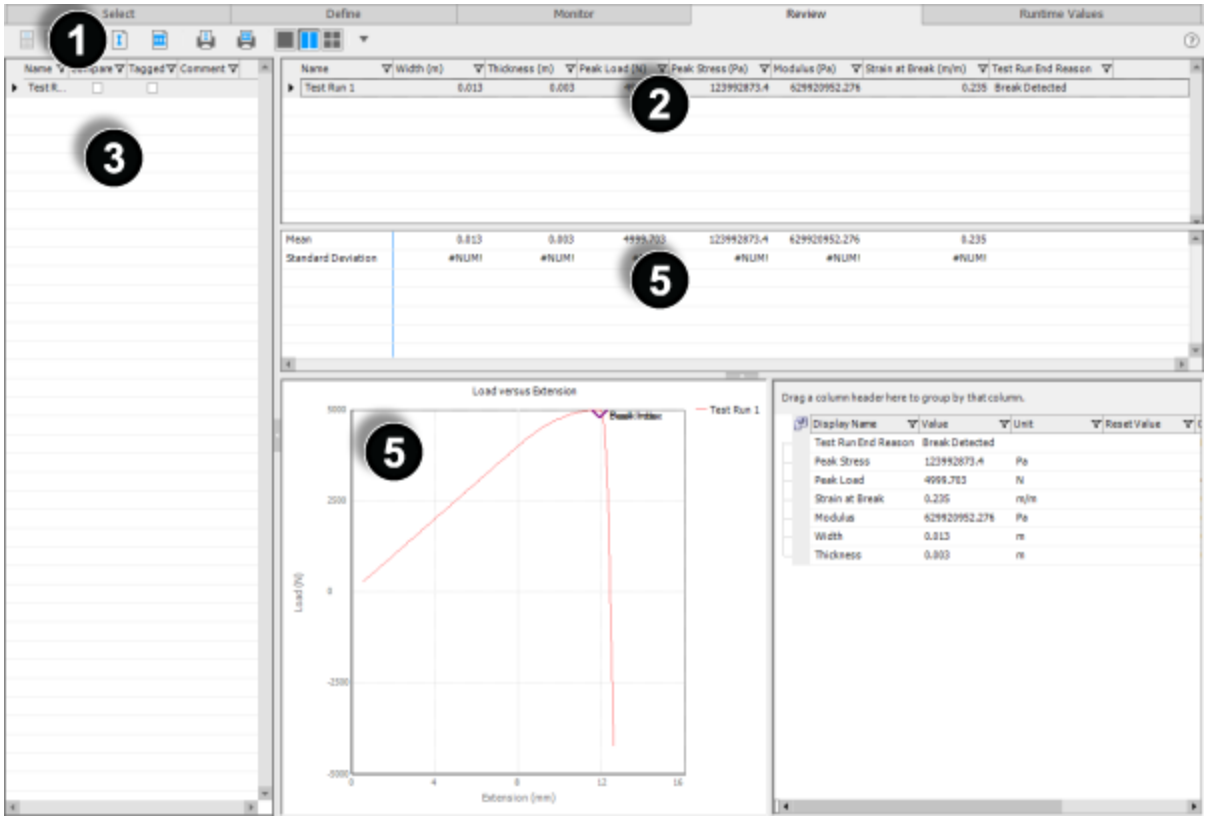
The review tab includes features that allow you to change variable values used to run tests and recalculate the results. These features can be used to correct incorrect operator entries (for example, specimen dimensions) or to create what-if scenarios in R&D applications.

- Select the statistics calculations that are applied to the test.
- Change variable values to recalculate statistics.
- Change chart markers to recalculate statistics.
- Tag test runs (manually or by using the Autotag feature) to remove specific test runs from the test statistics calculations.
- Manually generate test run and test reports.
- Copy data or chart images to the clipboard for use in other applications.
- Enter post-test variables to complete test analysis.

Review Tab Layout

The **Review** tab displays after a test run ends showing test run statistics, charts, and so on.

Review Tab Layout



Review Tab Description

Number	Item	Description
1	Toolbar	Provides icons that allow you skip to the activities in the Finish section of the test, revert and recalculate variables, generate and print reports, and configure and save the layout of the Review tab.
2	Test run results table	Shows test results from individual test runs, allows you to compare one test run to another, and allows you to exclude (referred to as tagging) individual test runs from statistical computations. Each test run row in the Results table includes a Compare check box that controls which test run data appears in the charts and tables.
3	Test run information table	Shows information about each test run, such as what test run information is available in the charts and tables on the Review tab, what test run data is included in calculations, and additional comments for each test run (if available).

Number	Item	Description
4	Test statistics table	The statistics table shows statistical data compiled from all of the test runs that are not tagged (excluded). The statistics calculations are user-defined by right-clicking the statistics area and clicking **Configure Statistics**.
5	Graph	The graph shows a graphical representation of test run or test results. You can add, remove, and edit markers on the graph.

Views

A view is chart or table that you can place in a panel on the Review tab. A panel can contain multiple views but only one view can be active (visible) at a time. The following views are available (the cycle-based views are not shown in the drop-down list if the test does not contain cycle data):

	Description of Views
Variable Table	Table that shows non-array-variables (in a single active test-run) and their properties. For more information, see "Variable Table" on page 283.
Variable Table for Multiple Runs	Table that shows data from two or more analysis runs selected as an analysis set. This table is activated, along with the **Variable-Column Table for Multiple Runs**, when an analysis set is created.
Array-Variable Table	Table that shows array-variables (in a single active test-run) and their properties. For more information, see "Array-Variable Table" on page 286.
Fixed-Column Boundary Table	Table that shows the value and other properties for each cycle (or boundary) of each non-array-variable in a single active test-run.
Data Acquisition Variable Table	Table that shows values for each cycle (or boundary) and each array index of each non-array-variable in a single active test-run.
Variable-Column Boundary Table for Multiple Runs	Table that shows values for each cycle (or boundary) of each variable in the multiple selected test runs. This table is activated, along with the **Variable Table for Multiple Runs**, when an analysis set is created.
Array-Variable Marker Chart for Multiple Runs	Chart that shows array-variable values versus the array index or another array-variable with optional markers.

History Marker Chart for Multiple Runs	Chart that shows values of variables versus the cycles (or boundaries) with optional markers.
Cycle Marker Chart for Multiple Runs	Chart that shows array-variable values versus the array index or another array-variable at specific cycles (or boundaries) with optional markers.
Cycle Time Marker Chart for Multiple Runs	Chart that shows array-variable values versus a time array-variable (where the time values are shifted so the first index is at zero to allow overlaying data) at specific cycles (or boundaries) with optional markers.

Add a View to a Panel

To add a view to a panel:

1. Right-click on a panel.
2. On the **Views** menu, click **Add View**.
3. Select a view type from the pop-up menu.

Switch Views Within a Panel

To switch views within a panel:

1. Right-click on a panel.
2. On the **Views** menu, click **Switch** to **View**.
3. Select the view that you want to display.

Select Test Runs to Appear in Charts and Tables

Active test run

The active test run in the results table will appear in each chart and table. The active test run is indicated by a black triangle in the left column of the results table.

Compare check boxes

The results table includes a **Compare** check box for each test run. These check boxes determine if the data for a test run appears in a chart or table view. If multiple Compare check boxes are selected, a **Test Run** drop-down list appears at the top of each table.

Right-click options

You can right-click on the results table for other test run selection options.

Delete View

To use the Display Manager window to delete a view:

Important: You cannot delete active (visible) displays. Any view that you delete will not appear in any display that included that view.

1. Make sure that the view that you want to delete is not visible.
2. On the **Actions** drop-down list, click **Display** and then click **Open Display Manager**.
3. In the **View** list, select the view you want to delete and click **Delete**.

View Test-Run Data in Chart and Table Views

As each test run is completed, it becomes the active test run in the Results table and its data is shown in the various charts and tables.

To make additional test run data available to view, use the **Compare** check boxes that appear next to each test run in the Results table.

- If none of the **Compare** check boxes are selected, the active test run determines the test run for which data is shown in each table and chart.
- If more than one **Compare** check box is selected, the data for each test run is displayed in all charts.
- If the **Compare** check box for more than one test run is selected, a test-run list (containing all test runs) appears at the top of each table. Use this list to select the test-run data that you want to display in the table.

Note: If the view supports multiple test runs, the data for each test run is displayed in the table or chart.

Copy and Paste Views from One Test to Another

If you create a custom view for a test (for example, a table that only includes certain columns presented in a particular order), you can use the Display Manager window to copy that view from the test and paste it into another test.

1. Use the Display Manager window to copy the view.
    A. Open the test that contains the view that you want to copy.
    B. Click the **Review** tab.
    C. Click the **Actions** drop-down arrow, hover over **Display**, and select **Open Display Manager**.
    D. In the **Views** column, select a view, right click and click **Copy**.
    E. Close the Display Manager window and close the test.
2. Open another test and paste the view into the Display Manager window's **View** column.
    A. Open the test to which you want to add the copied view.
    B. Click the **Review** tab.
    C. Click the **Actions** drop-down arrow, hover over **Display**, and select **Open Display Manager**.
    D. Right-click in the **View** column and click **Paste**.

Table Views

Add and Configure Table Views

To add and configure table views:

1. Add a table to the current display.
    A. Right-click in the chart/table area and click **Views** and then **Add View**.
    B. Select the type of table that you want to add.
2. Customize the columns that appear in the table.
    A. Click the **Column Chooser** icon.
    B. In the Column Chooser window, select the check box for each column that you want to appear in the table.
3. Optional - Rearrange the column order by clicking on a column heading and dragging it to the desired location.

    Note: Rearranging columns in the Statistics table is temporary. The next time you run the test, the order will revert to the order that the test designer set in the application Set Variables Order window.

Add Variables to Statistics and Results Table

Variables whose properties have the **Results** check box selected will appear as columns in the Statistics table and as rows in the Results table.

1. Click the **Test Definition** tab.
2. Click the **Variables** tab.
3. Select the variable that you want to include in Statistics and Results tables.
4. In the Properties panel for the selected variable, select the **Results** check box.

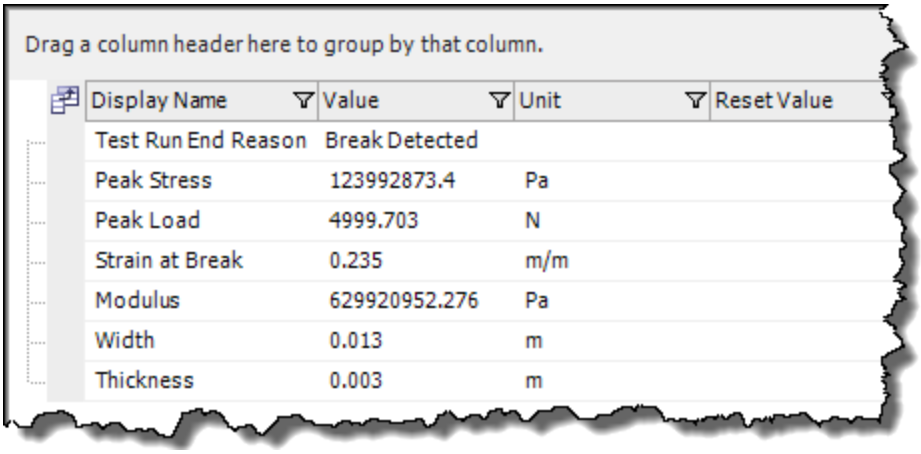
Variable Table

The Variable Table shows the current value of the selected analysis run variables. The table supports direct value overrides that can be used for error correction or “what if” speculation.

You can override values for data not measured from a signal directly in the Value column. However, you cannot override and recalculate calculations, unless you type a number in the Value column. In that case, the calculation is handled as an assigned constant value.

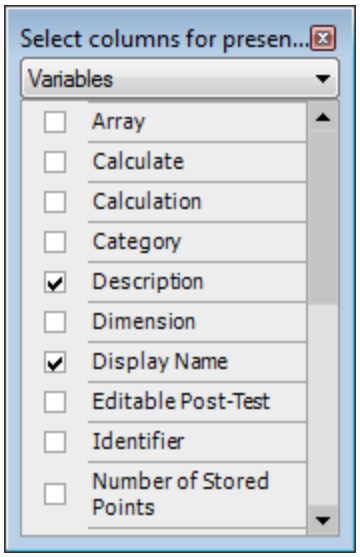
Changes made in the Variable Table are processed only in the current analysis run.

Variable Table



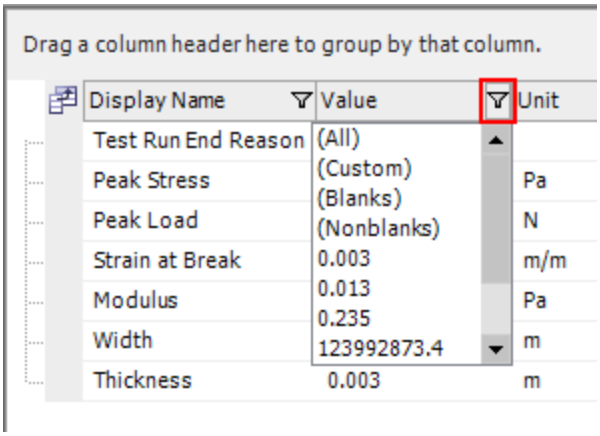
By default, the One Panel View is shown. If you click the Two Panel View button, the second panel for the Variable Table is pre-configured to show the columns selected in the following figure, by default.

Default Columns for Variable Table



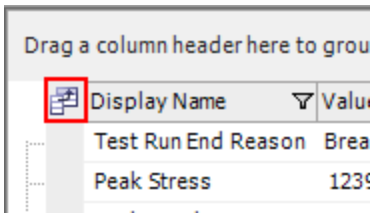
In these tables, You can filter the results that appear in any column. The funnel turns blue when filtering is on; click the funnel to see a drop-down list of filtering options.

Column Filtering



You can configure the columns to be shown for the variables and variable array values in the table by clicking the Column Chooser icon, as shown in the following figure.

Column Chooser Button

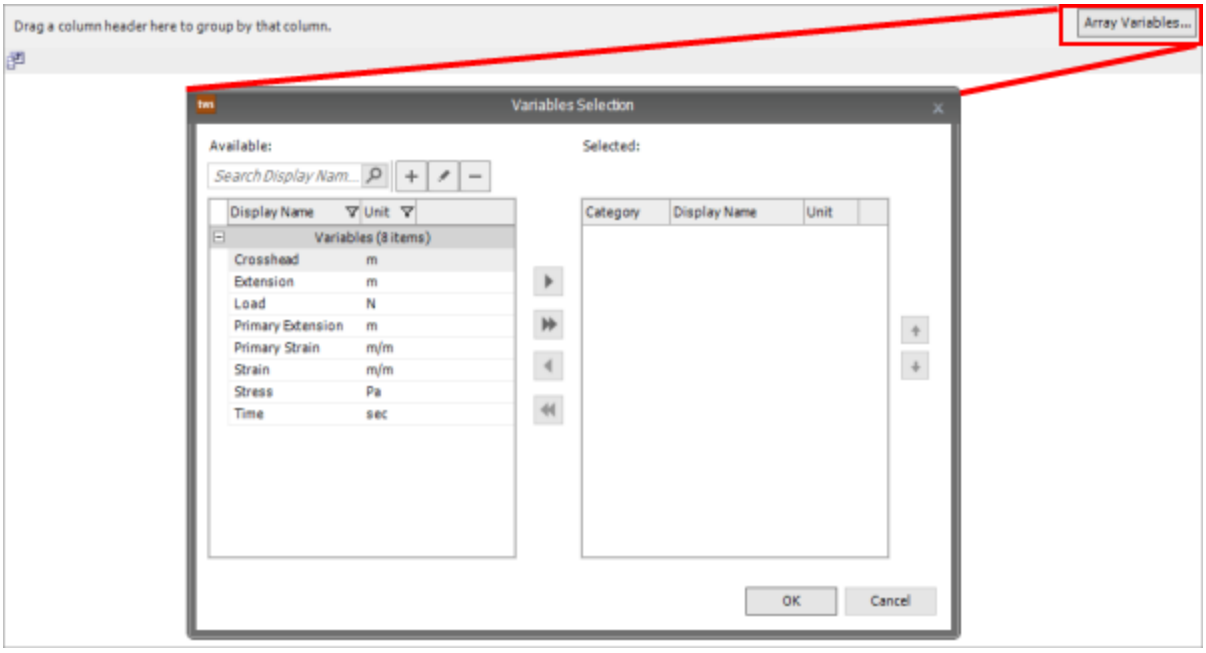


You can override values for data not measured from a signal directly in the Value column. However, you cannot override and recalculate calculations, unless you type a number in the Value column. In that case, the calculation is handled as an assigned constant value.

