Quick Start

--V202410131

Contents

1. TSMaster

1.1. Project Management

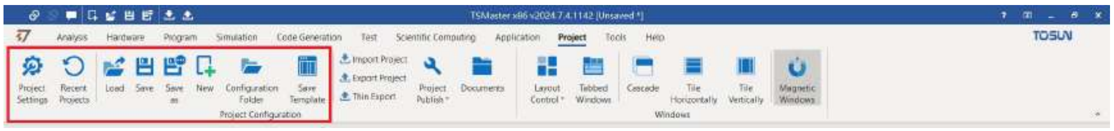
The project configuration file suffix for TSMaster is .T7z, which is a configuration file with compression and encryption capabilities. The T7z file contains all the contents of the current configuration: windows, information on the windows, databases, panels, and C scripts. The project configuration files for TSMaster have the following advantages:

- The project configuration file contains all the information, and the distribution file is very concise, requiring only a single T7z file. Currently, most project configuration files on the market store file links (such as links to databases, panels, etc.). Therefore, when distributing, there will be various supporting files in addition to the configuration file under the project configuration directory.

- The project file itself is encrypted.

1.1.1. Project File Management

Project files mainly involve the following operations: Load, Save, Save as, and New.



TSMaster's project management is in two forms: project directories and T7Z files. The differences are as follows:

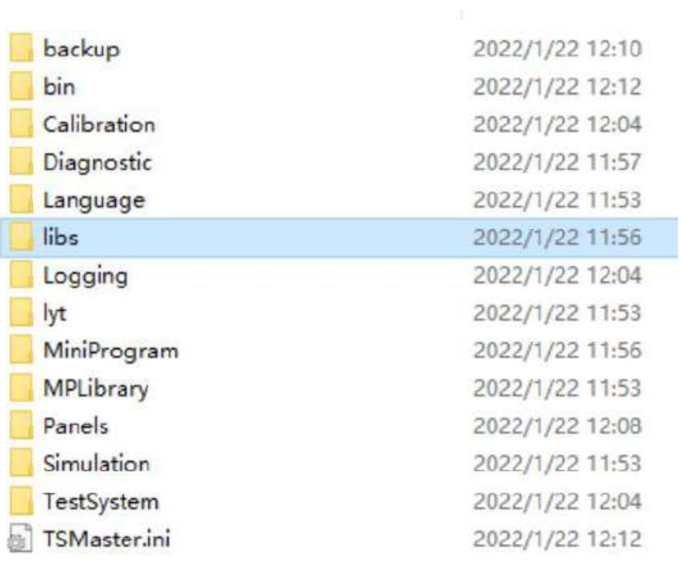
- All original configuration files related to the project are stored in a separate directory, making it convenient for management using tools such as Git, SVN, and others.

- T7Z File: The original configuration files related to the project are packaged and compressed into a single independent T7Z project. The advantages are: the entire project directory is compressed into a single file, including all DBC, A2L, Panel, and other configurations, making it convenient for project distribution and collaboration.

Both project directories and T7Z files are supported simultaneously. Users can use them according to their needs flexibly.

1.1.2. TSMaster Project Directory

Starting from version V2022.1.21.693, TSMaster introduced the project directory. Similar to conventional software, it stores all project-related files within a single project directory. A typical TSMaster project directory is shown in the following figure:

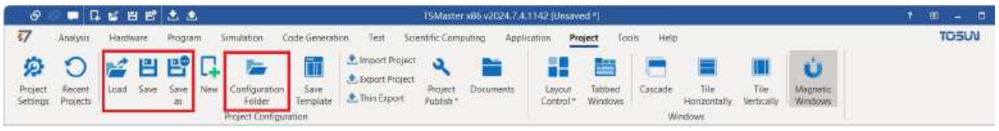


backup	2022/1/22 12:10
bin	2022/1/22 12:12
Calibration	2022/1/22 12:04
Diagnostic	2022/1/22 11:57
Language	2022/1/22 11:53
libs	2022/1/22 11:56
Logging	2022/1/22 12:04
lyt	2022/1/22 11:53
MiniProgram	2022/1/22 11:56
MPLibrary	2022/1/22 11:53
Panels	2022/1/22 12:08
Simulation	2022/1/22 11:53
TestSystem	2022/1/22 12:04
TSMaster.ini	2022/1/22 12:12

As shown in the figure above, the basic functional module configurations stored in each directory are as follows:

- Calibration: stores calibration-related configurations.
- Diagnostic: stores diagnostic-related configurations.
- Language: stores multilingual-related configurations. Unless special circumstances, this folder should not be modified.
- libs: The default directory for storing C script projects.
- Logging: The default directory for storing message data such as BLF files.
- MiniProgram: stores directories related to mini-programs.
- MPLibrary: stores directories related to mini-program libraries.
- Simulation: stores configurations related to RBS simulations.
- Panels: stores configurations related to panels.
- TestSystem: stores configurations related to panels.

1.1.2.1.Project Directory Operation Panel



Load: load application configuration.

Save: save application configuration.

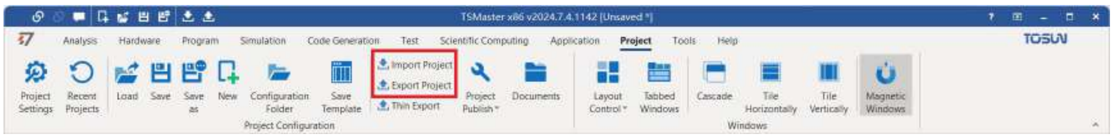
Save as: save application configuration as.

Configuration Folder: Open TSMaster configuration folder.

1.1.3. T7Z Project File

1.1.3.1.T7Z Operation Panel

T7Z Operation Panel is as shown below:

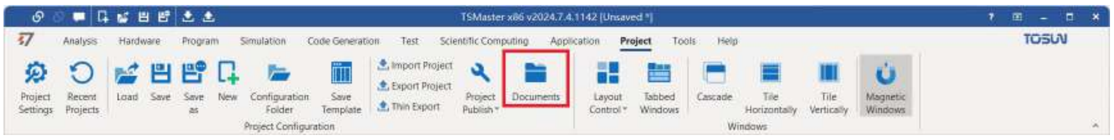


Import Project: import project from a T7z file.

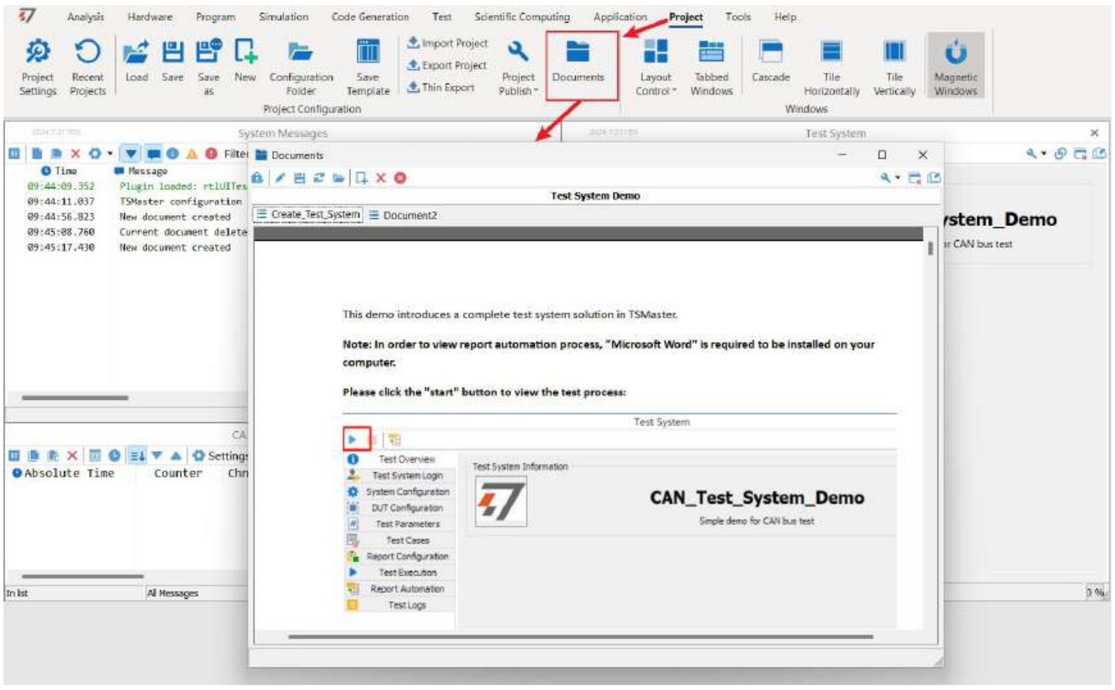
Export Project: export current project to a T7z file.

1.1.4. Add Documentation for the Project

After a user completes the project development, if the project is to be released for customer use, it is necessary to write a user manual specifically for that project. TSMaster has a built-in project documentation management module for this purpose.



Through this module, users can add usage documents specifically for the configured project with texts and images, and multiple documents can be added at the same time. When saving the project, the project documentation will be packaged together into the project file (t7z file). After the user loads the t7z file, if they want to access the usage instructions for the project, they simply need to open the documentation module from here.



```
Time          Message
09:44:09.352  Plugin loaded: rtlUITes
09:44:11.037  TSMaster configuration
09:44:56.823  New document created
09:45:08.760  Current document delete
09:45:17.430  New document created
```

Test System Demo

This demo introduces a complete test system solution in TSMaster.

Note: In order to view report automation process, "Microsoft Word" is required to be installed on your computer.

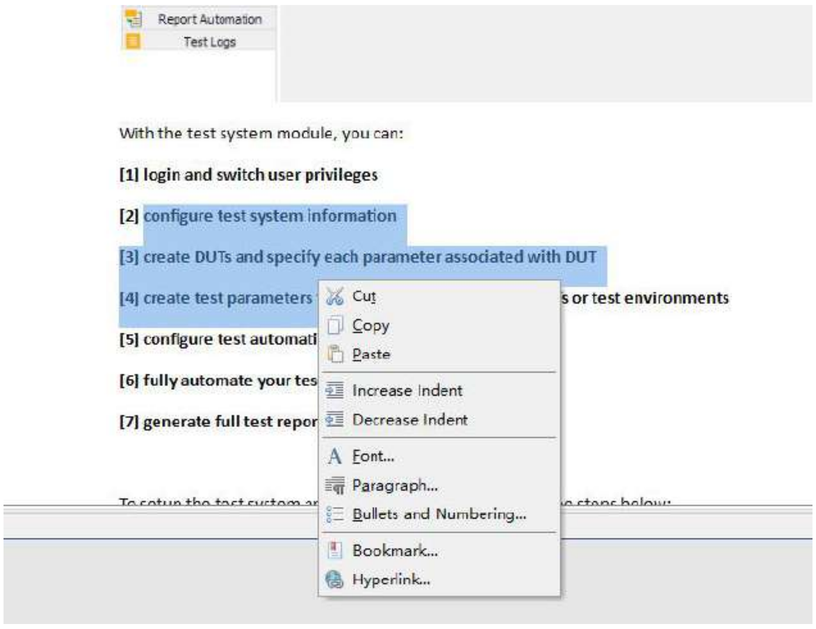
Please click the "start" button to view the test process:

CAN_Test_System_Demo

Simple demo for CAN bus test

In the documentation, if you want to modify the text display format or other aspects, there are two ways to do it:

1. Directly edit in the built-in editor of TSMaster: Select the corresponding text, right-click, and you will see the properties setting window. From there, you can enter the settings to modify the attributes.



With the test system module, you can:

[1] login and switch user privileges

[2] configure test system information

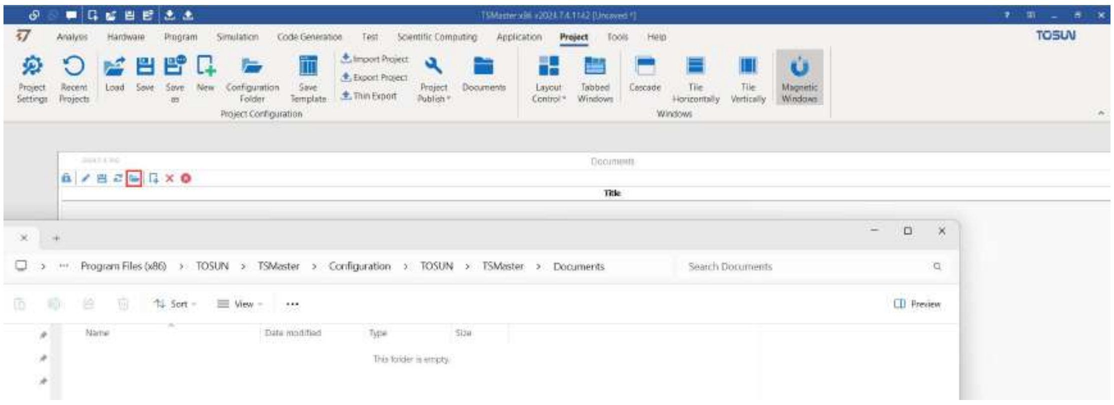
[3] create DUTs and specify each parameter associated with DUT

[4] create test parameters **s or test environments**

[5] configure test automati

[6] fully automate your tes

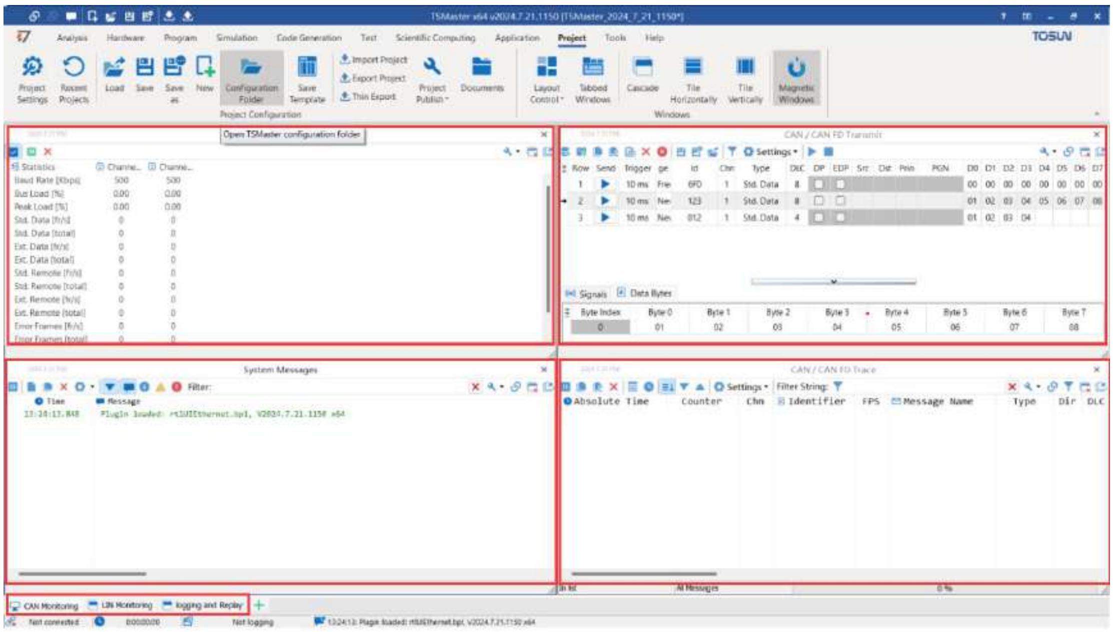
[7] generate full test repor



1.2. Add Management and Control Permissions

Most companies have an internal permission management system. After installing TSMaster, users need to contact the company's IT department to add the relevant TSMaster internal modules to the whitelist for normal operation. The main modules that need to be added to the whitelist include:

1.3. Workspace and Windows

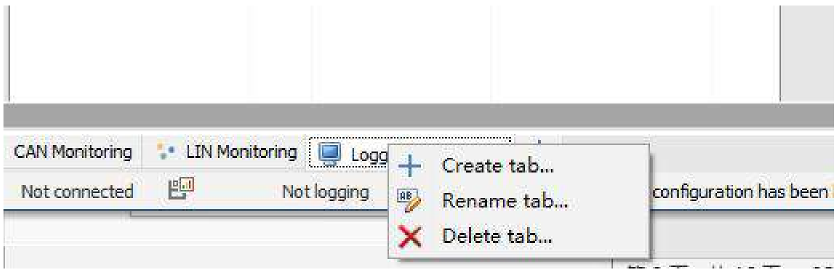


```
Time          Message
13:24:13.848  Plugin loaded: rtUSEthernet.bpl, V2024.7.21.1150 x64
```

One of the key design points of TSMaster is to provide users with maximum flexibility as much as possible, which is why it offers the workspace and windows mechanism.

1.3.1. Workspace

It can be understood as a window container, which facilitates users to place function-related windows into the same container. Upon installation, the software includes three default workspaces: CAN Monitoring, LIN Monitoring, and Logging and Replay. Workspaces support operations such as adding, deleting, and renaming. After configuring the workspaces, users can save them as templates for others to use.



Supports adding, deleting, editing, and other operations.

After creating a workspace, users can add windows to it.

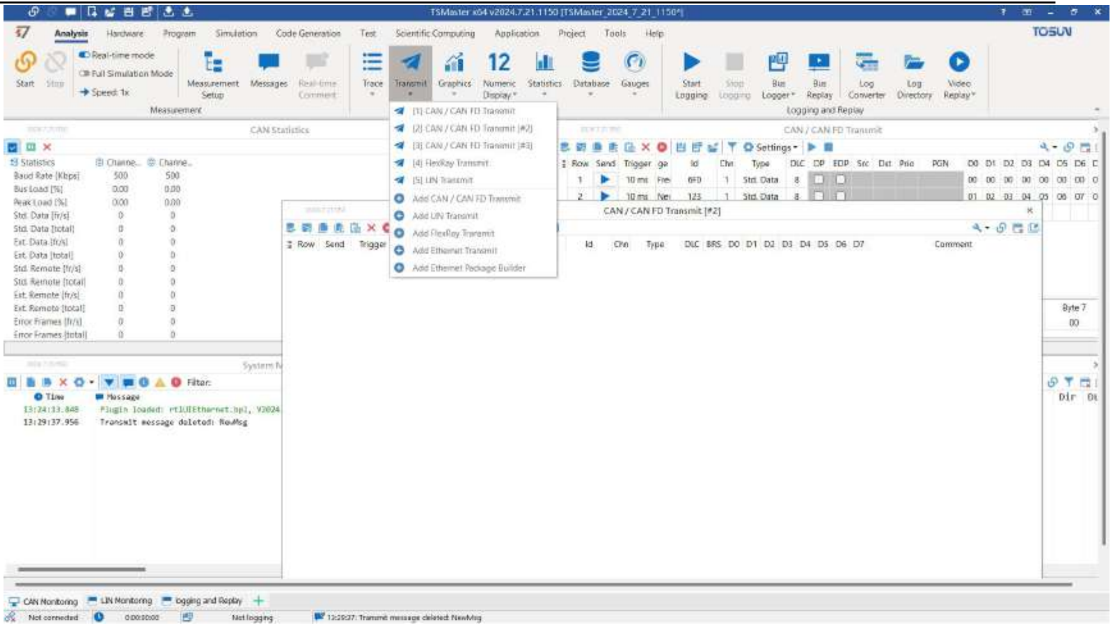
1.3.2. Window

Each window represents a functional module, such as CAN Trace, LIN Trace, CAN Transmit, LIN Transmit, CCode, and so on.

