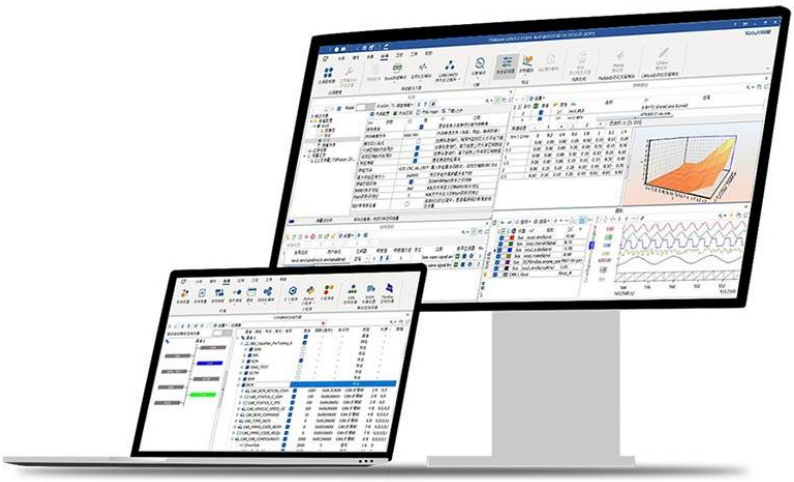
TSAutotest Test Case Editor User Manual



Key words	TSAutotest, Test Case Editor

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

Revision History:

Version	Date	Description
V1.0.0	2026.08.06	Initial release

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

Contents

1 Introduction 5
1.1 About this User Manual 5
1.2 Product Overview 5
1.3 Terms and Abbreviations 6
2 Functional Overview 7
3 Quick Start 8
4 Features 9
4.1 ALM Cloud Platform 9
4.1.1 Test Resource Management 9
4.1.2 Documents Management 10
4.1.3 Remote Project Management 11
4.1.4 Edit Mode 12
4.1.5 AI-Assisted Test Case Design 13
4.2 Local Project Management 13
4.3 Test Case Editing 14
4.3.1 Local Mappings 16
4.3.2 Local Variables 17
4.4 Test Liabrary 17
4.5 Signal Management 18
4.6 Language Switching 19
4.7 Real-Time Validation 20
5 Example 21
5.1 Configure the TSMaster Project 21
5.2 Configure the TSFMU Resource Package 22
5.3 Open the Project 23
5.4 Create a Test Case File 23
5.5 Add Test Steps 24
5.6 Import the Test Case 25
5.7 Execute the Test Case 26
6 Appendix 28
6.1 Read and Write Steps 28
6.1.1 Read Step 28
6.1.2 Write Step 30
6.2 Test Library Step Description 31
6.2.1 Data Operations 32
6.2.2 Flow Control 32
6.3 Common Steps Description 36
6.3.1 MAPort 36
6.3.2 EES Pin 37
6.4 Mini API Description 38
6.5 Expression Description 38
7 Troubleshooting 41

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

7.1 Git Access Denied 41
8 Contact Information 43

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

1 Introduction

This manual describes the currently implemented features of the test case editor — TSAutotest, including test case editing, project and resource loading, test step functionality, and operation examples.

1.1 About this User Manual

There are formatting conventions and symbols used in this manual:

Style	Usage
【】	Indicates clickable UI elements such as menus, windows, and dialog names.
bold	Used for emphasis on text information.
Hyperlink	Represents links and cross-references.
<CTRL>+<S>	Indicates keyboard shortcuts.

Symbol	Usage
	Indicates a warning message to alert users to important safety risks.
	Warns of potential electrostatic discharge that may cause equipment damage.
	Provides supplementary or additional information.
	Presents an example for clarification.
	Offers step-by-step guidance.

1.2 Product Overview

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

TSAutotest is a test case editor designed primarily for creating and editing test cases. Users can organize test steps in a graphical editor, manage local mappings and local variables, and use signal read/write steps and Mini API calls based on resource files exported from TSMaster. Edited test case files (.cas) can be imported into TSMaster and converted into graphical test programs for execution.

When used with an ALM system, TSAutotest can serve as part of an ECU software automated testing system that provides bidirectional traceability and continuous integration capabilities. Test resources on the ALM system can be edited, and TSMaster can be scheduled by the ALM system to execute test cases.

1.3 Terms and Abbreviations

Term	Description
Test Case	Automated test logic consisting of multiple test steps. The current version primarily uses a tree-based step editor.
Test Step	The smallest editable unit in a test case, such as Wait, Loop, IfThenElse, Read, Write, MiniApi, or MAPort.
Local Mapping	A mapping label used within a test case to associate a short name used by a step with an actual signal or resource path.
Local Variable	A variable used within a test case to store API return values, signal read values, or data related to expression calculations.
TSMaster Project	A TSMaster .TSProj project file. Currently, TSAutotest reads exported resources from the project directory through a temporary entry point.
TSFMU	A resource package exported from TSMaster. It is currently used to provide signal files and Mini API description files.
ALM	Application Lifecycle Management. A system used to manage test resources and schedule test execution.
EES	Electrical Error Simulation. A function used to inject electrical faults into ECU pins during testing.
.cas	The test case file format used by TSAutotest. It stores test steps, local mappings, and local variables in XML format.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

2 Functional Overview

The current version of the TSAutotest test case editor provides test step editing and basic project management features. Edited test cases can be imported into TSMaster and converted into flowchart programs for test execution.

The main features include:

- ALM Cloud Platform
- Local Project Management
- Test Case Editing
- Signal Management
- Test Library
- Language Switching
- Real-Time Validation

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

3 Quick Start

1. Open the test case editor, click **Open Local Project**, and select a TSMaster project.
2. In the Project Explorer, select the **Test Cases** folder.
3. Right-click to select **Add File** from the context menu, and enter the test case file name.
4. Double-click the newly added test case file.
5. Drag a step from the Test Library to the test case editing area.

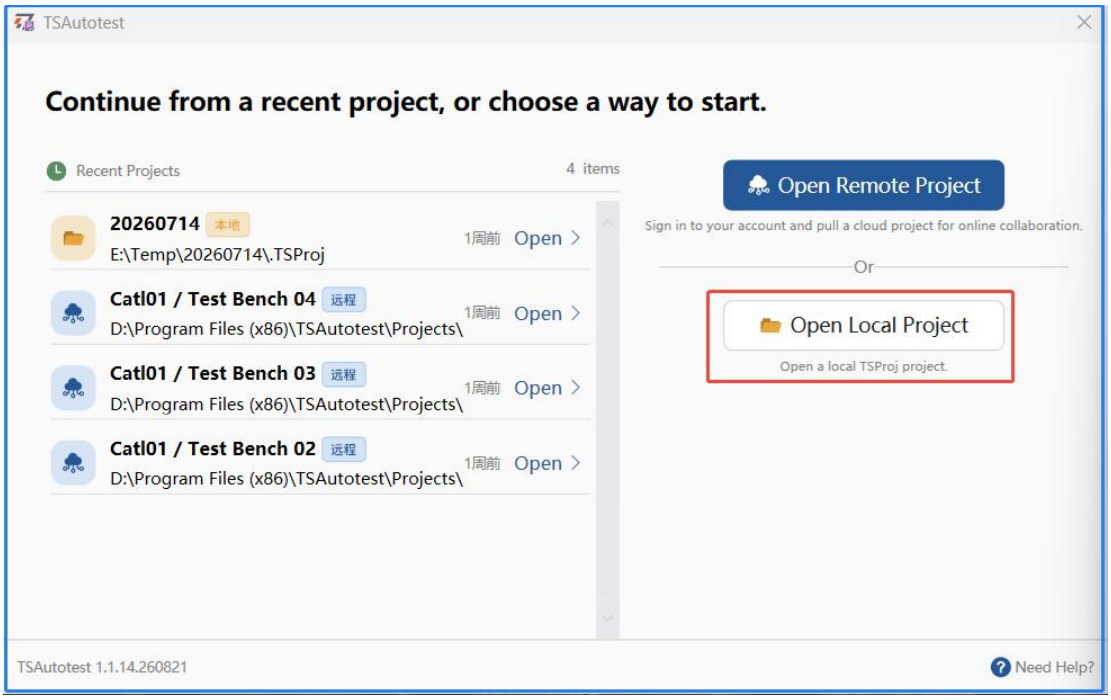


Figure1 Quick Start

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

4 Features

The core functionality is test case editing. Other modules, such as the Test Library and Signal Tree, enhance test case editing. For example, the Signal Tree provides signal read/write steps, the Test Library provides flow control and data operation steps, and project management provides the foundation for editing multiple test cases.

When used with the ALM cloud platform, the test case editor supports remote project management and synchronization, AI-assisted test case design, and other capabilities.

4.1 ALM Cloud Platform

When used with the ALM cloud platform, the test case editor can retrieve test resources from the cloud platform for local editing and manually synchronize the edited resources back to the cloud platform.

4.1.1 Test Resource Management

After logging in to the ALM cloud platform, you can select an ALM project and a test bench configured for that project. Each test bench is associated with a TSMaster project. After selecting and confirming a test bench, you can use Git to pull the associated TSMaster project from the repository to the local computer for editing. If the selected test bench is not associated with a TSMaster project, you can upload a locally configured TSMaster project to the repository and associate it with the test bench.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

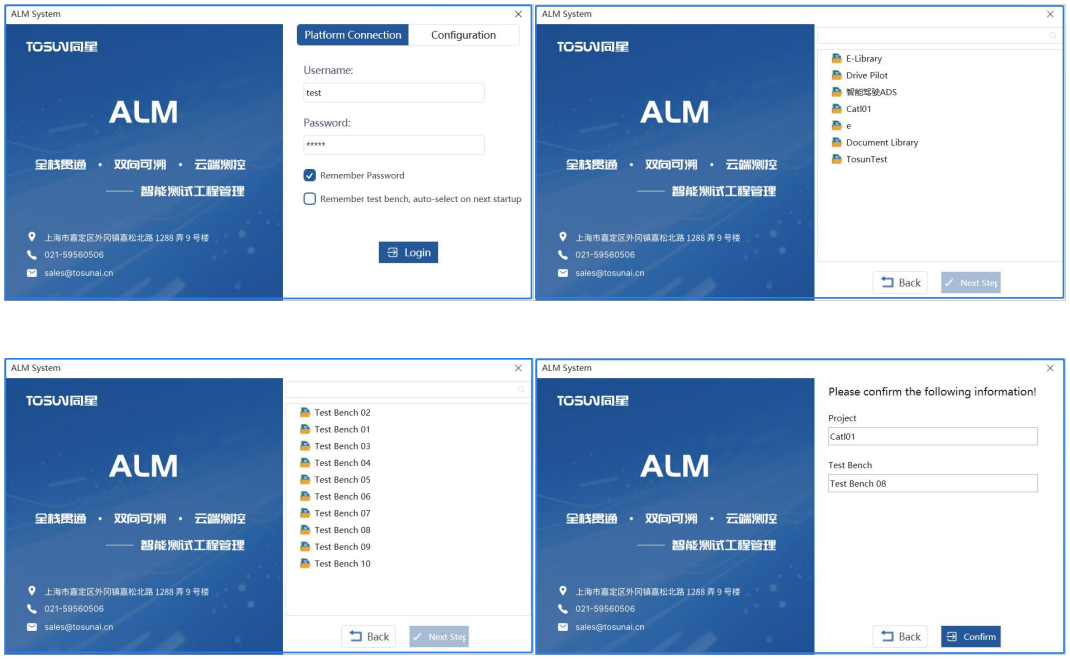


Figure2 Test Resource Management

4.1.2 Documents Management

Unlike opening a local project, when a remote project is retrieved from the ALM cloud platform, the files in the TSFMU resource package must be manually added to the document toolbox the first time. The document toolbox contains two categories: **Project Documents** and **Mapping Documents**. To add a file, right-click the corresponding category node and select **Import**, then select a JSON file extracted from the TSFMU package, such as can.json, lin.json, or apis_all.json in the file selection dialog box. These files provide data sources for signal read/write operations and Mini API calls used by test steps.