Array-Variable Table

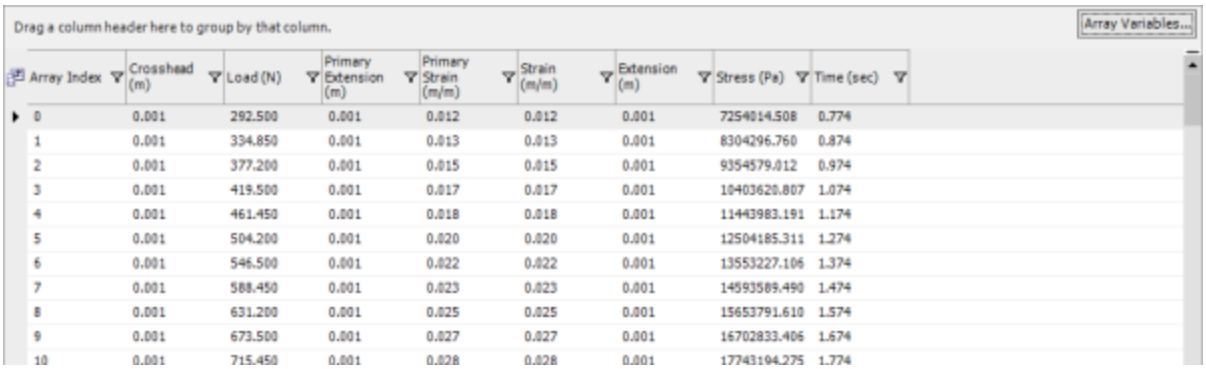
The Array-Variable table shows you the element values in one or more selected array variables. Click the **Array Variables** button to open the Variables Selection window. Use the arrows to move **Available Variables** to the **Selected Variables** box and click **OK**.

Variables Selection Window



- Right Arrow button—Click this button to add a variable that you have selected on the Available Variables panel to the Selected Variables panel.
- Double Right Arrow button—Click this button to add all variables to the Selected Variables panel.
- Left Arrow button—Click this button to delete a single variable from the Selected Variables panel.
- Double Left Arrow button—Click this button to delete all the variables from the Selected Variables panel.

The default left-hand column of the table is the Array Index. Subsequent columns contain the values of the array variables.


Drag a column header here to group by that column.

Array Variables...

Array Index	Crosshead (m)	Load (N)	Primary Extension (m)	Primary Strain (m/m)	Strain (m/m)	Extension (m)	Stress (Pa)	Time (sec)
0	0.001	292.500	0.001	0.012	0.012	0.001	7254014.508	0.774
1	0.001	334.850	0.001	0.013	0.013	0.001	8304296.760	0.874
2	0.001	377.200	0.001	0.015	0.015	0.001	9354579.012	0.974
3	0.001	419.500	0.001	0.017	0.017	0.001	10403620.807	1.074
4	0.001	461.450	0.001	0.018	0.018	0.001	11443983.191	1.174
5	0.001	504.200	0.001	0.020	0.020	0.001	12504185.311	1.274
6	0.001	546.500	0.001	0.022	0.022	0.001	13553227.106	1.374
7	0.001	588.450	0.001	0.023	0.023	0.001	14593589.490	1.474
8	0.001	631.200	0.001	0.025	0.025	0.001	15653791.610	1.574
9	0.001	673.500	0.001	0.027	0.027	0.001	16702833.406	1.674
10	0.001	715.450	0.001	0.028	0.028	0.001	17743194.275	1.774

You can see the units for each selected array variable using the **Column Chooser** button.

Variable-Column Boundary Table

The Variable-Column Boundary table shows the current values of the selected variables. The number of possible table columns depends on the number of data-collection variables and types of indexes. Each row contains data from one cycle or data group.

Variable Table for Multiple Runs

The Variable Table for Multiple Runs shows data from two or more analysis runs.

Values and calculations cannot be changed or recalculated in the table. Values that have been previously changed in an analysis run are indicated by a check mark in the Modified column of the table row.

The table columns available for the table are determined by the analysis definition. For display, columns are removed or added using the Column Chooser.

The number of rows for each analysis run is determined by the number of identifiers in the analysis run. Rows can be grouped, sorted, and filtered for ease of comparison.

The Variable Table for Multiple Runs cannot be saved for future use.

Variable-Column Table for Multiple Runs

The New Variable-Column Table for Multiple Runs shows data from two or more analysis runs selected as a multi-run analysis. The table is typically used for comparing the data of the analysis runs. This table is activated with the Variable Table for Multiple Runs when a multi-run analysis is created.

The table has one row for each analysis run. The table columns are determined by the analysis definition. For display, table columns are added or removed using the **Column Chooser** button.

The New Variable-Column Table for Multiple Runs cannot be saved for future use. However, data can be printed and exported.

Chart Views

Active Test Run

Charts can display data for multiple test runs but activities such as adding and removing markers, moving markers, and adding text annotations are only applied to the active test run in the results table. The active test run is indicated by a black triangle in the left column of the results table.

Add Text to a Chart

To add text to a chart, move text, or attach text to a different data point:

Add text

1. In the Results Table at the top of the Review window, select the test run that you want to annotate with a text callout.

   The black triangle on the left of the Results table indicates the active test-run selection.

2. Right click on the chart near the point of the curve where you want to attach a text box and arrow and click **Add Text**.
3. Use the Edit window to define the text and line attributes.

   The default text is the name of the test run that you selected (the active test run) in the Results table.

Move text

Click and drag the text to reposition it on the chart.

Attach text to a different data point

1. Right-click on the text and click **Attach to different data point**.
2. Move the cross hairs to a point on the curve where you want to attach the text and click.

   When the cross hair is positioned on the test-run curve associated with the text, the color of the cross hair circle will match the color of the box.

3. After the text is attached to the desired data point, right-click on the text box and deselect (uncheck) the **Attach to different data point** option to hide the cross hair.

Add and Configure a Chart

To add and configure a chart:

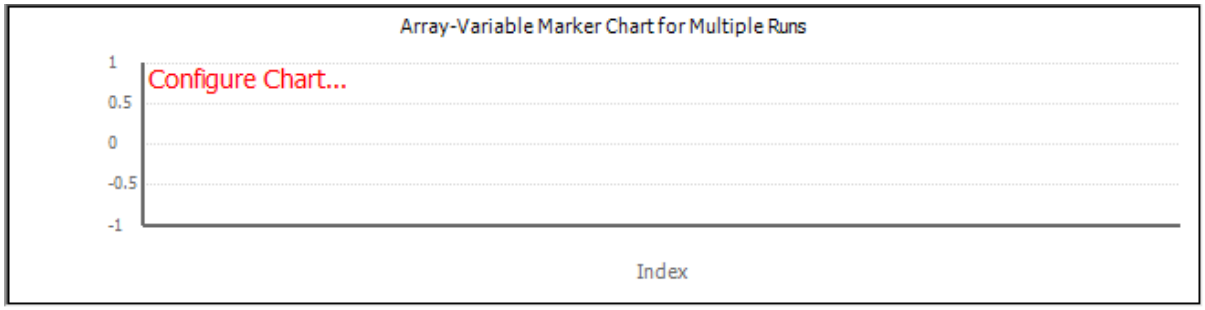
1. Add a chart to the current display.
   A. Right-click in the chart/table area and click **Add View**.
   B. Select the type of chart that you want to add.
2. Right-click and click **Configure Chart**.

Note: For more information about configuring charts, see “Using Charts” on page 257.

Array-Variable Marker Chart for Multiple Runs

The Array-Variable Marker Chart for Multiple Runs shows array-variable values versus the array index of another array-variable. You can also add optional markers.

Array-Variable Marker Chart for Multiple Runs

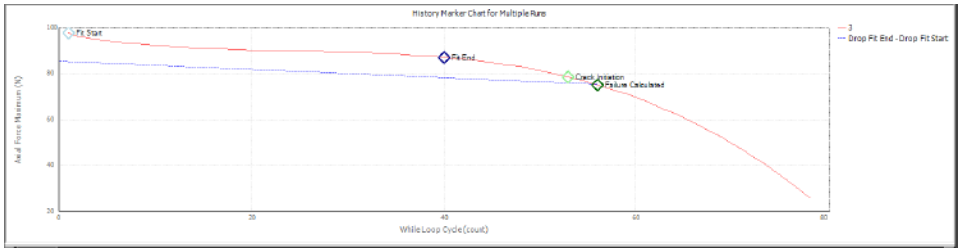


To configure the chart, right-click on the chart and select **Configure Chart**.

History Marker Chart for Multiple Runs

The History Marker Chart for Multiple Runs allows you to add markers and lines to history charts.

Example History Marker Chart for Multiple Runs



Right-click on the chart and select **Configure Chart**.

Markers

Right-click on a marker to open Add or Remove Markers, Edit Marker, Edit Variable, and Move Selected Marker.

Marker Properties

Item	Description
Add or Remove Markers	Opens the Variables Selection window where you can use the arrows to add or remove variables. When you add a variable to the Selected Variables box, it appears on the graph. To remove a marker, right-click on the marker to be removed and select **Add or Remove Markers**. In the Variables Selection window, move the variable to be removed from the **Selected Variables** box to the **Available Variables** box. The marker is removed from the chart.
Edit Marker	Select to change the **Display Name** and **Symbol Color** of the marker.
Edit Variable	Select to open the Edit Variable window where you can modify various aspects of the selected variable.
Move Selected Marker	To reposition the marker, right-click on the marker to reposition and select the **Move Selected Marker**. Move your cursor along the curve. When the cursor touches a data point, an orange circle surrounds the point and shows the X and Y values. To set the marker, click the curve at the data point. The orange circle changes to an orange diamond shape. Right-click and select **Move Selected Marker** to clear the move action. The marker changes to a purple diamond shape. To move the marker to the original location, right-click on the moved marker and select **Reset Marker**.

Cycle Marker Chart for Multiple Runs

The Cycle Marker Chart for Multiple Runs on the **Review** tab panel, shows array-variable values versus the array index or another array-variable at specific cycles (or boundaries). You can also add optional markers.

To add the chart, right-click on the panel and select **Views** > **Add View**. To configure the chart, right-click on the chart and select **Configure Chart**.

Cycle Time Marker Chart for Multiple Runs

The Cycle Time Marker Chart for Multiple Runs on the **Review** tab panel shows array-variable values versus a time array-variable, where the time values are shifted so the first index is at zero to allow overlaying date, at specific cycles (or boundaries). You can also add optional markers.

To add the chart, right-click on the panel and select **Views** > **Add View**. To configure the chart, right-click on the chart and select **Configure Chart**. The Configure Chart window opens.

Analyze Test Results

Results Table

The Results table shows the results of each test run and user-selected statistical calculations that include all non-tagged test runs.

Test runs

Each test run appears as a row in the Results table.

Statistics

At the bottom of the results table are calculated statistical values (such as Mean and Standard Deviation) for the test. The types of statistics displayed and their order is user defined. Statistics calculations include data from all the test runs that are not tagged (excluded) from the test.

Note: The list of available statistics calculations is hard coded and cannot be edited.

Results columns

Each variable whose Results properties check box is selected will appear as a column in the results table. Once a column is added, the value calculated from the test runs appears and is included in any statistical calculations.

Add and Configure Statistics Calculations in Results Table

1. Right-click the statistics portion of the results table and click **Configure Statistics**.
2. Use the Configure Statistics window to add or remove statistics calculations and change the order in which they appear.

Add Columns to Results Table

To add columns to the results table:

1. Click the **Test Definition** tab.
2. Click the **Review** > **Results table** node of the test definition tree.
3. In the Results list, add the variables that you want to appear in the results table.

Change Test Variable Values for Post-Test Analysis

The procedures in the following section assume that you have added the **Variable Table** view to the panels section on the **Review** tab. This table allows you to create what-if analysis scenarios by changing test variable values and observing the test results.

Change Test Variable Values

Important: Only variables whose properties have the **Editable Post-Test** check box selected can be changed in the **Review** tab.

1. Click the **Review** tab.
2. On the Results table, select a test run.

   A black arrow appears to the left of the selected test run.
3. In the Variables table, click in the Value cell of the variable that you want to change and type the new value.

   If necessary, right-click and add a Variable Table view to the display.
4. Click outside the cell to replace the current variable value with the new value.
   - In the Results table, any statistical calculation that includes the modified variable changes to reflect the new value. Any charts that are affected by the new value will also reflect the changes.
   - In the Variable Table, a **Reset** button appears next to the changed variable in the **Reset Value** column. Click this button to undo any changes made to the variable.
   - In the Variable Table for Multiple Runs, the **Modified column** check box for the variable that you just changed is selected.
5. Optional - Generate a report that shows the changes or use the Export Raw Data function to analyze the data in another application.

Revert Changes to Test-Run Variable Values

The **Review** tab provides a number of ways to revert changes that you make to test run variable values.

Reset Value column

When a variable value is changed in a variable table, a **Reset** button for that variable appears in the Reset Value Column.

Click the **Reset** button to revert any changes to the variable.

Revert and Recalculate the Test Variables button

Click the **Revert** and **Recalculate the Test Variables** button to revert all variable changes (in all test runs) and recalculate the statistics in the Results table.

 **Note:** This button can also be used to apply changes made to the Auto Tag rules.

Revert and Recalculate the Test-Run Variables button

1. Click on a test run to select it.

    **Note:** In the Results table, the selected test run is indicated by a black arrow to the left of the test run.
2. Click the **Revert** and **Recalculate the Test Run Variables** button to revert all variable changes in the selected test run and recalculate the statistics in the Results table.

Tag Test Runs

At the end of each test run, the **Review** tab displays to allow you to evaluate the data to ensure that the test run is valid.

- Tagging a test run removes the test run results from the statistics calculations.
- The Autotag feature allows you to create rules that determine when a test run is automatically tagged. The Autotag rules are applied as each test run is executed or can be defined and applied post-test.

Exclude Test Run from Statistical Calculations

To exclude the data of a test run from statistics calculations:

1. Select the test run.
2. Select the **Tagged** check box for the selected test run. When tagged, the test run results are removed from statistics calculations.

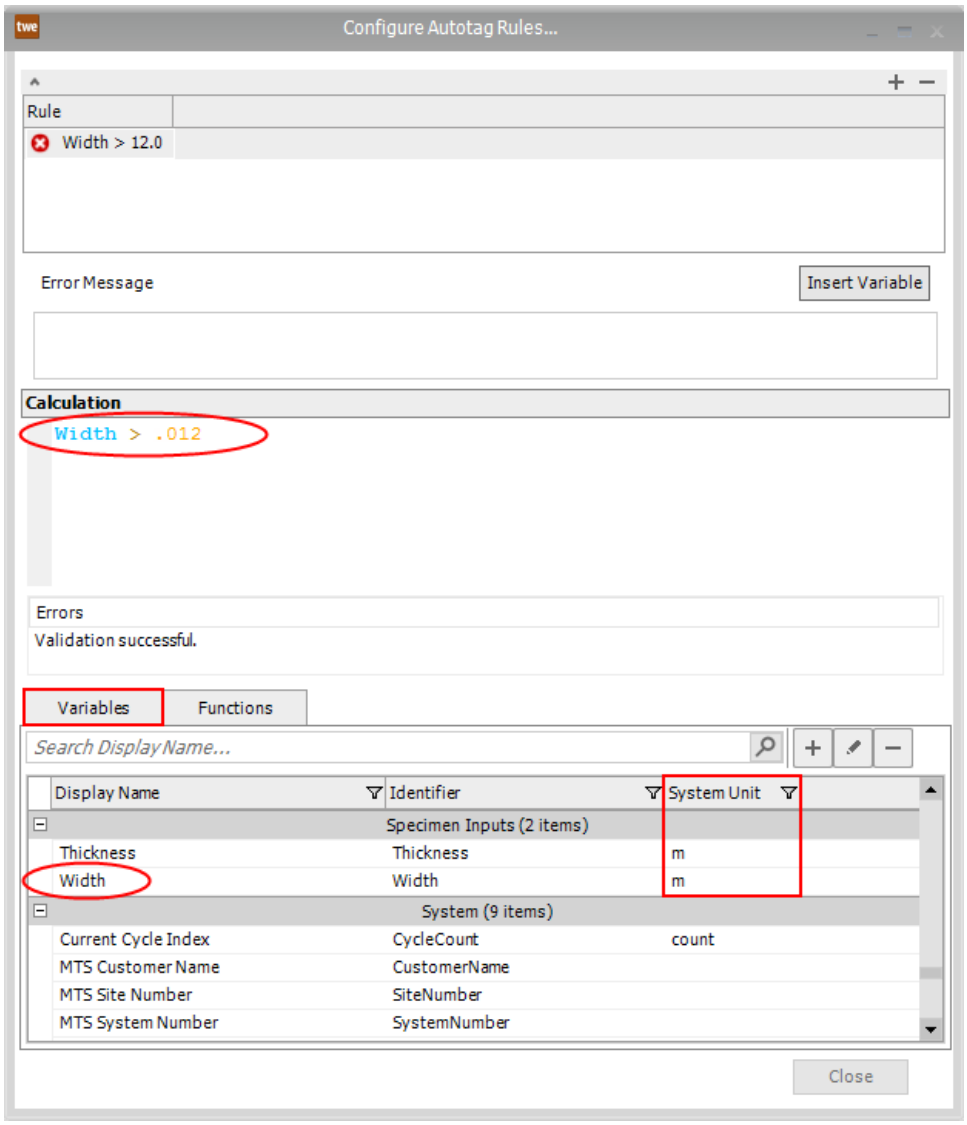
Configure Autotag Rules

To configure an autotag rule:

1. Right-click the Results table and click **Configure Autotag Rules**.
2. Use the Configure Autotag Rules window to define rules that will result in a test run being automatically tagged.
    A. In the **Rule** list, click + to add an autotag rule.
    B. In the **Calculation** list, use the Calculation Editor to define rules that will result in a test run being automatically tagged.