1.3.2.1. Add Window

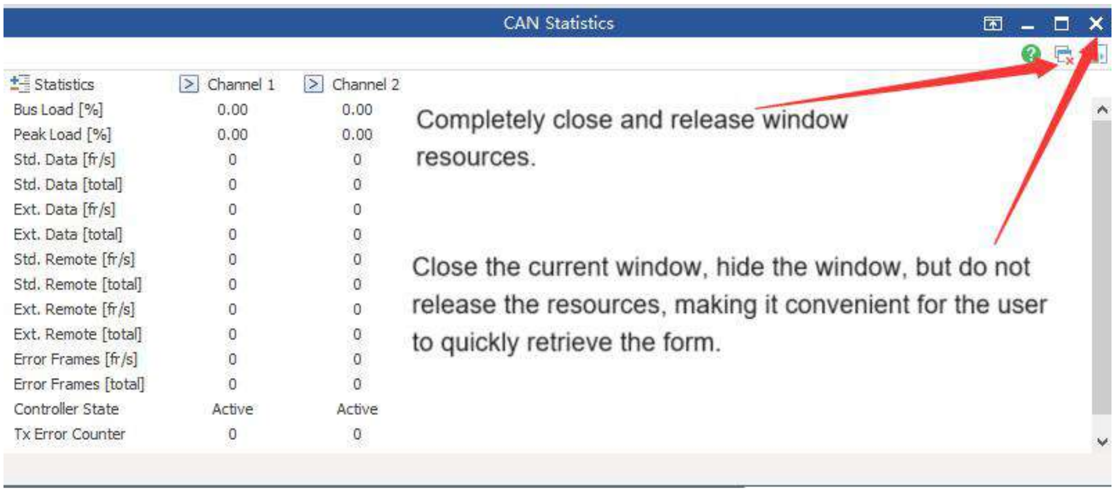
Users can create multiple windows of the same type simultaneously, and they exist independently from each other. Taking the CAN Trace window as an example, a user can add a CAN Trace window to the CAN Monitor workspace, and at the same time, they can also add a CAN Trace window to the Logging and Replay workspace. During the program's runtime, the two windows work in parallel and independently.

For an existed window, after clicking it, for example, the second option as shown in below figure, then the corresponding window pop up. For the options with + , such as “Add CAN/CAN FD Transmit”, it will create the new windows and place it in the current workspace.


Transmit
[1] CAN / CAN FD Transmit
[2] CAN / CAN FD Transmit [#2]
[3] CAN / CAN FD Transmit [#3]
[4] FlexRay Transmit
[5] LIN Transmit
Add CAN / CAN FD Transmit
Add LIN Transmit
Add FlexRay Transmit
Add Ethernet Transmit
Add Ethernet Package Builder

1.3.2.2.Close Window

TSMaster provides two levels of window closing. Clicking on the icon at the top right corner only hides the window but does not release resources, allowing users to quickly retrieve the window. To completely close the window, click the icon.



CAN Statistics

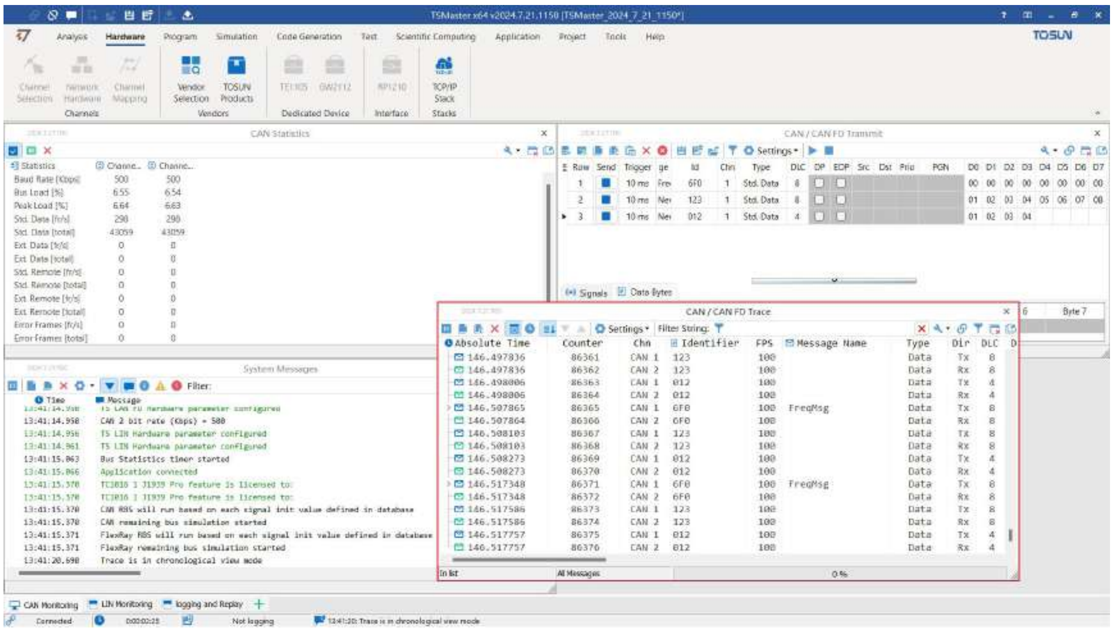
Statistics	Channel 1	Channel 2
Bus Load [%]	0.00	0.00
Peak Load [%]	0.00	0.00
Std. Data [fr/s]	0	0
Std. Data [total]	0	0
Ext. Data [fr/s]	0	0
Ext. Data [total]	0	0
Std. Remote [fr/s]	0	0
Std. Remote [total]	0	0
Ext. Remote [fr/s]	0	0
Ext. Remote [total]	0	0
Error Frames [fr/s]	0	0
Error Frames [total]	0	0
Controller State	Active	Active
Tx Error Counter	0	0

Completely close and release window resources.

Close the current window, hide the window, but do not release the resources, making it convenient for the user to quickly retrieve the form.

1.3.2.3.Window Pop-up/Pop-in

TSMaster employs the MDI window mode, where each sub window operates within the workspace provided by the overall parent window. If users wish to focus on monitoring specific windows, for instance, to view detailed content in the Trace window, the form of sub windows may limit the scope of observation, as demonstrated below:



CAN Statistics

Statistics	Channe...	Channe...
Baud Rate [Kbps]	500	500
Bus Load [%]	6.55	6.54
Peak Load [%]	6.64	6.63
Std. Data [fr/s]	290	290
Std. Data [total]	43059	43059
Ext. Data [fr/s]	0	0
Ext. Data [total]	0	0
Std. Remote [fr/s]	0	0
Std. Remote [total]	0	0
Ext. Remote [fr/s]	0	0
Ext. Remote [total]	0	0
Error Frames [fr/s]	0	0
Error Frames [total]	0	0

CAN / CAN FD Transmit

Row	Send	Trigger	ge	Id	Chn	Type	DLC	D0	D1	D2	D3	D4	D5	D6	D7
1	■	10 ms	Fre	6F0	1	Std. Data	8	00	00	00	00	00	00	00	00
2	■	10 ms	Nev	123	1	Std. Data	8	01	02	03	04	05	06	07	08
3	■	10 ms	Nev	012	1	Std. Data	4	01	02	03	04				

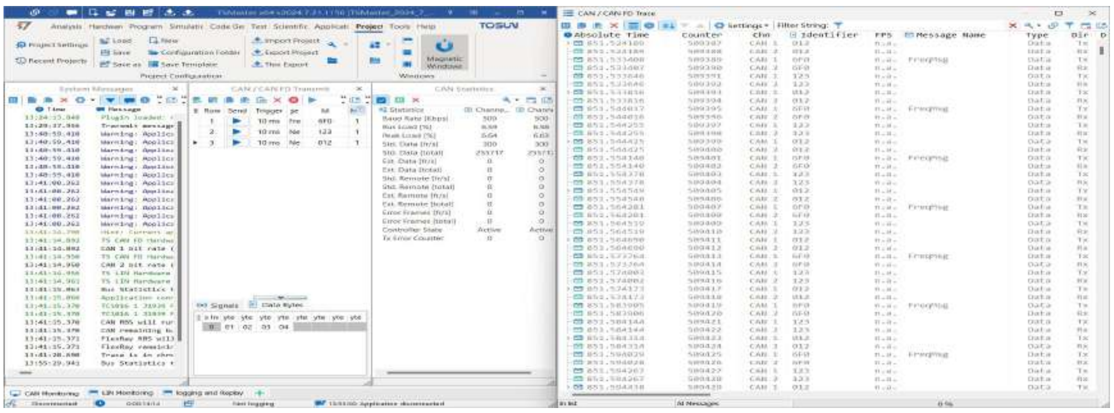
System Messages

```
Time           Message
13:41:14.958   TS CAN FD Hardware parameter configured
13:41:14.958   CAN 2 bit rate (Kbps) = 500
13:41:14.956   TS LIN Hardware parameter configured
13:41:14.961   TS LIN Hardware parameter configured
13:41:15.063   Bus Statistics timer started
13:41:15.066   Application connected
13:41:15.370   TC1016 1 J1939 Pro feature is licensed to:
13:41:15.370   TC1016 1 J1939 Pro feature is licensed to:
13:41:15.370   CAN RBS will run based on each signal init value defined in database
13:41:15.370   CAN remaining bus simulation started
13:41:15.371   FlexRay RBS will run based on each signal init value defined in database
13:41:15.371   FlexRay remaining bus simulation started
13:41:20.698   Trace is in chronological view mode
```

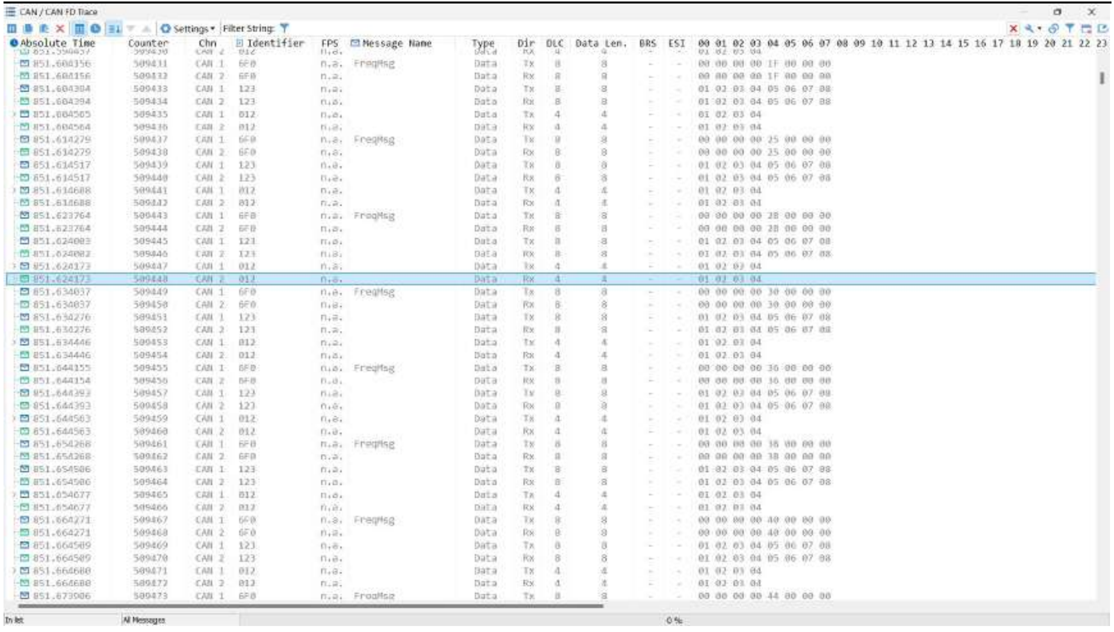
CAN / CAN FD Trace

```
Absolute Time   Counter   Chn     Identifier   FPS   Message Name   Type   Dir   DLC
146.497836      86361     CAN 1   123          100                  Data   Tx    8
146.497836      86362     CAN 2   123          100                  Data   Rx    8
146.498006      86363     CAN 1   012          100                  Data   Tx    4
146.498006      86364     CAN 2   012          100                  Data   Rx    4
146.507865      86365     CAN 1   6F0          100   FreqMsg        Data   Tx    8
146.507864      86366     CAN 2   6F0          100                  Data   Rx    8
146.508103      86367     CAN 1   123          100                  Data   Tx    8
146.508103      86368     CAN 2   123          100                  Data   Rx    8
146.508273      86369     CAN 1   012          100                  Data   Tx    4
146.508273      86370     CAN 2   012          100                  Data   Rx    4
146.517348      86371     CAN 1   6F0          100   FreqMsg        Data   Tx    8
146.517348      86372     CAN 2   6F0          100                  Data   Rx    8
146.517586      86373     CAN 1   123          100                  Data   Tx    8
146.517586      86374     CAN 2   123          100                  Data   Rx    8
146.517757      86375     CAN 1   012          100                  Data   Tx    4
146.517757      86376     CAN 2   012          100                  Data   Rx    4
In list   All Messages   0 %
```

Therefore, TSMaster also provides a window pop-out mechanism. When the user clicks the button in the top-right corner of the sub window, the sub window will pass the constraints of the parent window and become a window running in parallel with the parent window, as shown in the image below:



In this mode, the sub window becomes a completely independent window, no longer constrained by the parent window, and can extent it to the entire screen.



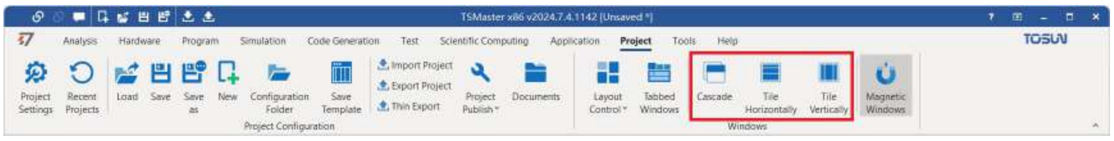
Absolute Time	Counter	Chn	Identifier	FPS	Message Name	Type	Dir	DLC	Data Len.	BRS	ESI	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
851.604156	509431	CAN 1	6F0	n.a.	FreqMsg	Data	Tx	8	8	-	-	00	00	00	00	1F	00	00	00																
851.604156	509432	CAN 2	6F0	n.a.		Data	Rx	8	8	-	-	00	00	00	00	1F	00	00	00																
851.604384	509433	CAN 1	123	n.a.		Data	Tx	8	8	-	-	01	02	03	04	05	06	07	08																
851.604394	509434	CAN 2	123	n.a.		Data	Rx	8	8	-	-	01	02	03	04	05	06	07	08																
851.604565	509435	CAN 1	012	n.a.		Data	Tx	4	4	-	-	01	02	03	04																				
851.604564	509436	CAN 2	012	n.a.		Data	Rx	4	4	-	-	01	02	03	04																				
851.614279	509437	CAN 1	6F0	n.a.	FreqMsg	Data	Tx	8	8	-	-	00	00	00	00	25	00	00	00																
851.614279	509438	CAN 2	6F0	n.a.		Data	Rx	8	8	-	-	00	00	00	00	25	00	00	00																
851.614517	509439	CAN 1	123	n.a.		Data	Tx	8	8	-	-	01	02	03	04	05	06	07	08																
851.614517	509440	CAN 2	123	n.a.		Data	Rx	8	8	-	-	01	02	03	04	05	06	07	08																
851.614688	509441	CAN 1	012	n.a.		Data	Tx	4	4	-	-	01	02	03	04																				
851.614688	509442	CAN 2	012	n.a.		Data	Rx	4	4	-	-	01	02	03	04																				
851.623764	509443	CAN 1	6F0	n.a.	FreqMsg	Data	Tx	8	8	-	-	00	00	00	00	28	00	00	00																
851.623764	509444	CAN 2	6F0	n.a.		Data	Rx	8	8	-	-	00	00	00	00	28	00	00	00																
851.624083	509445	CAN 1	123	n.a.		Data	Tx	8	8	-	-	01	02	03	04	05	06	07	08																
851.624082	509446	CAN 2	123	n.a.		Data	Rx	8	8	-	-	01	02	03	04	05	06	07	08																
851.624173	509447	CAN 1	012	n.a.		Data	Tx	4	4	-	-	01	02	03	04																				
851.624173	509448	CAN 2	012	n.a.		Data	Rx	4	4	-	-	01	02	03	04																				
851.634037	509449	CAN 1	6F0	n.a.	FreqMsg	Data	Tx	8	8	-	-	00	00	00	00	30	00	00	00																
851.634037	509450	CAN 2	6F0	n.a.		Data	Rx	8	8	-	-	00	00	00	00	30	00	00	00																
851.634276	509451	CAN 1	123	n.a.		Data	Tx	8	8	-	-	01	02	03	04	05	06	07	08																
851.634276	509452	CAN 2	123	n.a.		Data	Rx	8	8	-	-	01	02	03	04	05	06	07	08																
851.634446	509453	CAN 1	012	n.a.		Data	Tx	4	4	-	-	01	02	03	04																				
851.634446	509454	CAN 2	012	n.a.		Data	Rx	4	4	-	-	01	02	03	04																				
851.644155	509455	CAN 1	6F0	n.a.	FreqMsg	Data	Tx	8	8	-	-	00	00	00	00	36	00	00	00																
851.644154	509456	CAN 2	6F0	n.a.		Data	Rx	8	8	-	-	00	00	00	00	36	00	00	00																
851.644393	509457	CAN 1	123	n.a.		Data	Tx	8	8	-	-	01	02	03	04	05	06	07	08																
851.644393	509458	CAN 2	123	n.a.		Data	Rx	8	8	-	-	01	02	03	04	05	06	07	08																
851.644563	509459	CAN 1	012	n.a.		Data	Tx	4	4	-	-	01	02	03	04																				
851.644563	509460	CAN 2	012	n.a.		Data	Rx	4	4	-	-	01	02	03	04																				
851.654268	509461	CAN 1	6F0	n.a.	FreqMsg	Data	Tx	8	8	-	-	00	00	00	00	38	00	00	00																
851.654268	509462	CAN 2	6F0	n.a.		Data	Rx	8	8	-	-	00	00	00	00	38	00	00	00																
851.654506	509463	CAN 1	123	n.a.		Data	Tx	8	8	-	-	01	02	03	04	05	06	07	08																
851.654506	509464	CAN 2	123	n.a.		Data	Rx	8	8	-	-	01	02	03	04	05	06	07	08																
851.654677	509465	CAN 1	012	n.a.		Data	Tx	4	4	-	-	01	02	03	04																				
851.654677	509466	CAN 2	012	n.a.		Data	Rx	4	4	-	-	01	02	03	04																				
851.664271	509467	CAN 1	6F0	n.a.	FreqMsg	Data	Tx	8	8	-	-	00	00	00	00	40	00	00	00																
851.664271	509468	CAN 2	6F0	n.a.		Data	Rx	8	8	-	-	00	00	00	00	40	00	00	00																
851.664509	509469	CAN 1	123	n.a.		Data	Tx	8	8	-	-	01	02	03	04	05	06	07	08																
851.664509	509470	CAN 2	123	n.a.		Data	Rx	8	8	-	-	01	02	03	04	05	06	07	08																
851.664680	509471	CAN 1	012	n.a.		Data	Tx	4	4	-	-	01	02	03	04																				
851.664680	509472	CAN 2	012	n.a.		Data	Rx	4	4	-	-	01	02	03	04																				
851.673906	509473	CAN 1	6F0	n.a.	FreqMsg	Data	Tx	8	8	-	-	00	00	00	00	44	00	00	00																

In list | All Messages | 0 %

To return to the MDI mode, similarly click on the icon of the window again, and the window will pop back into its previous workspace.