Project documents can serve as a knowledge base for AI-assisted test case design, enabling the AI to provide more accurate test case designs.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

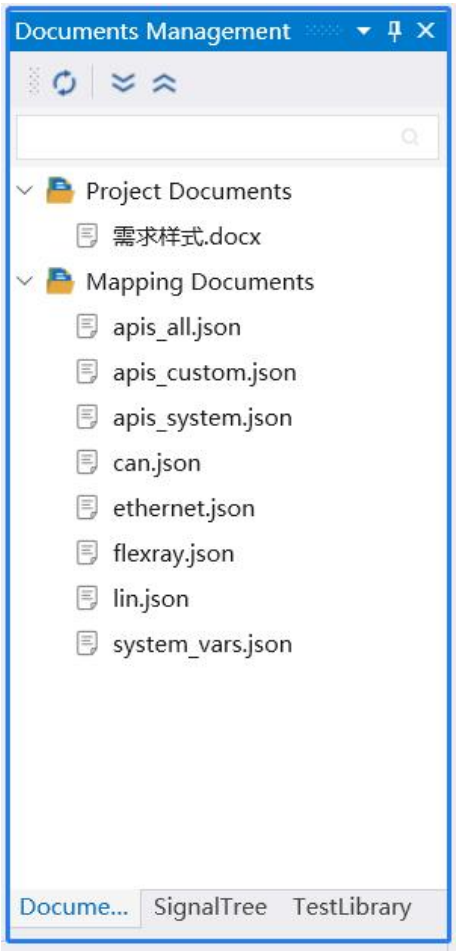


Figure3 Documents Management

4.1.3 Remote Project Management

After a TSMaster project has been retrieved to the local computer and test cases have been edited, you can synchronize the project using Update and Commit. This keeps the local project up to date and allows modifications to be pushed to the repository in a timely manner.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

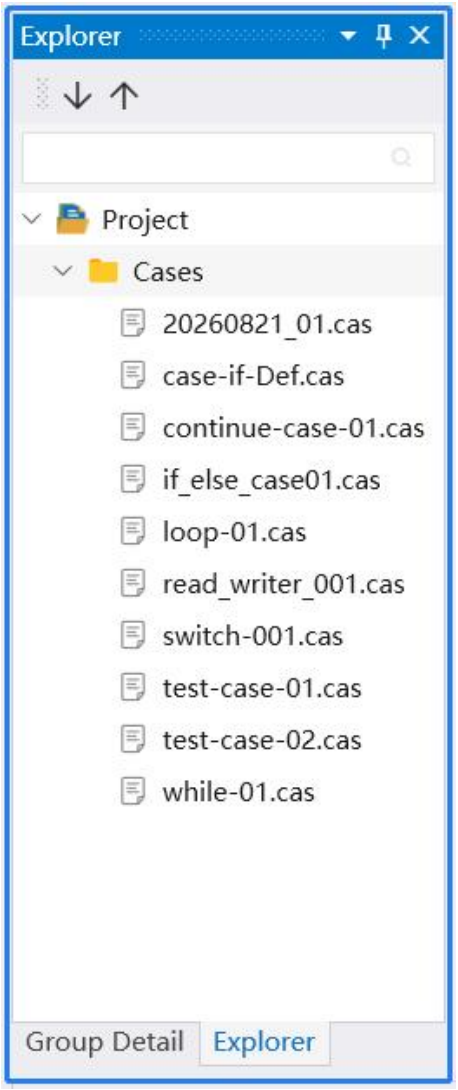


Figure4 Remote Project Synchronization

4.1.4 Edit Mode

The test case editor supports multiple users editing test cases in the same remote project. However, only one user can edit an individual test case file at a time. Therefore, the file is locked when entering edit mode. The default mode is read-only mode.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

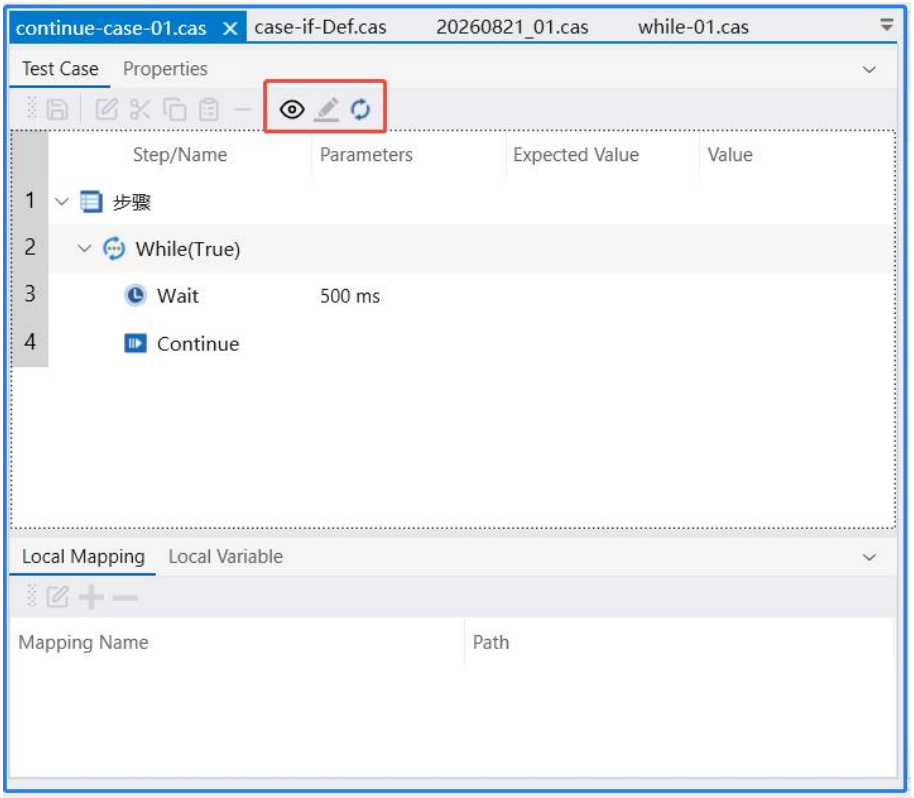


Figure5 Read-Only Mode and Locked Mode

4.1.5 AI-Assisted Test Case Design

AI can combine the requirement files and signal files in the current test resources to design test cases based on user requirements. After the AI generates test case groups, they can be applied to the current test case file.

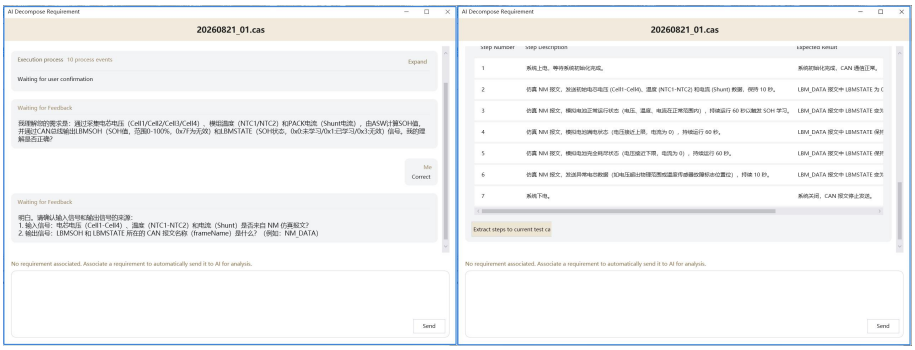


Figure6 AI-Assisted Test Case Design

4.2 Local Project Management

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

Project management is implemented through the project Explorer, which displays project files in a tree structure and provides management operations through the context menu. These operations include creating, cutting, copying, pasting, and deleting directories and test case files, as well as importing existing test case files from external sources into a specified directory.

After a TSMaster project is opened using **Open Local Project** on the start page or **Open** in the main interface, the project files are loaded from the relative path ./TSTest/Project. The system automatically saves the most recently opened projects, up to 20 entries. Previously opened projects can be opened directly from the history list on the start page. Remote projects support automatic login.

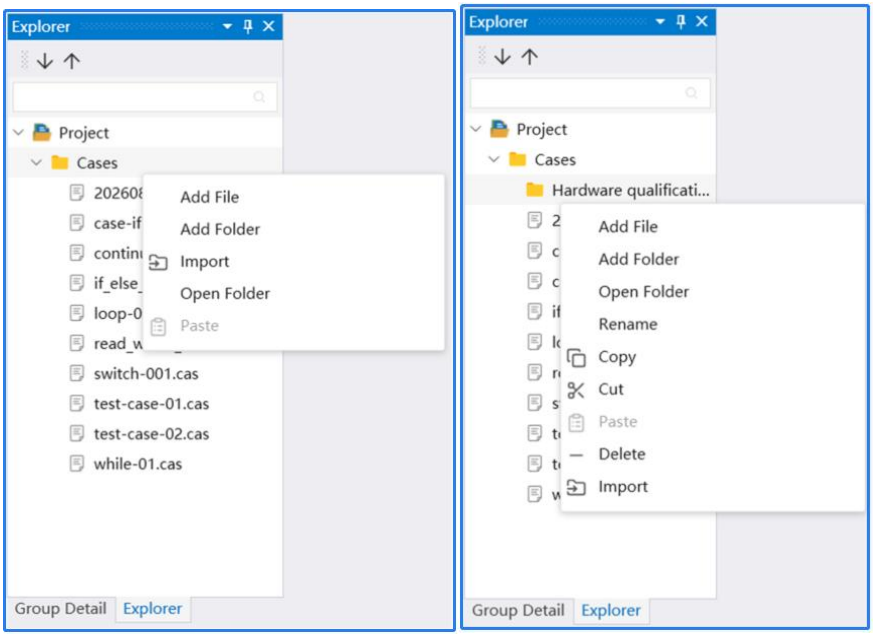


Figure7 Project Management

4.3 Test Case Editing

After double-clicking a test case file to edit it in the project Explorer, its editing view is displayed in a document tab. When a test case is modified, an * is displayed at the end of the title to indicate unsaved changes. Click the Save button to save the test case file.

Test cases are displayed and edited as a tree that represents the order, position, and hierarchy of test steps. Some steps can contain child steps. For example, when the condition of an IfDef step is met, the child steps within the IfDef step are executed.

Test cases can be edited in the following ways:

- Drag a target step from the Test Library to the editing area to add a Test Library step.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

- Click Read or Write on the Signal Tree to add a read or write step.
- Double-click a target step to edit an existing step.
- Drag an existing step to another location to change its position, order, or hierarchy.
- Use the context menu to edit, cut, copy, paste, delete, and perform other operations.
- Use the toolbar to save, edit, cut, copy, paste, and delete.
- Use keyboard shortcuts: Edit: <CTRL>+<E>; Cut: <CTRL>+<X>; Copy: <CTRL>+<C>; Paste: <CTRL>+<V>; Delete: <Delete>.