    When the calculation is true for a particular test run, the test run is automatically tagged.
    C. In the **Error Message** box, enter the message that will appear on the **Review** tab when you move the mouse over an autotagged test run.
    D. When done, click **Close** to apply the autotag rule. (You do not need to manually recalculate.)

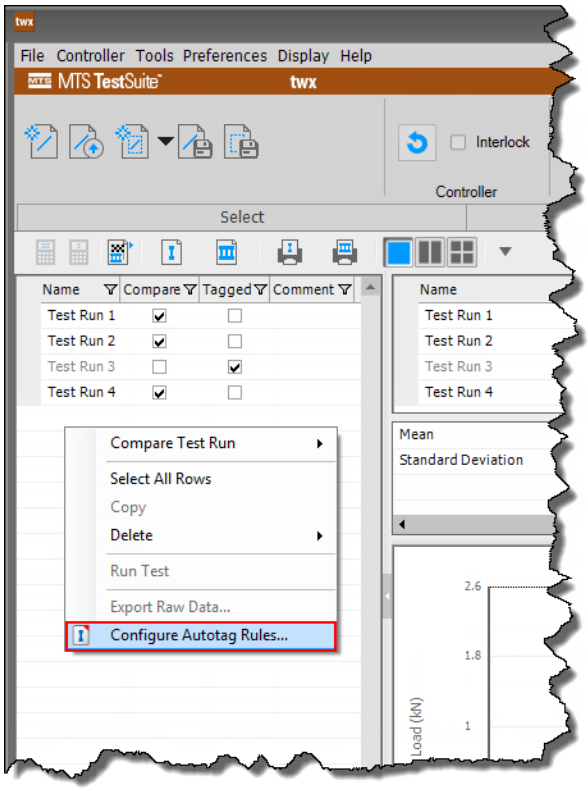
Important: If you get unexpected results, it may be because the autotag rule uses system variable units as opposed to project variable units. Make sure that your rule uses system units. You can find the system units by looking on the **Variables** tab provided at the bottom of the Configure Autotag Rules window as shown in the following illustration.



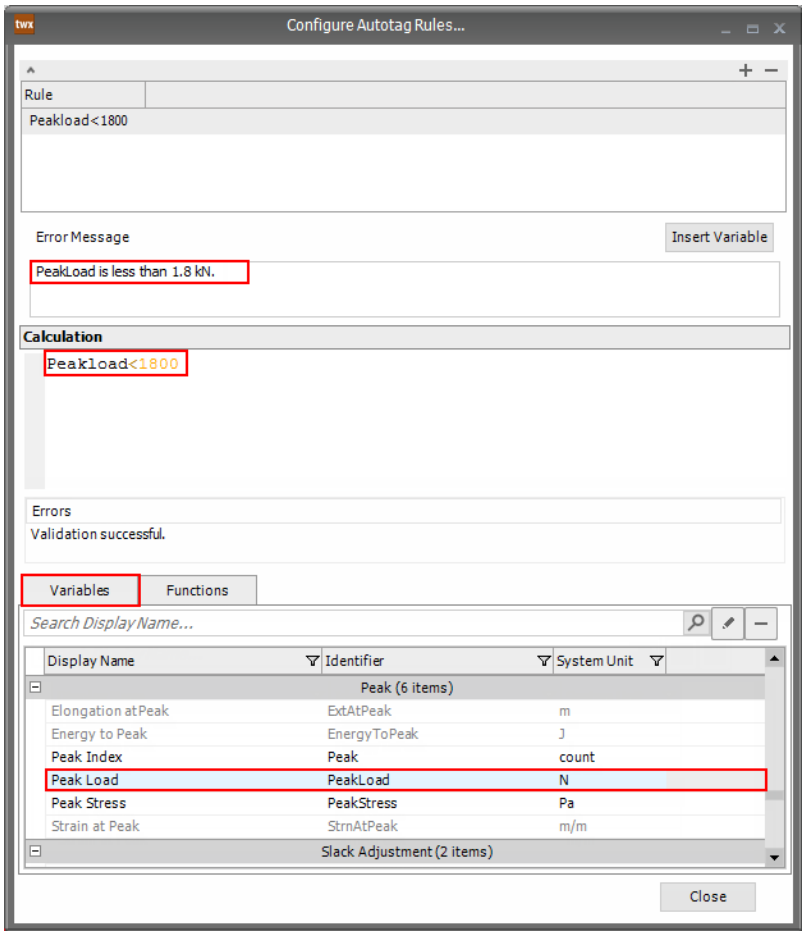
Example

Suppose you want to automatically exclude test runs with grip slippage. For this example, assume that peak loads less than 1.8 kN indicate grip slippage. Setting up an autotag rule will automatically tag those test runs so that they do not affect your statistical results and reports. The following procedure shows how to set up an autotag rule.

1. Open the autotag window by right-clicking anywhere in the leftmost panel and selecting **Configure Autotag Rules**.


File Controller Tools Preferences Display Help
MTS TestSuite
twx
Interlock
Controller
Select
Name
Compare
Tagged
Comment
Test Run 1
Test Run 2
Test Run 3
Test Run 4
Compare Test Run
Select All Rows
Copy
Delete
Run Test
Export Raw Data...
Configure Autotag Rules...
Mean
Standard Deviation
2.6
1.8
1
Load (kN)

2. Set up a rule.



A. Add a rule.

   Click the green + sign in the upper right corner of the Configure Autotag Rules window.

B. Enter a variable.

   Click the **Variables** tab, and then select **PeakLoad** from the list. **PeakLoad** appears in the **Calculation** panel.

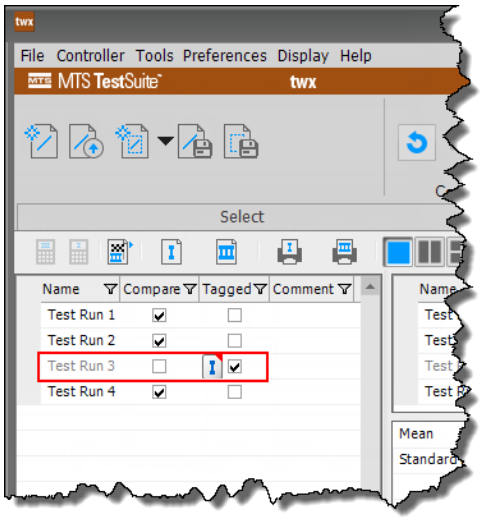
C. Enter a function and a value.

   In the **Calculation** panel, enter **<1800** immediately following **PeakLoad**.

D. Enter an error message.

   Enter **PeakLoad is less than 1.8kN** in the **Error Message** text box.

E. Click **Close**.



Notice that Test Run 3 is autotagged. While autotagging can conveniently tag test runs with numerical data, it will not tag test runs without numerical data. Always review autotagging results to ensure the results are as expected.

Apply Autotag Rules

Autotag rules can be changed and applied before or after test runs.

1. Change or add autotag rules.
2. Click the **Revert** and **Recalculate the Test Variables** button to apply the modified auto-tag rules to all test runs.

User-defined Variables in Auto-Tag Calculations Overview

If your Auto Tag rules contain calculations that contain constant(s) that could change for different specimen or material types, you can create a variable to represent the constant.

For example, if you want to autotag modulus calculations that fall outside of an acceptable range, you could create two variables, one for the lower limit and one for the upper limit and create two auto-tag rules.

Rule Number One: Calculation: Modulus<(Lower Limit Variable) Error Message: “Modulus is below the lower limit.”

Rule Number Two: Calculation: Modulus>(Upper Limit Variable) Error Message: “Modulus is above the upper limit.”

Optional - If you set each variable availability property to Pretest, the operator can enter the value for each variable in the Setup Variables window that appears for each test run.

Important: To avoid unit-conversion issues with calculations, it is good practice to create a variable to hold any constant value (that has a dimension and units associated with it) used in a calculation. You can assign any units that you want to the variable and the application converts them to the base units when the calculation is performed.

Use User-Defined Variables in Auto-Tag Calculations

1. Create variables to represent the constants that you want to use in the auto-tag calculation.

   If you want the operator to enter a value for the auto-tag variable in a Setup Variables window at the beginning of a test: In the variable properties, select the **Pretest** check box.

2. Configure the Auto Tag Rules for the test.

   In the Configure Auto Tag Rules window, the auto-tag variable that you just created appears in the **Variables** list. This variable can be inserted into any calculation that you create.

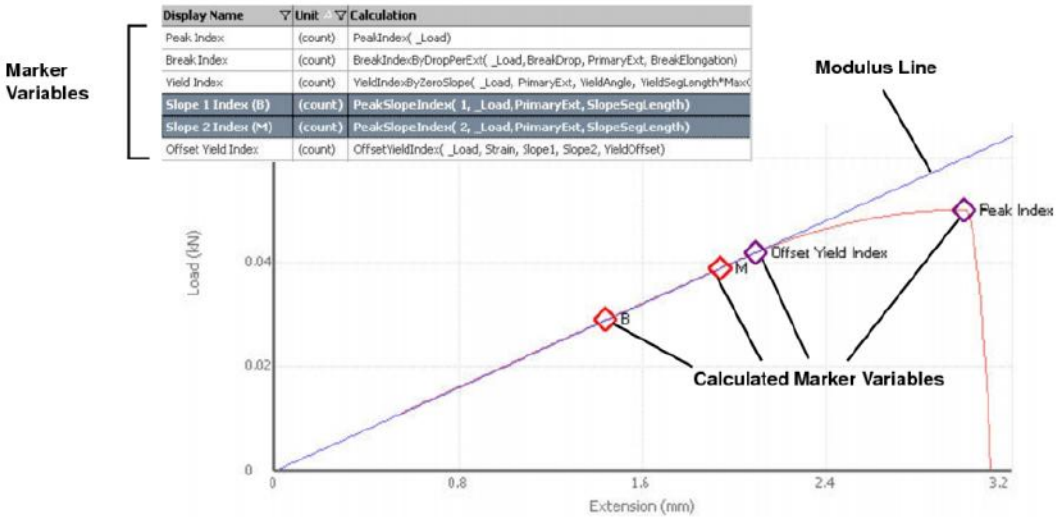
Markers

A marker is a variable that defines a data point in a test run. Selected markers appear in charts as a symbol placed on a data point.

A marker variable can be defined by a calculation or when a run-time event occurs. In either case, the marker variable saves an index into a data array. When the marker is added to a chart, its position represents the x and y axis values associated with the index.

- Calculated - Marker values are typically derived from a calculation and are often used in a calculation. For example, the least squares calculation for the modulus line uses two calculated marker variables (Slope1 and Slope2) that appear on a chart.
- Run-time events- Markers may also be assigned during a test by an operator through a digital input to provide a visual cue that an event has occurred. For example, an operator may create markers by pressing a key on the handset during a pull test to mark an event. A digital input triggered during a test run can also mark an event.

Marker Variables



How are markers used?

Markers are primarily used in post-test analysis, reports, and calculations.

- Run-time markers provide a visual cue that an event has occurred during a test run.
- Yield, break, and peak load are examples of calculated markers that mark events and whose values are used in other calculations.

Can markers be moved?

Marker variables that are configured as Editable Post-Test can be moved to a different location on the curve.

For example, if there is an anomaly in test-run data at a marker data point, you can move the marker to a different location.

Because the marker represents an index value, moving the marker changes its index value and will automatically change the value of any calculation that includes the marker.

Note:
To revert any changes caused by moving markers, click the Revert and Recalculate the Test Variables icon in the Review tab toolbar.

How do you create markers?

Many markers are predefined as part of an MTS template. For example, tensile templates include predefined yield, break, peak-load markers. In addition to predefined template markers, there are other ways to create markers:

- Create a new variable and configure it as a marker.
- Add run-time marker capability to your test so that an operator or digital input can add markers to a test run.

  Note:
  You can create markers after a test run to annotate a chart.

How do you display markers?

Each chart in the **Review** tab has a right-click option to Add or Remove Markers.

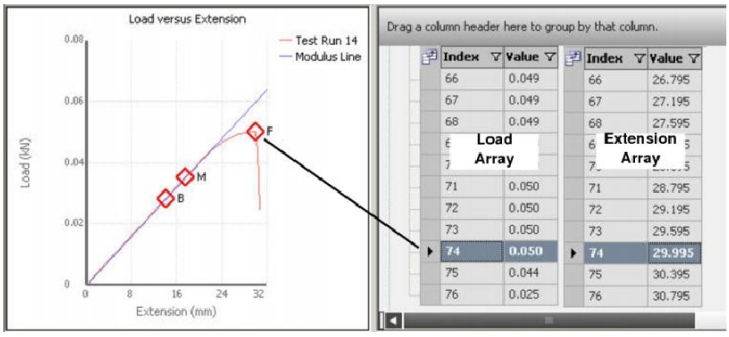
Rules for marker calculations

Calculation results for marker variables must return an index value.

For example the break Index Marker expression BreakIndexByDropPerExt(_Load, BreakDrop, PrimaryExt, BreakElongation) returns the peak index of the specified channel (in this instance, the _ Load channel).

In the following example, the break index marker (F) returned an index value of 74. When the break index marker is added to the Load versus Extension chart, the value for index 74 from the load and extension arrays are used to plot the marker.

Marker Calculations



Marker examples

Peel tests - Move the start and end markers (usually a calculation that calculates the average value between these two markers). You can move the markers to compensate for anomalies that occurred during the test run.

Modulus calculations - Place B and M markers in the chart, the slope of the modulus line is calculated using a least squares algorithm.

Run-Time Markers Overview

Run-time markers are user-initiated markers used to mark an event that occurs while a test segment is performed. For instance, if a specimen begins to tear before it fails, you may want to use a run-time marker to mark the point when the tear began. To use run-time markers, you must modify your test.

Run-time markers are typically set by a digital input triggered by one of the F keys (F1 or F2) on the handset. You can also use hardwired digital inputs to trigger a marker.

Add or Remove Markers from a Chart

To add or remove a marker from a chart:

1. Right-click on a chart and select **Add or Remove Markers**.
2. Use the Variable Selection window to add or remove markers.
3. Click **OK**.

   The changes you made to the markers do not affect the markers in the other charts.

Move Markers in a Chart

You can move chart markers to recalculate statistics and results.

1. Right-click on a marker and select **Move Selected Marker**.
2. Move the cross hairs to the point on the curve where you want to move the marker to and click.

   The circle and the coordinates indicate the position.
3. After the marker is moved to the desired position, right-click the marker and deselect the **Move Selected Marker** selection to turn off the cross hair feature.

Note: To revert any changes caused by moving markers, click the **Revert** and **Recalculate the Test Variables** icon in the **Review** tab toolbar.

Edit Chart Markers

To edit a chart marker:

1. Right-click a marker and select **Edit Marker**.
2. Use the Marker window to change the display name or the marker color.
3. Click **OK**.

Invalid Markers Overview

An invalid (or missing) marker has a value that causes the marker to not appear on a chart.

Moving Invalid Markers

To move an invalid marker:

1. Right-click on the chart that has an invalid marker and click **Move Invalid Markers** and select the marker that you want to move.
2. Move the cross hairs to the point on the curve where you want the marker to and click.
3. After the marker is moved to the desired position, right-click the marker and deselect the **Move Selected Marker** selection to turn off the cross hair feature.

Extract Test Results

Create Test Reports

Reports Overview

The **Review** tab toolbar can be used to manually generate a report for a single test run or for a test (all test runs).

To generate a report, you must select a report template file that defines what information is shown in a test report and how that information is formatted. The report template and the resulting report are Microsoft Excel files.

The MTS TestSuite application creates a test report by opening the report template, populating it with the data from the test run, formatting the data, and saving the report as a separate report file.

Note: A report can be generated without Excel installed. The Reporter Add-In option is required to create report templates.

Report Templates Overview

A report template is an Excel template file that defines what information is shown in a test report and how that information appears. Report templates are created and modified using the MTS TestSuite **Report** tab that appears in the Microsoft Excel application when the Reporter Add-In option is installed.You can design report templates to generate reports for a single test run or for a test (all test runs).

Because the report template defines the information that appears in the report, different test types (with unique sets of variables and other information) may require their own custom report templates. Another use of report templates is to customize the type of information shown for a specific audience.

For more information

For more information on how to create report templates, see the *Reporter Add-In User Guide*.

Select Default Report Template and Report Directories

To select default report template and report directories:

1. From the **Preferences** menu, click **Configuration**.
2. Click the **Project** tab and select the Project that you want to configure.
3. Select the default directories for storing report templates and generated reports.

   Important: If you leave the **Report Directory** setting blank, test run reports are saved with the test run.

Open, Print, Rename, or Delete Report

To open, print, rename, or delete a report:

1. On the **Actions** drop-down list, click **View Reports**.
2. The Reports window lists all of the reports generated for the current test.
3. Right-click any of the listed reports and select the appropriate action to open, print, rename, or delete the report.

Generate a Report for a Test Run

To generate a report for a test run:

1. Select the test-run template that you want to use to create the report.
   A. Click the drop-down arrow next to the Generate a report for the selected test run(s) icon.
   B. Click the report template that you want to use.
2. Click the **Test Run Report** icon.