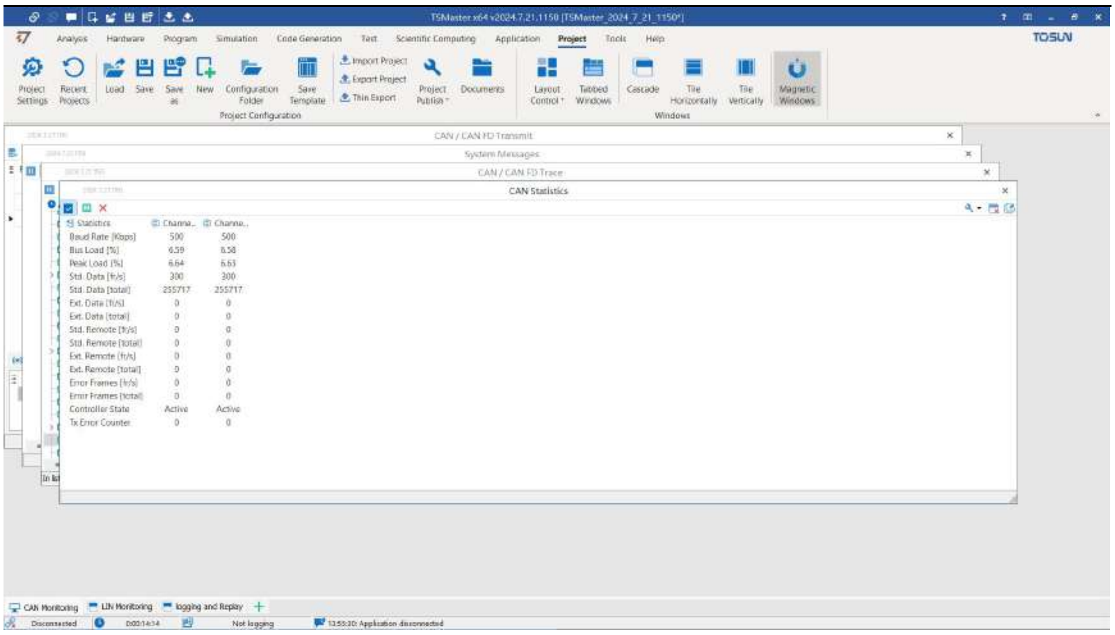
1.3.2.4. Window Alignment

In a workspace, it's common to have multiple sub windows working simultaneously. After dragging and repositioning the windows multiple times, their order can become quite messy. TSMaster provides shortcut keys for window alignment, which can be accessed via the following operation path: Project -> Window.



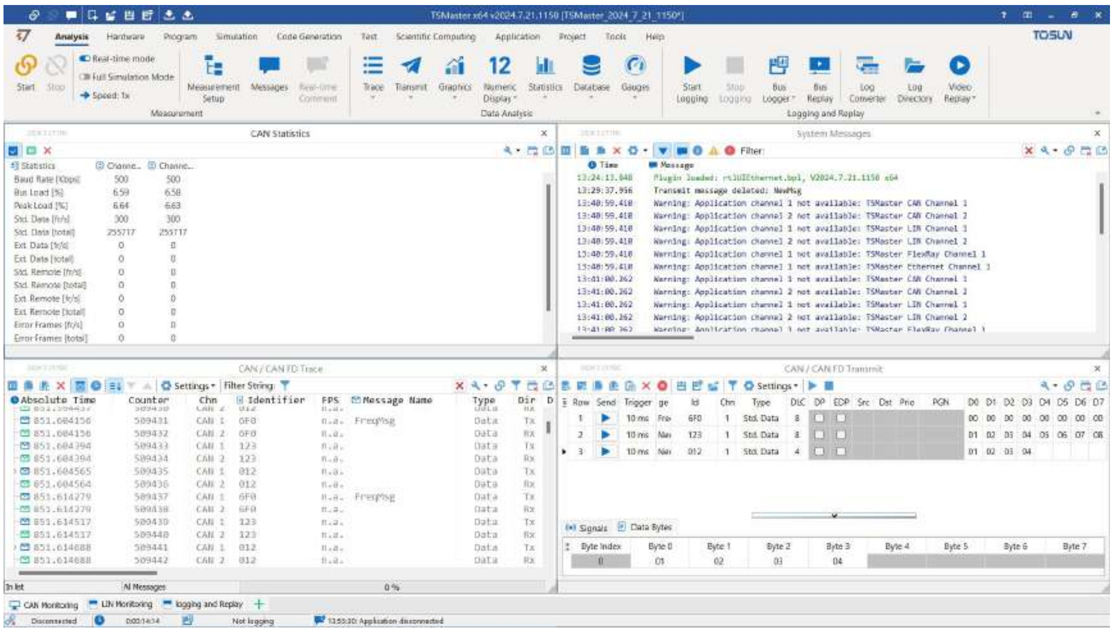
The effects of the three alignment methods are as follows:

- ➢ Cascade: cascade arrangement.



Statistics	Channe...	Channe...
Baud Rate [Kbps]	500	500
Bus Load [%]	6.59	6.58
Peak Load [%]	6.64	6.63
Std. Data [fr/s]	300	300
Std. Data [total]	255717	255717
Ext. Data [fr/s]	0	0
Ext. Data [total]	0	0
Std. Remote [fr/s]	0	0
Std. Remote [total]	0	0
Ext. Remote [fr/s]	0	0
Ext. Remote [total]	0	0
Error Frames [fr/s]	0	0
Error Frames [total]	0	0
Controller State	Active	Active
Tx Error Counter	0	0

➢ Tile Horizontally: horizontal arrangement.



```
Time          Message
13:24:13.040  Plugin loaded: rtlUIEthernet.bpl, V2024.7.21.1150 x64
13:29:37.956  Transmit message deleted: NewMsg
13:40:59.418  Warning: Application channel 1 not available: TSMaster CAN Channel 1
13:40:59.418  Warning: Application channel 2 not available: TSMaster CAN Channel 2
13:40:59.418  Warning: Application channel 1 not available: TSMaster LIN Channel 1
13:40:59.418  Warning: Application channel 2 not available: TSMaster LIN Channel 2
13:40:59.418  Warning: Application channel 1 not available: TSMaster FlexRay Channel 1
13:40:59.418  Warning: Application channel 1 not available: TSMaster Ethernet Channel 1
13:41:00.262  Warning: Application channel 1 not available: TSMaster CAN Channel 1
13:41:00.262  Warning: Application channel 2 not available: TSMaster CAN Channel 2
13:41:00.262  Warning: Application channel 1 not available: TSMaster LIN Channel 1
13:41:00.262  Warning: Application channel 2 not available: TSMaster LIN Channel 2
13:41:00.262  Warning: Application channel 1 not available: TSMaster FlexRay Channel 1
```

```
Absolute Time   Counter  Chn    Identifier  FPS   Message Name  Type  Dir
851.604156      509431   CAN 1  6F0         n.a.  FreqMsg       Data  Tx
851.604156      509432   CAN 2  6F0         n.a.                Data  Rx
851.604394      509433   CAN 1  123         n.a.                Data  Tx
851.604394      509434   CAN 2  123         n.a.                Data  Rx
851.604565      509435   CAN 1  012         n.a.                Data  Tx
851.604564      509436   CAN 2  012         n.a.                Data  Rx
851.614279      509437   CAN 1  6F0         n.a.  FreqMsg       Data  Tx
851.614279      509438   CAN 2  6F0         n.a.                Data  Rx
851.614517      509439   CAN 1  123         n.a.                Data  Tx
851.614517      509440   CAN 2  123         n.a.                Data  Rx
851.614688      509441   CAN 1  012         n.a.                Data  Tx
851.614688      509442   CAN 2  012         n.a.                Data  Rx
```

Row	Send	Trigger	ge	Id	Chn	Type	DLC	DP	EDP	Src	Dst	Prio	PGN	D0	D1	D2	D3	D4	D5	D6	D7
1	▶	10 ms	Fre	6F0	1	Std. Data	8	☐	☐					00	00	00	00	00	00	00	00
2	▶	10 ms	Nai	123	1	Std. Data	8	☐	☐					01	02	03	04	05	06	07	08
3	▶	10 ms	Nai	012	1	Std. Data	4	☐	☐					01	02	03	04				

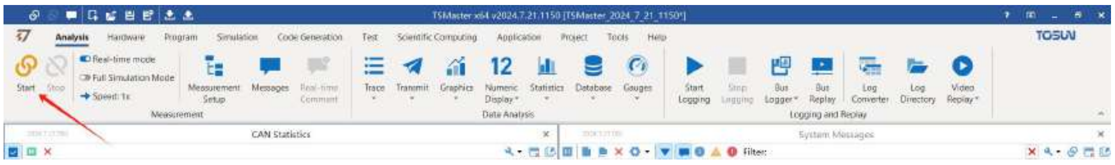
Byte Index	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0	01	02	03	04				

Disconnected 0:00:14:34 Not logging 13:53:30: Application disconnected

➢ Tile Vertically: vertically arrangement.

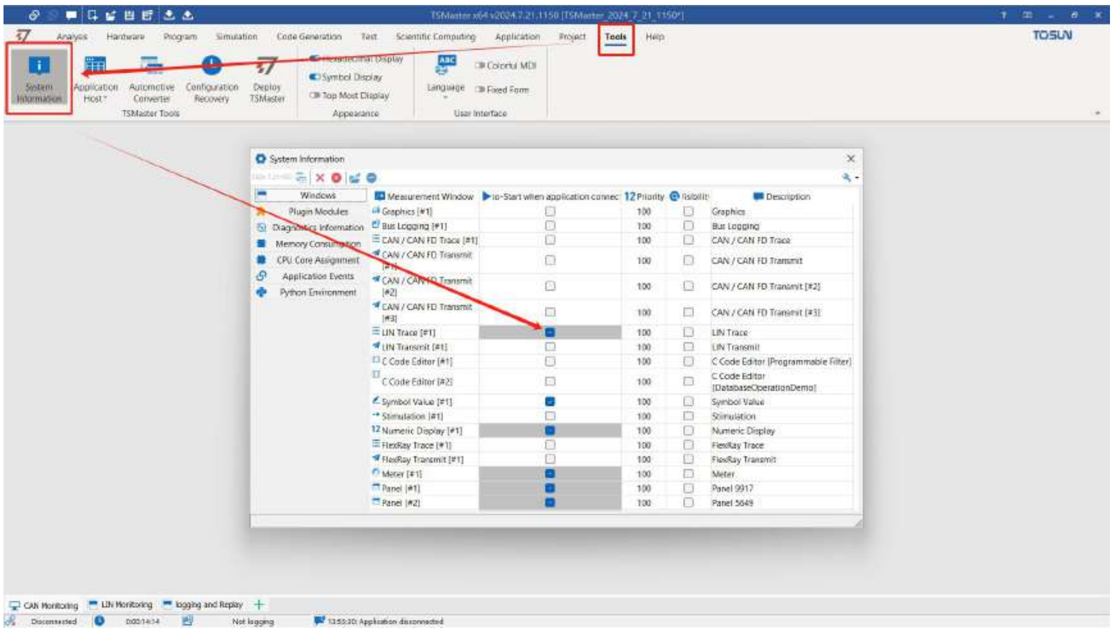
1.3.3. Start Application（Start）

Click the "Start" button to launch the TSMaster environment, as shown below:



During the startup process, it's important to note that in some scenarios, certain functional modules need to start simultaneously upon initiation, while others do not have this requirement. Therefore, TSMaster provides a "Start" function and a module for associating/disassociating functional modules. These settings can primarily be configured in two locations:

1.3.3.1.Associate in System Information



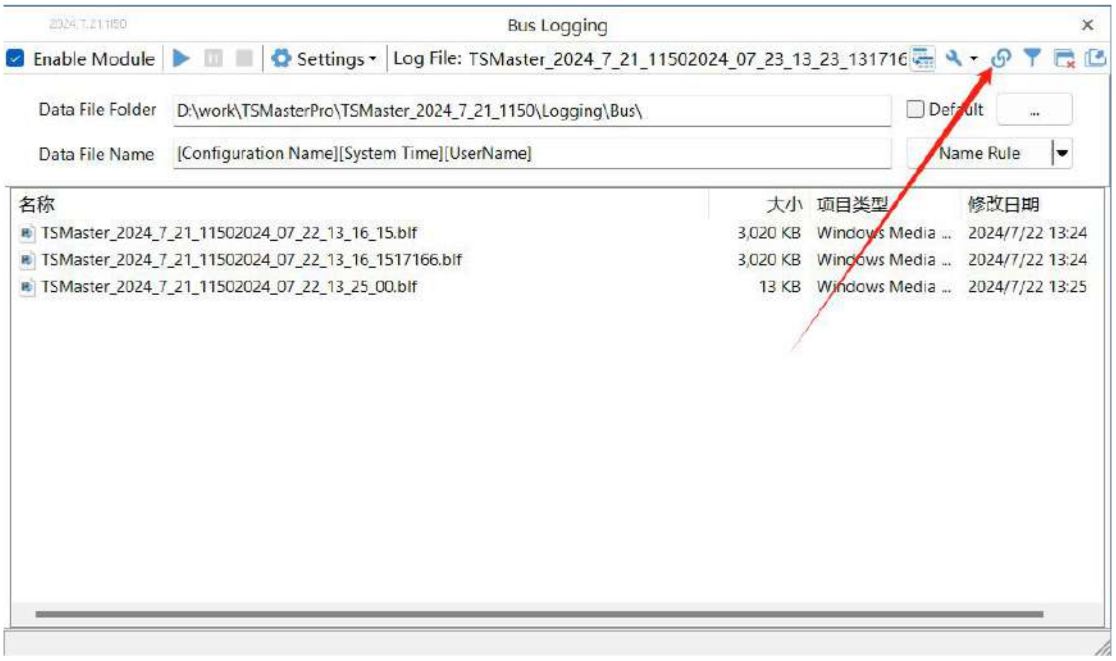
Measurement Window	Auto-Start when application connec	Priority	Visibility	Description
Graphics [#1]	☐	100	☐	Graphics
Bus Logging [#1]	☐	100	☐	Bus Logging
CAN / CAN FD Trace [#1]	☐	100	☐	CAN / CAN FD Trace
CAN / CAN FD Transmit [#1]	☐	100	☐	CAN / CAN FD Transmit
CAN / CAN FD Transmit [#2]	☐	100	☐	CAN / CAN FD Transmit [#2]
CAN / CAN FD Transmit [#3]	☐	100	☐	CAN / CAN FD Transmit [#3]
LIN Trace [#1]	☑	100	☐	LIN Trace
LIN Transmit [#1]	☐	100	☐	LIN Transmit
C Code Editor [#1]	☐	100	☐	C Code Editor [Programmable Filter]
C Code Editor [#2]	☐	100	☐	C Code Editor [DatabaseOperationDemo]
Symbol Value [#1]	☑	100	☐	Symbol Value
Stimulation [#1]	☐	100	☐	Stimulation
Numeric Display [#1]	☑	100	☐	Numeric Display
FlexRay Trace [#1]	☐	100	☐	FlexRay Trace
FlexRay Transmit [#1]	☐	100	☐	FlexRay Transmit
Meter [#1]	☑	100	☐	Meter
Panel [#1]	☑	100	☐	Panel 9917
Panel [#2]	☑	100	☐	Panel 5649

Access through the operation path: Tools -> System Information. On this panel, check the relevant functional modules.

☑ : this indicates that the checked modules are defaulted to start up automatically during initiation and cannot be configured otherwise. Checking other nodes will cause the corresponding modules to start working immediately upon initiation; if unchecked, those modules will remain dormant during initiation.

1.3.3.2.Directly Associate on Functional Modules

In the upper right corner of each functional module, there is a button to select whether to start automatically. The configuration principle is as follows:



Log File: TSMaster_2024_7_21_11502024_07_23_13_23_131716

Data File Folder D:\work\TSMasterPro\TSMaster_2024_7_21_1150\Logging\Bus\

Data File Name [Configuration Name][System Time][UserName]

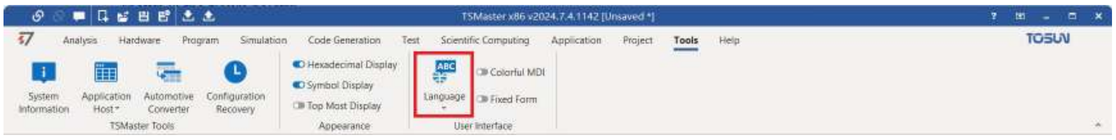
名称	大小	项目类型	修改日期
TSMaster_2024_7_21_11502024_07_22_13_16_15.blf	3,020 KB	Windows Media ...	2024/7/22 13:24
TSMaster_2024_7_21_11502024_07_22_13_16_1517166.blf	3,020 KB	Windows Media ...	2024/7/22 13:24
TSMaster_2024_7_21_11502024_07_22_13_25_00.blf	13 KB	Windows Media ...	2024/7/22 13:25

The icon in the upper right corner indicates whether it will start automatically. When pressed, it will start automatically; otherwise it will not start automatically.

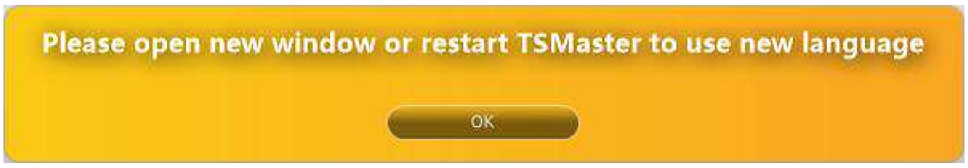
The operations mentioned above are essentially the same as the settings in the System Information.