In remote project mode, the test case editor also supports file locking and unlocking. After a file is locked, other users enter read-only mode, ensuring that only one user edits the same test case file at a time. The Lock, Unlock, and Refresh Lock Status buttons on the toolbar are available only in remote project mode.

Step Drag-and-Drop Restrictions:

- Break and Continue steps can only be placed inside While or Loop loops.
- Branch child steps such as Then, Else, Case, and Default cannot be moved to other locations.
- IfThenElse and SwitchCase steps do not allow child steps to be dragged into them.
- When pasting across test case files, the system automatically creates missing local variables and local mappings in the target test case and decrements the corresponding reference counts in the source test case.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

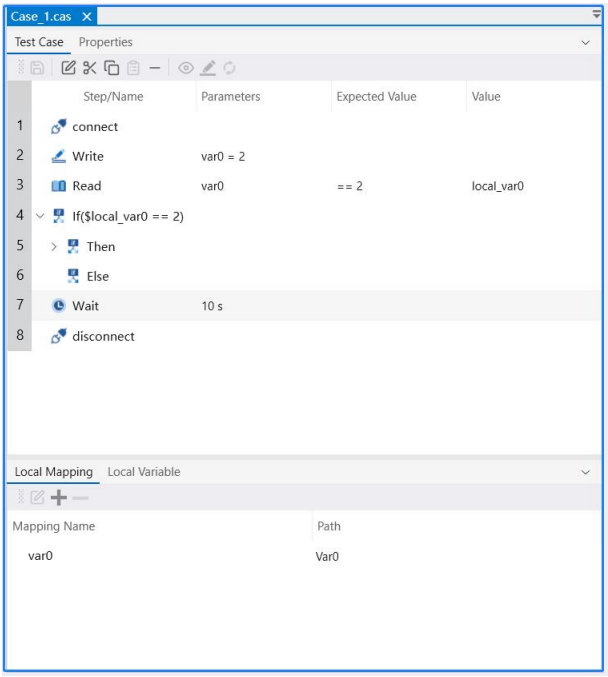


Figure8 Test Case Editing

4.3.1 Local Mappings

A mapping name is a short alias associated with a specific signal through its path. The associated signal can be read or written using the alias.

Mapping names must be unique within an individual test case. Existing mappings can be modified, deleted, or added in the area below the test case editing area. In addition to manually adding mappings in the Local Mappings area, you can right-click a signal in the Signal Tree and select **Local Mapping** to directly create the selected signal as a local mapping for the current test case. When an existing mapping is edited, the system recursively updates all Read and Write steps that reference the mapping.



Figure9 Local Mapping

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

4.3.2 Local Variables

A local variable can be understood as a variable whose scope is limited to an individual test case. It can be created by saving the value from a Read step or by manually adding a local variable.

Local variables can be used by Test Library steps such as Bit extract, IfThenElse, IfDef, While, and SwitchCase. The supported data types are Int32, UInt32, Int64, UInt64, Double, String, UInt8 Array, Int32 Array, Int64 Array, and Double Array.

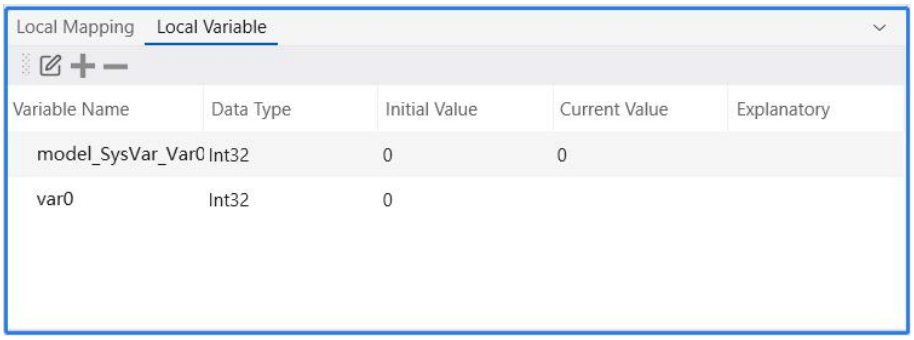


Figure10 Local Variable

4.4 Test Liabrary

The Test Library provides steps for flow control, data operations, common steps, and Mini APIs.

- Flow Control: includes Wait, Group, Exit, IfThenElse, IfDef, While, Loop, Break, Continue, SwitchCase, and Return. These steps are used to control test case execution or support test case design.
- Data Operations: currently include the Bit extract step. Other types of data operation steps, such as Calculation, will be added in future versions
- Common Steps: include EES Pin, as well as Start Simulation and Stop Simulation under MAPort.
- Mini API: the step can call interfaces provided by TSMaster. As with signal management, the data source for callable Mini APIs depends on the resource package exported from TSMaster. Custom interfaces can also be exported. The Mini API library uses JSON format (apis_all.json) and is organized in a three-level hierarchy: Scope, Group, and API. The Test Library supports search and filtering. Enter a keyword in the search box to display matching steps. Click the Refresh button on the toolbar to reload API resources. The Expand and Collapse buttons expand or collapse all tree nodes.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

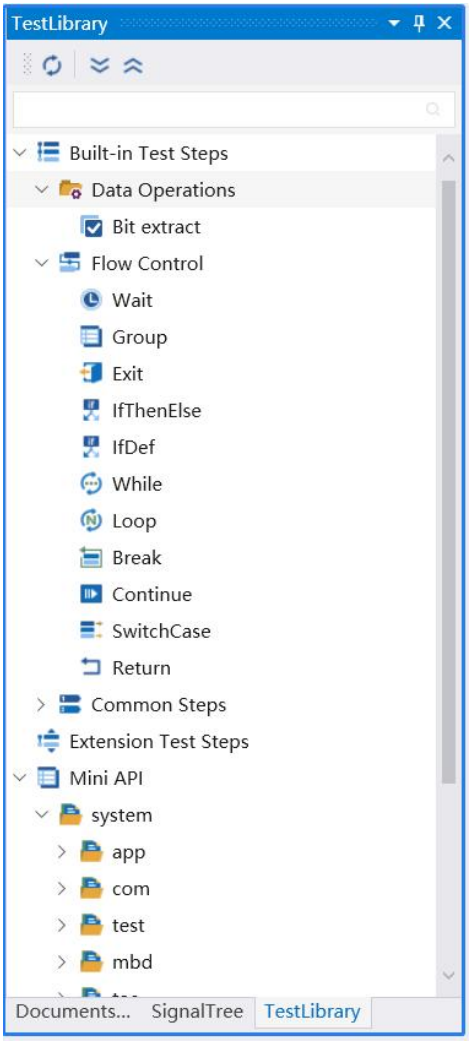


Figure11 Test Library

4.5 Signal Management

The **Signal Tree** module displays signal data sources and provides steps for reading and writing signal values.

The Signal Tree currently obtains its data from TSFMU resource packages exported from TSMaster. When a TSMaster project is opened using the **Open** button, the system searches for and loads the most recently dated TSFMU resource package from the fixed path ./TSTest/Resources/. Supported signal types include system variables (SysVar) and database signals for bus types such as CAN, LIN, FlexRay, and Ethernet. The Signal Tree is organized hierarchically by bus type, channel (Network), node (Node), frame (Frame), PDU, and signal. System variables support scalar types (int8, uint8, int16, uint16, int32, uint32, int64, uint64, float, double, and string) and the corresponding array types.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

After selecting a target signal in the Signal Tree, you can use the context menu to perform the following operations: Read (create a Read step), Write (create a Write step), Global Mapping (add the signal to the Global Mapping Editor), and Local Mapping (add the signal to the local mappings of the current test case). The Signal Tree supports search and filtering, as well as expand/collapse operations.

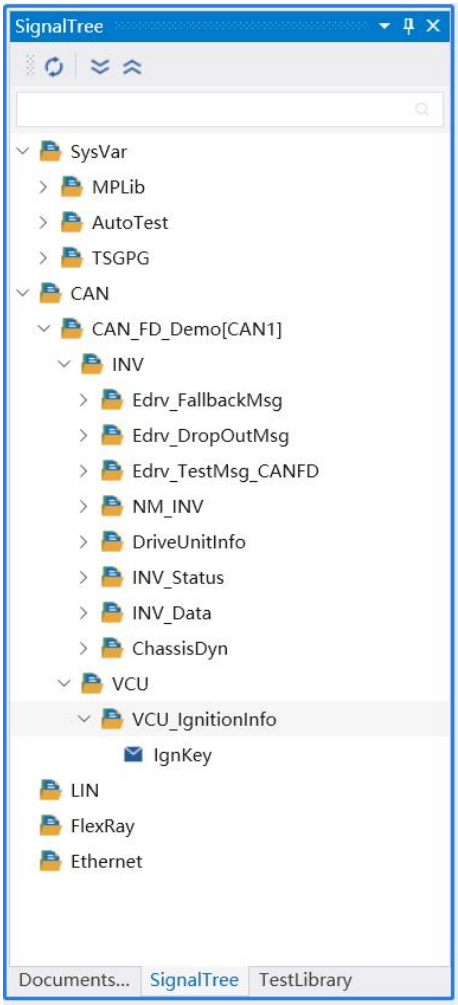


Figure12 Signal Tree

4.6 Language Switching

The test case editor currently supports Chinese and English. Click the **Language Settings** button on the toolbar to open the settings interface and switch languages. The software must be restarted for the change to take effect.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

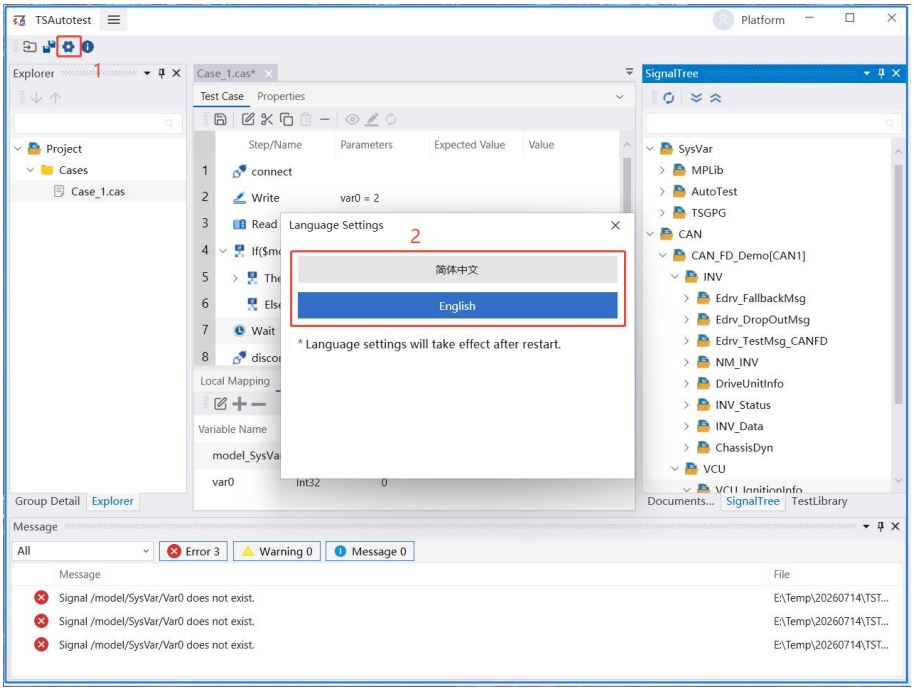


Figure13 Language Switching

4.7 Real-Time Validation

The Real-Time Validation module displays validation results for test case files in the message panel at the bottom. The system automatically detects changes to .cas files through file system monitoring and performs validation automatically after a file is modified.