   The Excel application automatically starts and the test data for the test run is read into the template.

   Important: If you select (highlight) multiple test runs and click the report icon, you will generate a separate report for each selected test run.

Generate a Single Report for the Test (all test runs)

1. Select the test-run template that you want to use to create the report.
   A. Click the drop-down arrow next to the Generate a report that includes all test runs icon.
   B. Click the report template that you wish to use.

2. Click the **Test Report** icon.

   The Excel application automatically starts and the test data for the test run is read into the template.

Extract Data and Images

Copy and Paste Rows

To copy and paste rows from the Results table to another application:

1. Click the **Review** tab.
2. In the Results table at the top of the **Review** tab, highlight the rows that you want to copy.

   If you want to copy all the test runs, right-click on the Results table and click **Select all Rows**.
3. Right-click on the Results table and click **Copy**.
4. Paste the copied cells into another application, such as Microsoft Excel.

Copy Chart Data to Clipboard

To copy chart data to the clipboard:

1. Click the **Review** tab.
2. Right-click on a chart and click **Copy Values**.
3. Paste the copied cells into another application, such as Microsoft Excel.

Copy Chart Image

To copy a chart image:

1. Click the **Review** tab.
2. Right-click on a chart and click **Copy Image**.
3. Paste the copied image into another application.

Save Chart Image

To save a chart image:

1. Click the **Review** tab.
2. Right-click on a chart and click **Save as Image**.

Export Raw Data

You can export raw test-run data as tab- or comma-delimited text (.txt) files that can be used by other applications.

Export Raw Data for a Test Run

To export raw data for a test run:

1. Click the **Review** tab.
2. In the Test Run table, right-click a test run and select **Export Raw Data**.

3. In the Export Raw Data Window, define the **Export Raw Data Properties**. For property definitions, see "Export Raw Data Properties" on page 304.
4. Click **OK**.

Export Raw Data Properties

Export Raw Data Properties

Item	Description
Folder Path	Defines the path to the directory in which the application writes the data file. The default directory path is: "<Data Files>". By default, "<Data Files>" points to C:\MTS TestSuite\Data Files. You can set the default directory path in the Configuration window (**Preferences > Configuration > Project > Data Export Directory**).
Folder Save	Defines whether the application saves the data file in a new folder or overwrites an existing folder.
Format	Defines whether the data is written as tab delimited text or comma separated values.
Unit Set	Defines the unit set in which the data is written.
Data Acquisition List	Defines the DAQ (data acquisition) activities whose data will be included in the data export. Each data export activity selected will result in a separate export file.
Signal List	Defines which signal data will be included in the data export file.
Combine Peak-Valley on One Line	Writes peak and valley values side-by-side in different columns on the same line to facilitate comparison.
Filter Minimum-Maximum Data	Writes a single minimum and a single maximum value to the data file. This feature is useful if you want only the single-most minimum and the single-most maximum values in instances where the application produces multiple values for each (due to looping or restarting the test). If this feature is not enabled, the data file will contain a minimum and a maximum value for each time activity run during the test

Displays

A Display is a convenient way to save the layout of the **Review** tab including the number of panels, panel sizing, and the views associated with each panel. Once saved, use the Display Manager window to select and manage displays.

Save a Display

To save a display:

1. Arrange the layout and appearance of the charts and tables in the **Review** tab.
2. Click the **Actions** drop-down list and click **Display** and then **Save Display As**.

Switch to Previously Saved Display

To switch to a previously saved display:

1. Click the **Actions** drop-down list and click **Display** and then **Switch to Display**.
2. On the **Display** menu, click a previously saved display.

Delete a Display

To delete a display:

1. Click the **Actions** drop-down list and click **Display** and then **Open Display Manager**.
2. Use the Display Manager window to delete a previously saved display.

Note: You cannot delete the active display.

Appendix: Converting and Importing from TestWorks 4

Converting TestWorks 4 Methods and Sample Files

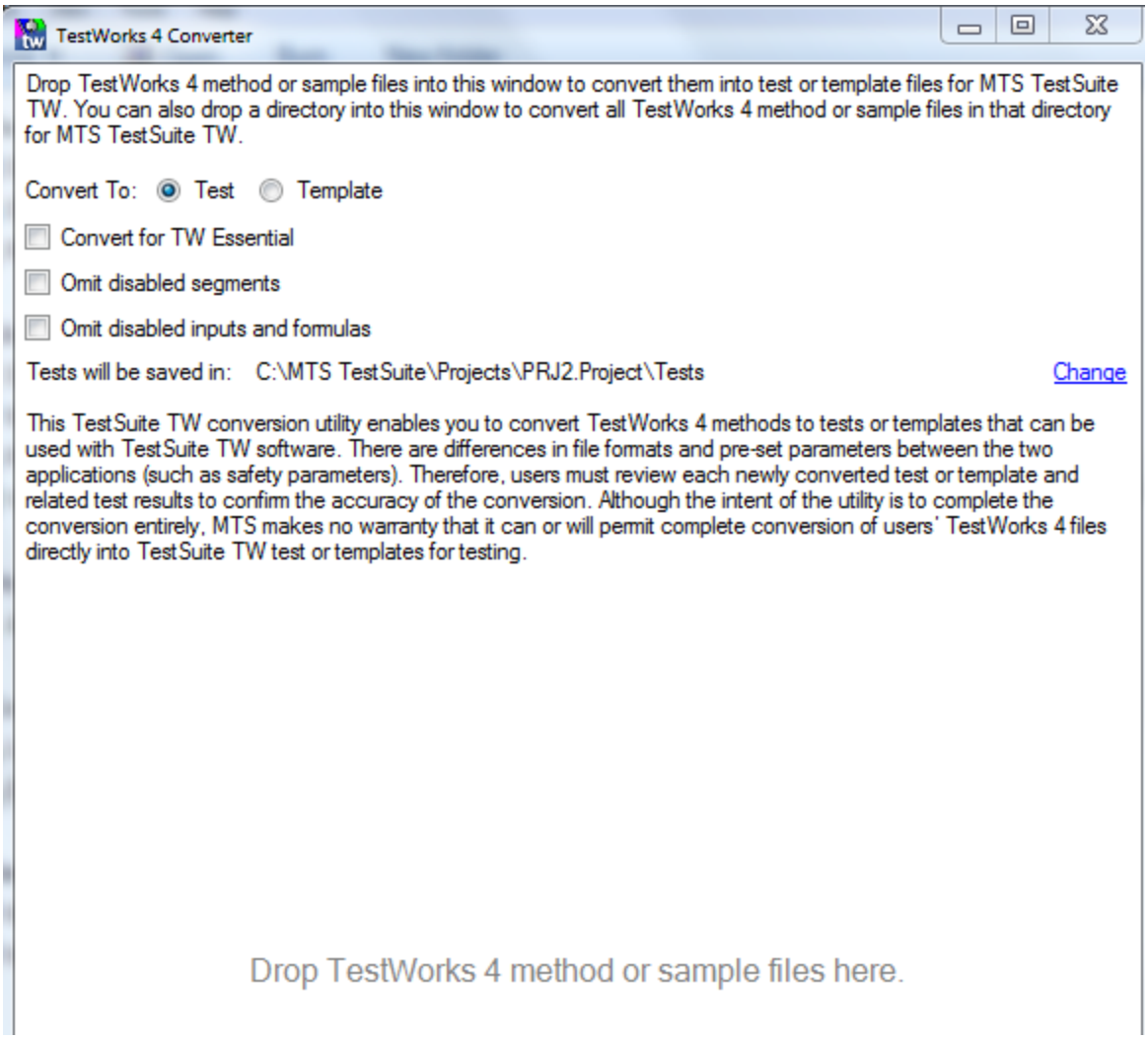
Fastpath:
Start menu > **All Programs** > **MTS TestSuite** > **TestWorks 4 Converter**

The TestWorks 4 Converter allows you to convert TestWorks 4 methods and sample files, or entire folders, so they can be used with the MTS TestSuite TW application.

There are differences in file formats and pre-set parameters (such as safety parameters) between the two programs, therefore, you must review each converted template or test to confirm the accuracy of the conversion. Although the intent of the converter is to provide a completely converted file, MTS does not guarantee that it can or will permit complete conversion of your TestWorks 4 method or sample file into an MTS TestSuite TW template or test.

Converting a TestWorks 4 Method or Sample File

TestWorks 4 Method and Sample File Converter



1. Open the TestWorks 4 converter from Start menu > **All Programs** > **MTS TestSuite** > **TestWorks 4 Converter**.
2. Click the radio button for **Test** or **Template** conversion.
3. If applicable select any of the following options:
    A. **Convert for TW Essential**—Creates a template that is editable in the MTS TestSuite TWE application.
    B. **Omit disabled segments**—Any disabled test segments in the TW4 method will not be converted into the resulting template.
    C. **Omit disabled inputs and formulas**—Any disabled input or formula int he TW4 method will not be converted into the resulting template
4. Save to the default file location, or click **Change** and browse for a different folder.
5. Drag your method or sample file or folder onto the converter window.
6. After the conversion process is complete, you will see a message indicating success or failure. Once converted, the test or template is stored in a sub-folder with the same name in the default file location.
7. Open the converted test or create the test from the converted template and verify the conversion results.

Importing TestWorks 4 Text Files

Fastpath:
Start menu > **All Programs** > **MTS TestSuite** > **TestWorks 4 Converter**
The MTS TestSuite applications allow you to import information from a TestWorks 4 text file, such as a LIMS file, into a TW test template to create a TW test file. You can use the TestWorks 4 conversion utility to first convert your TestWorks 4 method to a TW template.

After importing the text file, the TW test template will be populated with the imported information for the correct number of test runs, global variable values, test run variable values, and the correct number of points in array variables in the test runs. As with all import and conversion procedures, you should review and/or run the test after the import process to verify that all information was imported properly.

Note: When you import the file, the imported resources may not map to the controller resources of the control system networked with your session or workstation. You will have to correct these resource validation errors before using the test. For more information about resources, see "Working with Resources" on page 82.

Before you import the file, verify the following:

- The MTS TestSuite template must have the same name as the TestWorks 4 method and be placed in the templates folder (by default, it is C:\MTS TestSuite\Templates).
- TestWorks 4 Global variables must be found in the Common variables section in the loaded template or an error will be generated and the import process will stop.

Importing a TestWorks 4 Text File

1. On the TW application, select **File** menu > **New** > **Test from TestWorks 4 text file**. The TW File Open window opens.
2. Select a TestWorks 4 text file. Click **Open**. A progress window appears as the new test is created. Some errors or warnings may be generated during the import process, which will be logged at the end of the process. If one or more errors is found in the Global Variables section of the text file, the test will not be created. If one or more errors is found in the Specimen (Test Run) section of the text file, the corresponding Test Run will not be added to the test.
3. After the new MTS TestSuite test has been created, it opens in the TW application workspace where you can test its validity.

Appendix: Eurotherm 2000 Series Temperature Controller

Overview

Configuring MTS TestSuite TW to communicate with a Eurotherm 2000 Series Temperature Controller consists of five steps.

1. Connecting to the computer
2. Configuring the external device file
3. Importing controller resources
4. Adding meters
5. Adding temperature control to the test flow

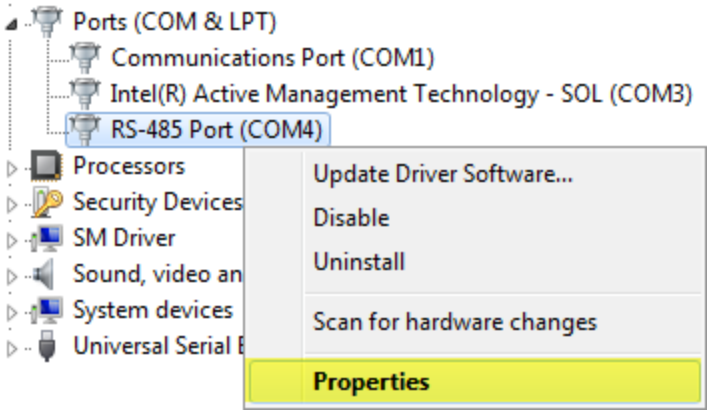
Note: To communicate with a single Eurotherm 2000 Series Temperature Controller, use **Configuring the External Device File for EI-Bisynch Single Zone**. To communicate with more than one Eurotherm 2000 Series Temperature Controllers, use **Configuring the External Device File for EI-Bisynch Multi-Zone**. Configured external device files can be exported and imported onto a different system.

Connecting to the Computer

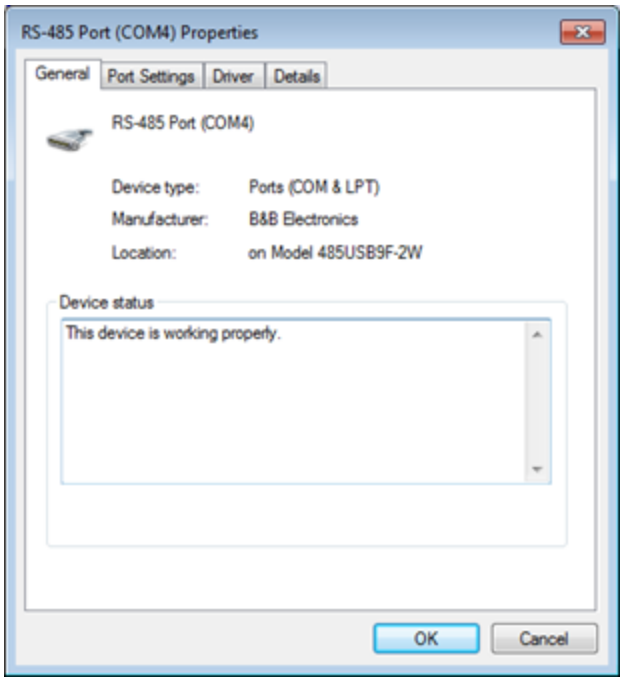
Connect the serial cable from the Eurotherm 2000 Series Temperature Controller to a serial port on your computer. If you are connecting to your computer through a B&B USB-to-Serial Converter such as MTS Part Number 100-306-657, see the following instructions.

To connect to your computer using a B&B USB-to-Serial Converter:

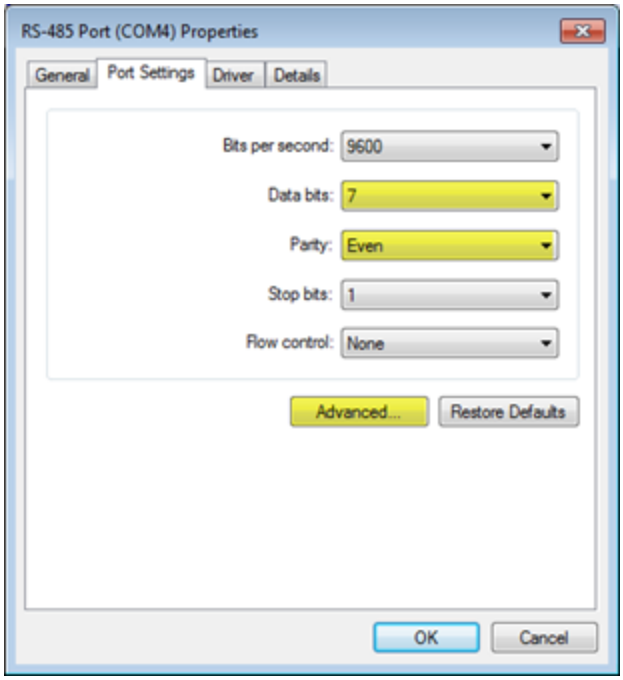
1. Before connecting the B&B USB-to-Serial Converter (B&B) to the computer, insert the USB Serial Driver CD.
2. Install the serial driver
3. Remove the USB Serial Driver CD
4. Open Device Manager in Windows.
5. Plug in the B&B.
6. Right-click the port with the B&B connection to configure the **Properties**.



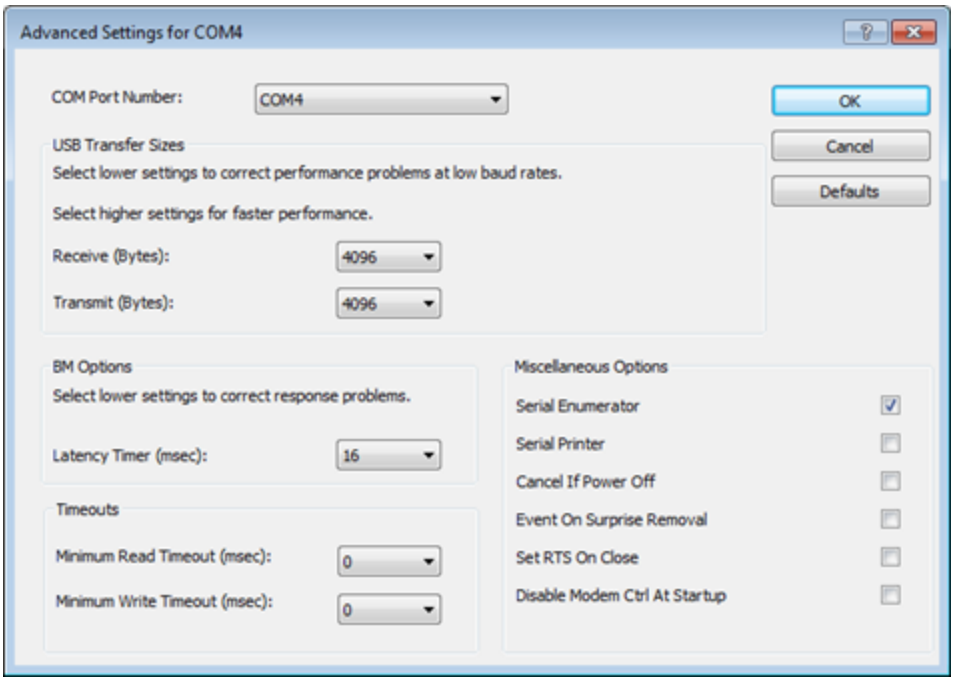
7. Verify that the B&B is working properly.