1.3.4. Multi-languages Support

TSMaster supports multiple languages. It can be switched through the window buttons as shown in the figure below:



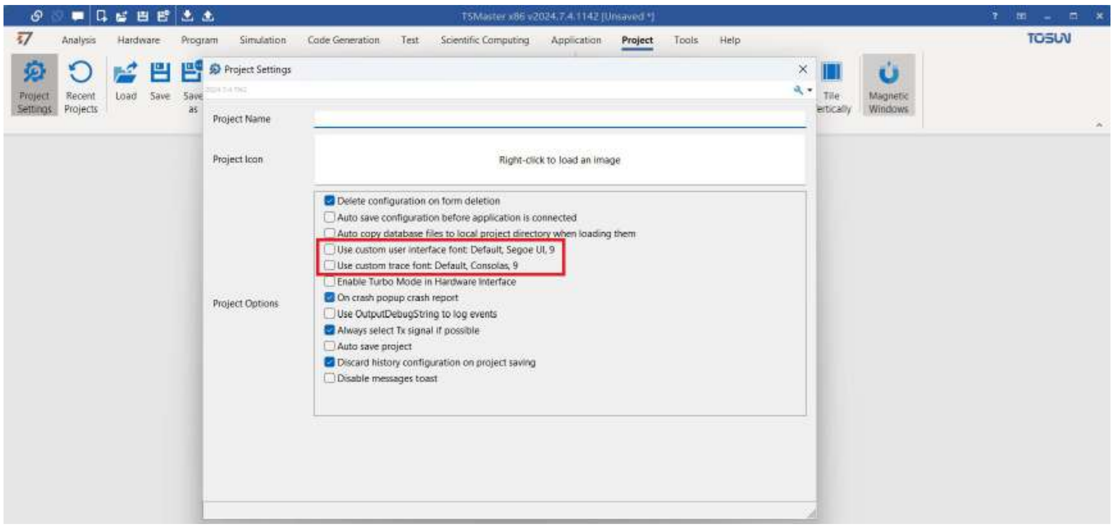
After clicking the "Switch Language" button, the system will display the following prompt:


Please open new window or restart TSMaster to use new language

After the user restarts the software, the software will fully switch to the new language environment. At the same time, TSMaster will remember this option and default to this language the next time it starts.

1.3.5. Change Window Font

If the user's screen is too large or too small, and the default font of the software cannot meet the user's needs, TSMaster allows users to adjust the font size according to their requirements. This mainly includes the general font of the software and the font of the message Trace interface. The adjustment path is shown in the figure below: Project -> Project Settings -> Project Options -> Use custom user interface font / Use custom trace font.



Project Settings

Project Name

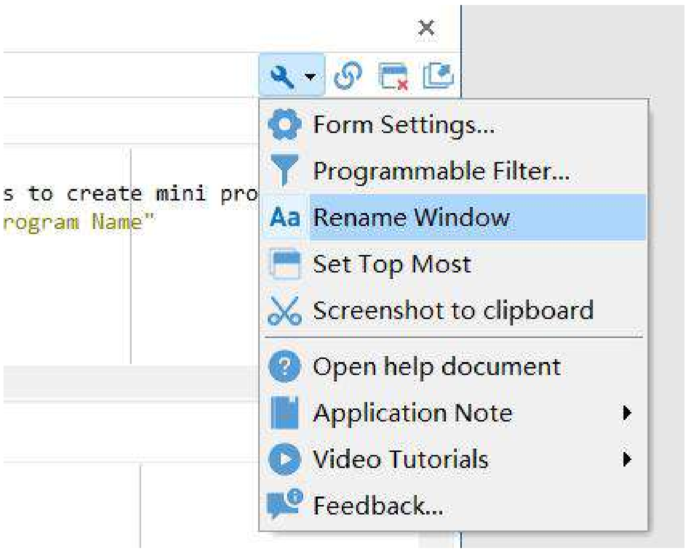
Project Icon: Right-click to load an image

Project Options:
- [✓] Delete configuration on form deletion
- [ ] Auto save configuration before application is connected
- [ ] Auto copy database files to local project directory when loading them
- [ ] Use custom user interface font: Default, Segoe UI, 9
- [ ] Use custom trace font: Default, Consolas, 9
- [ ] Enable Turbo Mode in Hardware Interface
- [✓] On crash popup crash report
- [ ] Use OutputDebugString to log events
- [✓] Always select Tx signal if possible
- [ ] Auto save project
- [✓] Discard history configuration on project saving
- [ ] Disable messages toast

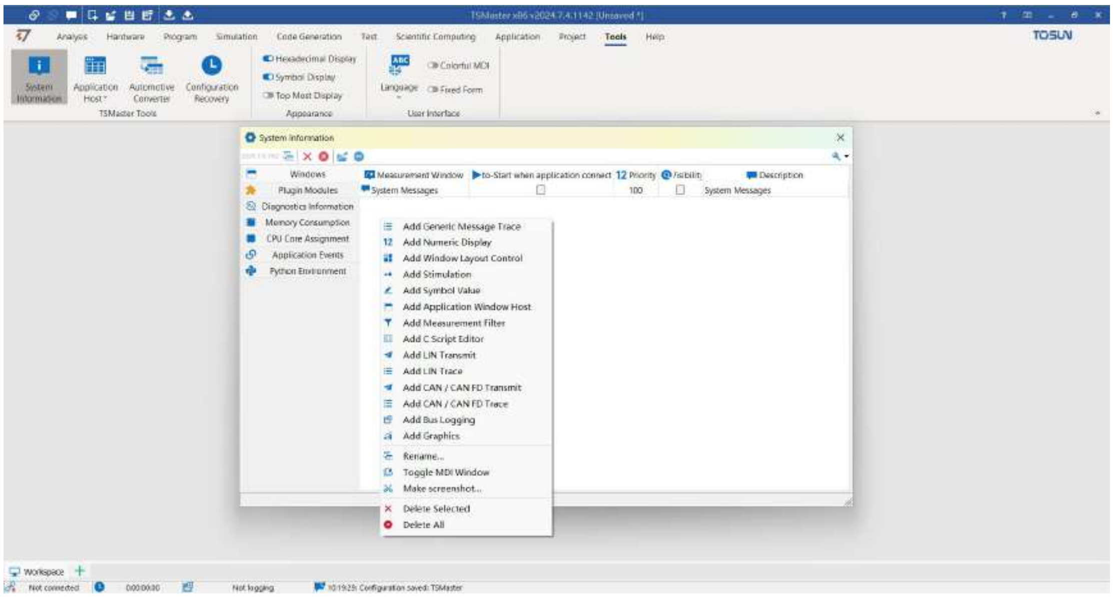
1.3.6. Rename Window

TSMaster supports renaming windows, and there are mainly the following methods:

- Method 1: In the upper right corner of the window, click the configuration button, and select "Rename Window" from the expanded drop-down menu, as shown below:

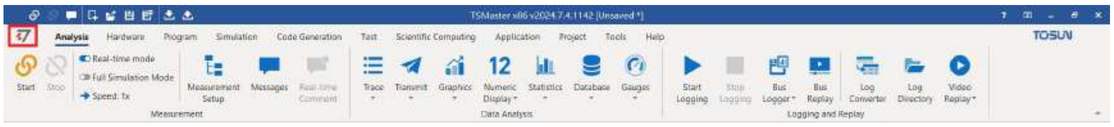


- Method 2: Through the path: Tools -> System Information -> Windows -> Select the window, right-click -> Rename.

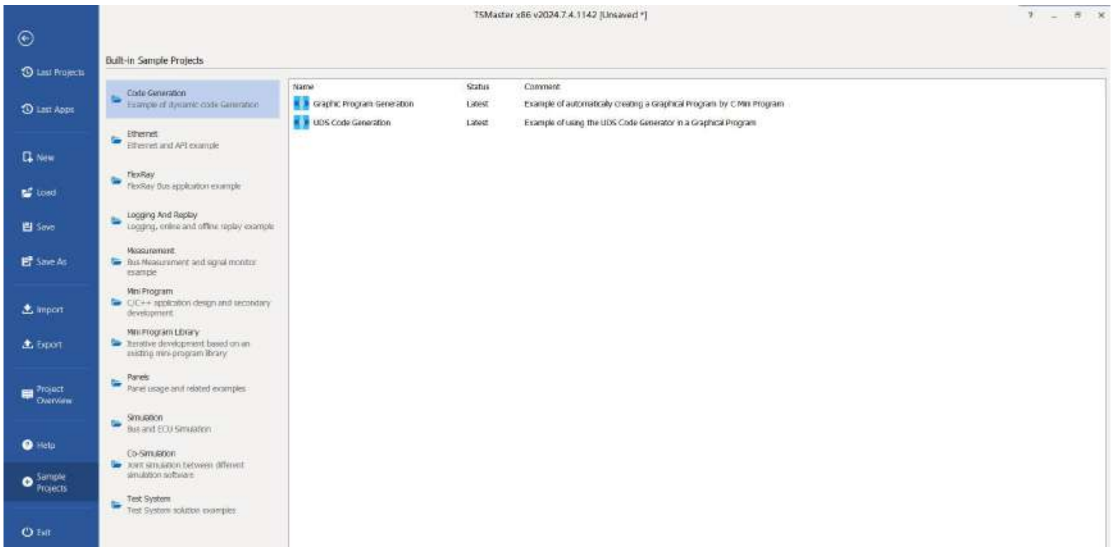


1.3.7. Open Example Project

- First, click the icon:



➢ Select Example Project:



TSMaster x86 v2024.7.4.1142 [Unsaved *]

Built-in Sample Projects

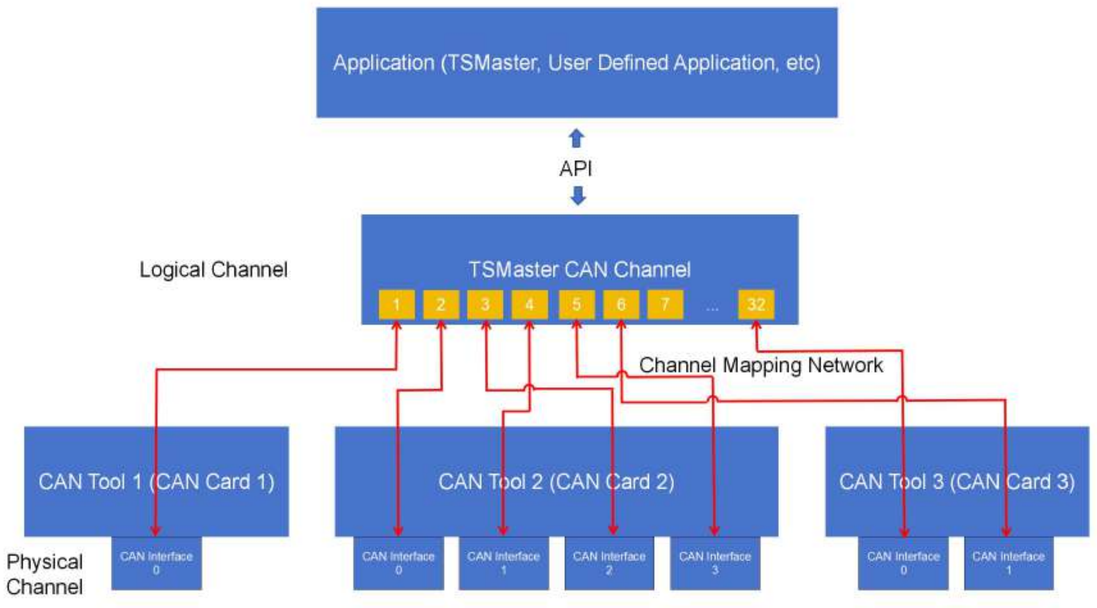
- Code Generation – Example of dynamic code Generation
- Ethernet – Ethernet and API example
- FlexRay – FlexRay Bus application example
- Logging And Replay – Logging, online and offline replay example
- Measurement – Bus Measurement and signal monitor example
- Mini Program – C/C++ application design and secondary development
- Mini Program Library – Iterative development based on an existing mini-program library
- Panels – Panel usage and related examples
- Simulation – Bus and ECU Simulation
- Co-Simulation – Joint simulation between different simulation software
- Test System – Test System solution examples

Name	Status	Comment
Graphic Program Generation	Latest	Example of automatically creating a Graphical Program by C Mini Program
UDS Code Generation	Latest	Example of using the UDS Code Generator in a Graphical Program

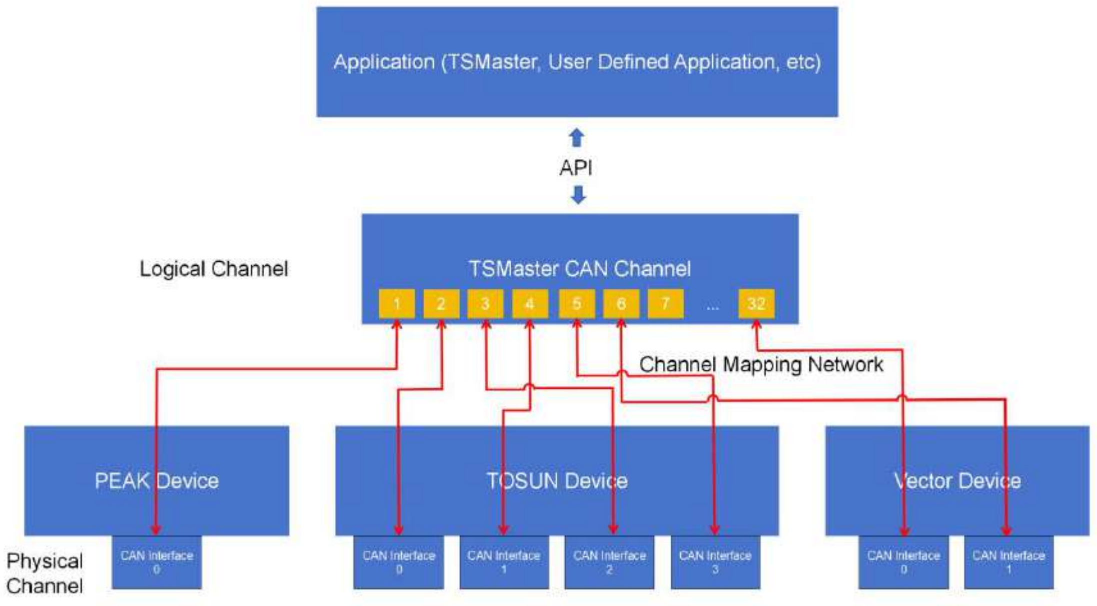
1.4. Hardware Configuration

1.4.1. Hardware Channel Mapping

The application program directly calls the logical channels within the TSMaster driver, and these logical channels are associated with the physical channels of the actual CAN tool (CAN card) through a mapping mechanism. The mapping mechanism is illustrated in the diagram below:



To present the effect of mapping in a more intuitive manner:



1.4.2. Purposes of TSMaster Mapping

TSMaster employs a mapping mechanism, primarily for the following purposes:

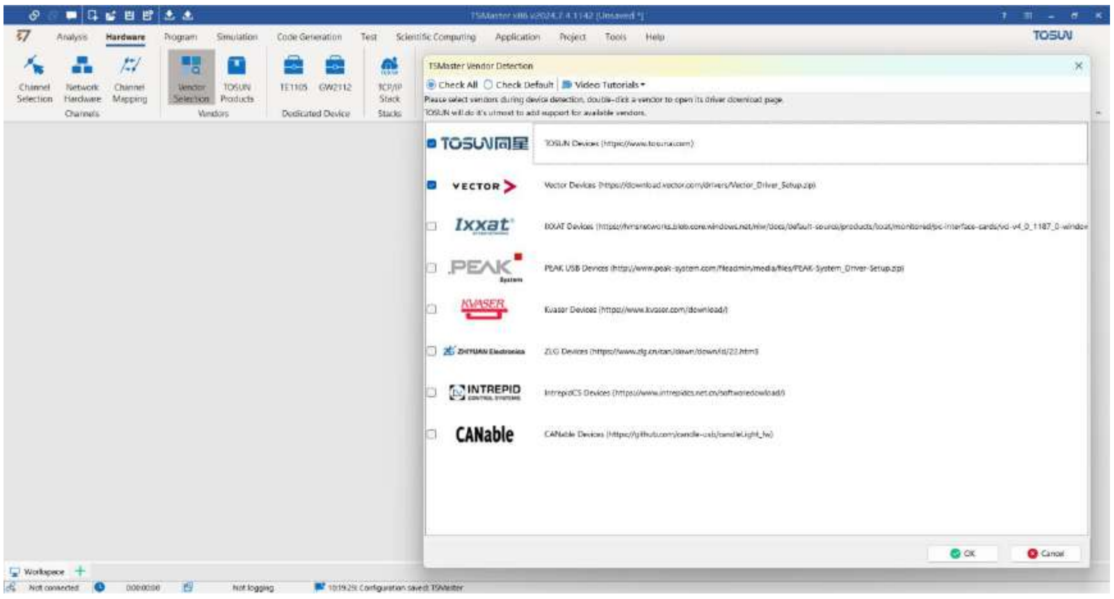
1. To maximize compatibility with the customer's existing hardware. Whether the customer is using Vector, Peak, Kvaser, or TOSUN's own TSCAN series tools, they can all be directly utilized in TSMaster through a mapping approach.

2. Logical channels provide a unified interface for the upper application layer. When users develop test programs using TSMaster scripts, they do not have to worry about previous work being wasted when switching between different tools.

3. The logical channel extends the physical channel's limitation on the number of channels. Due to the physical space constraints of interfaces, common CAN tools often have 1, 2, or 4-channel configurations. However, in certain application scenarios where more than 4, 6, or even more CAN channels are required to work collaboratively, the direct use of hardware may not meet these requirements. In such cases, logical channels can be employed to map multiple physical channels from various devices into TSMaster for coordinated operation. Precise synchronization between channels will be elaborated on in further detail later.

