SoMachine
PD_ETest
Library Guide

06/2017

EIO0000002394.00

www.schneider-electric.com

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

Table of Contents

Safety Information

Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

⚡ The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

⚠ This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

BEFORE YOU BEGIN

Do not use this product on machinery lacking effective point-of-operation guarding. Lack of effective point-of-operation guarding on a machine can result in serious injury to the operator of that machine.

⚠ WARNING

UNGUARDED EQUIPMENT

- Do not use this software and related automation equipment on equipment which does not have point-of-operation protection.
- Do not reach into machinery during operation.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

This automation equipment and related software is used to control a variety of industrial processes. The type or model of automation equipment suitable for each application will vary depending on factors such as the control function required, degree of protection required, production methods, unusual conditions, government regulations, etc. In some applications, more than one processor may be required, as when backup redundancy is needed.

Only you, the user, machine builder or system integrator can be aware of all the conditions and factors present during setup, operation, and maintenance of the machine and, therefore, can determine the automation equipment and the related safeties and interlocks which can be properly used. When selecting automation and control equipment and related software for a particular application, you should refer to the applicable local and national standards and regulations. The National Safety Council's Accident Prevention Manual (nationally recognized in the United States of America) also provides much useful information.

In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the operator's hands and other parts of the body are free to enter the pinch points or other hazardous areas and serious injury can occur. Software products alone cannot protect an operator from injury. For this reason the software cannot be substituted for or take the place of point-of-operation protection.

Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

NOTE: Coordination of safeties and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the Function Block Library, System User Guide, or other implementation referenced in this documentation.

START-UP AND TEST

Before using electrical control and automation equipment for regular operation after installation, the system should be given a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such a check be made and that enough time is allowed to perform complete and satisfactory testing.

⚠ WARNING

EQUIPMENT OPERATION HAZARD

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters, and debris from equipment.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

Software testing must be done in both simulated and real environments.

Verify that the completed system is free from all short circuits and temporary grounds that are not installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Close the equipment enclosure door.
- Remove all temporary grounds from incoming power lines.
- Perform all start-up tests recommended by the manufacturer.

OPERATION AND ADJUSTMENTS

The following precautions are from the NEMA Standards Publication ICS 7.1-1995 (English version prevails):

- Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly operated.
- It is sometimes possible to misadjust the equipment and thus produce unsatisfactory or unsafe operation. Always use the manufacturer’s instructions as a guide for functional adjustments. Personnel who have access to these adjustments should be familiar with the equipment manufacturer’s instructions and the machinery used with the electrical equipment.
- Only those operational adjustments actually required by the operator should be accessible to the operator. Access to other controls should be restricted to prevent unauthorized changes in operating characteristics.

About the Book

At a Glance

Document Scope

This document describes the PD_ETest library.

The PD_ETest library provides a collection of assistant function blocks for the ETEST framework of the Logic Builder. For further information on the ETEST framework and its use, refer to the ETEST User Guide.

Validity Note

This document has been updated for the release of SoMachine V4.3.

Related Documents

Title of documenation	Reference number
ETEST User Guide	*EIO0000002388 (ENG)*; *EIO0000002389 (FRE)*; *EIO0000002390 (GER)*; *EIO0000002391 (ITA)*; *EIO0000002392 (SPA)*; *EIO0000002393 (CHS)*

You can download these technical publications and other technical information from our website at *http://www.schneider-electric.com/ww/en/download*.

Product Related Information

⚠ WARNING

LOSS OF CONTROL

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.[1]
- Each implementation of this equipment must be individually and thoroughly tested for proper operation before being placed into service.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

[1] For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in this manual, or that appear in or on the products themselves, are generally derived from the terms or definitions of international standards.

In the area of functional safety systems, drives and general automation, this may include, but is not limited to, terms such as *safety*, *safety function*, *safe state*, *fault*, *fault reset*, *malfunction*, *failure*, *error*, *error message*, *dangerous*, etc.