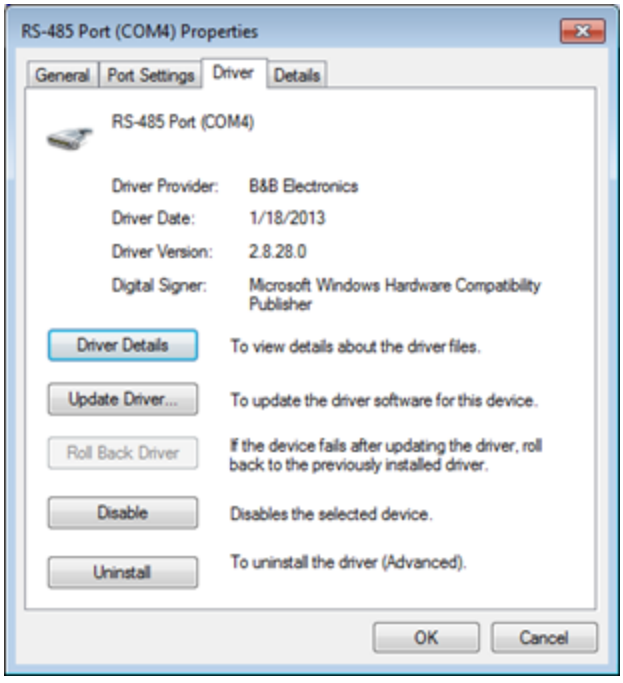
8. Change the **Data bits** to 7, the **Parity** to Even, and click **Advanced**.



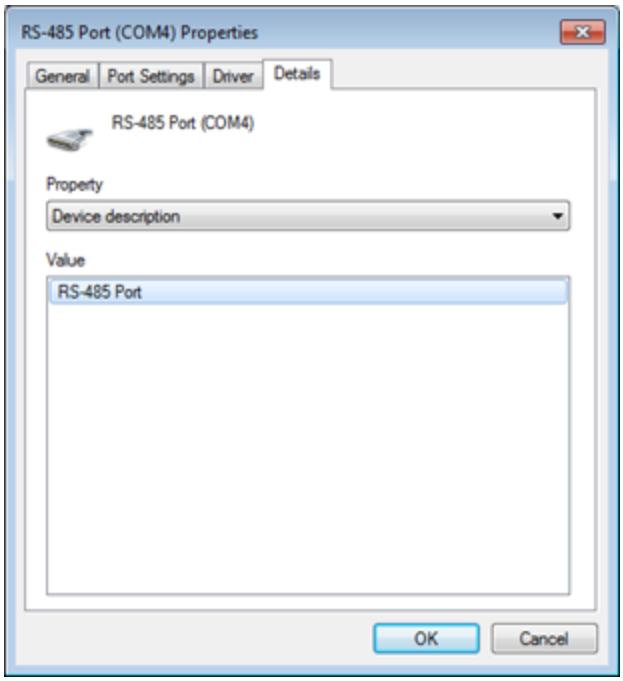
9. Verify that the advanced settings match the following image.



10. Verify that the **Driver** tab is similar to the following image.



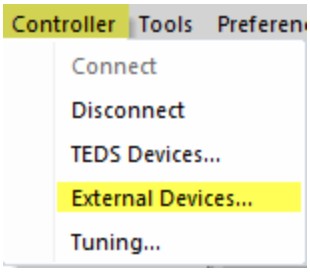
11. Verify that the **Details** tab is similar to the following image.


RS-485 Port (COM4) Properties
General
Port Settings
Driver
Details
RS-485 Port (COM4)
Property
Device description
Value
RS-485 Port
OK
Cancel

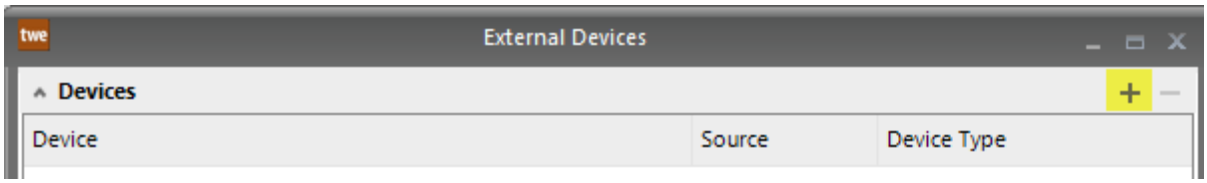
Configuring the External Device File for EI-Bisynch Single Zone

To configure an external device file to communicate with a single Eurotherm 2000 Series Temperature Controller using the EI-Bisynch serial communication protocol:

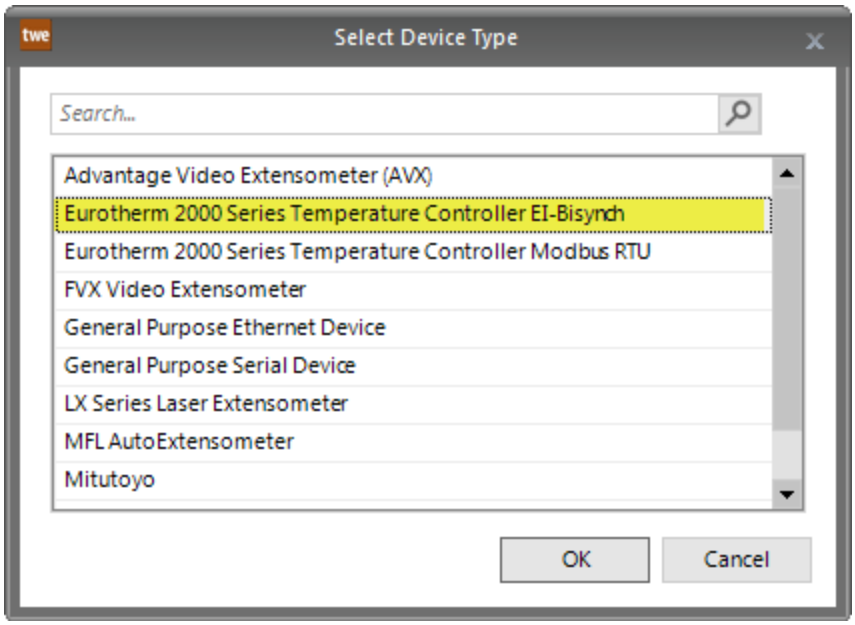
1. On the **Controller** menu, click **External Devices**.



2. In the External Devices window, click the plus sign to add an external device file.

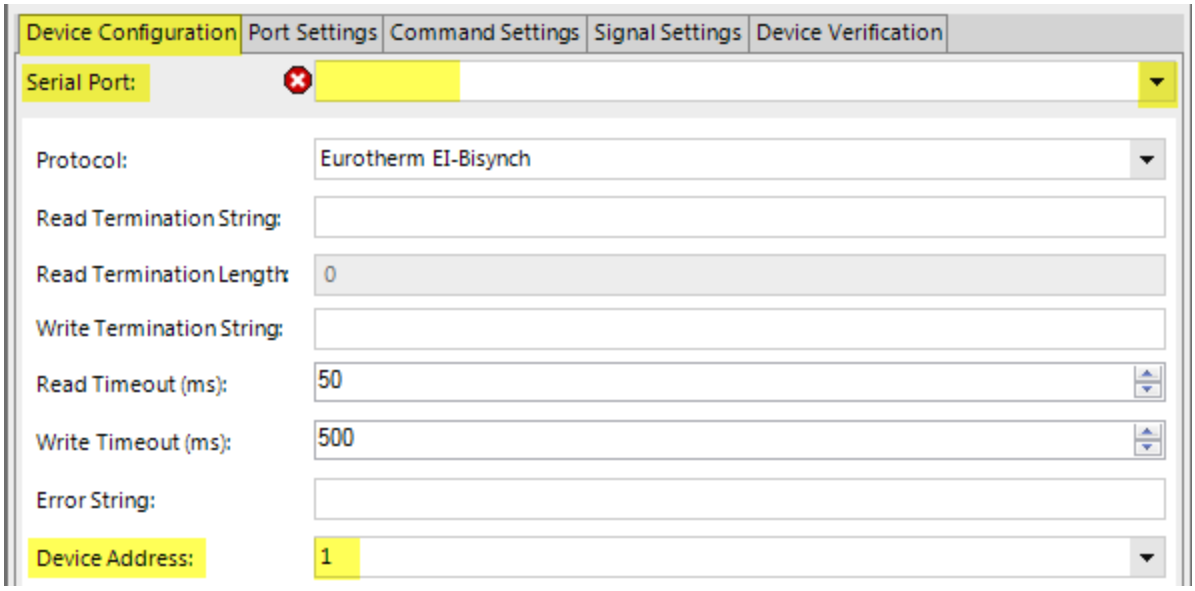


3. In the Select Device Type window, select **Eurotherm 2000 Series Temperature Controller EI-Bisynch**.

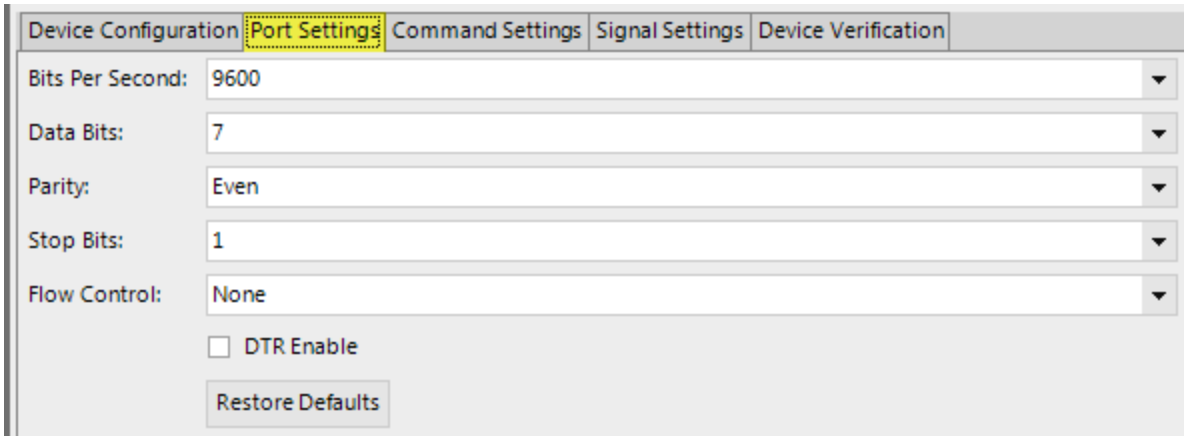


4. On the **Device Configuration** tab, use the drop-down menu to select the appropriate serial

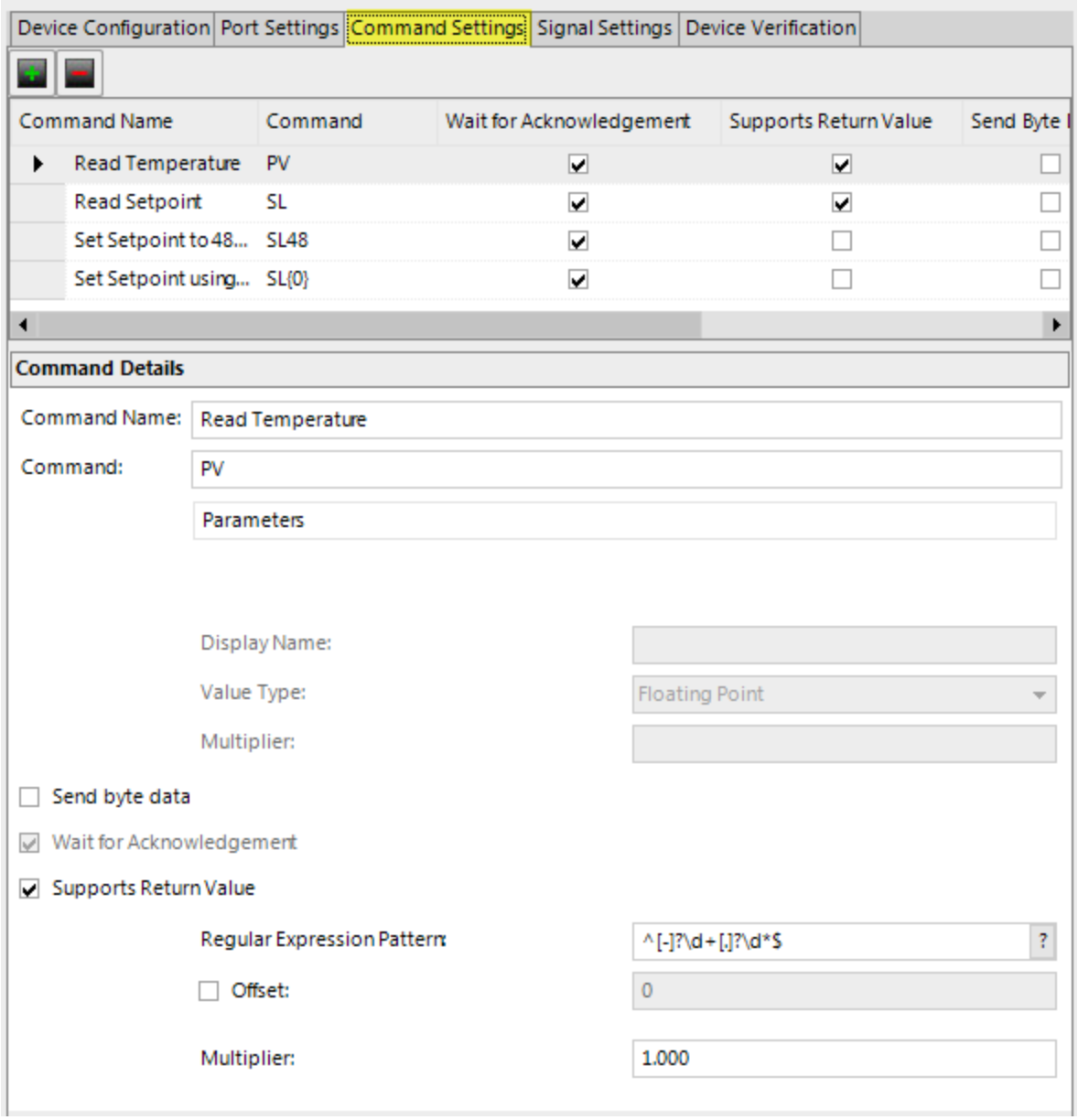
port. Verify the **Device Address**. Typically, for a single zone system the **Device Address** is 1.



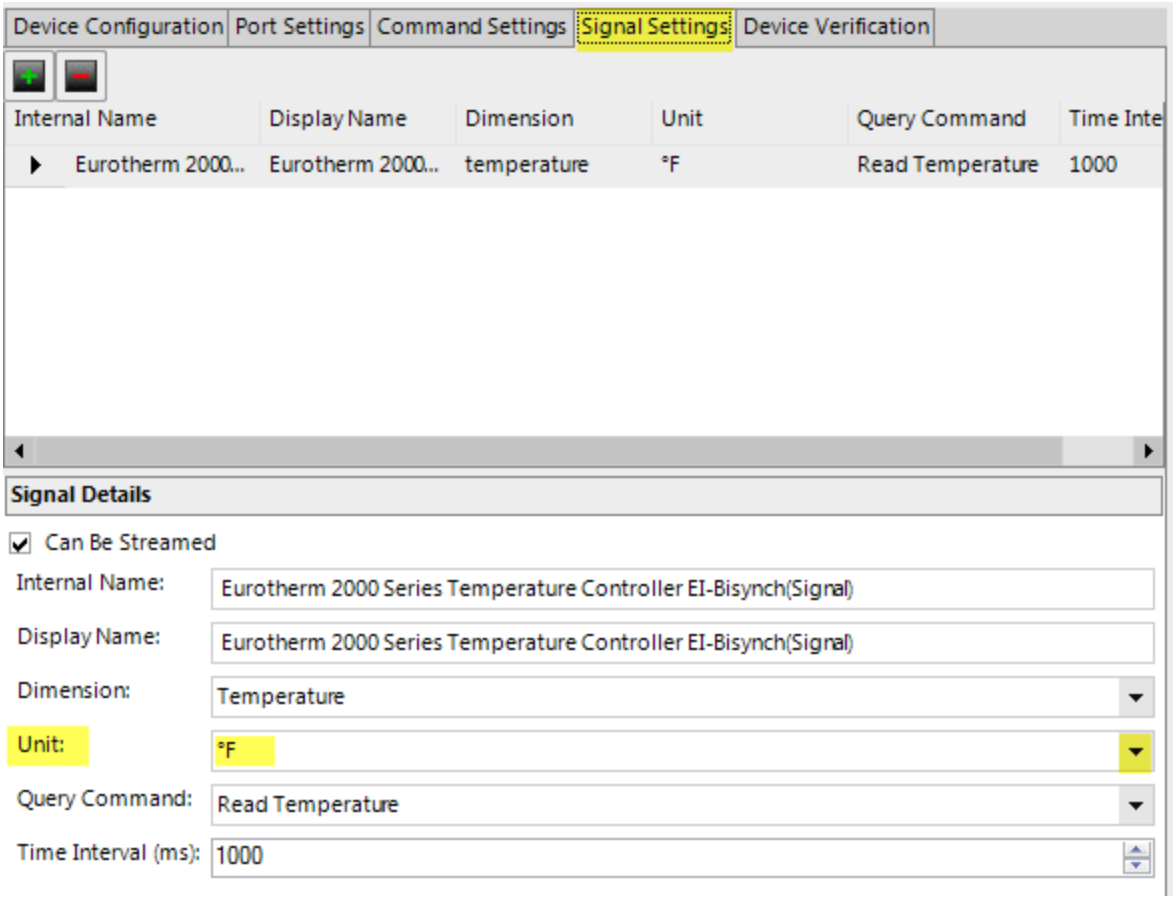
5. On the **Port Settings** tab, use the default values.