1.4.3. Load Hardware Driver

TSMaster is compatible with a wide range of tools, but by default, it only loads the hardware drivers for TOSUN and Vector. If you wish to use CAN drivers from other brands, you need to enable the corresponding hardware drivers on the hardware tool provider's page, as shown below:



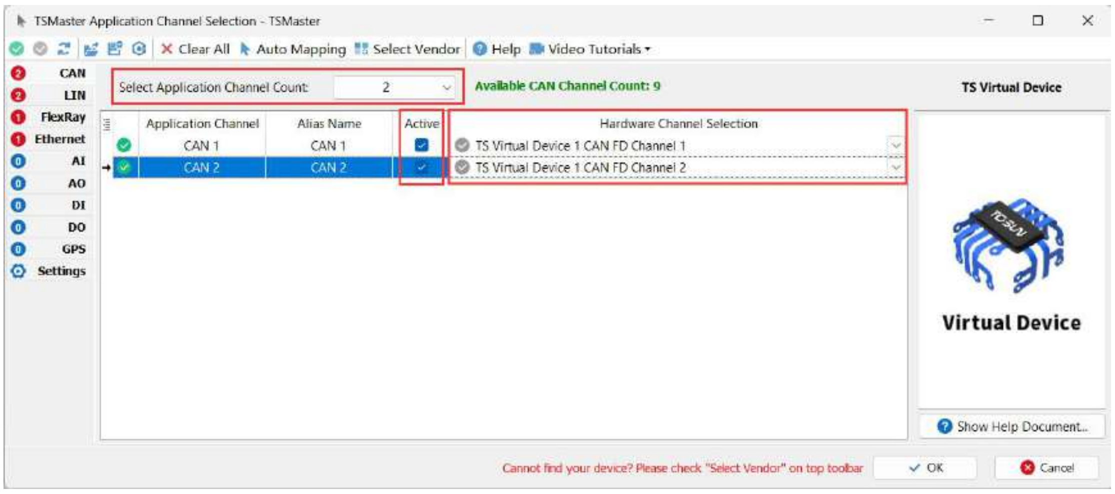
[✓ Check All] Check Default

Please select vendors during device detection, double-click a vendor to open its driver download page.
TOSUN will do it's utmost to add support for available vendors.

- [✓] TOSUN Devices (https://www.tosunai.com)
- [✓] Vector Devices (https://download.vector.com/drivers/Vector_Driver_Setup.zip)
- [ ] IXXAT Devices (https://hmsnetworks.blob.core.windows.net/nlw/docs/default-source/products/ixxat/monitored/pc-interface-cards/vci-v4_0_1187_0-window
- [ ] PEAK USB Devices (http://www.peak-system.com/fileadmin/media/files/PEAK-System_Driver-Setup.zip)
- [ ] Kvaser Devices (https://www.kvaser.com/download/)
- [ ] ZLG Devices (https://www.zlg.cn/can/down/down/id/22.html)
- [ ] IntrepidCS Devices (https://www.intrepidcs.net.cn/softwaredownload/)
- [ ] CANable Devices (https://github.com/candle-usb/candleLight_fw)

10:19:29 Configuration saved: TSMaster

1.4.4. Select Hardware Channel

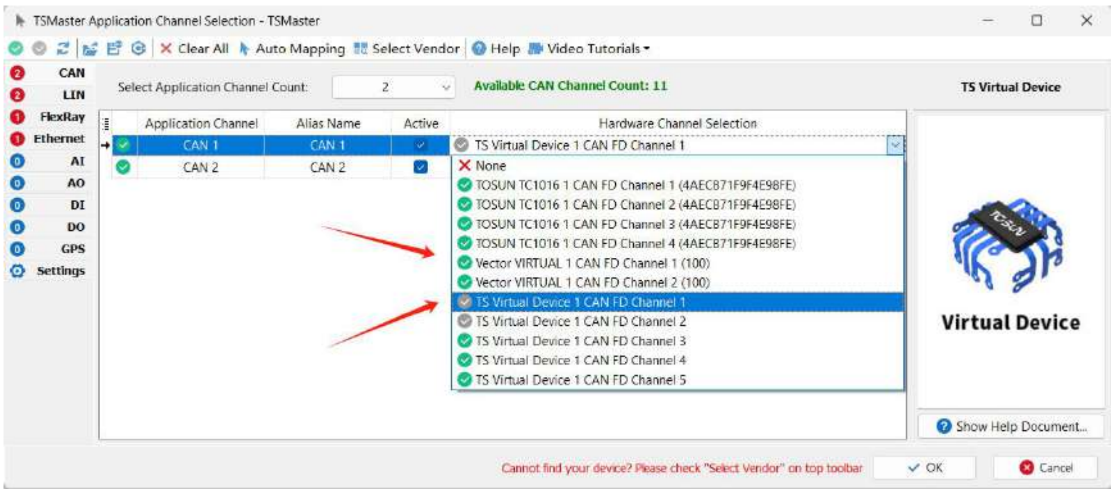

TSMaster Application Channel Selection - TSMaster
Clear All
Auto Mapping
Select Vendor
Help
Video Tutorials
CAN
LIN
FlexRay
Ethernet
AI
AO
DI
DO
GPS
Settings
Select Application Channel Count: 2
Available CAN Channel Count: 9
TS Virtual Device
Virtual Device
Show Help Document...
Cannot find your device? Please check "Select Vendor" on top toolbar
OK
Cancel

Application Channel	Alias Name	Active	Hardware Channel Selection
CAN 1	CAN 1	✓	TS Virtual Device 1 CAN FD Channel 1
CAN 2	CAN 2	✓	TS Virtual Device 1 CAN FD Channel 2

Select Application Channel Count: select the number of application channel that required.
Active: whether to active the channel.
Hardware Channel Selection: select the corresponding hardware channel.

1.4.4.1.Select Hardware Channel

In the hardware channel selection interface, users can choose the channels of hardware from different manufacturers that are plugged into the computer. Once the selection is made, the channel mapping on TSMaster is completed.


TSMaster Application Channel Selection - TSMaster
Clear All
Auto Mapping
Select Vendor
Help
Video Tutorials
CAN
LIN
FlexRay
Ethernet
AI
AO
DI
DO
GPS
Settings
Select Application Channel Count: 2
Available CAN Channel Count: 11
TS Virtual Device
Virtual Device
Show Help Document...
Cannot find your device? Please check "Select Vendor" on top toolbar
OK
Cancel

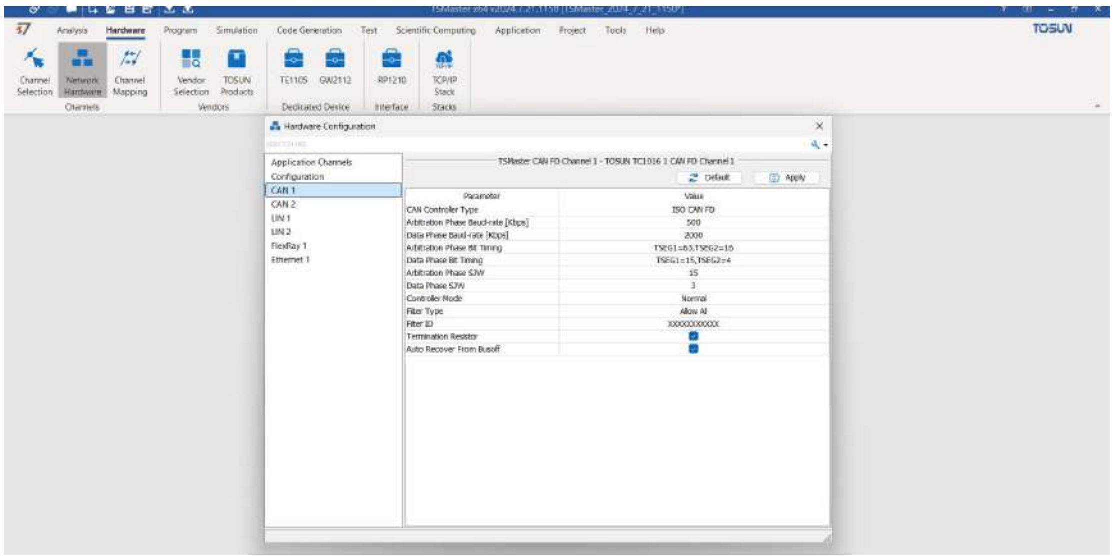
Application Channel	Alias Name	Active	Hardware Channel Selection
CAN 1	CAN 1	✓	TS Virtual Device 1 CAN FD Channel 1
CAN 2	CAN 2	✓	

None
TOSUN TC1016 1 CAN FD Channel 1 (4AECB71F9F4E98FE)
TOSUN TC1016 1 CAN FD Channel 2 (4AECB71F9F4E98FE)
TOSUN TC1016 1 CAN FD Channel 3 (4AECB71F9F4E98FE)
TOSUN TC1016 1 CAN FD Channel 4 (4AECB71F9F4E98FE)
Vector VIRTUAL 1 CAN FD Channel 1 (100)
Vector VIRTUAL 1 CAN FD Channel 2 (100)
TS Virtual Device 1 CAN FD Channel 1
TS Virtual Device 1 CAN FD Channel 2
TS Virtual Device 1 CAN FD Channel 3
TS Virtual Device 1 CAN FD Channel 4
TS Virtual Device 1 CAN FD Channel 5

As shown in the figure above: For logical channel CAN1, you can choose hardware from TOSUN or from Vector.

1.4.4.2.Configuration Parameters

General settings for hardware channel parameters include bitrate, sampling rate, bit timing, etc.

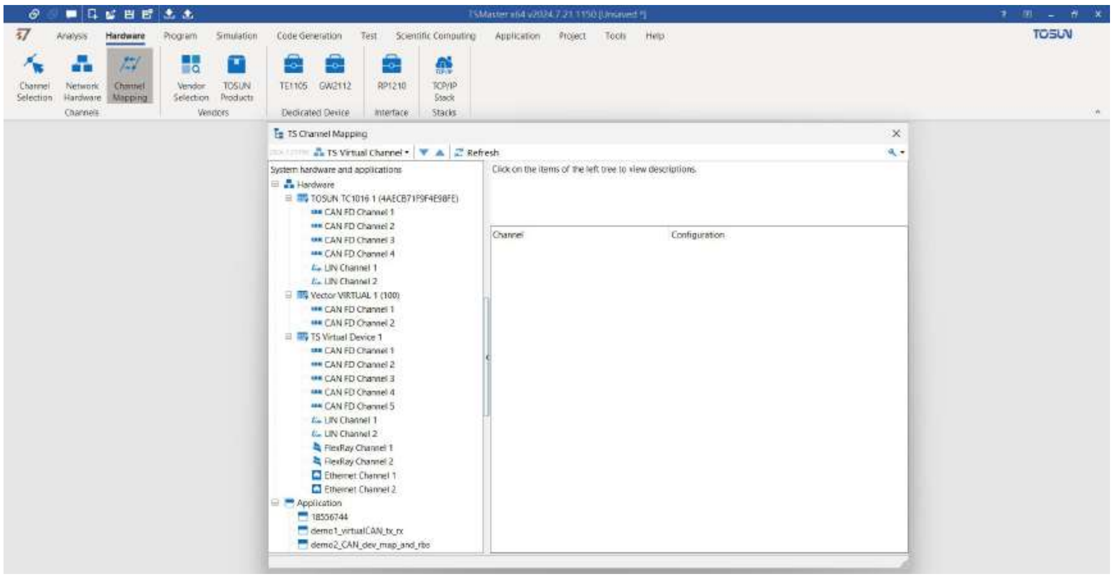

Hardware Configuration

Application Channels
Configuration
CAN 1
CAN 2
LIN 1
LIN 2
FlexRay 1
Ethernet 1

TSMaster CAN FD Channel 1 - TOSUN TC1016 1 CAN FD Channel 1

Parameter	Value
CAN Controller Type	ISO CAN FD
Arbitration Phase Baud-rate [Kbps]	500
Data Phase Baud-rate [Kbps]	2000
Arbitration Phase Bit Timing	TSEG1=63,TSEG2=16
Data Phase Bit Timing	TSEG1=15,TSEG2=4
Arbitration Phase SJW	15
Data Phase SJW	3
Controller Mode	Normal
Filter Type	Allow All
Filter ID	XXXXXXXXXXX
Termination Resistor	✓
Auto Recover From Busoff	✓

1.4.4.3.Channel Mapping


TS Channel Mapping

System hardware and applications
- Hardware
  - TOSUN TC1016 1 (4AECB71F9F4E98FE)
    - CAN FD Channel 1
    - CAN FD Channel 2
    - CAN FD Channel 3
    - CAN FD Channel 4
    - LIN Channel 1
    - LIN Channel 2
  - Vector VIRTUAL 1 (100)
    - CAN FD Channel 1
    - CAN FD Channel 2
  - TS Virtual Device 1
    - CAN FD Channel 1
    - CAN FD Channel 2
    - CAN FD Channel 3
    - CAN FD Channel 4
    - CAN FD Channel 5
    - LIN Channel 1
    - LIN Channel 2
    - FlexRay Channel 1
    - FlexRay Channel 2
    - Ethernet Channel 1
    - Ethernet Channel 2
- Application
  - 18556744
  - demo1_virtualCAN_tx_rx
  - demo2_CAN_dev_map_and_rbs

Click on the items of the left tree to view descriptions.

Channel	Configuration

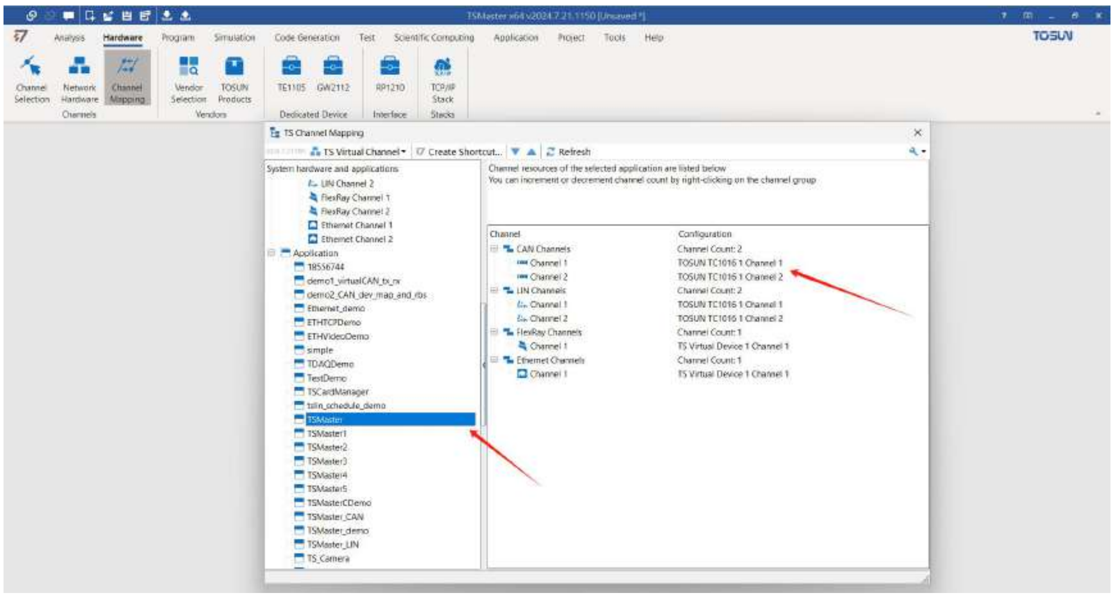
Channel mapping provides a graphical mapping manager, which mainly includes two main components:

1. Hardware: It displays all available hardware device information currently plugged into the user's computer. As shown below, it includes both TOSUN's CANMini hardware and

Vector's CANCASEXL hardware.

2. Application: It displays all applications that are based on the TSMasterAPI underlying layer (including TSMaster itself, of course).

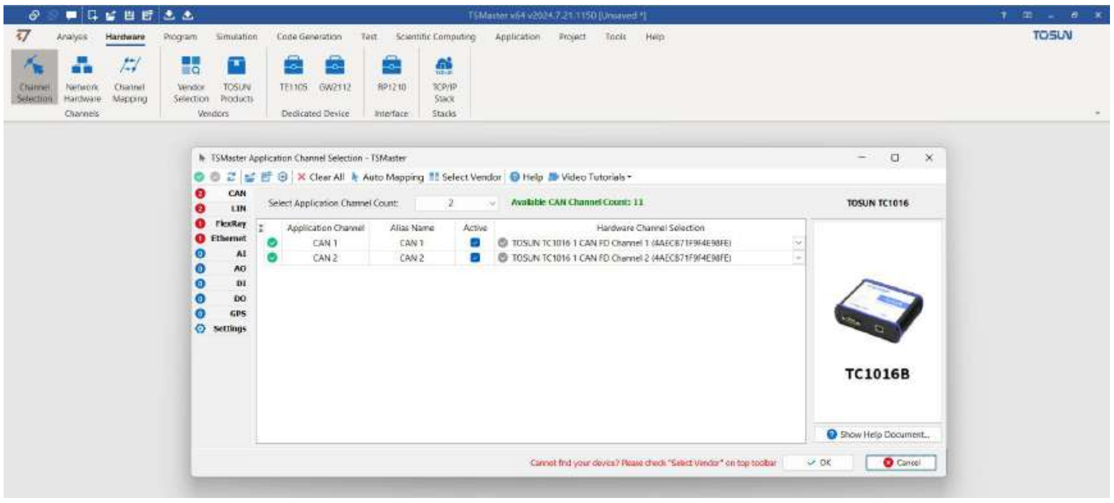
   Take TSMaster application itself for example:



System hardware and applications
- LIN Channel 2
- FlexRay Channel 1
- FlexRay Channel 2
- Ethernet Channel 1
- Ethernet Channel 2
- Application
  - 18556744
  - demo1_virtualCAN_tx_rx
  - demo2_CAN_dev_map_and_rbs
  - Ethernet_demo
  - ETHTCPDemo
  - ETHVideoDemo
  - simple
  - TDAQDemo
  - TestDemo
  - TSCardManager
  - tslin_schedule_demo
  - TSMaster
  - TSMaster1
  - TSMaster2
  - TSMaster3
  - TSMaster4
  - TSMaster5
  - TSMasterCDemo
  - TSMaster_CAN
  - TSMaster_demo
  - TSMaster_LIN
  - TS_Camera

Channel resources of the selected application are listed below
You can increment or decrement channel count by right-clicking on the channel group

Channel	Configuration
CAN Channels	Channel Count: 2
Channel 1	TOSUN TC1016 1 Channel 1
Channel 2	TOSUN TC1016 1 Channel 2
LIN Channels	Channel Count: 2
Channel 1	TOSUN TC1016 1 Channel 1
Channel 2	TOSUN TC1016 1 Channel 2
FlexRay Channels	Channel Count: 1
Channel 1	TS Virtual Device 1 Channel 1
Ethernet Channels	Channel Count: 1
Channel 1	TS Virtual Device 1 Channel 1