Among others, these standards include:

Standard	Description
EN 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2008	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
EN 1088:2008 ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2006	Safety of machinery - Emergency stop - Principles for design
EN/IEC 62061:2005	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2008	Digital data communication for measurement and control: Functional safety field buses.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

In addition, terms used in the present document may tangentially be used as they are derived from other standards such as:

Standard	Description
IEC 60034 series	Rotating electrical machines
IEC 61800 series	Adjustable speed electrical power drive systems
IEC 61158 series	Digital data communications for measurement and control – Fieldbus for use in industrial control systems

Finally, the term *zone of operation* may be used in conjunction with the description of specific hazards, and is defined as it is for a *hazard zone* or *danger zone* in the *Machinery Directive* (*2006/42/EC*) and *ISO 12100:2010*.

NOTE: The aforementioned standards may or may not apply to the specific products cited in the present documentation. For more information concerning the individual standards applicable to the products described herein, see the characteristics tables for those product references.

Part I
General Information

What Is in This Part?

This part contains the following chapters:

Chapter 1
Specific Safety Information

Overview

This section contains information regarding working with the PD_ETest library. Qualified personnel working with the PD_ETest library must read and observe this information.

What Is in This Chapter?

This chapter contains the following topics:

Qualification of Personnel

Overview

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel.

No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and the installation, and has received safety-related training to recognize and avoid the hazards involved.

The qualified person must be able to detect possible hazards that may arise from parameterization, modifying parameter values and generally from mechanical, electrical, or electronic equipment. The qualified person must be familiar with the standards, provisions, and regulations for the prevention of industrial accidents, which they must observe when designing and implementing the system.

Proper Use

Overview

This product is a library to be used together with the control systems and servo amplifiers intended solely for the purposes as described in the present documentation as applied in the industrial sector.

Always observe the applicable safety-related instructions, the specified conditions, and the technical data.

Perform a risk evaluation concerning the specific use before using the product. Take protective measures according to the result.

Since the product is used as a part of an overall system, you must ensure the safety of the personnel by means of the concept of this overall system (for example, machine concept).

Any other use is not intended and may be hazardous. Electrical devices and equipment must only be installed, operated, maintained, and repaired by qualified personnel.

Product Related Information

Product Related Information

⚠ WARNING

LOSS OF CONTROL

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.[1]
- Each implementation of this equipment must be individually and thoroughly tested for proper operation before being placed into service.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

[1] For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

Before you attempt to provide a solution (machine or process) for a specific application using the POUs found in the library, you must consider, conduct and complete best practices. These practices include, but are not limited to, risk analysis, functional safety, component compatibility, testing and system validation as they relate to this library.

⚠ WARNING

IMPROPER USE OF POUS

- Perform a safety-related analysis for the application and the devices installed.
- Ensure that the POUs are compatible with the devices in the system and have no unintended effects on the proper functioning of the system.
- Use appropriate parameters, especially limit values, and observe machine wear and stop behavior.
- Verify that the sensors and actuators are compatible with the selected POUs.
- Thoroughly test all functions during verification and commissioning in all operation modes.
- Provide independent methods for critical control functions (emergency stop, conditions for limit values being exceeded, etc.) according to a safety-related analysis, respective rules, and regulations.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Chapter 2
Presentation of the Library

General Information

Introduction

The PD_ETest library provides a collection of assistant function blocks for the ETEST framework of the Logic Builder. For further information on the ETEST framework and its use, refer to the ETEST User Guide.

When you create a test case in the Logic Builder, the PD_ETest library is automatically integrated in your project. You cannot use ETEST framework without this library.

NOTE: The PD_ETest library is only used in connection with the ETEST framework that is integrated in Logic Builder. The functions and function blocks contained in this library are used implicitly, and are not programmed in the user application.
The POUs of the PD_ETest library can only be accessed from test case objects and resource objects, which have been created in the ETEST framework.
If you attempt to use these functions and functions blocks directly in your application, it can lead to compile errors.

Part II
Global Variables

What Is in This Part?

This part contains the following chapters:

Chapter 3
Global Constant List

Global Constant List (GCL)

Overview

Type:	Global constant
Available as of:	V1.2.5.0

Description

The global constant list contains the global constants of the PD_ETest library.