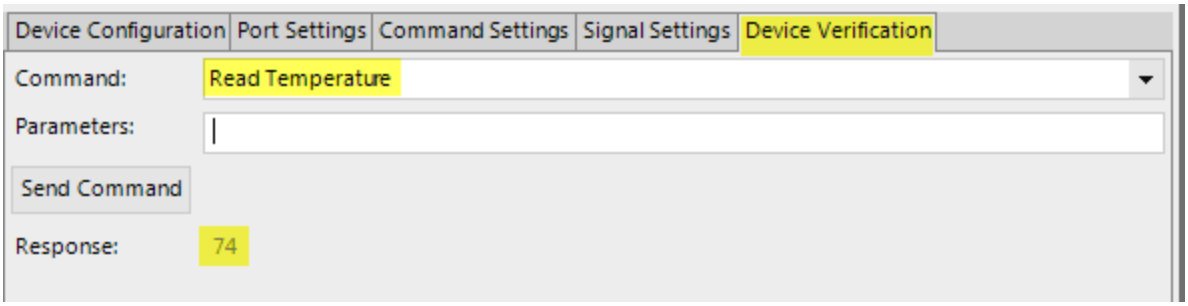
6. On the **Command Settings** tab, use the default commands.



7. On the **Signal Settings** tab, verify that the **Unit** selection is the same as the Eurotherm units.



8. On the **Device Verification** tab, select the **Read Temperature** command and click **Send Command** to verify communication.

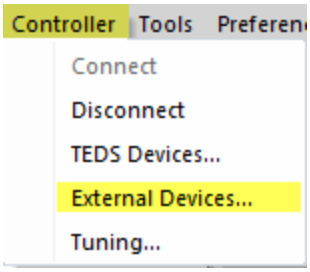


Configuring the External Device File for EI-Bisynch Multi-Zone

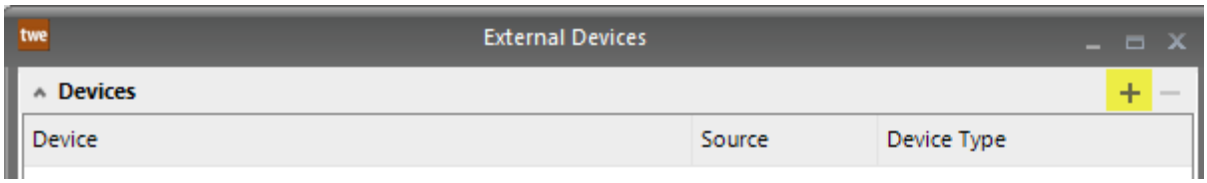
To configure an external device file to communicate with multiple Eurotherm 2000 Series Temperature Controllers using the EI-Bisynch serial communication protocol:

Note: Each Eurotherm 2000 Series Temperature Controller controls one temperature zone.

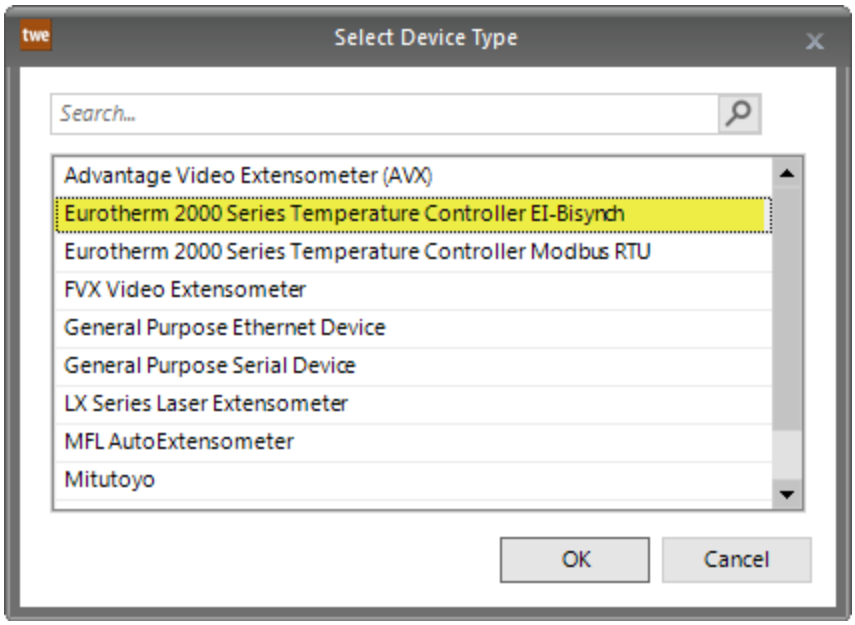
1. On the **Controller** menu, click **External Devices**.



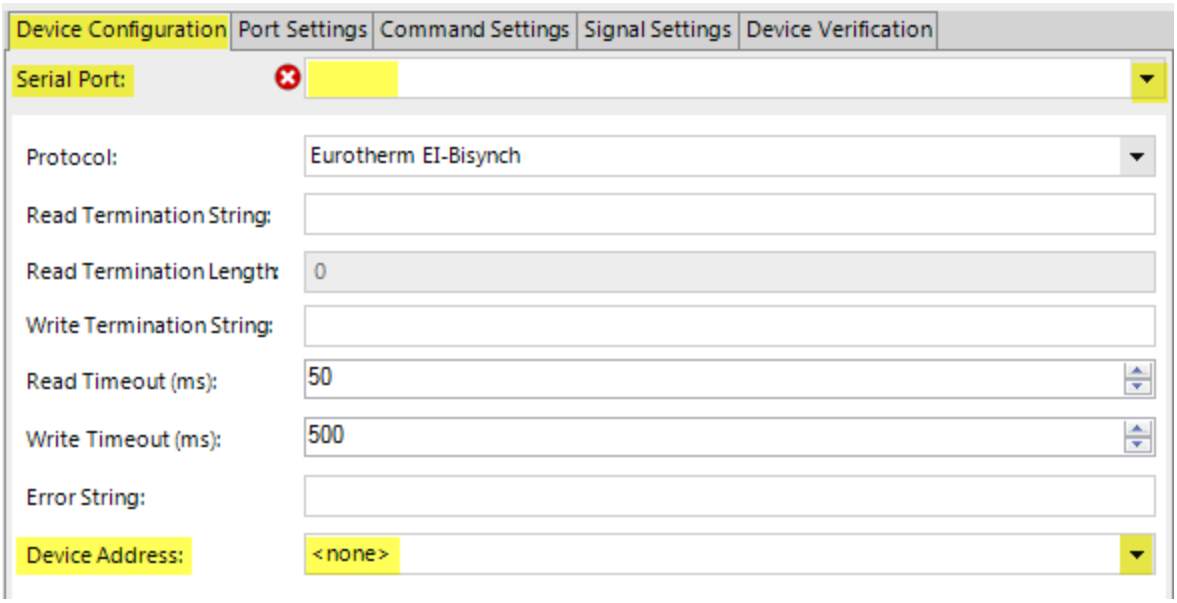
2. In the External Devices window, click the plus sign to add an external device file.



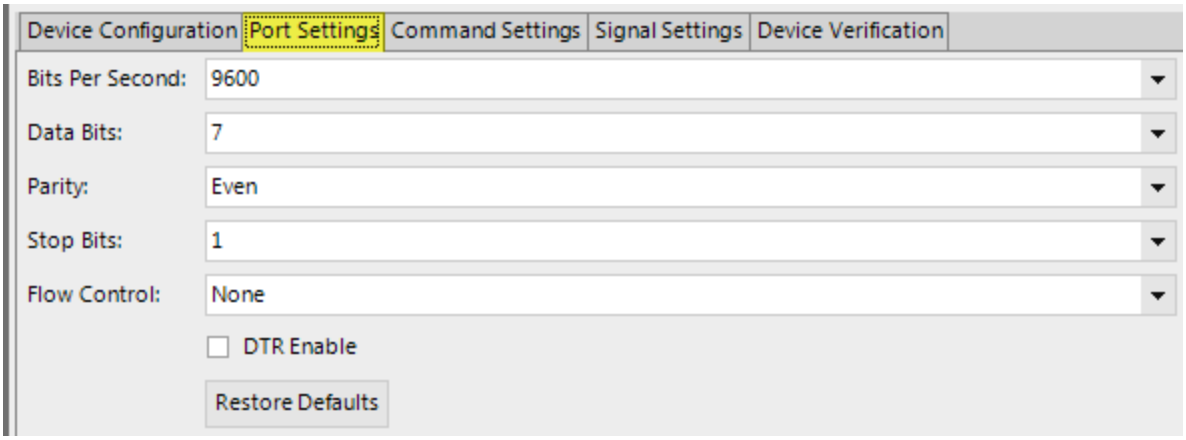
3. In the Select Device Type window, select **Eurotherm 2000 Series Temperature Controller EI-Bisynch**.



4. On the **Device Configuration** tab, use the drop-down menus to select the appropriate serial port and set the **Device Address** to **<none>**.

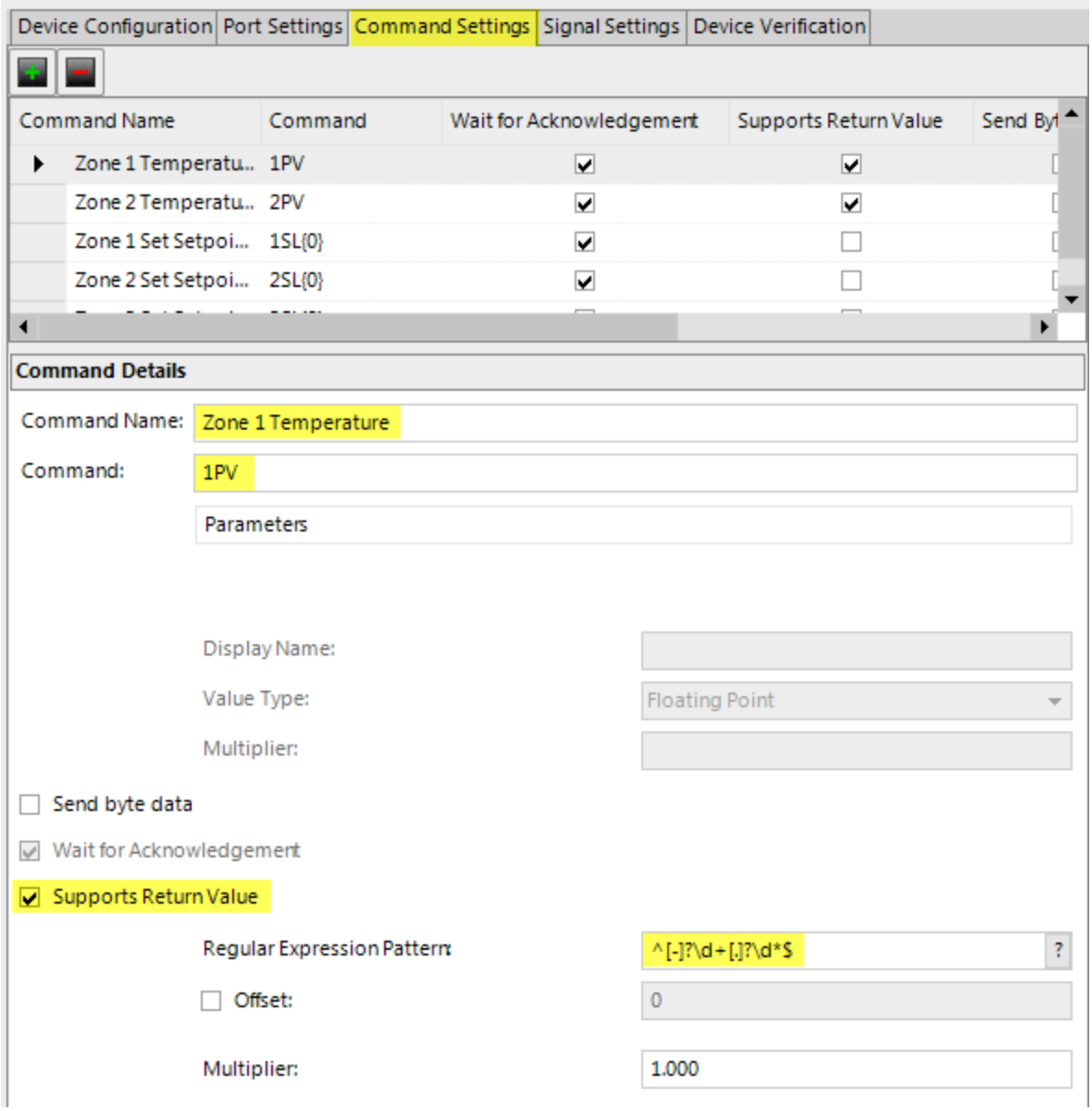


5. On the **Port Settings** tab, use the default values.

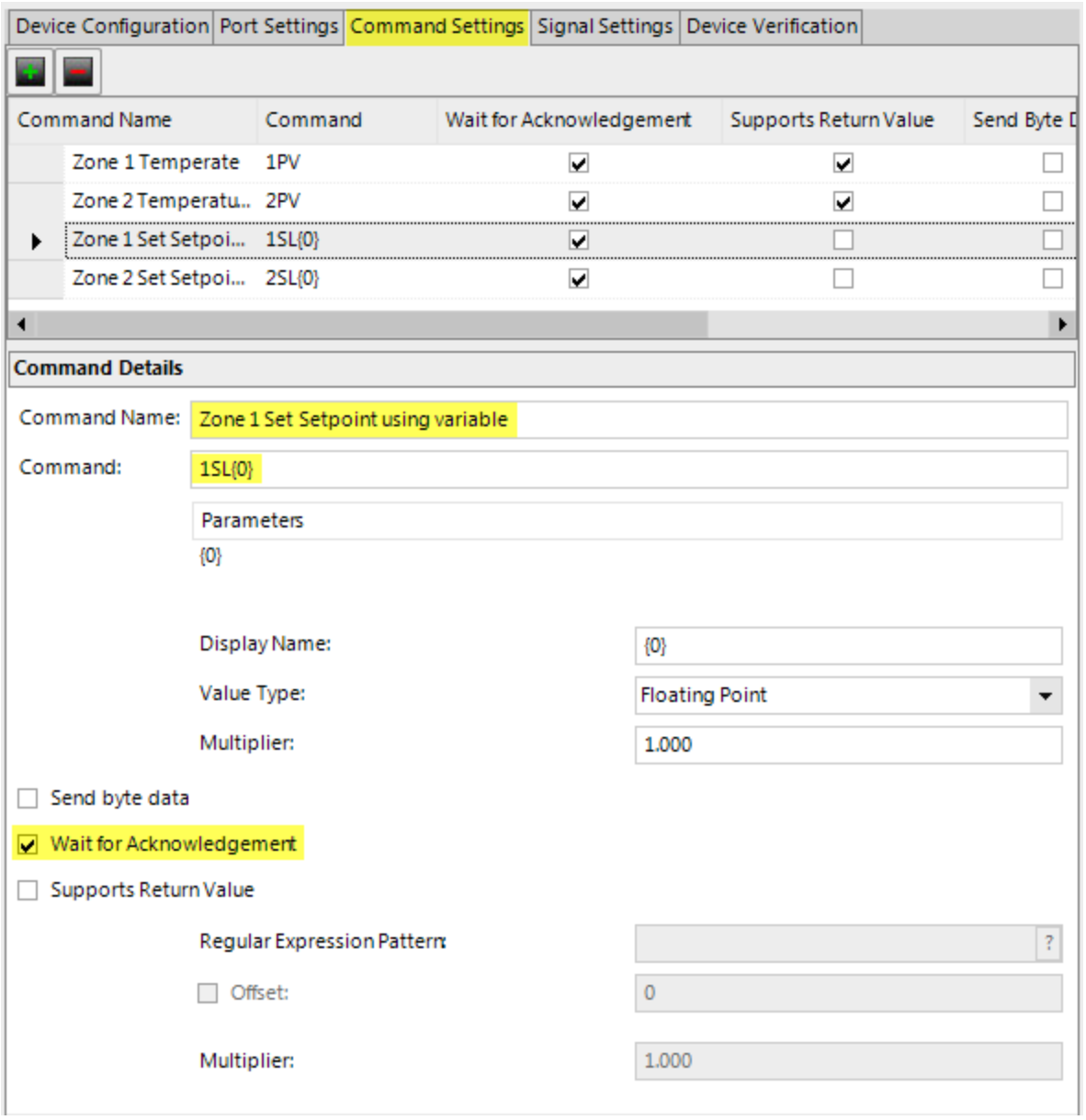


6. On the **Command Settings** tab, add commands to read temperature and set setpoint using variable for each temperature zone. The command to read temperature is PV. The command to set setpoint using variable is SL{0}. The address of the device for each temperature zone precedes the command: 1PV, 1SL{0}, 2PV, 2SL{0}...