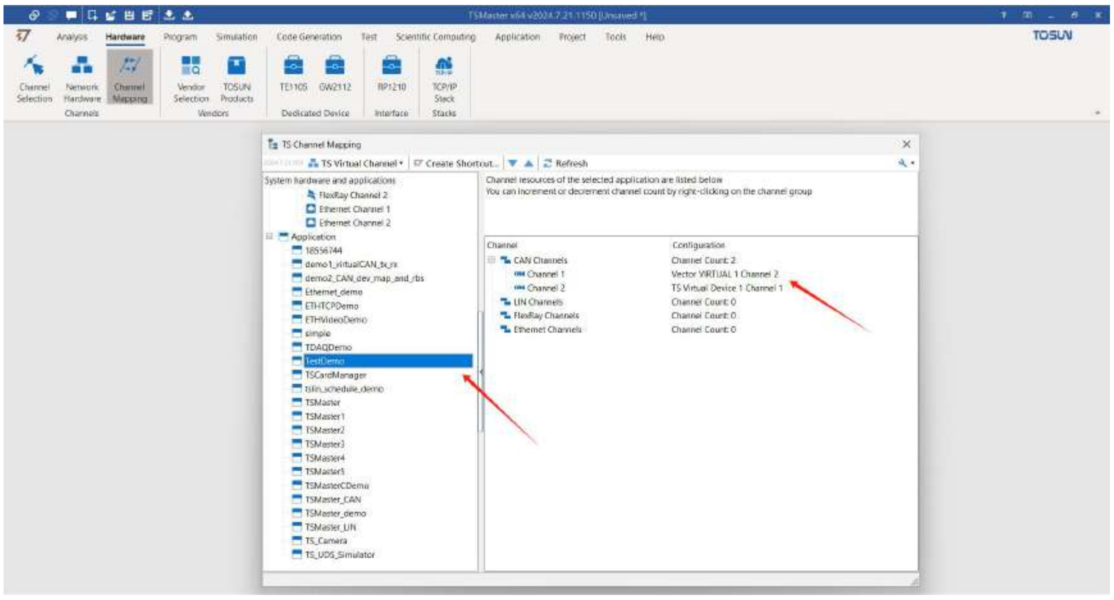


Select Application Channel Count: 2
Available CAN Channel Count: 11

	Application Channel	Alias Name	Active	Hardware Channel Selection
✓	CAN 1	CAN 1	✓	TOSUN TC1016 1 CAN FD Channel 1 (4AECB71F9F4E98FE)
✓	CAN 2	CAN 2	✓	TOSUN TC1016 1 CAN FD Channel 2 (4AECB71F9F4E98FE)

Cannot find your device? Please check "Select Vendor" on top toolbar

The parameter settings visible within the mapping manager are fully correspondent to the channel settings in TSMaster. Taking the TestDemo application as an example:



Channel resources of the selected application are listed below
You can increment or decrement channel count by right-clicking on the channel group

Channel	Configuration
CAN Channels	Channel Count: 2
Channel 1	Vector VIRTUAL 1 Channel 2
Channel 2	TS Virtual Device 1 Channel 1
LIN Channels	Channel Count: 0
FlexRay Channels	Channel Count: 0
Ethernet Channels	Channel Count: 0

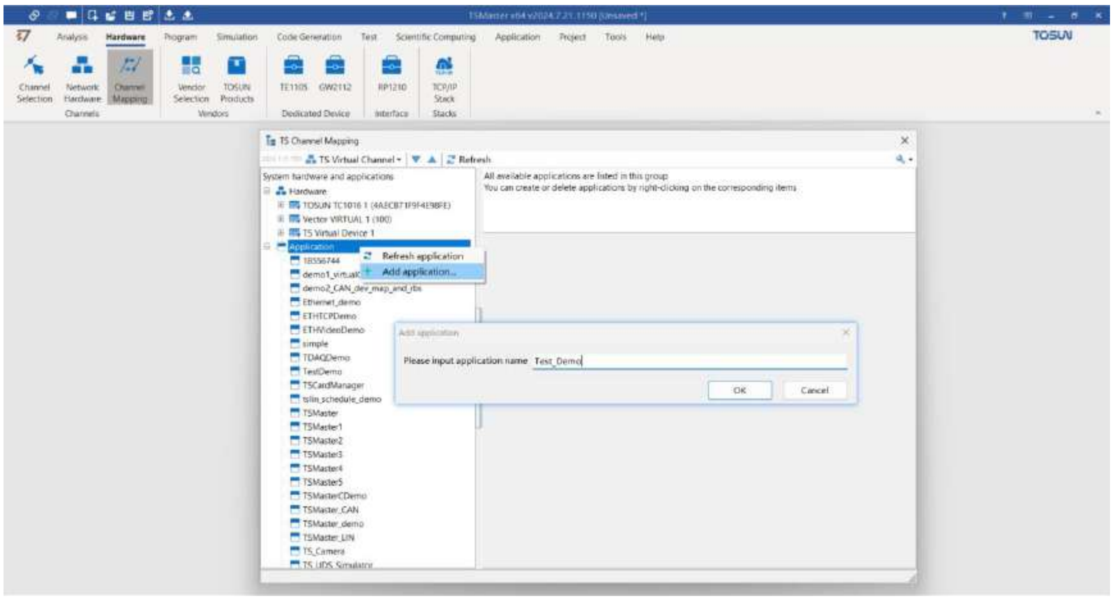
The application named TestDemo utilizes two CAN channels and zero LIN channels. The CAN1 channel is mapped to channel 2 of the Vector VRTUA Channel2, while the CAN2 channel is mapped to the TS Virtual Device1 Channel1.

1.4.4.4.Configure Hardware Channel

Similar to any application based on the TSMaster API, channel mapping can be configured graphically or through code before connecting the application. Below, we will explain both methods separately:

1.4.4.4.1. Configuration by GUI:

1. First, add the application by Hardware-> Channel Mapping->Application->right click the application and select “Add application”:



System hardware and applications

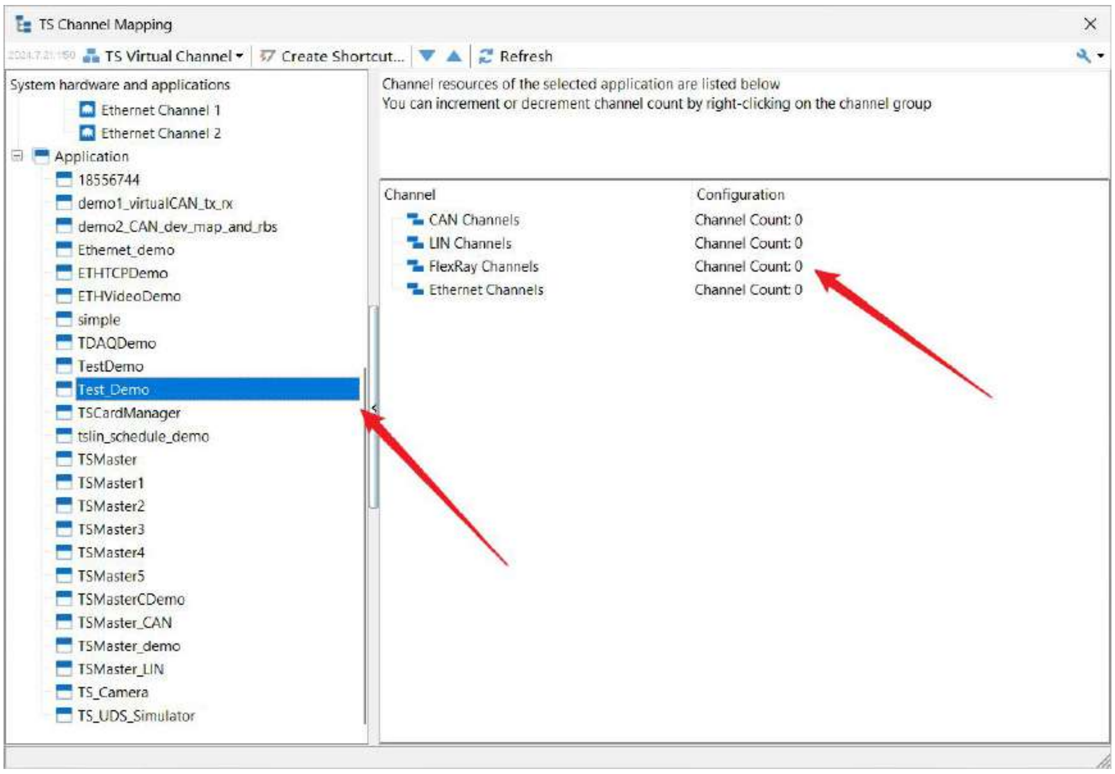
All available applications are listed in this group
You can create or delete applications by right-clicking on the corresponding items

Please input application name Test_Demo

Once an application is added, it remains in the system and does not require recreation the next time it is used.

2. Map hardware for the application.

Select the application added in the previous step. If it is a newly created program, the corresponding hardware channels for CAN Channels and LIN Channels should both equal 0, meaning no actual hardware is assigned, as shown in the figure below:

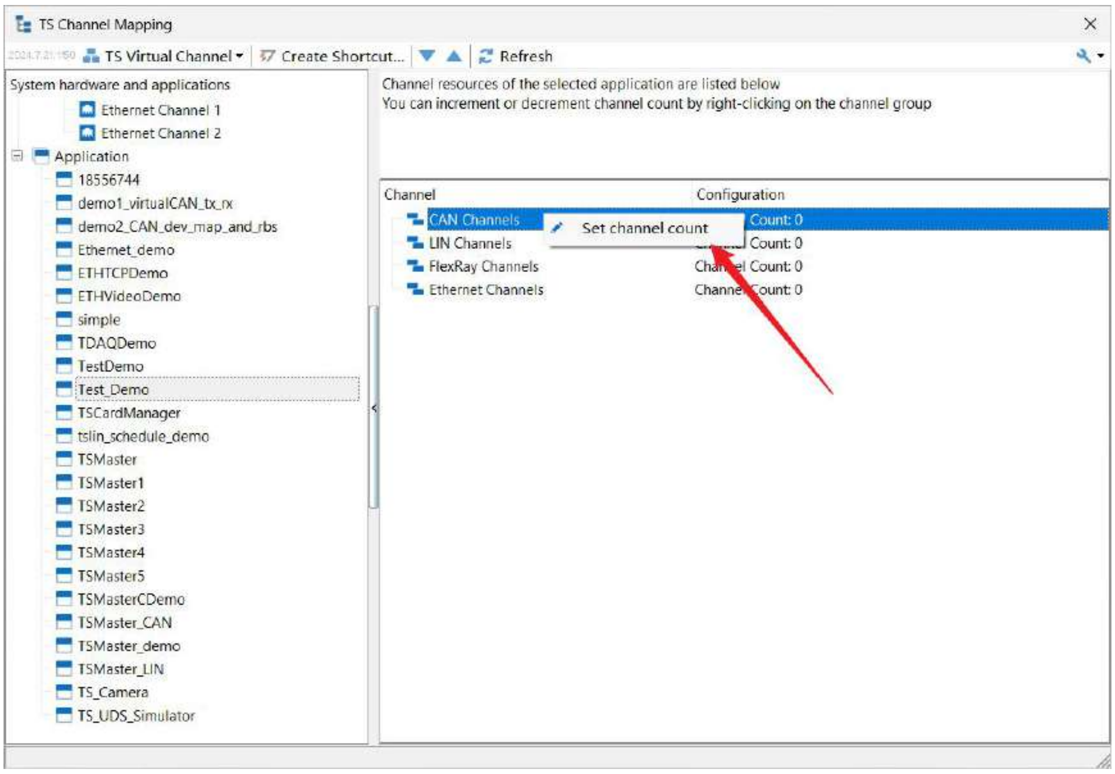


System hardware and applications
- Ethernet Channel 1
- Ethernet Channel 2
- Application
  - 18556744
  - demo1_virtualCAN_tx_rx
  - demo2_CAN_dev_map_and_rbs
  - Ethernet_demo
  - ETHTCPDemo
  - ETHVideoDemo
  - simple
  - TDAQDemo
  - TestDemo
  - Test_Demo
  - TSCardManager
  - tslin_schedule_demo
  - TSMaster
  - TSMaster1
  - TSMaster2
  - TSMaster3
  - TSMaster4
  - TSMaster5
  - TSMasterCDemo
  - TSMaster_CAN
  - TSMaster_demo
  - TSMaster_LIN
  - TS_Camera
  - TS_UDS_Simulator

Channel resources of the selected application are listed below
You can increment or decrement channel count by right-clicking on the channel group

Channel	Configuration
CAN Channels	Channel Count: 0
LIN Channels	Channel Count: 0
FlexRay Channels	Channel Count: 0
Ethernet Channels	Channel Count: 0

At this point, select the CAN channels for the application and right-click to set the number of channels, as shown below:



TS Channel Mapping

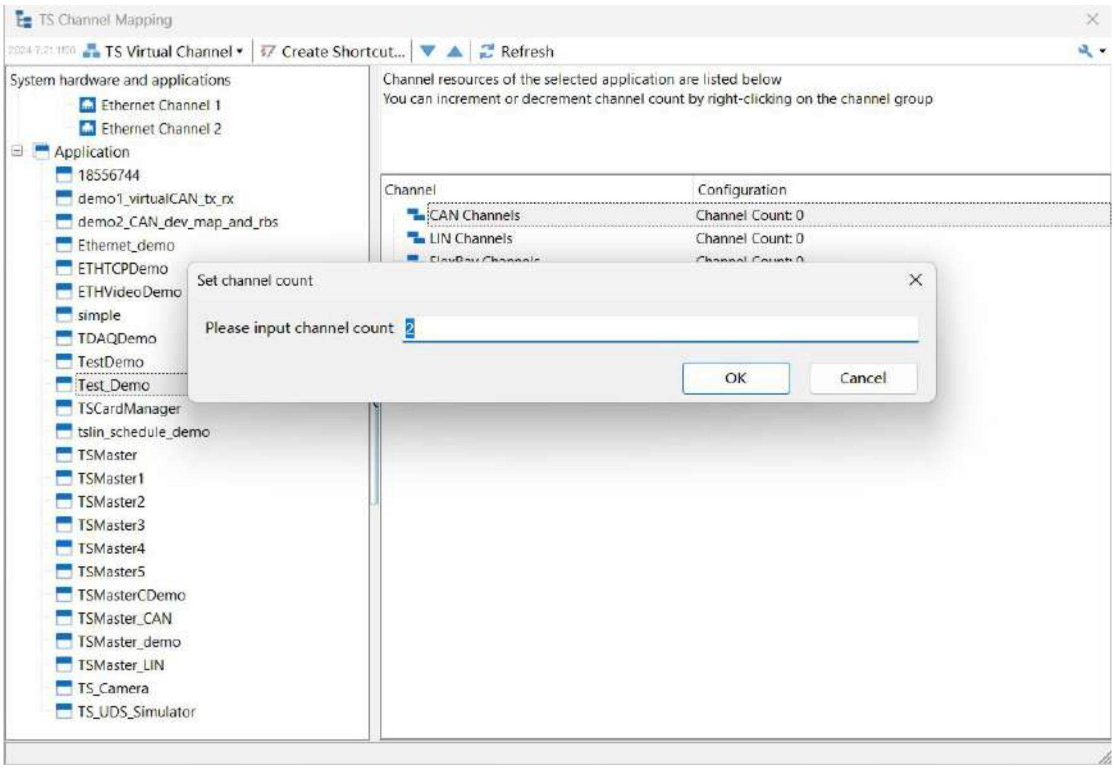
TS Virtual Channel | Create Shortcut... | Refresh

System hardware and applications
- Ethernet Channel 1
- Ethernet Channel 2
- Application
  - 18556744
  - demo1_virtualCAN_tx_rx
  - demo2_CAN_dev_map_and_rbs
  - Ethernet_demo
  - ETHTCPDemo
  - ETHVideoDemo
  - simple
  - TDAQDemo
  - TestDemo
  - Test_Demo
  - TSCardManager
  - tslin_schedule_demo
  - TSMaster
  - TSMaster1
  - TSMaster2
  - TSMaster3
  - TSMaster4
  - TSMaster5
  - TSMasterCDemo
  - TSMaster_CAN
  - TSMaster_demo
  - TSMaster_LIN
  - TS_Camera
  - TS_UDS_Simulator

Channel resources of the selected application are listed below
You can increment or decrement channel count by right-clicking on the channel group

Channel	Configuration
CAN Channels	Count: 0
LIN Channels	Count: 0
FlexRay Channels	Count: 0
Ethernet Channels	Count: 0

An interface for setting the number of channels appears. For an application, up to 64 channels are supported at the same time, here it is set to 2 channels, as shown below:

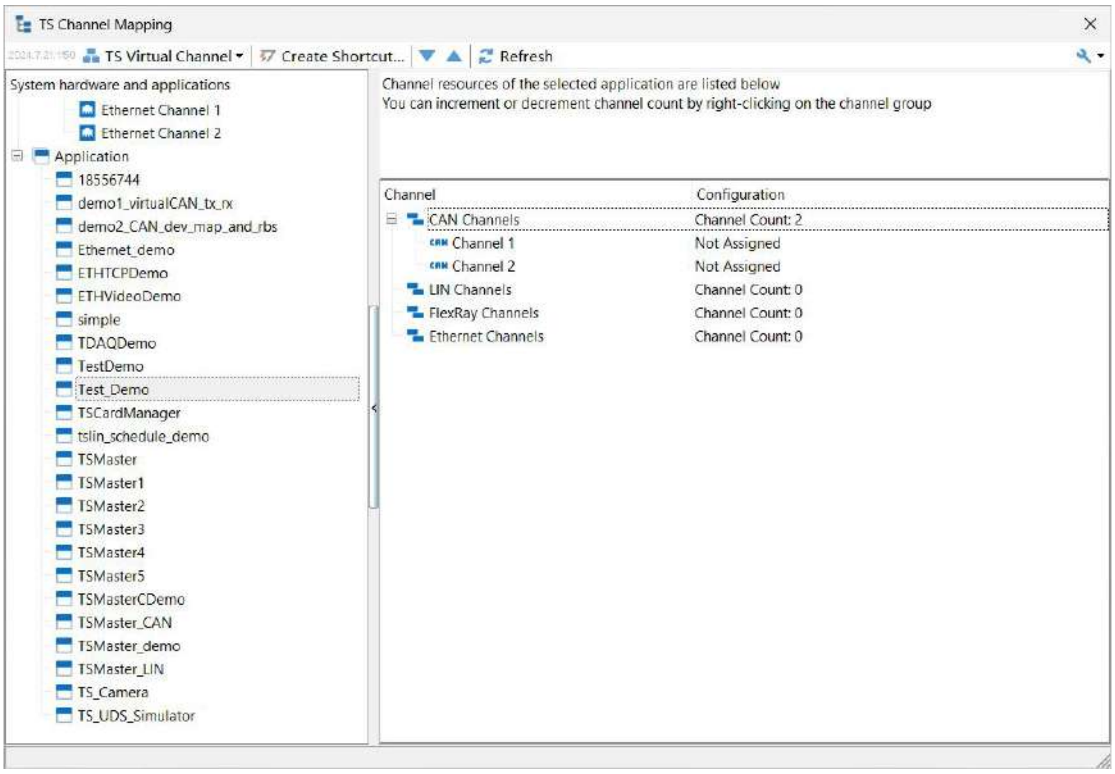


Channel resources of the selected application are listed below
You can increment or decrement channel count by right-clicking on the channel group

Channel	Configuration
CAN Channels	Channel Count: 0
LIN Channels	Channel Count: 0