Global Constant

Variable	Data type	Description
`Gc_sLibraryVersion`	STRING[80]	Library version

Chapter 4
Global Parameter List

Global Parameter List (GPL)

Overview

Type:	Global parameter
Available as of:	V1.2.5.0

Description

The global parameter list contains the global parameters of the PD_ETest library. They can be overwritten by project-specific values in the **Library Manager**.

Global Parameters

Variable	Data type	Value	Description
`Gc_uiMaxNumberOfTestCases`	UINT	500	Maximum number of test cases that can be stored.
`Gc_uiMaxNumberOfTcResAssignments`	UINT	800	Maximum number of resources that can be assigned to a test case.
`Gc_uiMaxNumberOfFailedTc`	UINT	300	Maximum number of test cases not successfully completed that can be stored.
`Gc_uiMaxNumberOfResources`	UINT	300	Maximum number of resources that can be stored.
`Gc_uiMaxNumberOfResResAssignments`	UINT	800	Maximum number of test resource assignments to resources that can be stored.
`Gc_uiResourceListSize`	UINT	50	Maximum size of the resources list that can be stored.
`Gc_uiMaxNumberOfMeasurands`	UINT	50	Maximum number of measured values that can be stored.

Part III
Execution

What Is in This Part?

This part contains the following chapters:

Chapter 5
Enumerations

ET_TestReturn

Overview

Type:	Enumeration
Available as of:	V1.2.5.0
Default value:	E_Working

Description

The enumeration ET_TestReturn includes the states of the test. It is relevant for internal execution of the test.

Enumeration Elements

Name	Value	Description
E_Working	0	Test case / test sequence is executed.
E_Done	1	Test case / test sequence completed successfully.
E_Failed	2	Test case / test sequence not successful.

Used By

ET_TestReturn is the datatype of the return value of IF_TestCase *(see page 35)*.

Chapter 6
Interfaces

What Is in This Chapter?

This chapter contains the following topics:

IF_ErrorLogger

Overview

Type:	Interface
Available as of:	V1.2.5.0

NOTE: This interface type is not intended for use in user logic.

Task

The `IF_ErrorLogger` interface is used to report detected errors.

Functional Description

The methods and properties of this interface are not visible or documented because it is only supposed to be used via the macros (for example, `ASSERT(…)`) of the ETEST framework in Logic Builder.

`IF_ManagedTestObject`

Overview

Type:	Interface
Available as of:	V1.2.5.0

NOTE: This interface type is not intended for use in user logic.

Task

The `IF_ManagedTestObject` interface combines joint methods of the objects that have to be prepared and adjusted. It is used for the data inventory in the PD_ETest library.

`CleanUp` Method

The `CleanUp` method is used to reinitialize the object(s) of the test cases and test resources so that the object(s) can be re-used.

Interface

Input	Data type	Description
`i_xFirstCall`	BOOL	TRUE: The method is called for the first time. FALSE: After the first call to the method. Can be used to initialize state machines.
`i_ifErrorLogger`	`IF_ErrorLogger`	Interface to report detected errors.

Return value

Data type	Description
`ET_TestReturn`	Refer to the `ET_TestReturn` enumeration *(see page 29)*.

Prepare Method

The Prepare method is used for preparation measures for the tests, such as the initialization of variables or the activation of the position control of an axis.

Interface

Input	Data type	Description
i_xFirstCall	BOOL	TRUE: The method is called for the first time. FALSE: After the first call to the method. Can be used to initialize state machines.
i_rfbResources	Reference to FB_ResourceList	List of the resources used.
i_ifErrorLogger	IF_ErrorLogger	Interface to report detected errors.

Return value

Data type	Description
ET_TestReturn	Refer to the ET_TestReturn enumeration *(see page 29)*.

IF_TestCase

Overview

Type:	Interface
Available as of:	V1.2.5.0

NOTE: This interface type is not intended for use in user logic.

Task

The IF_TestCase interface is automatically implemented in every test case.

Execute Method

The Execute method contains the actions and verifications of the tests.

Interface

Input	Data type	Description
i_xFirstCall	BOOL	TRUE: The method is called for the first time. FALSE: After the first call to the method. Can be used to initialize state machines.
i_ifErrorLogger	IF_ErrorLogger	Interface to report detected errors.