   For the read temperature commands, add a descriptive **Command Name**, add the **Command**, check **Supports Return Value**, and add the **Regular Expression Pattern**: ^[-]?\d+[.]?\d*$

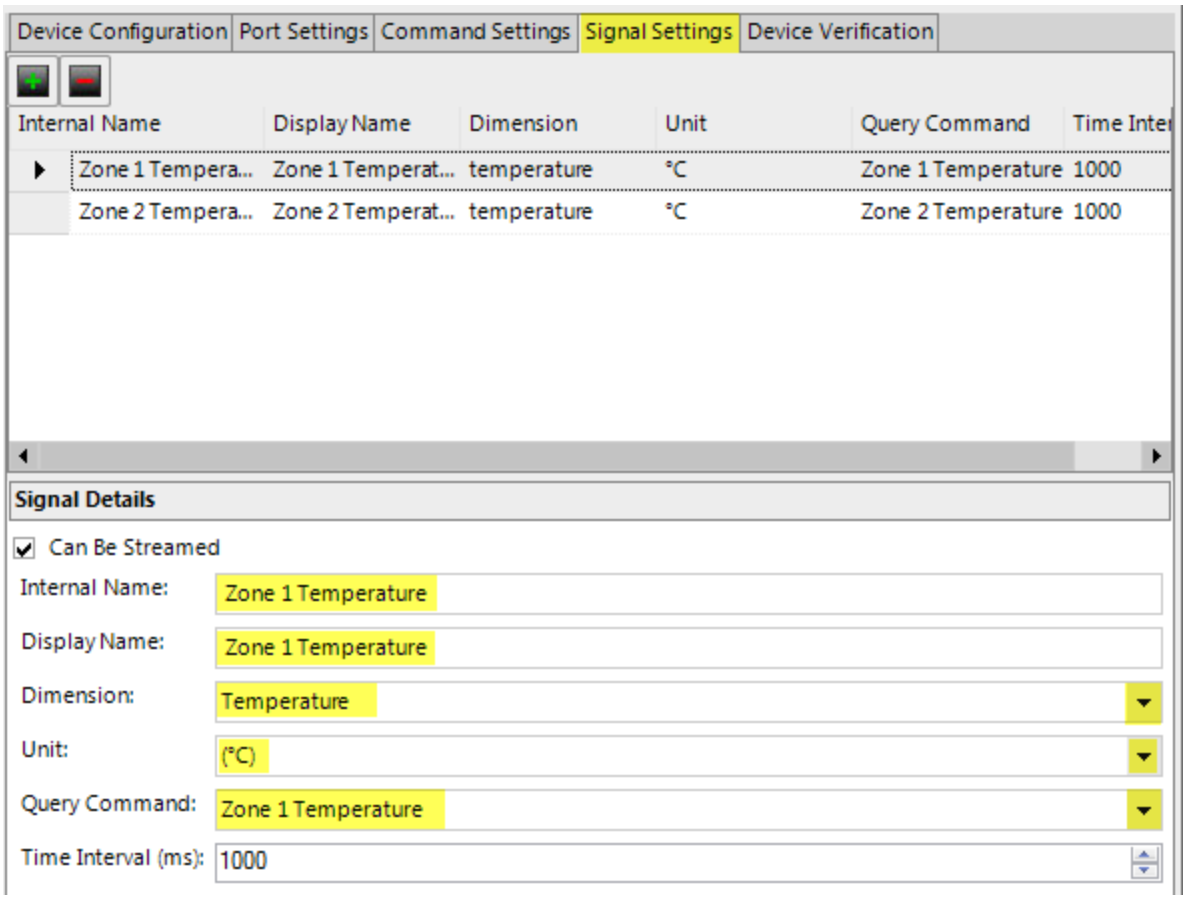


For the set setpoint using variable commands, add a descriptive **Command Name**, add the **Command**, and select the **Wait for Acknowledgment** check box.

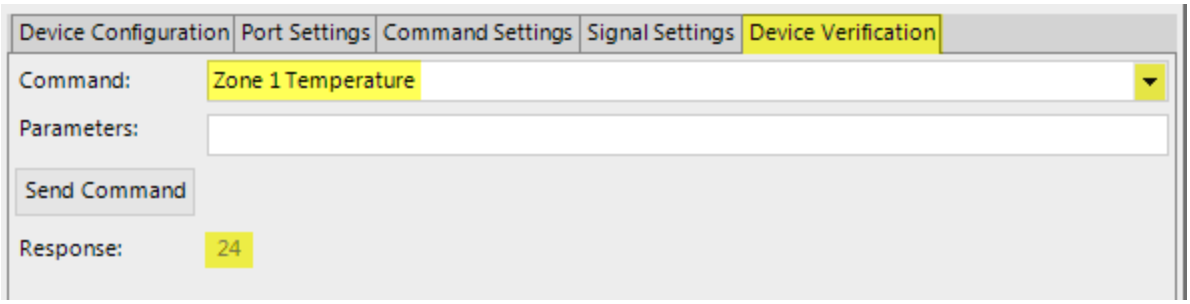


7. On the **Signal Settings** tab, add a signal for each read temperature command added on the **Command Settings** tab.

   Add a descriptive **Internal Name** and **Display Names**. Use the drop-down menus to set the **Dimension** to **Temperature**, to set **Unit** to match the units used by the Eurotherm, and to select the appropriate **Read Temperature** command for **Query Command**.



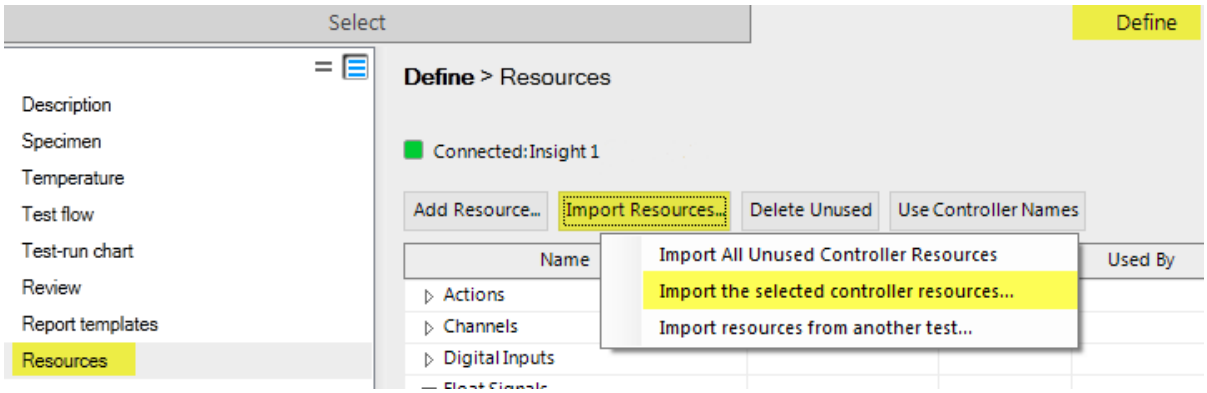
8. On the **Device Verification** tab, use the drop-down menu to select each **Read Temperature** command and click **Send Command** to verify communication.



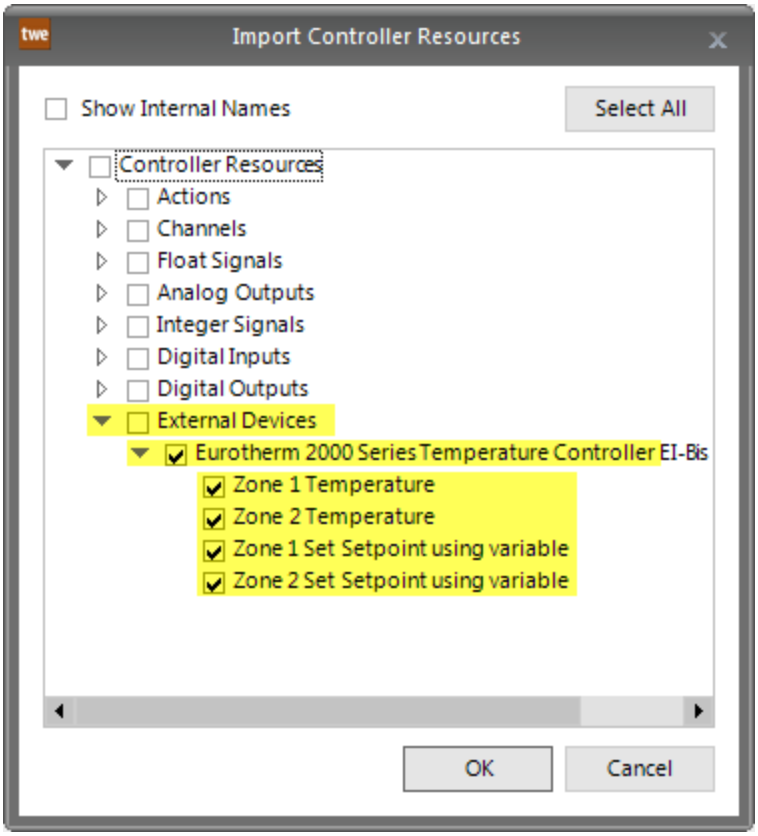
Importing Controller Resources

To import Eurotherm 2000 Series Temperature Controller controller resources into an MTS TestSuite TWS test or template:

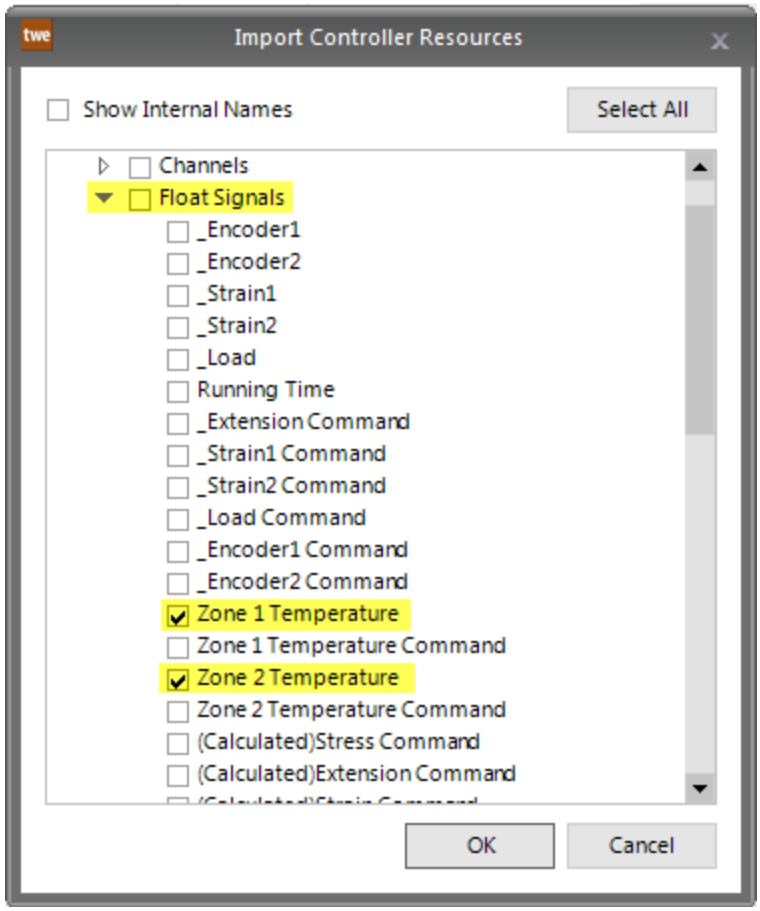
1. In the test or template on the **Define** page select **Resources**, click **Import Resources** > **Import the selected controller resources**



2. In the Import Controller Resources window, expand **External Devices** and check **Eurotherm 2000 Series Temperature Controller**, and expand **Float Signals** and check the read temperature signals configured on the **Signal Settings** tab of the external device file.



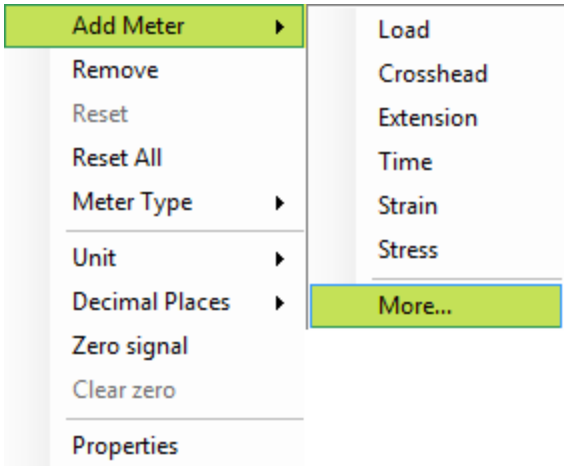
Note: Ensure that you select the read temperature signals and not the read temperature Command signals.



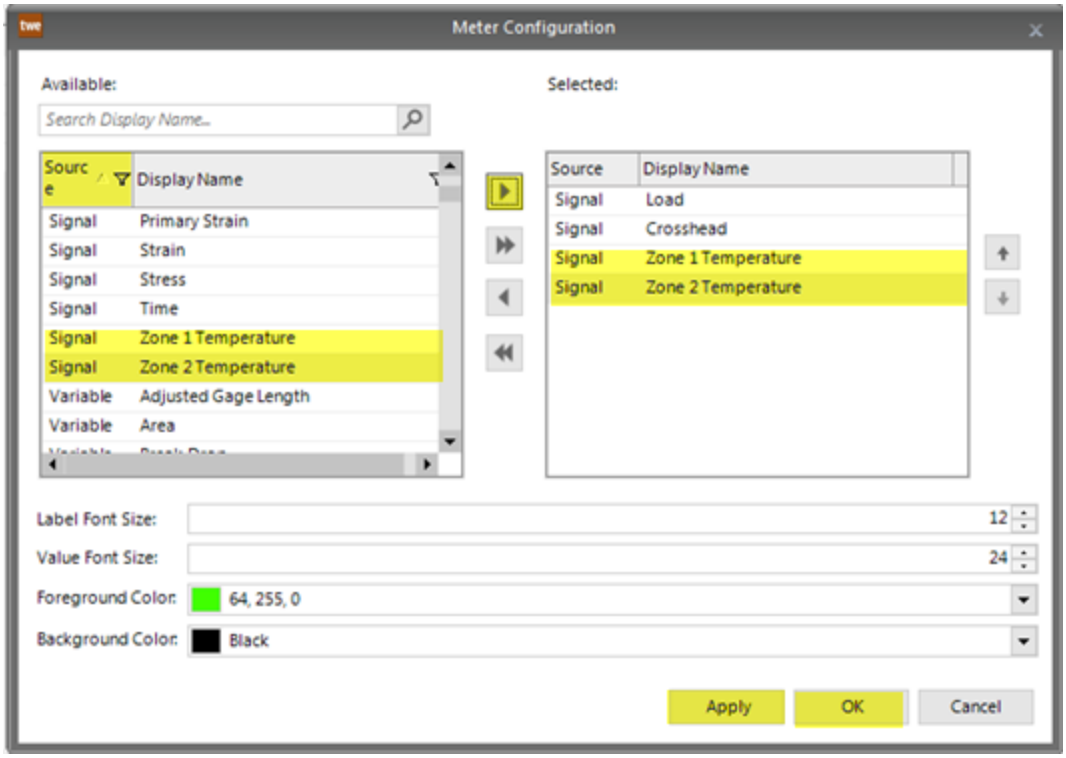
Adding Meters

To add meters for temperature float signals from a Eurotherm 2000 Series Temperature Controller:

1. Right-click a meter and select **Add Meter** > **More**.



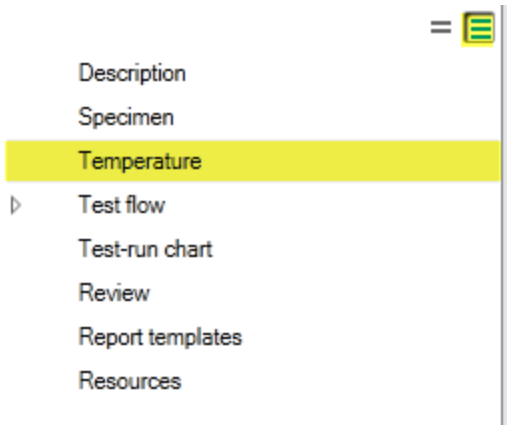
2. In the Meter Configuration window, click **Source** to show the signals first. Scroll down to the temperature signals. Click the arrow to add the temperature signals to the list of meters. Click **Apply** to add the meters. Click **OK** to close the Meter Configuration window.