Validation includes:

- Whether signals exist
- Whether signal data types match
- Whether Mini APIs exist and their parameters match
- Whether local mappings exist
- Whether variable references are valid
- Whether fields are empty, and other checks

Messages are classified as errors, warnings, and information. They can be filtered by file and type. Double-click a message to locate the corresponding step in the test case editor.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

5 Example

This section provides an example of configuring a resource package, creating a project, creating a test case, modifying the value of a CAN signal through the test case, and reading the value to verify whether it meets the expected result.

The procedure is as follows:

1. Configure the TSMaster project.
2. Configure the TSFMU resource package.
3. Open the project.
4. Create a test case file.
5. Add test steps.
6. Import the test case into TSMaster.
7. Execute the imported test case.

5.1 Configure the TSMaster Project

Example objective: create a simulation project and import a DBC database.

- Open TSMaster and click **Create new project**.
- Click **Simulation**.
- Click **CAN RBS Simulation**.
- Select a path.
- Click **Analysis** -> **Database** -> **Display CAN Database**.
- In the CAN Database interface, right-click **Channel 1 [CAN]** and select **Load CAN Database...**.

Select the CAN_FD_Demo.dbc file under the TSMaster installation path.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

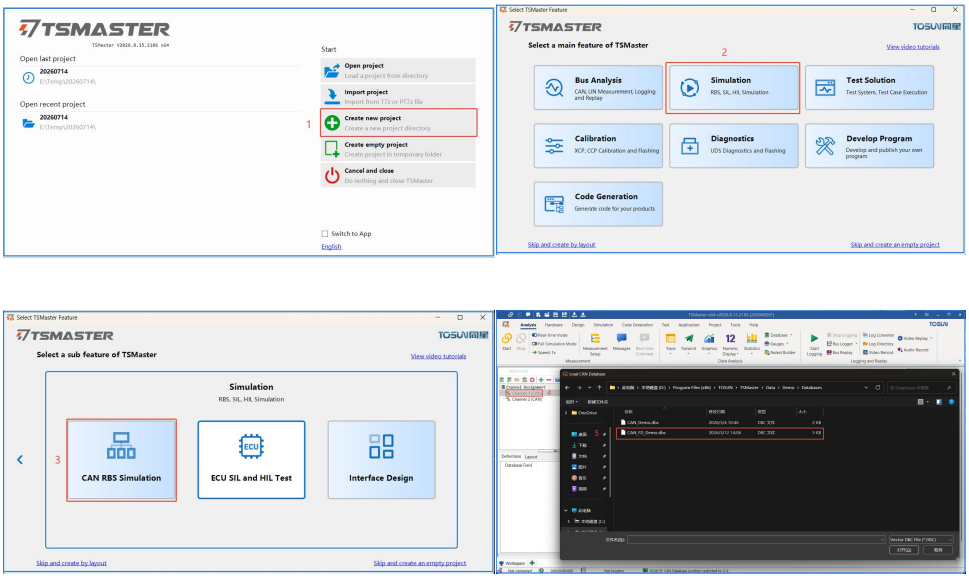


Figure14 Creating a TSMaster Project

5.2 Configure the TSFMU Resource Package

Prerequisites: A project is open in TSMaster, a CAN database has been added to the project, and TSMaster supports the TSMaster XIL feature.

- Click **TSMaster XIL** under **Simulation**.
- Click the **Export TSFMU** tab.
- Select the required signals and APIs.
- Click **Export TSFMU**.
- In the folder selection dialog box, select the TSTest/Resources directory under the current project. If the directory does not exist, create it manually.
- Click **Save**.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

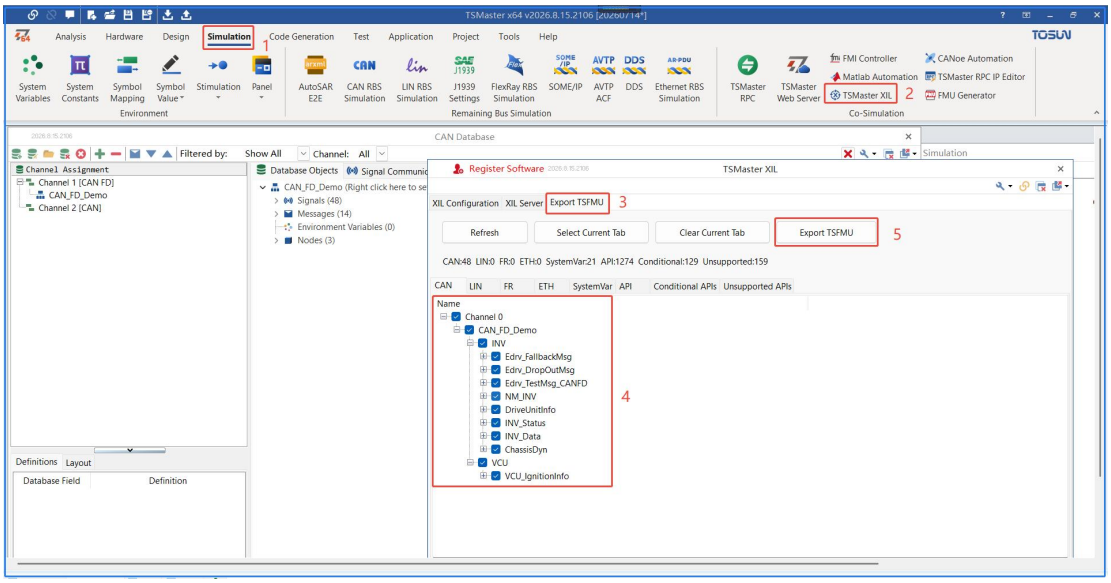


Figure15 TSFMU Resource Package Configuration Example

5.3 Open the Project

Note: This is a temporary solution for quickly trying out the test case editing features.

Prerequisites: The required signal files and APIs have been exported as a TSFMU resource package and saved under the project-relative path TSTest/Resources/.

1. Click **Open Load Project** in TSAutotest.
2. In the file selection dialog box, select the TSMaster project to open. The file type is .TSProj.
3. Click **Open** in the file selection dialog box.

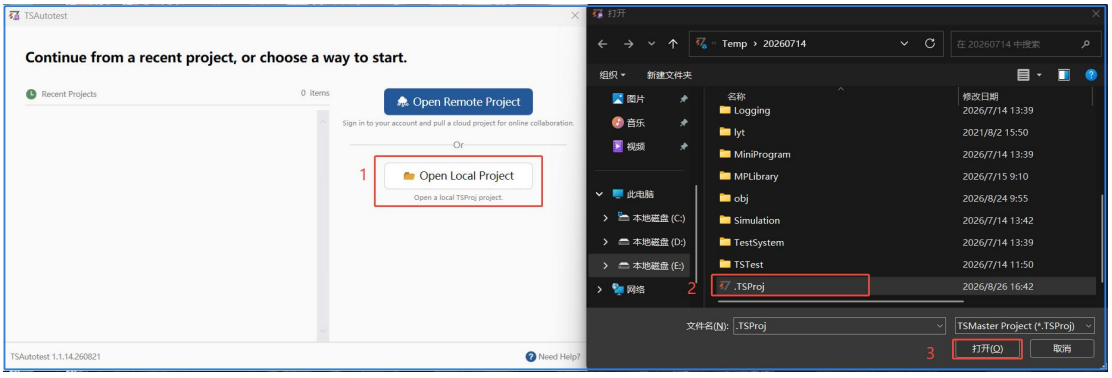


Figure16 Opening a Project

5.4 Create a Test Case File

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

- Select the **Test Cases** node in the project **Explorer**.
- Right-click to select **Add File** from the context menu.
- Enter the file name in the **Add File** dialog box.
- Click **Confirm**.
- Double-click the newly added test case file.

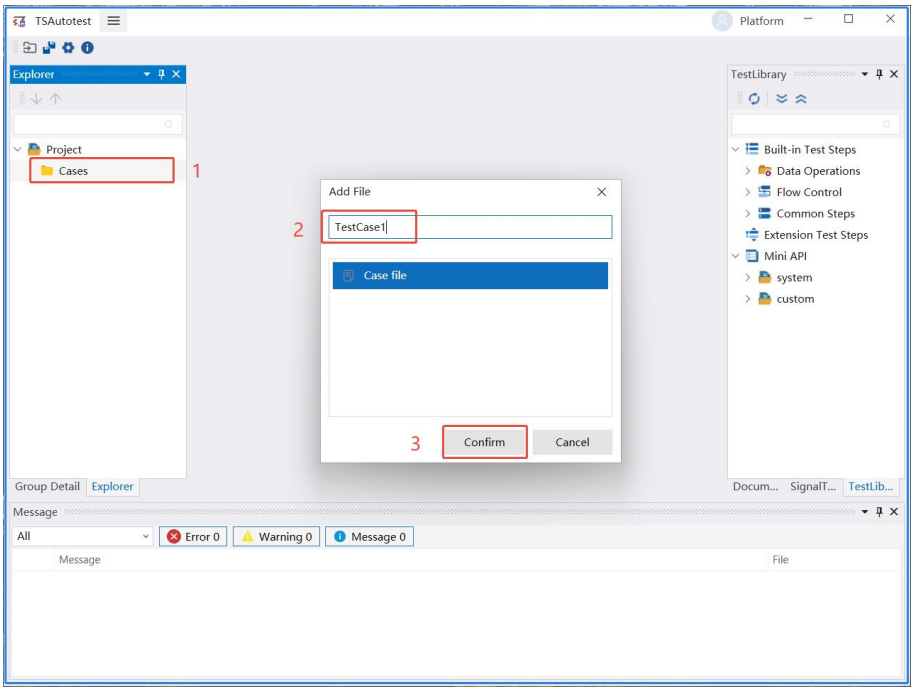


Figure17 Adding a Test Case File

5.5 Add Test Steps

Prerequisites: A test case file is open, the TSFMU resource package has been loaded, and the resource package contains signals.

- In the **Signal Tree**, search for the **IgnKey** signal and select it -> click the context menu -> click **Write**.
- Enter 1 in the **Value** input box.
- Right-click the IgnKey signal again and click **Read**.
- Select **Assessment**.
- Enter 1 in the **Measure** input box.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

- Click **Confirm**.