Output	Data type	Description
q_rProgress	REAL	Used to display the test case progress. Write-enabled parameter.
q_sState	STRING[255]	Textual description of the status of a test case. Write-enabled parameter.

Return value

Data type	Description
ET_TestReturn	Refer to the ET_TestReturn enumeration *(see page 29)*.

Finalize Method

The Finalize method is performed when the Execute method has been completed successfully or not, or when it has been canceled. In contrast to CleanUp, the Finalize method is performed in the same cycle as the Execute method.

IF_TestResource

Overview

Type:	Interface
Available as of:	V1.2.5.0

NOTE: This interface type is not intended for use in user logic.

Task

The `IF_TestResource` interface has to be implemented from a test resource.

It is used to manage test resources.

Used By

`IF_ManagedTestObject` *(see page 33)*

Part IV
MemoryLists

What Is in This Part?

This part contains the following chapters:

Chapter 7
Enumerations

ET_MeasurandTypes

Overview

Type:	Enumeration
Available as of:	V1.2.5.0
Default:	E_NotClassified

Description

The enumeration includes the types of measurand.

Enumeration Elements

Name	Value	Description
E_NotClassified	0	The measured value type is not defined. You can use the input i_sName of the AddMeasurand method to provide information on the measured value.
E_ExecutionTime	1	The measured value is the execution time of the module in nanoseconds. You can use the input i_sName of the AddMeasurand method to provide more information on the time that is measured.
E_SizeOf	2	The measured value is the size of the module in bytes.

Used By

FB_MeasurandList *(see page 41)*

Chapter 8
Function Blocks

FB_MeasurandList

Overview

Type:	Function block
Available as of:	V1.2.5.0

Task

Recording the measured values.

Functional Description

The function block FB_MeasurandList manages the values measured with the AddMeasurand method.

AddMeasurand Method

The AddMeasurand method stores external recorded measured values as a STRING value. For example, the execution times of a function while using specific parameters or the data size of a structure (determined with SIZEOF) can be recorded or stored.

The method can be called at any position within a test case. A test case can provide any number of measured values by calling the method several times. For further information, refer to the chapter *Record Measured Values (see page 53)*.

Interface

Input	Data type	Description
i_sModule	STRING[255]	Name of the unit that transfers the measured value. Used to assign the measured value to a library, function block, or function.
i_sName	STRING[255]	Name of the measured value.
i_etType	ET_MeasurandTypes	Type of the measured value. Also refer to the enumeration ET_MeasurandTypes *(see page 39)*. This entry is optional.
i_sValue	STRING[255]	Measured value.

Return value

Data type	Description
UDINT	Indicates the number of stored measured values. The return value is equal to zero if the maximum number of measured values that can be recorded was exceeded and no measured value can be stored anymore. Refer to the parameter `Gc_uiMaxNumberOfMeasurands` *(see page 25)*.

Part V
Using ETEST

Chapter 9
Using ETEST

What Is in This Chapter?

This chapter contains the following topics:

Add an ETEST Object

Add an ETEST Object

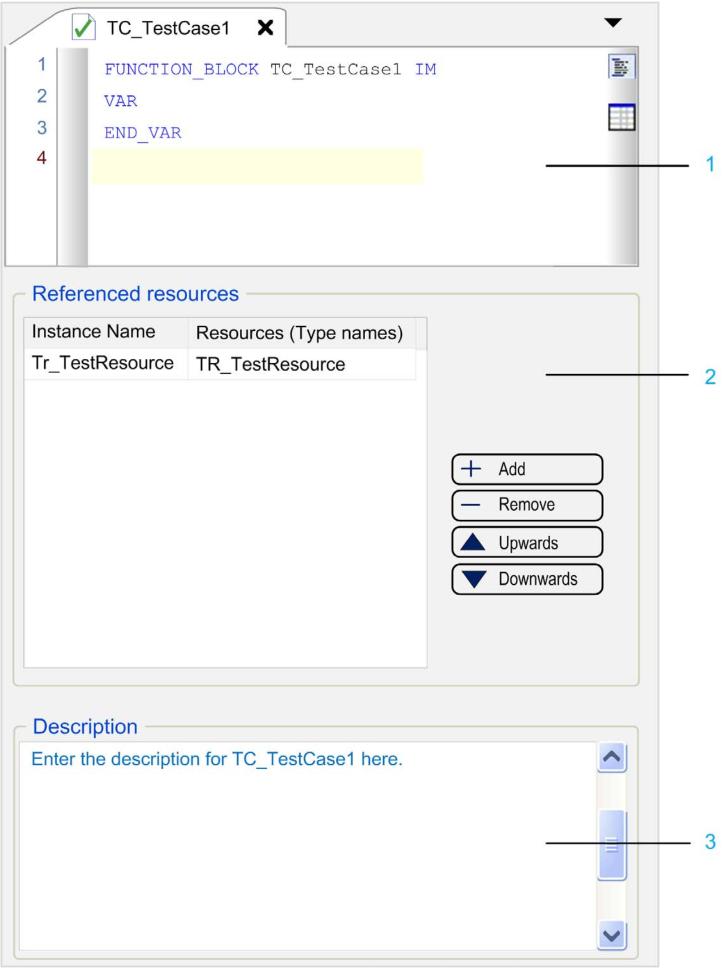
To add an ETEST object to your project, proceed as follows:

Step	Action
1	Right-click the **Application** node in the **Tools tree** and execute the **Add Object...** command from the context menu.
2	Select the ETEST object you want to add (**Test Case**, **Test resources**, **Test Series**). **Result:** The **ETEST Object** dialog box opens.
3	Rename your object by typing a name in the box **Name**. **NOTE:** Choose a name that complies to the IEC standard. Do not use special characters, leading digits, or spaces within the name. The name must not exceed a length of 32 characters. If you do not rename the object, a name is given by default.
4	Click the **Add** button. **Result:** The selected ETEST object is added to the project and appears as a new node in the **Tools tree**.

Edit Test Cases

Overview

The structure of test cases is similar to function blocks, but there is no editor available for the implementation section. Only the methods of test cases are implemented. To implement the test case, use the method editor by double-clicking the method in the **Tools tree**.



```
FUNCTION_BLOCK TC_TestCase1 IM
VAR
END_VAR
```

Instance Name	Resources (Type names)
Tr_TestResource	TR_TestResource

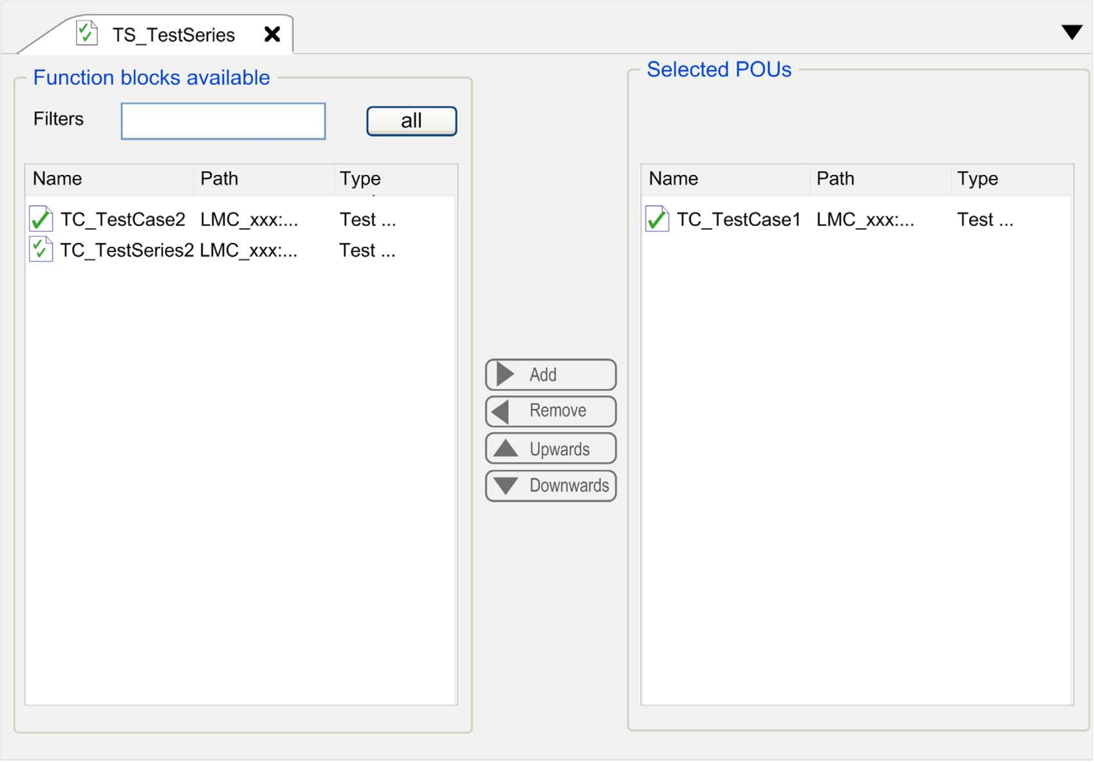
Enter the description for TC_TestCase1 here.