Adding Temperature Control Activity to Test Flow

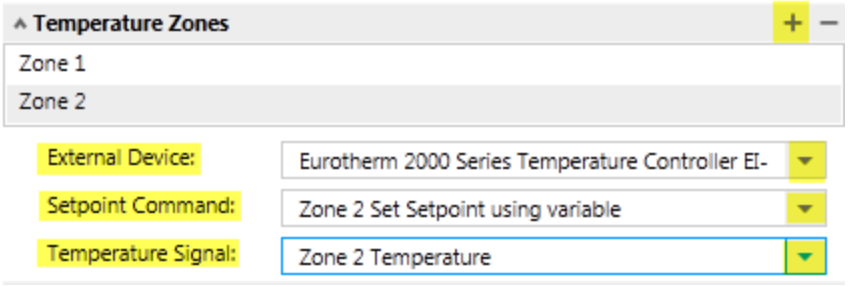
To add temperature control to the MTS TestSuite TWS test flow to bring the system to a target temperature, wait for a hold time, and monitor the temperature through the test:

1. In the test or template on the **Define** page in the test definition tree, select **Advanced Mode** navigation mode and select **Temperature**.



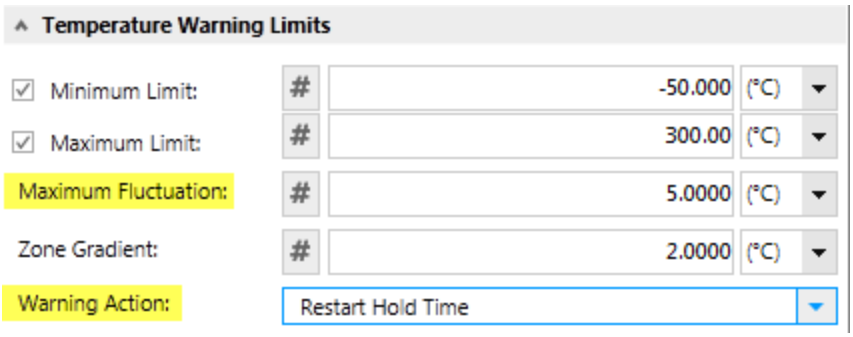
2. Click **Enable**, set the **Target** temperature, and set the **Hold Time**.

3. Configure the **Temperature Zones** by using the drop-down menus to select an **External Device**, **Setpoint Command**, and **Temperature Signal**. Add additional temperature zones if necessary.

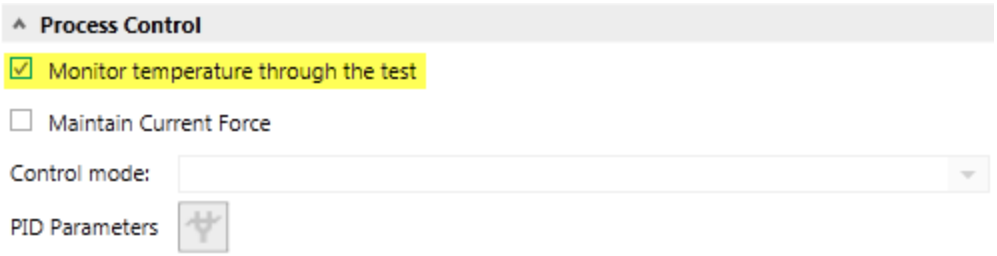


4. Configure the **Temperature Warning Limits**. Only the **Maximum Fluctuation** and

Warning Action are required. The **Zone Gradient** becomes active when a second temperature zone is added.



5. Select the **Monitor temperature through the test** check box.



Appendix: MTS Tuning Template Example

Tuning Template Example Overview

The Load and Strain control modes added by the Advanced Rate Control software option require correct PID and tuning parameter values. Determining the correct PID and tuning parameter values for a test setup is an iterative process.

This MTS EM Tuning Template Example can aid in determining the correct PID and tuning parameter values for the Load control mode. A test created from this template repeatedly runs **Command** activities while you adjust PID and tuning parameter values.

The **Test-Run Display** shows two scopes and the **Command** activity variables. The top scope shows the Load signal and Load Command. The bottom scope shows the Load Error, that is, the difference between the Load signal and Load Command. The **Command** activity variables are the Go To rates, Go To end levels, and Dwell duration.

The test Procedure Run section contains two **While Loops** in parallel. The first **While Loop** contains the **Command** activities. The **Command** activities are a **Go To**, a **Dwell**, and another **Go To**. The second **While Loop** contains an **Input Variables** Operator Entry and a **Tuning Parameters** Control Setting. While the **Command** activities run repeatedly, the Input Variables window shows. You adjust the PID and tuning parameter variables in the Input Variables window and apply then when you click OK. The Input Variables window shows repeatedly so you can further adjust the PID tuning parameter variables. The scopes show how the system responds to the new PID and tuning parameter values.

Once the correct PID and tuning parameter values are determined, use the Exit Variable in the Input Variables window to exit the **While Loops**. After the test exits the **While Loops**, the PID and tuning parameter variables are written to an XML file in the Data Export Directory of TestSuite. You can read this XML file into a different test requiring the same PID and tuning parameter values.

This MTS EM Tuning Template Example can be used in MTS TestSuite TWE, TWS, and TWX. However, MTS TestSuite TWE is required to make modifications to the MTS Procedure/Test Flow.

Tuning Template Example Key Features

The MTS EM Tuning Template Example:

• Can be used in MTS TestSuite TWE, TWS, and TWX. However, MTS TestSuite TWE is required to make modifications to the MTS Procedure/Test Flow.

• Can be used on either an MTS Insight or MTS InsightPlus based system.

• Provides an interface to determine the PID and tuning parameter values for a test setup running in the Load control mode added by the Advanced Rate Control software option.

• Can be used for load rate or constant load.

• Can be changed from tension to compression by inverting the polarity on the Load and Crosshead Float Signals.

• Writes PID and tuning parameter values to an XML file in the Data Export Directory. The resulting XML file can be read into a different test requiring the same PID and tuning parameter values.

• Provides an interface to adjust PID and tuning parameter values while the test is running.

• Shows scopes while the test is running, so the effects of the PID and tuning parameter value adjustments can be observed.

• Shows the command and feedback signals from the test run in the review chart.

How to Use the Tuning Template Example

The MTS EM Tuning Template Example is configured for a tension test using the Load Control Mode. To run a compression test, invert the polarity on the Load and Crosshead Float Signals.

1. Verify that the Advanced Rate Control software option is installed:
   A. Go to **Start** > **All Programs** > **MTS TestSuite** > **License Administrator**.
   B. Under **Other Features**, ensure that Custom.AdvancedRateControl appears.
2. Open the MTS TestSuite application.
3. Go to **MTS Templates** > **TW-EM** > **Tuning** > **MTS EM Tuning Template Example**.
4. Double-click the **MTS EM Tuning Template Example** to create a new test.
5. Install the specimen.
6. Clear the interlocks.
7. Click the run button.
8. Enter values for the **Material Name**, specimen dimensions, and **Command** activity variables. The **Material Name** is used to name the XML file containing the PID and tuning parameter values. Subsequent test runs will overwrite this file if the **Material Name** is not changed. To run a constant load test, set **End Level 1** and **End Level 2** to the same value and increase the **Dwell Duration**.

Name	Value	Unit
Material Name	Material 2011	
Width	12.700	mm
Thickness	3.175	mm
GoTo End Level 1 Rate	20.000	N/s
End Level 1	600.000	N
Dwell Duration	5.000	sec
GoTo End Level 2 Rate	20.000	N/s
End Level 2	100.000	N

9. Observe that initially the Load (blue line) does not respond to the Load Command (red line) until the **kP_Load** variable is increased.

   Note: The maximum speed for metals is about 1 mm/min. The maximum speed for rubber is about 10 to 100 mm/min.

10. Increase the value of **kP_Load** by a factor of 10 and click **OK** until a response is seen. Then increase the value more gradually.

    For example: 0.0010 **OK**; 0.010 **OK**; 0.1 **OK**, 0.2 **OK**...

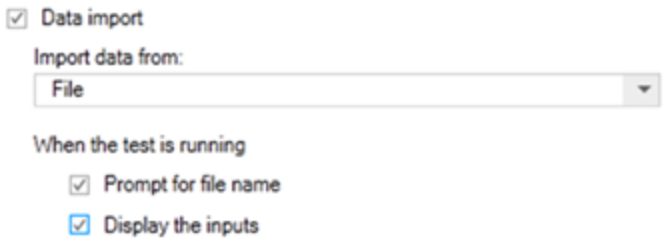
Name	Value	Unit
ExitVariable	Continue	
kP_Load	0.00010	unitless
kI_Load	0.0000	unitless
kD_Load	0.000	unitless
DerivativeInterval	2.000	sec
MaximumSpeed	1.000	mm/min
MaximumIntegral	30.000	mm/s

11. Increase kI_Load.
12. Increase kD_Load if necessary.
13. When tuning is complete, change the **Exit Variable** to “Done” and click **OK** to exit. The values and chart are results on the **Review** page. The PID and tuning parameter values are saved to an XML file in the Data Export Directory.

Name	Value	Unit
ExitVariable	Done	
kP_Load	1.20000	unitless
kI_Load	0.0020	unitless
kD_Load	0.000	unitless
DerivativeInterval	2.000	sec
MaximumSpeed	10.000	mm/min
MaximumIntegral	30.000	mm/s

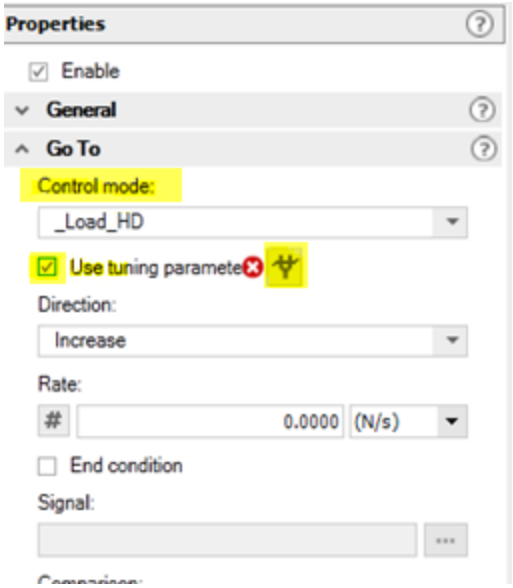
Modify a Template to Import Tuning Parameters from an XML File

1. Open the template to be modified.
2. If necessary, import the **Control Mode**.
3. Select the **Pre-test run Data import**. The **Pre-test run Data import** will prompt before every test run unless the **Data import** is disabled.
4. Configure the **Data import** as follows:

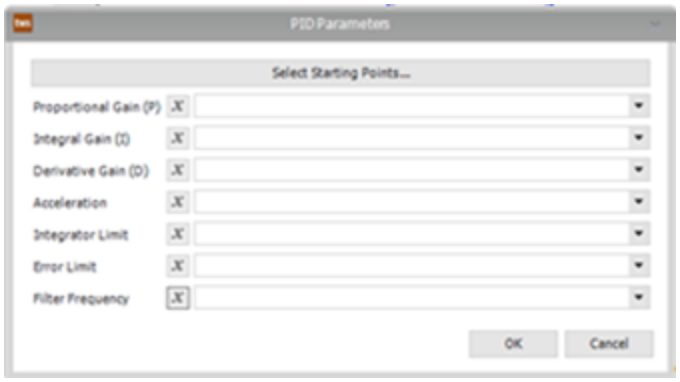


5. Select the **Command** activity that is using the Advanced Rate Control or HD Control mode.

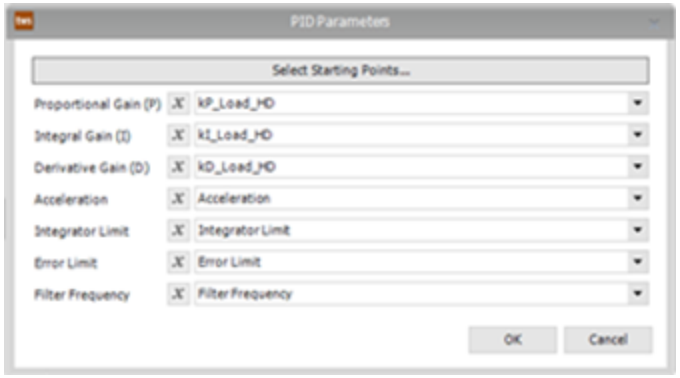
6. In the **Command** activity **Properties** panel, select the control mode and check **Use Tuning Parameters** and click the Tuning Parameters icon.



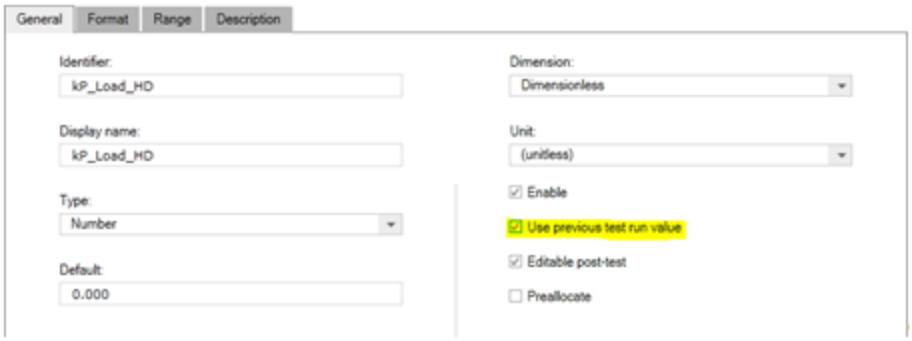
7. In the PID Parameters window, change the tuning parameters to variable selection.



8. To create variables for each tuning parameter, use the drop-down menu and select <new variable>.
9. Change the **Identifier** and **Display Name**. Ensure that the **Identifier** matches exactly the **Identifier** of the corresponding variable in the XML file.
10. Repeat Step 9 for the rest of the tuning parameters.



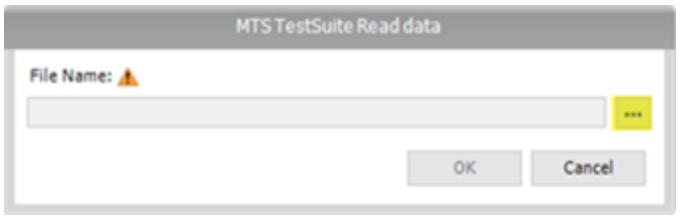
11. To use the same tuning parameters for multiple test runs, check Use previous test run value for each variable.

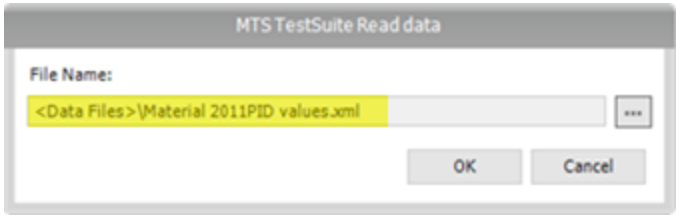


12. Go to **File** > **Save As** > **Template...**

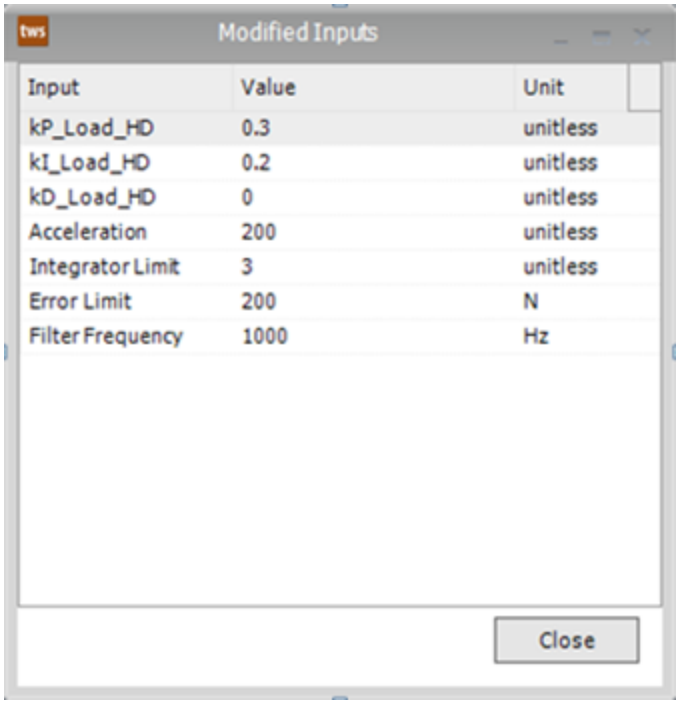
Using the Modified Template

1. Navigate to the XML file containing the applicable tuning parameters.





The imported values will display:



2. Click the **Close** button and the test continues as normal.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

MTS Systems Corporation Headquarters
14000 Technology Drive
Eden Prairie, MN 55344-2290 USA
Email: info@mts.com
www.mts.com

ISO 9001 Certified Quality Management System