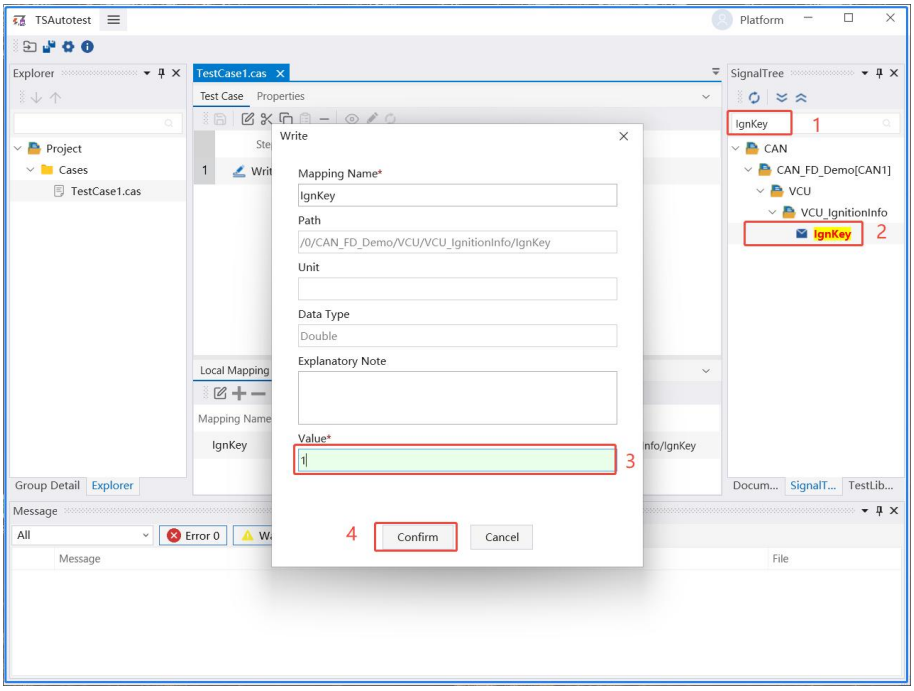


Figure18 Adding Read and Write Step 1

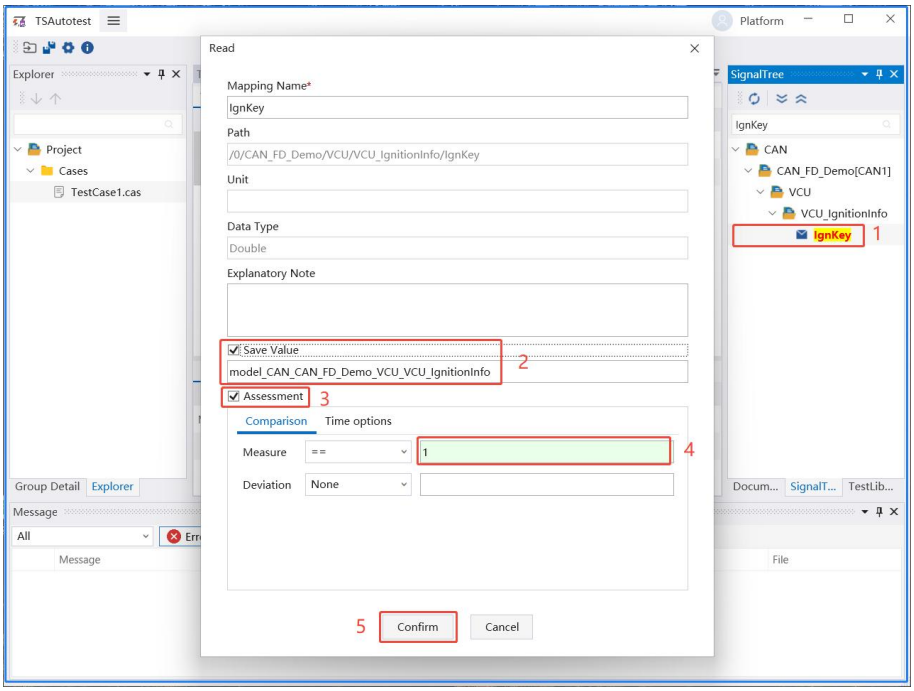


Figure19 Adding Read and Write Step 2

5.6 Import the Test Case

- Click **Design** -> **Graphic Program Builder**.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

- Select **TOSUN Importer**.
- Click **Add**.
- Select the edited .cas test case file.

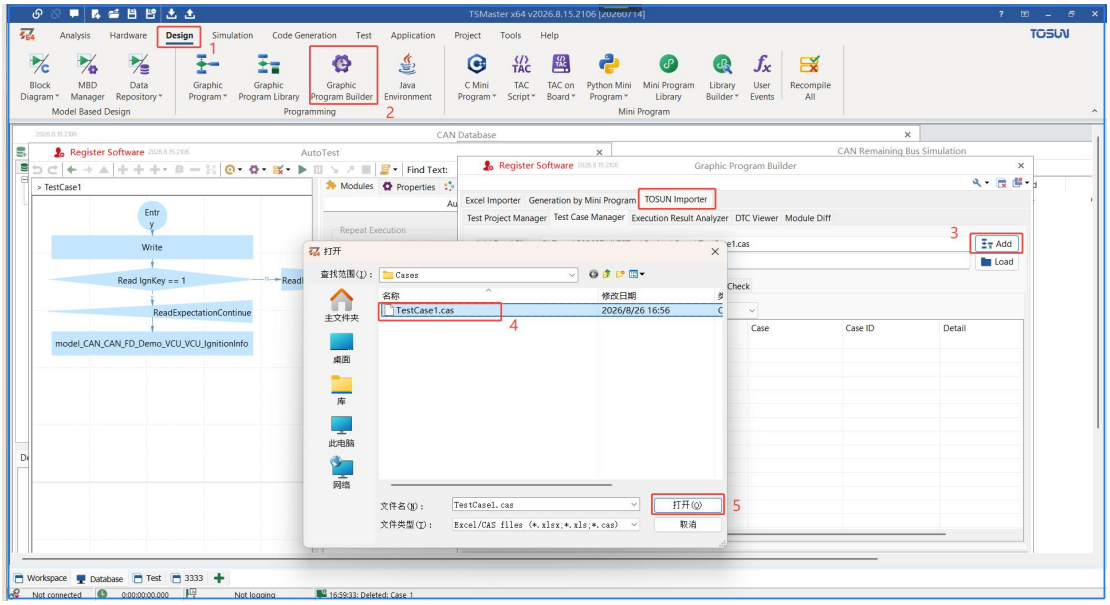


Figure20 Importing a Test Case

5.7 Execute the Test Case

- Click **Simulation** -> CAN Bus Simulation.
- Enable the **VCU_IgnitionInfo** frame under the **VCU** node.
- Click **Run** RBS simulation.
- Click **Run** current graphic program builder.
- Check whether the signal value meets the expected result.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

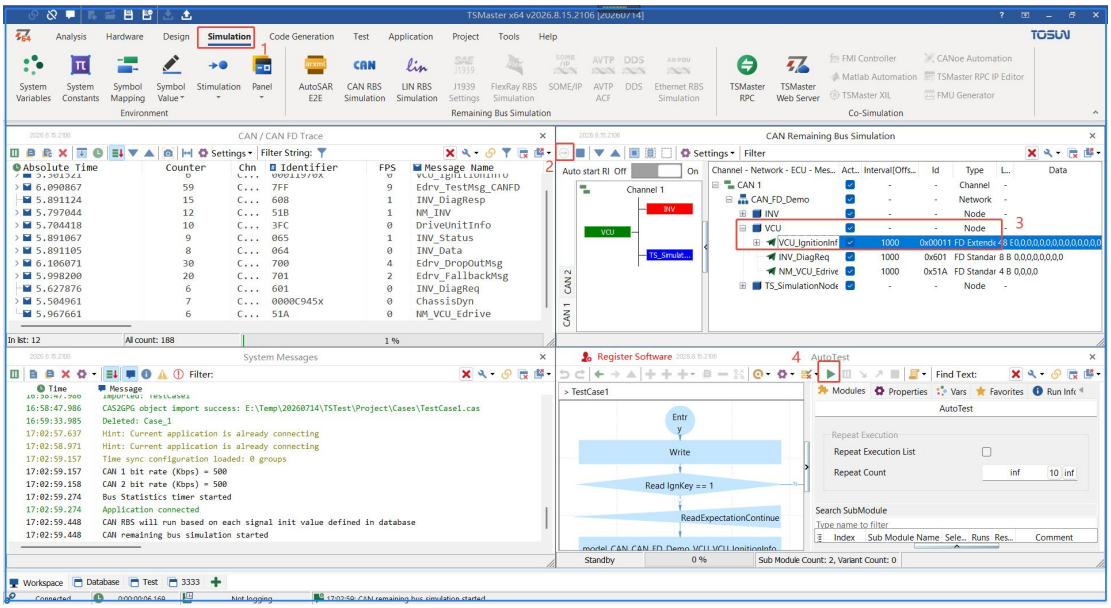


Figure21 Executing a Test Case Step 1

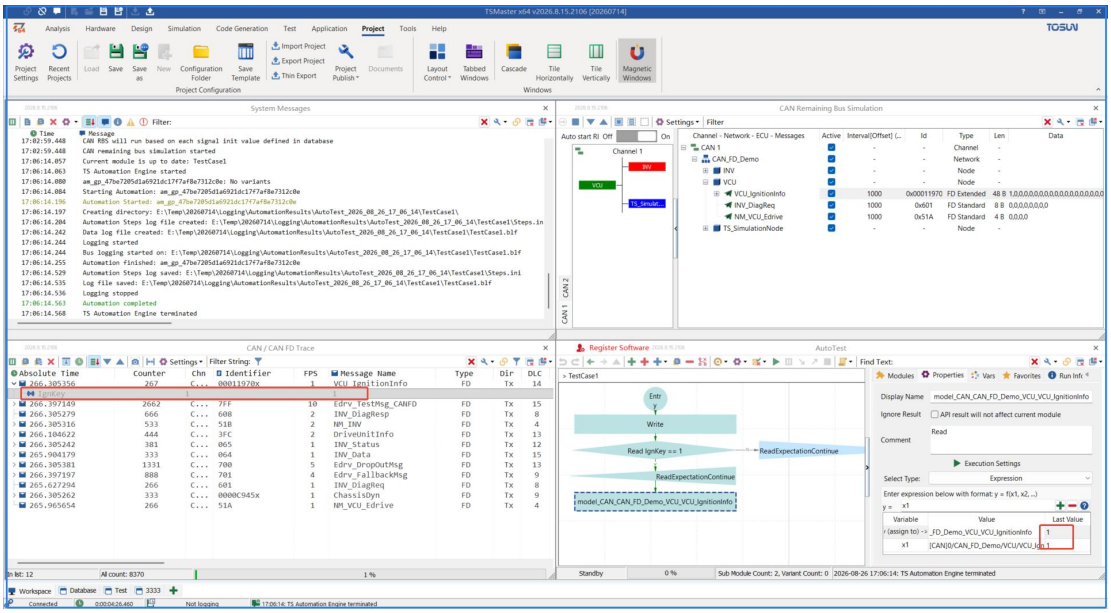


Figure22 Executing a Test Case Step 2

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

6 Appendix

This chapter describes the detailed functionality of the test steps. Refer to the corresponding test step sections for details about each step and its configuration options.

6.1 Read and Write Steps

6.1.1 Read Step

This step is used to read signal values. Supported signal types include TSMaster system variables and bus signals. The read value can be stored in a local variable, which can then be used by other types of steps, such as While and IfThenElse in flow control. The read value can also be evaluated.