1 The declaration editor corresponds to the declaration section of a function block. You can switch between the text view and the table view of the editor using the buttons **Textual** and **Tabular**.
2 The **Referenced resources** area provides integrated test resources.
3 The **Description** area provides space for explanations regarding the test case.

Use Test Series

Overview

One project comprises several test cases. The **TestSeries** editor allows you to group several test cases into a test series to execute them in a sequence.



Function blocks available

Name	Path	Type
✓ TC_TestCase2	LMC_xxx:...	Test ...
✓ TC_TestSeries2	LMC_xxx:...	Test ...

Selected POUs

Name	Path	Type
✓ TC_TestCase1	LMC_xxx:...	Test ...

The **TestSeries** editor consists of two areas:

Area	Description
Function blocks available	This list includes test cases and test series that are subnodes of the same **Application** node as the open test series. It also lists those test cases and test series that are located in the **Tools tree**. You cannot select a test series as a subnode of itself. Cyclical references between test series are not allowed.
Selected POUs	The selected POUs are displayed. The POUs are the test cases. A test series can select a POU directly only once. Indirectly, POUs can be selected several times. It is thus permissible that test series may embed other test series, and that these embedded test series in turn contain the same test case.

Button / box	Description
Add	Moves a selected test case from the list **Function blocks available** into the list **Selected POUs**.
Remove	Moves a selected test case from the list **Selected POUs** to the list **Function blocks available**.
Upwards	Moves a test case in the area **Selected POUs** upward in the list.
Downwards	Moves a test case in the area **Selected POUs** downward in the list.
Filters all	Enter a text in the **Filters** field to filter the list of **Function blocks available** in accordance with the entry. Click the button **all** to clear the filter.

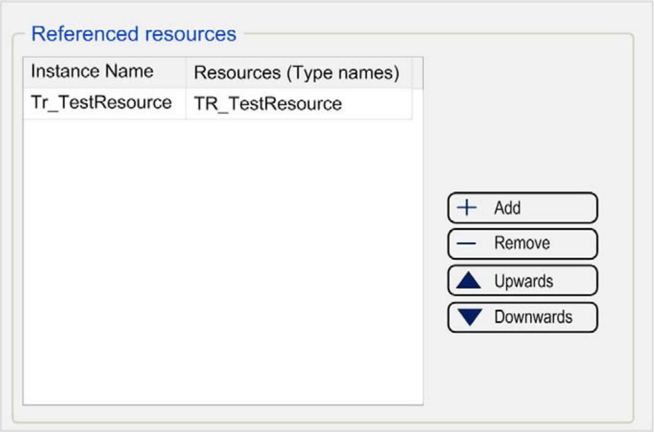
Mouse or Keyboard Action	Description
Double-click	Double-click an entry in the **Function blocks available** list to shift it to the **Selected POUs** list and vice versa.
Delete key	Moves a selected test case from the list **Selected POUs** to the list **Function blocks available**.
Ctrl+A shortcut	Selects all entries of the list.

Use Test Resources

Overview

A test resource encapsulates the access as well as the initialization of hardware and larger data structures used by test cases. Resources are created and processed just as test cases *(see page 47)*. A test case or a resource can bind a resource.

The declaration section for resources (**Referenced resources**) is located in the central area of the editor.