Please input channel count 2

Create two CAN channels for the application, but without pointing them to any actual hardware. Therefore, both channels will show as unassigned, as shown below:

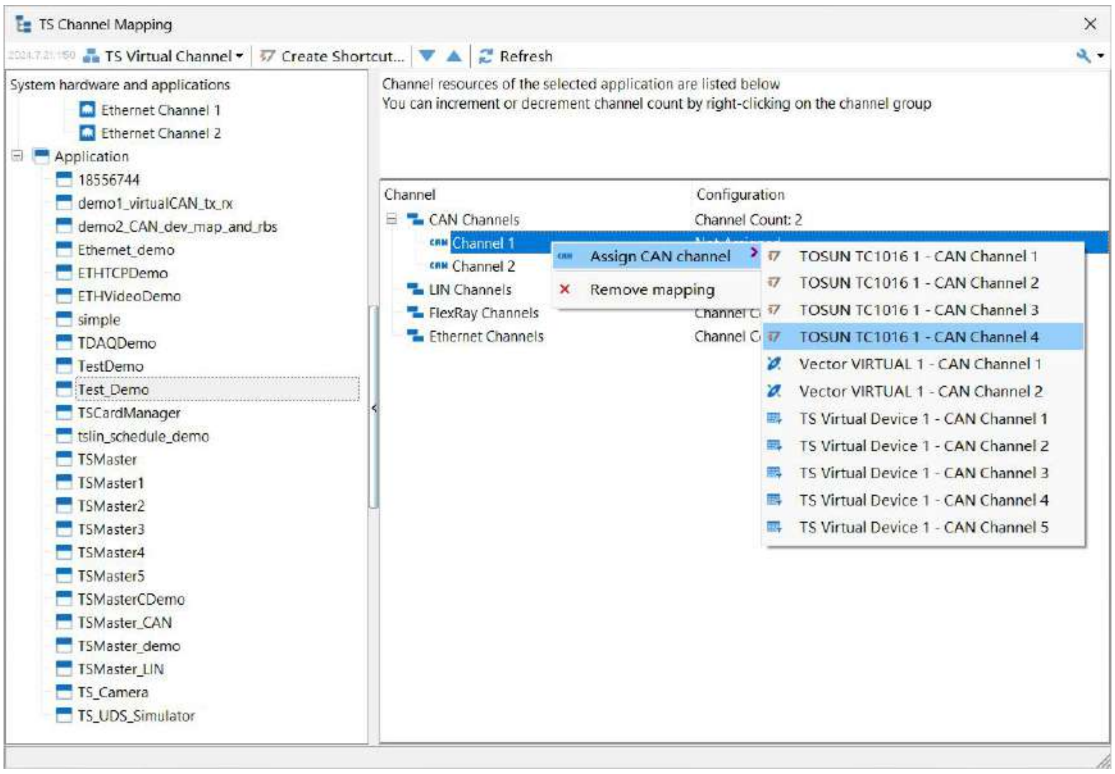


Channel resources of the selected application are listed below
You can increment or decrement channel count by right-clicking on the channel group

Channel	Configuration
CAN Channels	Channel Count: 2
Channel 1	Not Assigned
Channel 2	Not Assigned
LIN Channels	Channel Count: 0
FlexRay Channels	Channel Count: 0
Ethernet Channels	Channel Count: 0

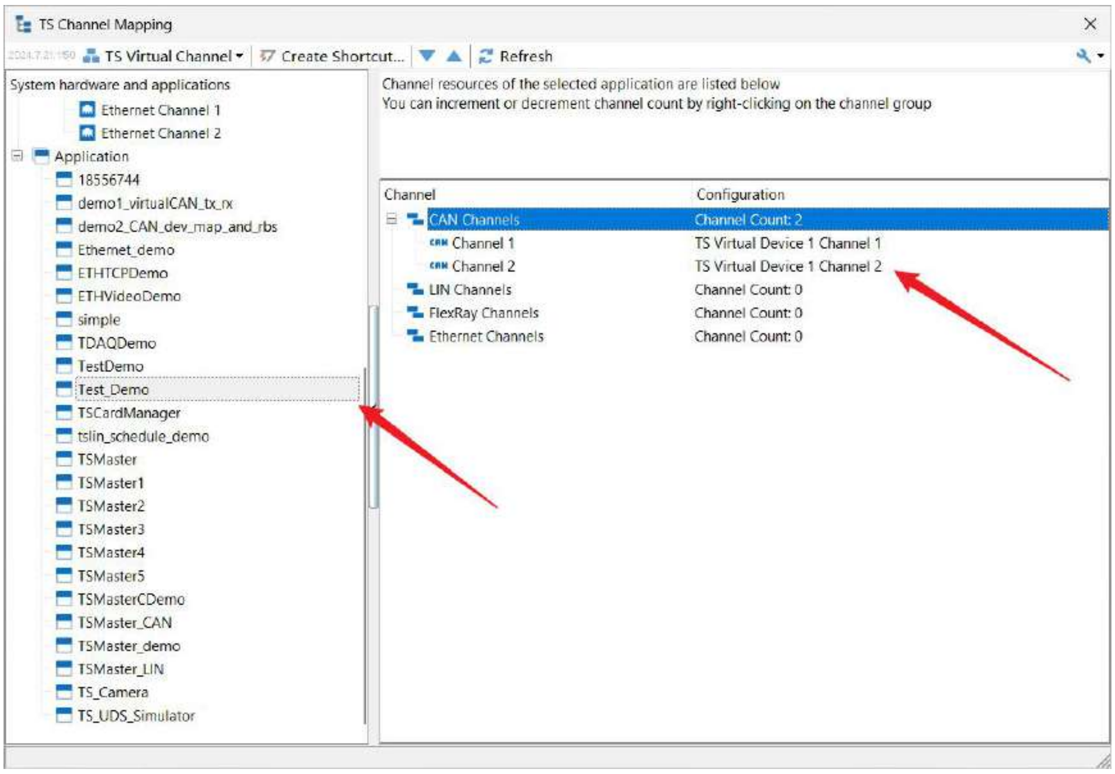
To map a specific hardware channel to the channel, follow the steps below: Select the channel -

> Right-click: Assign CAN channel -> choose a channel from the expanded hardware resources on the right side, and the CAN channel mapping is complete.



Channel resources of the selected application are listed below
You can increment or decrement channel count by right-clicking on the channel group

After mapping is completed, as shown in the figure below, the device indicates: the application named Test_Demo contains two CAN channels. CAN Channel 1 uses Channel 1 of the TOSUN TS.CAN Lite2 hardware, and CAN Channel 2 uses Channel 2 of the TOSUN TS.CAN Lite2 hardware.



TS Channel Mapping

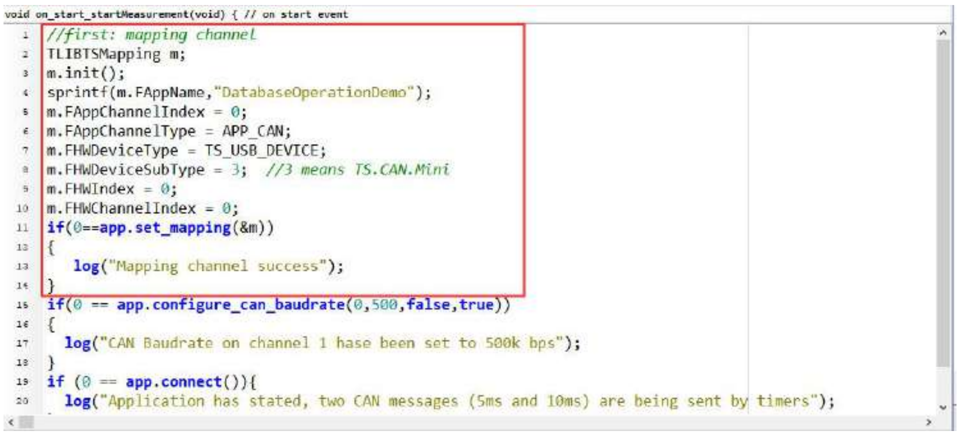
TS Virtual Channel ▾ | Create Shortcut... | Refresh

System hardware and applications
- Ethernet Channel 1
- Ethernet Channel 2
- Application
  - 18556744
  - demo1_virtualCAN_tx_rx
  - demo2_CAN_dev_map_and_rbs
  - Ethernet_demo
  - ETHTCPDemo
  - ETHVideoDemo
  - simple
  - TDAQDemo
  - TestDemo
  - Test_Demo
  - TSCardManager
  - tslin_schedule_demo
  - TSMaster
  - TSMaster1
  - TSMaster2
  - TSMaster3
  - TSMaster4
  - TSMaster5
  - TSMasterCDemo
  - TSMaster_CAN
  - TSMaster_demo
  - TSMaster_LIN
  - TS_Camera
  - TS_UDS_Simulator

Channel resources of the selected application are listed below
You can increment or decrement channel count by right-clicking on the channel group

Channel	Configuration
CAN Channels	Channel Count: 2
Channel 1	TS Virtual Device 1 Channel 1
Channel 2	TS Virtual Device 1 Channel 2
LIN Channels	Channel Count: 0
FlexRay Channels	Channel Count: 0
Ethernet Channels	Channel Count: 0

1.4.4.4.2. Configuration by Codes



```
void on_start_startMeasurement(void) { // on start event
//first: mapping channel
TLIBTSMapping m;
m.init();
sprintf(m.FAppName,"DatabaseOperationDemo");
m.FAppChannelIndex = 0;
m.FAppChannelType = APP_CAN;
m.FHWDeviceType = TS_USB_DEVICE;
m.FHWDeviceSubType = 3;  //3 means TS.CAN.Mini
m.FHWIndex = 0;
m.FHWChannelIndex = 0;
if(0==app.set_mapping(&m))
{
   log("Mapping channel success");
}
if(0 == app.configure_can_baudrate(0,500,false,true))
{
  log("CAN Baudrate on channel 1 hase been set to 500k bps");
}
if (0 == app.connect()){
  log("Application has stated, two CAN messages (5ms and 10ms) are being sent by timers");
```

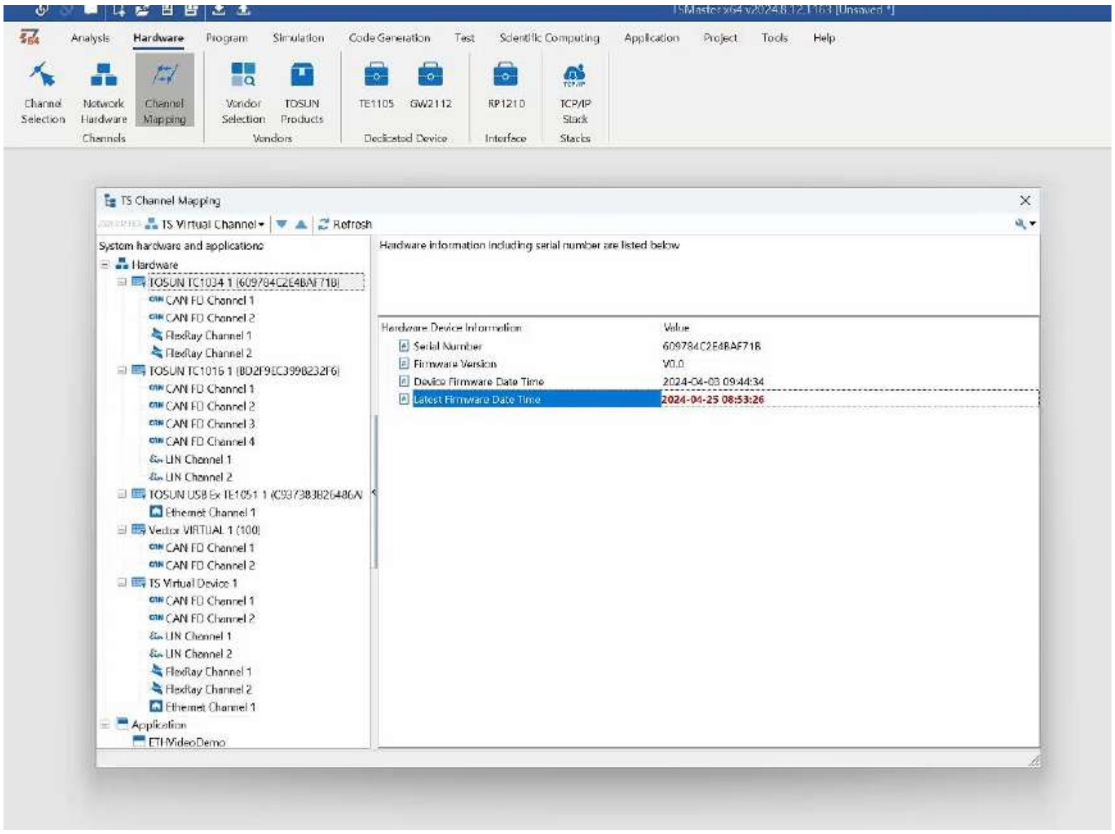
As shown by the code segment highlighted in the red box in the above figure, it accomplishes the task of mapping the channel of an application named "DatabaseOperation" to the physical channel of the TSCANMini device.

After the mapping is completed, the bitrate of this logical channel is configured, along with whether it operates in listen-only mode and whether the internal termination resistor is enabled. Once all settings are finalized, the connect function is called to establish the connection to the application. It is within the connect function that the actual initialization of the CAN tool's hardware parameters takes place. Prior to the application connecting, the CAN tool must remain in a silent state to prevent any interference with the CAN bus.

1.4.5. Firmware Upgrade

During the usage of hardware devices, there may be a need for adding new features, modifications, or optimizations to the hardware functionality. To address these needs, TSMaster's hardware management system includes a firmware management and upgrade module. Below, we will use the TC1034 device as an example to explain the firmware upgrade process.

➢ First, enter the Channel Mapping interface and select the device that requires firmware upgrade. As shown below:



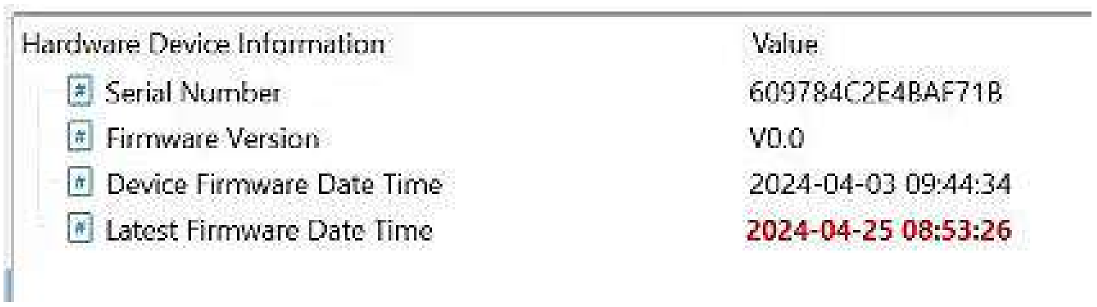
System hardware and applications
- Hardware
  - TOSUN TC1034 1 (609784C2E4BAF71B)
    - CAN FD Channel 1
    - CAN FD Channel 2
    - FlexRay Channel 1
    - FlexRay Channel 2
  - TOSUN TC1016 1 (BD2F9EC3998232F6)
    - CAN FD Channel 1
    - CAN FD Channel 2
    - CAN FD Channel 3
    - CAN FD Channel 4
    - LIN Channel 1
    - LIN Channel 2
  - TOSUN USB Ex TE1051 1 (C937383B26486A
    - Ethernet Channel 1
  - Vector VIRTUAL 1 (100)
    - CAN FD Channel 1
    - CAN FD Channel 2
  - TS Virtual Device 1
    - CAN FD Channel 1
    - CAN FD Channel 2
    - LIN Channel 1
    - LIN Channel 2
    - FlexRay Channel 1
    - FlexRay Channel 2
    - Ethernet Channel 1
- Application
  - ETHVideoDemo

Hardware information including serial number are listed below

Hardware Device Information	Value
Serial Number	609784C2E4BAF71B
Firmware Version	V0.0
Device Firmware Date Time	2024-04-03 09:44:34
Latest Firmware Date Time	2024-04-25 08:53:26

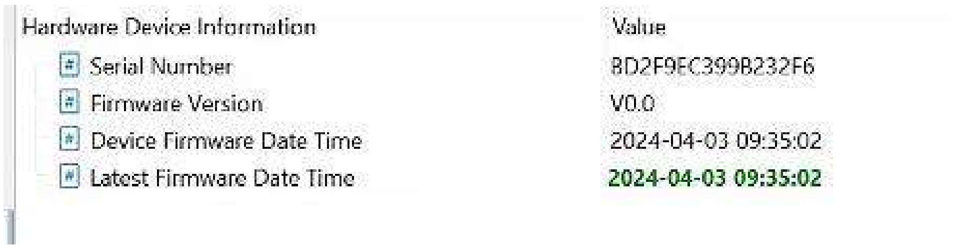
➢ Second，check firmware status.

If there is an updated firmware available, it will be displayed in red font, as shown in the figure below:



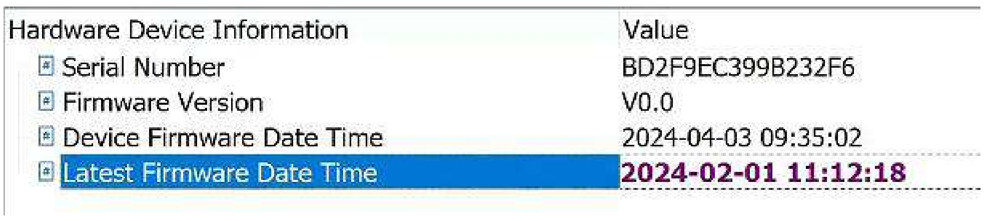
Hardware Device Information	Value
Serial Number	609784C2E4BAF71B
Firmware Version	V0.0
Device Firmware Date Time	2024-04-03 09:44:34
Latest Firmware Date Time	2024-04-25 08:53:26

If the current device firmware is already the latest version, it will be displayed in green, as shown in the figure below:



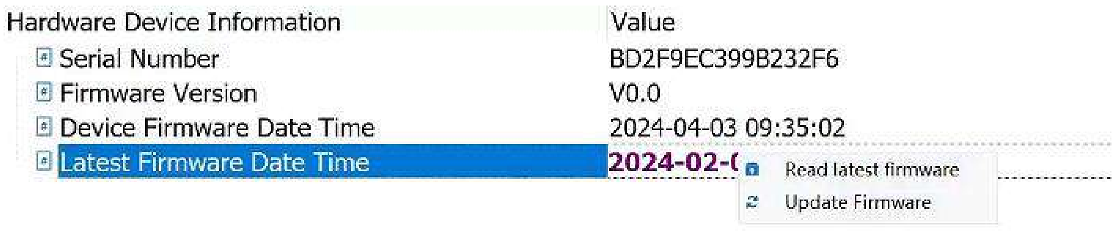
Hardware Device Information	Value
Serial Number	BD2F9EC399B232F6
Firmware Version	V0.0
Device Firmware Date Time	2024-04-03 09:35:02
Latest Firmware Date Time	2024-04-03 09:35:02

If the firmware of the current device is newer than the one queried by the software, it will be displayed in purple, as shown in the figure below:



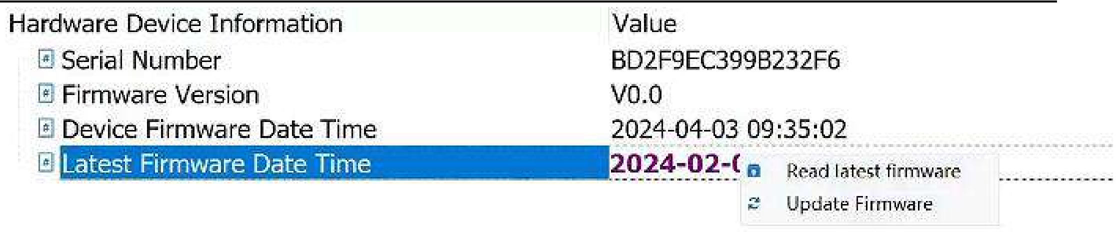
Hardware Device Information	Value
Serial Number	BD2F9EC399B232F6
Firmware Version	V0.0
Device Firmware Date Time	2024-04-03 09:35:02
Latest Firmware Date Time	2024-02-01 11:12:18

When the software firmware display indicates "Not Ready", right-click on the "Latest Firmware Date Time" column to bring up the context menu, and click "Read latest firmware" to query the currently available firmware information.