6.1.1.1 Numeric and String Types

The comparison operators available for the Measurement setting differ between numeric and string types. The following configuration options are provided:

- Mapping Name: Defines an alias used to access a signal at a specified path.
- Save Value: Specifies whether to save the read signal value. When enabled, the read value is assigned to a local variable.
- Assessment: Checks whether the read value meets the expected result. The evaluation condition is configured through Measure, the tolerance range through Deviation, and the evaluation timing through Time Options.
  - Comparison: includes **Measure** (numeric types: ==, !=, >=, <=, >, <; string types: identical, starts with, ends with, contains, does not contain, dissimilar), and **Deviation** (numeric types: None, Absolute)
  - Time Options: include Type, Timeout, Minimum duration, and Polling cycle, and available time option types include:
    - None
    - Wait until true (continuously read and evaluate until the condition becomes true or the timeout expires)
    - True for at least (the condition must remain true for at least a specified duration)

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

- True for at least within (the condition must remain true for at least a specified duration within the total Timeout period)
- Synchronization only (no evaluation)

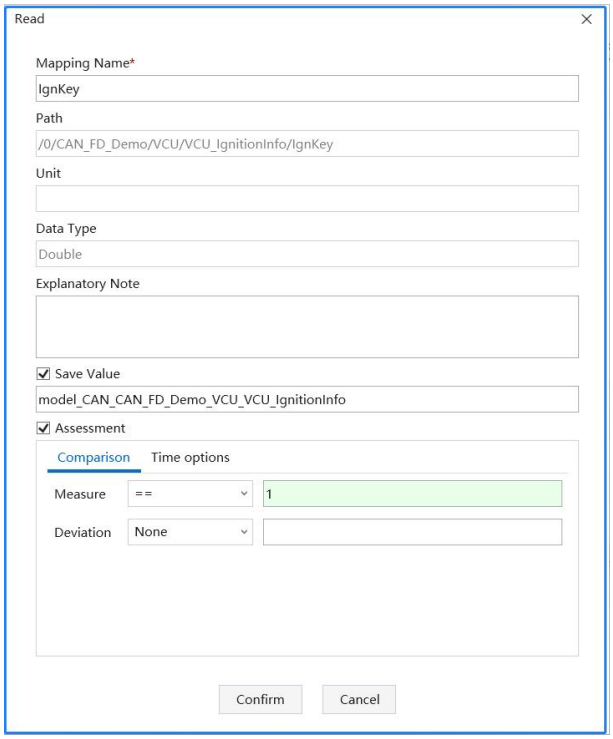


Figure23 Read Step — Numeric and String Types

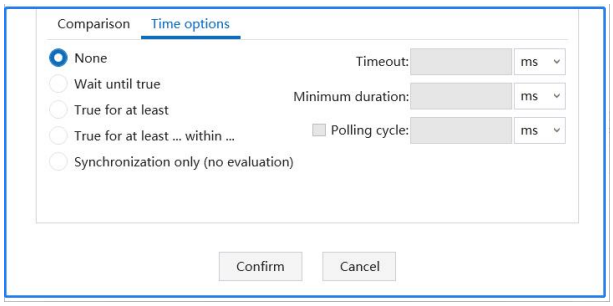


Figure24 Time Options

6.1.1.2 Array Types

When the data type is an array, comparison rules can be configured individually for each element or applied uniformly to all elements.

When **Single** is selected, modify the array length to specify the number of array elements to compare, then click the cell below each array index to configure its evaluation rule.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

When **Common** is selected, click **Edit** to configure a common evaluation rule for all array elements.

Compared with numeric types, array types provide the following additional configuration options:

- Assessment type (**Single** or **Common**): When **Single** is selected, each array element can have its own comparison rule. When **Common** is selected, all array elements share the same rule.
- Array Length
- Assessment settings for each array element

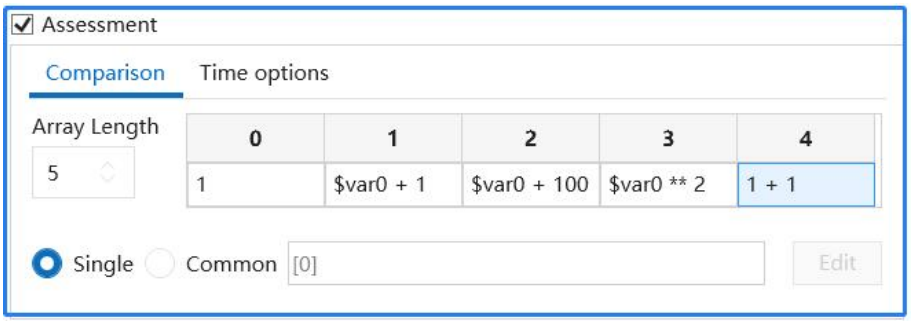


Figure25 Read Step — Array Type Assessment

6.1.2 Write Step

This step is used to write signal values. Supported signal types include TSMaster system variables and bus signals.

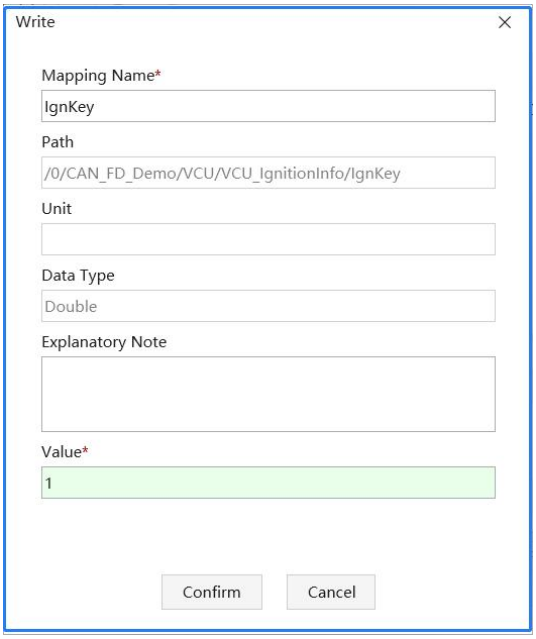


Figure26 Write Step

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

6.2 Test Library Step Description

The Test Library contains fixed test steps, including data operations, flow control, common steps, and Mini APIs loaded through the TSFMU resource package.

The Test Library is displayed as a tree. Click **Refresh** to reload Mini API resources, **Expand** to expand all tree nodes, and **Collapse** to collapse all tree nodes.

Select a target item in the tree and drag it to the test case editing area to add a new test step.

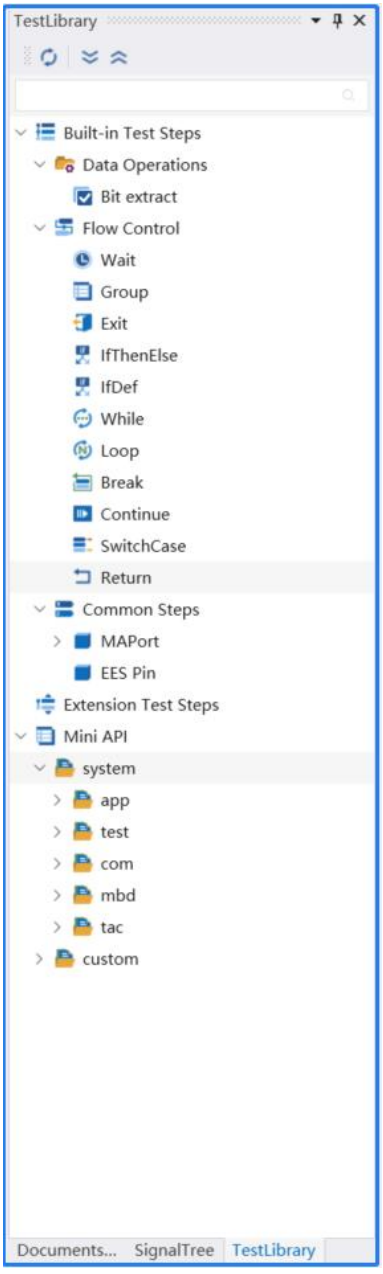


Figure27 Test Library

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

6.2.1 Data Operations

Data operations include the Bit extract step. This step extracts data from a local variable or constant according to the configured Start bit, Bit length, and Byte order properties, and assigns the extracted data to another local variable. The preview area displays the bits selected by the configuration.

The **Expression** input box accepts constants and references to local variables using the $ symbol.

The **Save in** input box specifies the variable in which to store the extracted data.

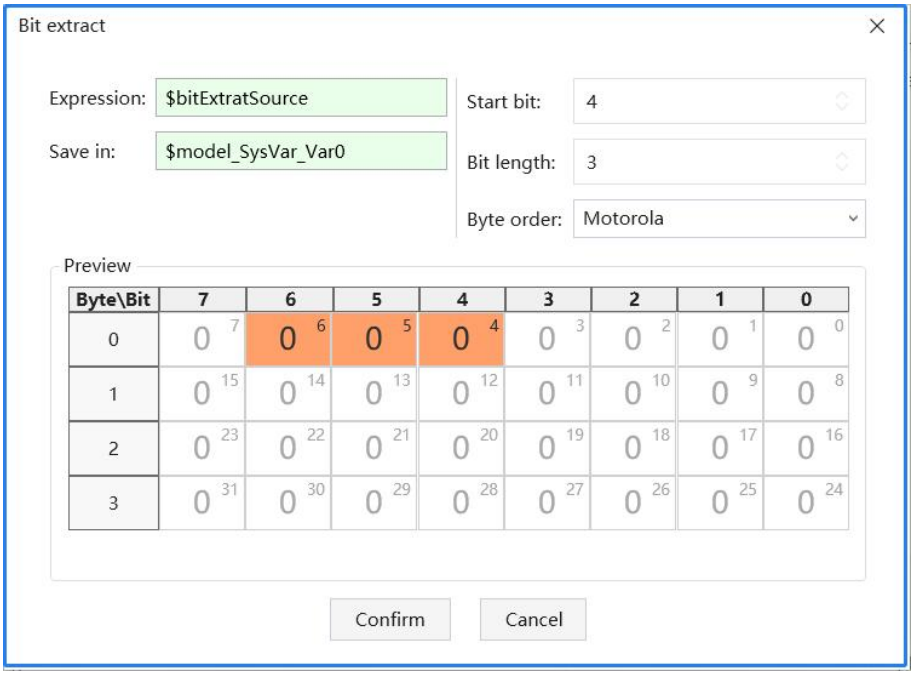


Figure28 Bit Extract Step

6.2.2 Flow Control

Flow control includes Wait, Group, Exit, IfThenElse, IfDef, While, Loop, Break, Continue, SwitchCase, and Return.

6.2.2.1 Wait

The Wait step contains two parameters, delay time and unit, and is used to wait for a specified duration. Available units are ms, s, min, and h.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

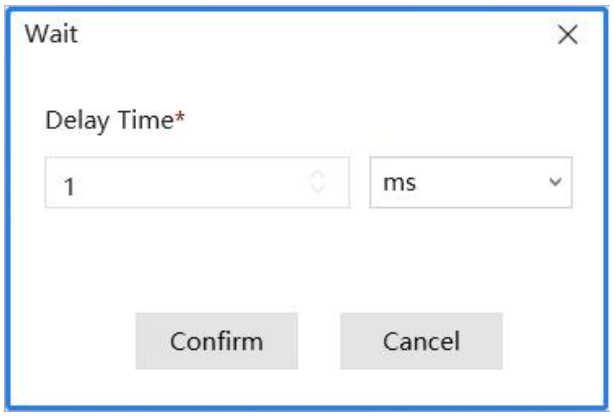


Figure29 Wait Step Editing

6.2.2.2 Group

The Group step does not perform any actual action. It is used only to group test cases and contain child steps. The step contains two key parameters, **Step/Name** and **Expected Result**, which can be used to describe the test case name and expected result.