Referenced resources

Instance Name	Resources (Type names)
Tr_TestResource	TR_TestResource

Button	Description
+ Add	Adds a test resource.
- Remove	Removes a test resource.
Upwards	Moves a selected test resource upward within the list.
Downwards	Moves a selected test resource downward within the list.

Mouse or Keyboard Action	Description
Double-click	Double-click an entry in the list of **Referenced resources** to edit it.
Delete key	Removes the test resource.
Ctrl+A shortcut	Selects the entries of the **Referenced resources** list.
Ctrl+click or **Shift**+click shortcut	Selects several entries of the **Referenced resources** list.

Each line of the table comprises a referenced resource. The column **Instance name** contains the name of the resource. This name is used if the resource is called from within a test object: A variable of this name is generated, which holds a reference for the resource.

The column **Resources (Type names)** contains the type name of the resource. This is the name of the test resource object which is to be integrated. In the online view, resources are displayed together with the other variables.

Execute Test

Overview

This section describes how to execute test cases and test series *(see page 48)*.

ETEST consists of two main test processes. One process runs within the Logic Builder on the PC. The other process runs on the controller. The controller process is integrated in the application which runs on the controller. Both processes communicate via a communication data structure.

Executing a Test

The following prerequisites must be fulfilled before a test can be executed:

- The application must contain a test object that is referenced in a task.
- The test case or the test series that shall be executed must be selected in the **Tools** tree.
- No other test must be running.

Execute a test as follows:

Step	Action
1	Right-click the node of the test case to be executed in the **Tools** tree. **Result**: The context menu opens.
2	Execute the command **Start test** from the context menu. **Result**: The test case is executed.

After you have started the test, the results view of the test opens. It indicates the progress of the test. You can stop a running test by clicking the **Cancel test run** icon in the toolbar.

NOTE: While a test is running, observe the following:

- Do not disconnect the controller.
- Do not log out.
- Do not stop or reset the application.
- Do not edit the project.

Doing so interrupts the test logic rendering any result invalid.

Record Measured Values

Overview

The example in this chapter illustrates how measured values can be recorded.

To achieve this, the following programming code must be called in a method that is executed within a test.

Example

```
TestManager.fbMeasurandList.AddMeasurand('module', 'name', ET_MeasurandTypes.E_NotClassified, 'value');
```

For further information, refer to:

- `FB_MeasurandList` *(see page 41)*
- `ET_MeasurandTypes` *(see page 39)*

Every time a command is executed, a new measured value is recorded.

`TestManager` is the name of TestManager-object *(see SoMachine, ETEST, User Guide)* of the application.

Evaluate Test Results

Overview

The ETEST framework provides the test results of a completed test and shows the progress of a running test in the **Test results** view.

NOTE: If test cases are built for function blocks contained in libraries, verify the compatibility of any library updates as it concerns your test cases.

Executed Tests

The **Tests runs** list contains the tests that have been executed since Logic Builder has been started.

Buttons	Description
Export	Stores selected test results in XML format. (**.testresult*).
Import	Opens test results stored in XML format.
Print report	Opens an HTML report in the web browser.

Test Results

Click an entry in the **Tests runs** list to open the test in the area **Test results**.

Imported test results are also opened in the area **Test results** and shown in the list of the most recently executed tests. The area **Test results** is automatically opened when a test is started. It shows the result (or the state) of a test.

The area **Test results** provides two sections.

- At the top, there is the summary of the test results that contains the name of the executed POU and the total result.
- Below, there is a section with various tabs:

Tab	Description
Test results	Tree view which shows the test series and test cases. It displays the individual results for each node.
User comment	User name and comment. Shows by default the Windows user name. You can add a comment to the **Comment** box. This data is stored together with the test result.
Products	Lists the libraries available in the project during the test.
Measured values	Lists measured values that were recorded during the execution of the test. Used to measure time values and the size of data structures.

Glossary

A

application

A program including configuration data, symbols, and documentation.

C

configuration

The arrangement and interconnection of hardware components within a system and the hardware and software parameters that determine the operating characteristics of the system.

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

E

expansion bus

An electronic communication bus between expansion I/O modules and a controller.

I

I/O

(*input/output*)

P

program

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

Index