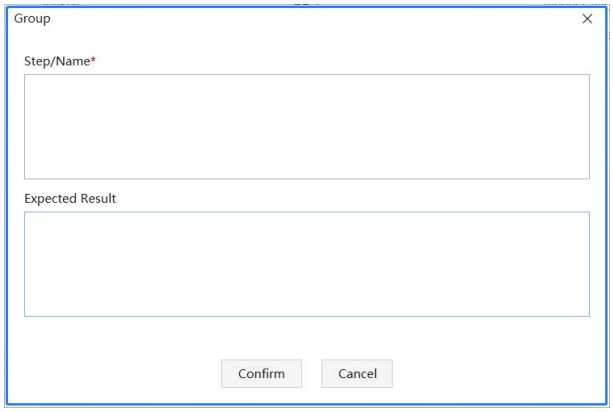


Figure30 Group Step Editing

6.2.2.3 Exit

This step is used to exit the execution of the current test case.

6.2.2.4 IfThenElse

IfThenElse is a conditional branching statement that contains a condition expression and two child branches, Then and Else. When the condition or its result is True, the steps under Then are executed; otherwise, the steps under Else are executed. Local variables can be referenced in the condition expression using the $ symbol.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

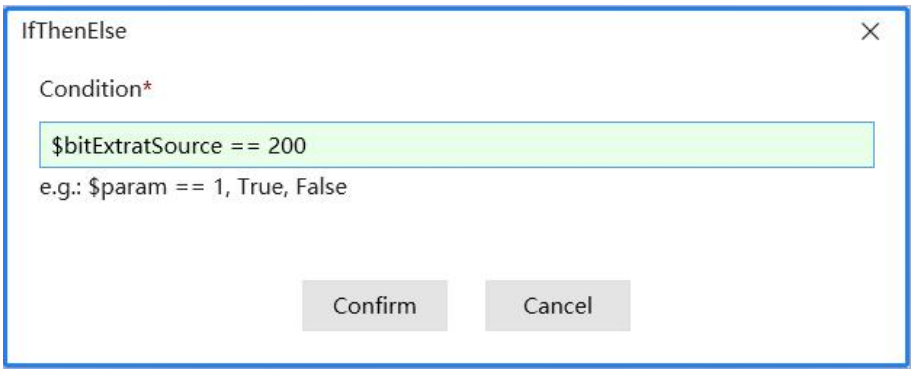


Figure31 IfThenElse Step Editing

6.2.2.5 IfDef

IfDef is a conditional statement that contains an expression. When the expression or its result is True, the child steps within IfDef are executed.

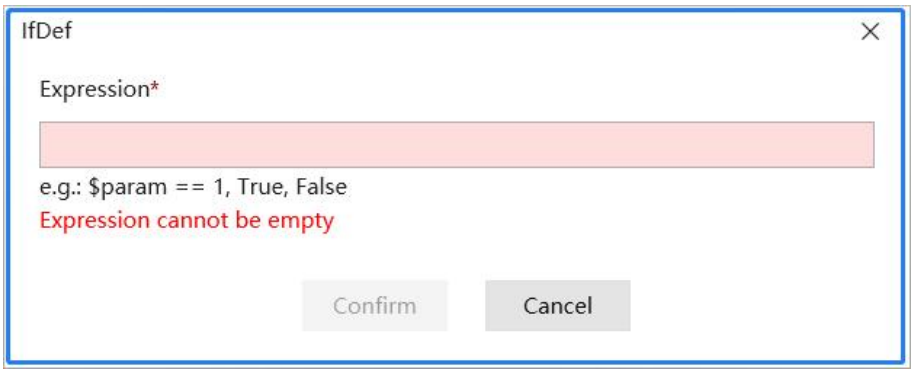


Figure32 IfDef Step Editing

6.2.2.6 While

While is a loop statement that contains a loop condition. When the loop condition or its result is True, the child steps within the While step are executed. Use a Break step to exit the loop or a Continue step to skip the remaining child steps and start the next iteration.

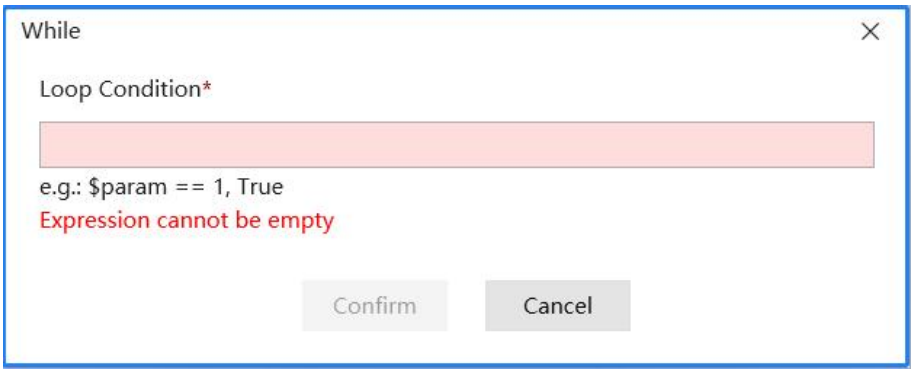


Figure33 While Step Editing

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

6.2.2.7 Loop

Loop is a loop statement that contains a loop count. During execution, the child steps are executed repeatedly for the configured number of iterations.

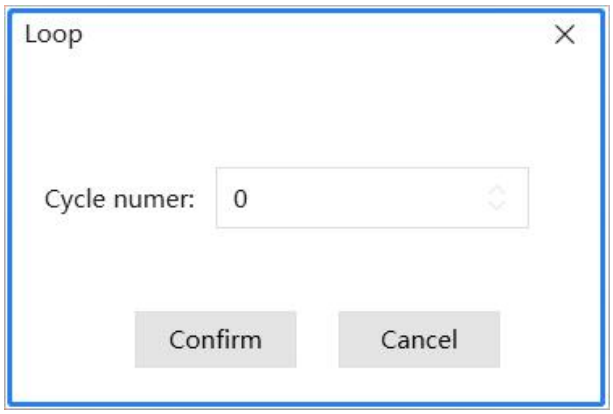


Figure34 Loop Step Editing

6.2.2.8 SwitchCase

SwitchCase is a conditional branching statement that contains an expression used to control branch execution. The step contains a default branch. Use the + and - buttons to add or remove branches. The Default branch cannot be deleted.

Local variables can be referenced using $ in the expression, or constants can be used directly to control branch execution.

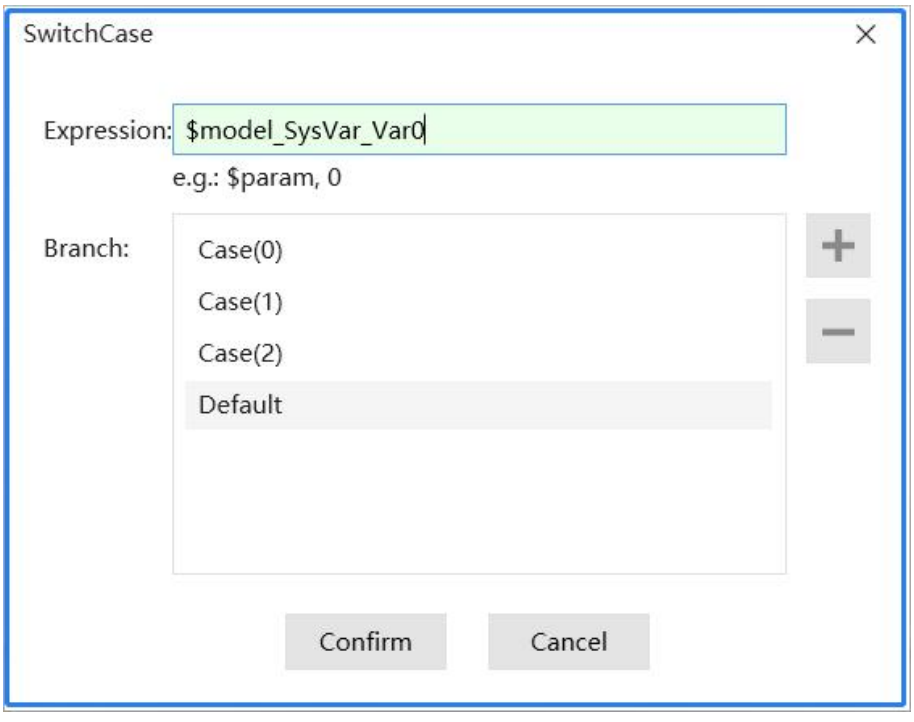


Figure35 SwitchCase Step Editing

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

6.2.2.9 Return

Return is a return statement that contains a Rating parameter used to control the return status. Available values are NONE, SUCCESS, FAILED, and ERROR.

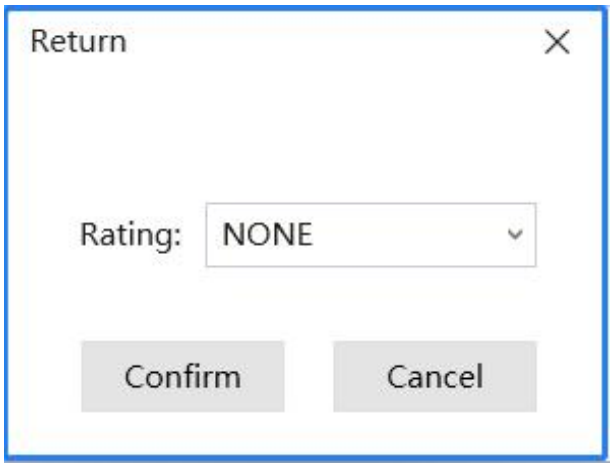


Figure36 Return Step Editing

6.3 Common Steps Description

6.3.1 MAPort

Start Simulation: Starts simulation on the XIL MAPort. Use the drop-down list to select a MAPort name available in the current TSMaster project.

Stop Simulation: Stops simulation on the XIL MAPort. Use the drop-down list to select a MAPort name available in the current TSMaster project.



Figure37 Start/Stop Simulation Step Editing

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

6.3.2 EES Pin

Electrical Error Simulation (EES) is a functional module in the test case editor used to simulate electrical faults in ECU wiring. An EES Pin (Electrical Error Simulation Pin) is a core concept of this function. It represents the logical pin of an error simulation unit in the test configuration and serves as the basic object for defining and operating electrical errors in a test case.

EES Pins can be used to inject various electrical faults into specific physical ECU pins during testing to verify the ECU's diagnostic logic and fault-handling capabilities.

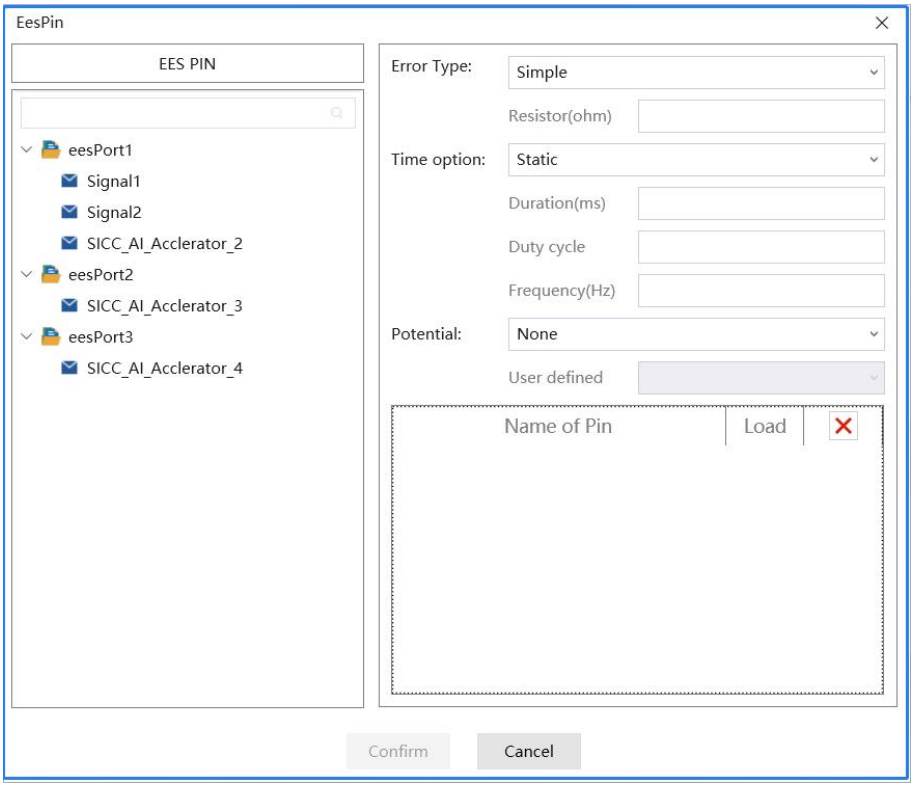


Figure38 EES Pin Step

EES Pin Step Configuration Parameters:

- Error Type: Supports Simple and Resistor. Resistor requires a resistance value to be configured.
- Time Option: Supports Static, Duration, and Loose contact. Loose contact requires the duty cycle and frequency to be configured.
- Potential Type: Supports None (open circuit), Ground (short to ground), Ubatt (short to the power supply), and User defined.

Select a signal by dragging a pin from the EESPort Signal Tree. Each step can operate on only one Port.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

6.4 Mini API Description

Mini API steps can be used to call interfaces provided by TSMaster. After an API method is dragged from the Test Library to the test case editing area, an editing dialog box opens automatically.

Configurable parameters include:

- Parameter list: Parameter name, type, and value input box.
- Save Value: Saves the API execution result to a local variable.
- Assessment
  - Measure (used to check whether the API return value meets the expected result)
  - Deviation (used to specify the tolerance range for the value check)

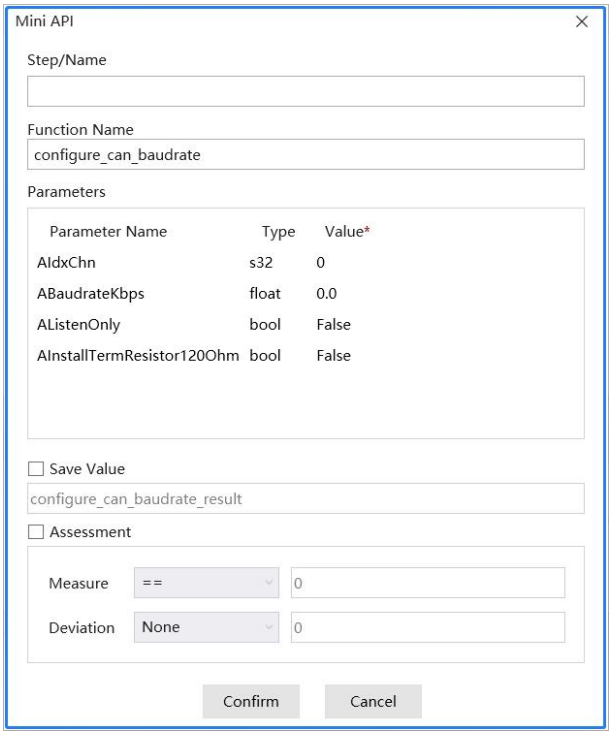


Figure39 Mini API Step

6.5 Expression Description

This section applies to condition expressions in flow control steps such as IfThenElse, IfDef, and While, as well as Measurement expressions in Read steps. The supported expression syntax and examples are described below.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

Note: Use the $ symbol to reference variables in expressions.

- Literal Constants:
  - Integer: `0x100`、`256`、`0b1010`
  - Floating-point number: `3.1415`、`-0.233`
  - String

```
'hello world'
"hello world"
```

  - Boolean: `True`、`False`
  - Null: `None`

- Arithmetic Operations: addition, subtraction, multiplication, division, floor division, modulo, exponentiation, and mixed operations
- Bitwise Operations: bitwise AND, bitwise OR, bitwise XOR, bitwise NOT, left shift, and right shift
- Logic Operations (Return Boolean Values): Logical AND, logical OR, and logical NOT
- Comparison Operations (Produce Boolean Values):
  - ==、!=、>、<、>=、<=
  - Identify comparison:

```
$var0 is None
$var1 is not True
```

  - Membership testing:

```
$state in (‘on’, ‘off’)
$value not in [0, 255, 512]
```

- Conditional Expressions (Ternary Operator):

```
$var0 if $model_SysVar_Var0 > 0 else 0
($signal + $offset) if $enabled else $default
$high_limit if $mode == 1 else $low_limit
```

- Sequence Operations (Indexing and Slicing): used when a variable is an array or another indexable object:
  - Index access
  - Slicing:

```
$data[1:3]  #the subsequence from index 1 to index 2
$data[:4]   #the first four elements
$data[::2]  #every other element
```

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

- Using Parentheses to Control Precedence: parentheses “() ” can be used to change the order of evaluation and can be nested to any depth.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

7 Troubleshooting

7.1 Git Access Denied

When the test case editor is installed to the default path `C:\Program Files (x86)\`, pulling a project from Git to a local folder may result in a “Git access denied” error.

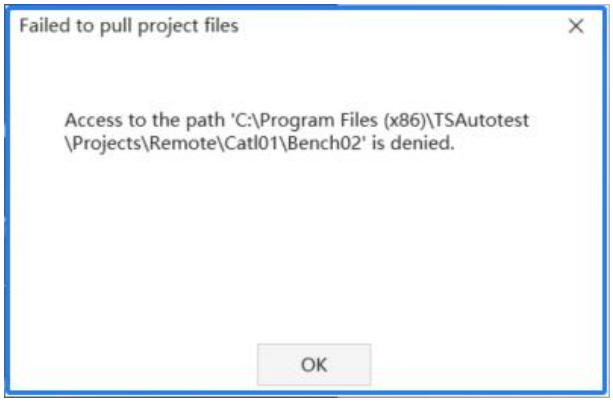


Figure40 Git Access Denied

Solutions:

1. Modify folder permissions (permanent solution).

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

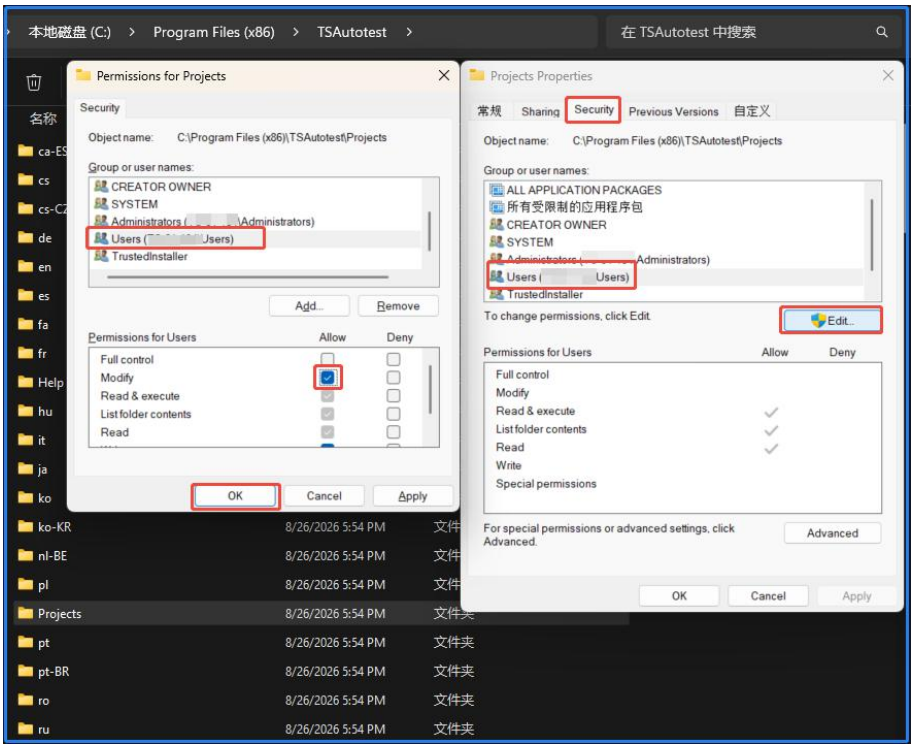


Figure41 Setting Directory Permissions

2. Install to a non-default path: install to another folder, such as `C:\Soft\TSAutotest`.

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

8 Contact Information

Shanghai TOSUN Technology Ltd.
No. 9 Building, 1288 Jiasong North Road, Jiading District, Shanghai (Headquarters)
Buildings 14-17, Lane 4849 Cao'an Highway (Shanghai Research Institute)
Website: www.tosunai.com
Tel: 021-59560506
Email: support@tosunai.cn

@ Copyright 2026-2027, Shanghai TOSUN Technology Ltd. All rights reserved.

TOSUN

Engineer Everything !

Software

Support CAN(FD)/LIN/FlexRay/SOME/IP and DoIP

UDS diagnostics/ECU flashing/CCP/XCP calibration

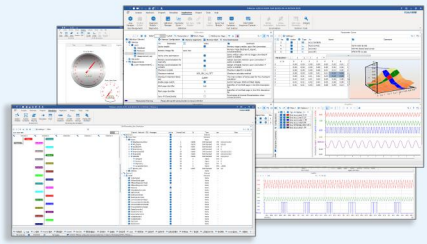
Embedded code generation/Application builder

Encrypted release/Logging and bus replay

Graphical programming/Residual bus simulation

C and Python scripting

Bus monitoring/Transmiting/Automated testing



TSMASTER

Hardware

1/2/4/8/12-channel CAN FD/CAN to USB/PCIe device

1/2/6-channel LIN to USB/PCIe device

Multi channel FlexRay/CAN FD to USB/PCIe device

Multi channel automotive Ethernet/CAN FD to USB/PCIe device

Automotive Ethernet media conversion device (T1 to Tx)

Multi-channel CAN FD/Ethernet/LIN datalogger



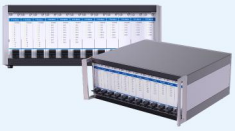
TTS test systems

-CAN FD/CAN/FlexRay/LIN communication boards

-Relay and fault injection boards

-Resistors for sensor simulation

-Digital I/O, Analog I/O boards available



CAN CAN FD LIN FlexRay OPEN ALLIANCE

Solutions

- Bus Conformance
- Network Automation Testing System
- Charging Testing System
- EMB Calibration Testing Equipment
- Information Security Solutions
- Steer-by-Wire Chassis Testing Solutions
- EOL Testing Equipment
- Motor Performance
- Durability Testing Solutions
- FCT







About TOSUN

The core product, TSMaster, is a comprehensive tool for automotive R&D, testing, production, and after-sales. It integrates essential functions with hardware support to streamline processes and ensure precision, making it ideal for automotive professionals.

International Organization

ASAM CiA

Quality Assurance

ISO9001:2015

CE Certification

CE



Contact Us :

+86 21-5956 0506

sales@tosunai.com

website :

www.tosunai.com

Beijing/Shanghai/Guangzhou/Chengdu/Changzhou/Taipei/Stuttgart/Seoul