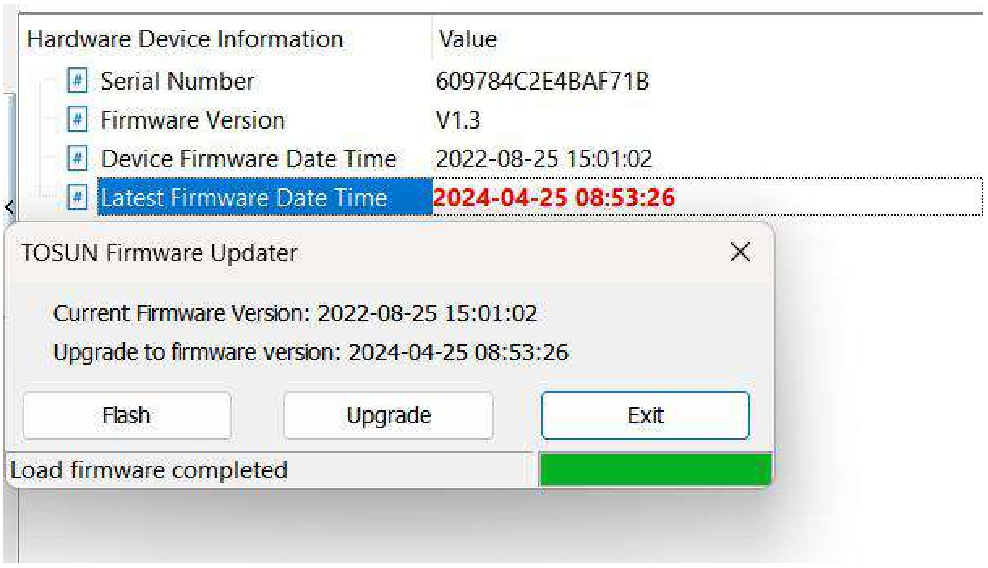
Hardware Device Information	Value
Serial Number	BD2F9EC399B232F6
Firmware Version	V0.0
Device Firmware Date Time	2024-04-03 09:35:02
Latest Firmware Date Time	2024-02-(

➢ Then, right-click on the "Latest Firmware Date Time" column to bring up the context menu, click "Update Firmware", and the firmware upgrade module will pop up.



Hardware Device Information	Value
Serial Number	BD2F9EC399B232F6
Firmware Version	V0.0
Device Firmware Date Time	2024-04-03 09:35:02
Latest Firmware Date Time	2024-02-(

Note that for some devices with larger firmware sizes, the upgrade module will first load the firmware data and then enable the upgrade button, allowing the upgrade process for the firmware to proceed. The firmware upgrade module is shown in the figure below:



Hardware Device Information	Value
Serial Number	609784C2E4BAF71B
Firmware Version	V1.3
Device Firmware Date Time	2022-08-25 15:01:02
Latest Firmware Date Time	2024-04-25 08:53:26

TOSUN Firmware Updater

Current Firmware Version: 2022-08-25 15:01:02

Upgrade to firmware version: 2024-04-25 08:53:26

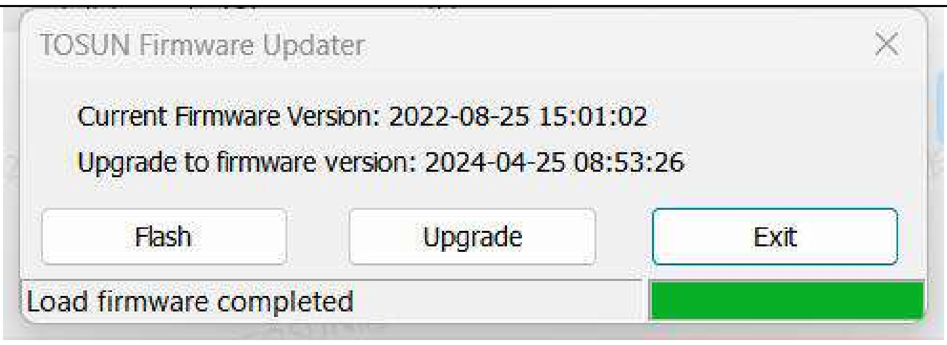
Load firmware completed

The firmware upgrade module has three buttons: Flash, Upgrade, and Exit.

1) Click Flash: the link light of the device currently selected for firmware upgrade will blink three times. This feature is particularly useful when multiple devices are online simultaneously, making it easier for users to locate the target hardware device. Please note that this functionality is only available in device firmware released after February 2024. Early devices will not respond to this button.

2) Click Exit: this will exit the upgrade module.

3) Click Upgrade: this will initiate the upgrade process. Once the progress bar completes, if the upgrade is successful, it will show the prompt message.


TOSUN Firmware Updater

Current Firmware Version: 2022-08-25 15:01:02

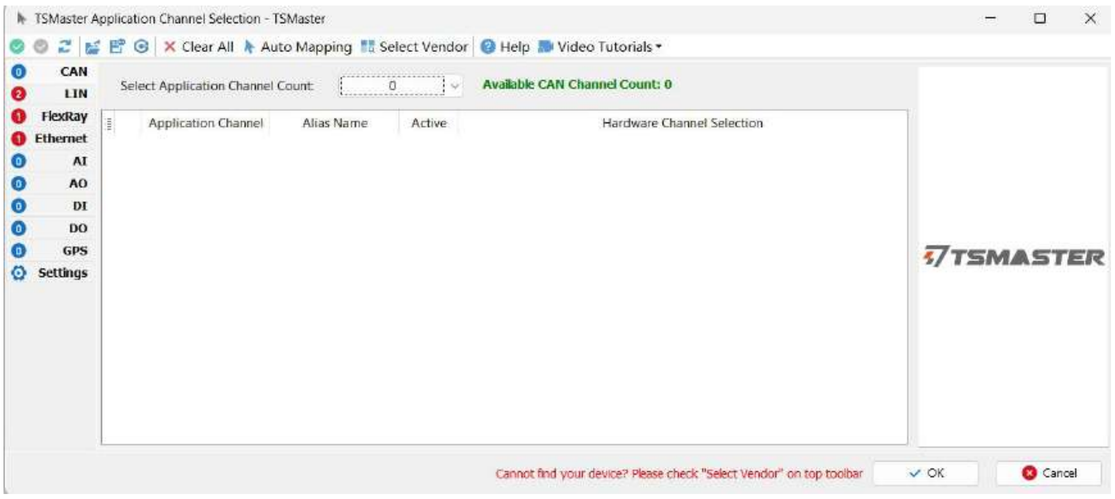
Upgrade to firmware version: 2024-04-25 08:53:26

Load firmware completed

1.4.6. Clarification

1.4.6.1. Hardware Not Found

The developer has correctly installed the device driver and plugged in the hardware device, but TSMaster is "not recognizing" the hardware device, as shown in the image below. This situation is typically caused by the following two reasons:


TSMaster Application Channel Selection - TSMaster

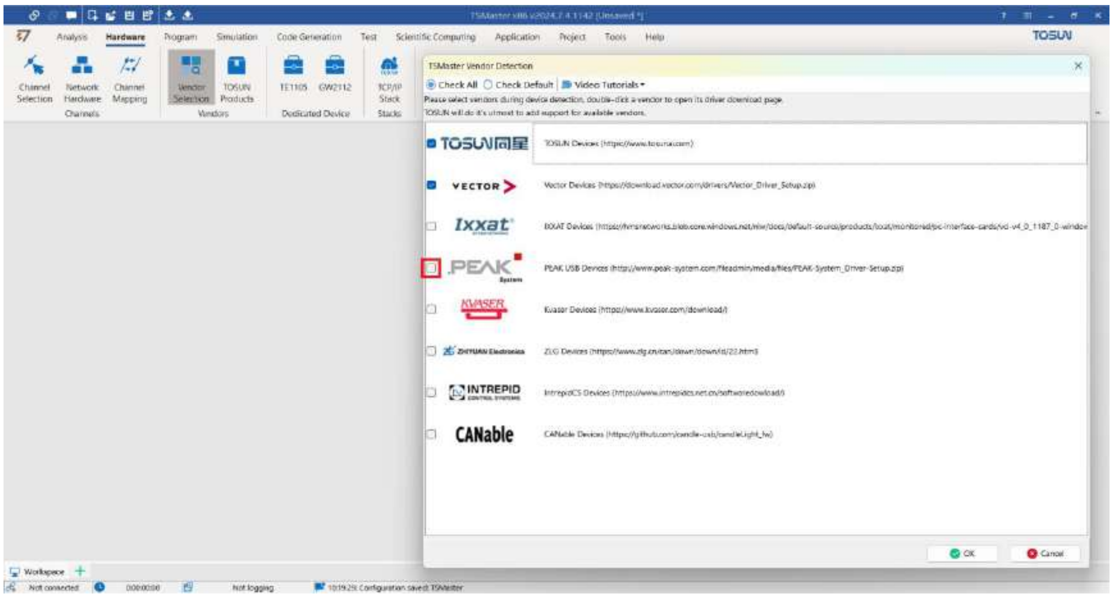
Select Application Channel Count: 0

Available CAN Channel Count: 0

Cannot find your device? Please check "Select Vendor" on top toolbar

1. Device driver not selected.

TSMaster is an open software platform that is compatible with mainstream CAN tools available in the market, such as TOSUN, Vector, VehicleSpy, Kvaser, and Peak. However, in order to reduce program initialization dependencies, only the CAN drivers for TOSUN and Vector are loaded by default. If this is the first time using TSMaster, user needs to enable the loading of the corresponding vendor's driver in the hardware configuration, as shown in the figure below:


TSMaster Vendor Detection

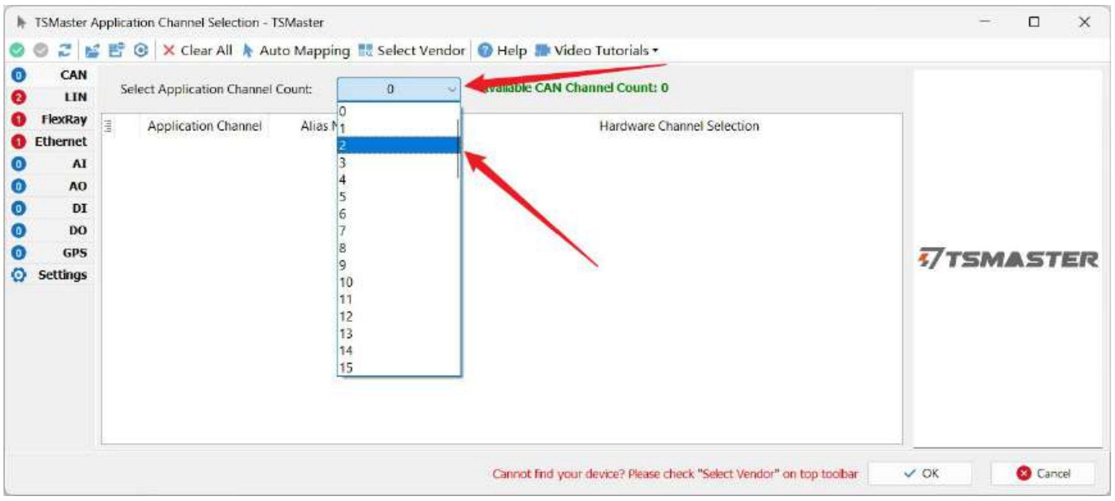
[✓ Check All] Check Default | Video Tutorials

Please select vendors during device detection, double-click a vendor to open its driver download page.
TOSUN will do it's utmost to add support for available vendors.

✓ TOSUN	TOSUN Devices (https://www.tosunai.com)
✓ VECTOR	Vector Devices (https://download.vector.com/drivers/Vector_Driver_Setup.zip)
☐ Ixxat	IXXAT Devices (https://hmsnetworks.blob.core.windows.net/nlw/docs/default-source/products/ixxat/monitored/pc-interface-cards/vci-v4_0_1187_0-window
☐ PEAK	PEAK USB Devices (https://www.peak-system.com/fileadmin/media/files/PEAK-System_Driver-Setup.zip)
☐ KVASER	Kvaser Devices (https://www.kvaser.com/download/)
☐ ZHIYUAN Electronics	ZLG Devices (https://www.zlg.cn/can/down/down/id/22.html)
☐ INTREPID	IntrepidCS Devices (https://www.intrepidcs.net.cn/softwaredownload/)
☐ CANable	CANable Devices (https://github.com/candle-usb/candleLight_fw)

2. The number of channels has not been set.

If the driver has been installed correctly and the hardware driver has been enabled, but you still cannot see the device in the hardware panel, please check if the number of channels has been set. If the CAN Channel Count is set to 0, it is of course that no online hardware can be displayed.

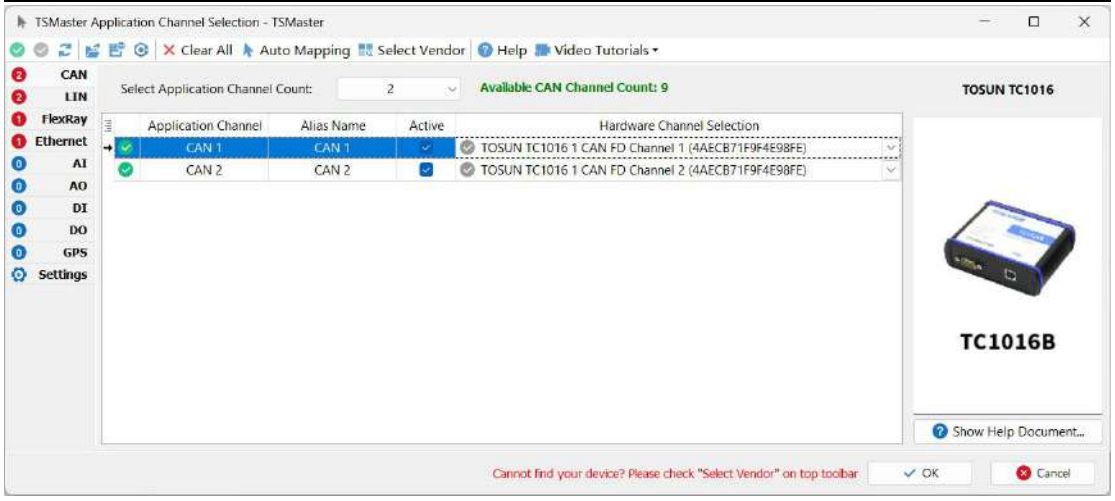

TSMaster Application Channel Selection - TSMaster

Select Application Channel Count: 0

Available CAN Channel Count: 0

Cannot find your device? Please check "Select Vendor" on top toolbar

The solution is to set the number of channels and map the channels to the physical channels of the actual CAN board. The principle can be found in the "Hardware Channel Mapping" section, as shown below:


TSMaster Application Channel Selection - TSMaster

Select Application Channel Count: 2

Available CAN Channel Count: 9

Application Channel	Alias Name	Active	Hardware Channel Selection
CAN 1	CAN 1	✓	TOSUN TC1016 1 CAN FD Channel 1 (4AECB71F9F4E98FE)
CAN 2	CAN 2	✓	TOSUN TC1016 1 CAN FD Channel 2 (4AECB71F9F4E98FE)

TOSUN TC1016

TC1016B

Cannot find your device? Please check "Select Vendor" on top toolbar

1.5. Measurement Setup

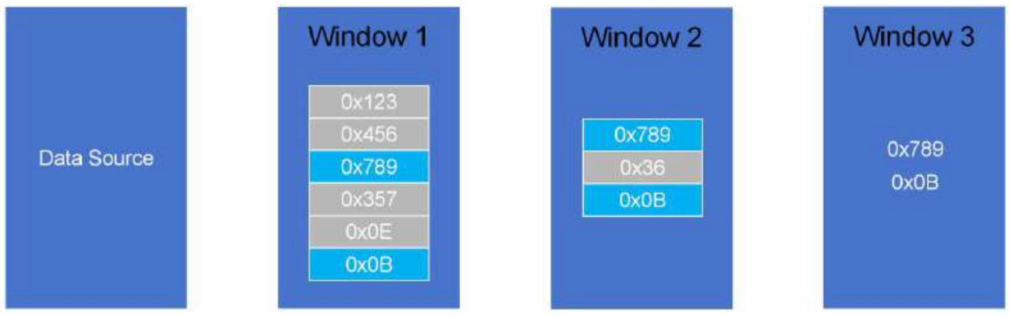
1.5.1. Feature Description

The Measurement Setup window mainly includes three functions:

- Provide a panel where users can quickly create functional module forms as needed.
- The Measurement Setup module summarizes all forms within the entire project. Users can quickly access the target form through Measurement Setup.
- To create a combination of data streams and implement data stream filtering.

1.5.2. Data Stream Filtering

TSMaster provides a Measurement Setup window, within which filtering can be achieved by combining and configuring the directions of data streams. The basic concept is as follows: data streams flow through a defined "gate" or window, and only the data contained within this window is allowed to pass through; other data is blocked. As shown in below figure:



1. The data contained in Window 1 includes: 0x123, etc. Only data with IDs included in this list is allowed to pass through to Window 2.

2. The data contained in Window 2 includes: 0x789, etc. Only data with IDs included in this list is allowed to proceed to the next Window 3.

3. After being filtered by the first two windows, the only message IDs that can reach the next window (window 3) are 0x789 and 0x0B.

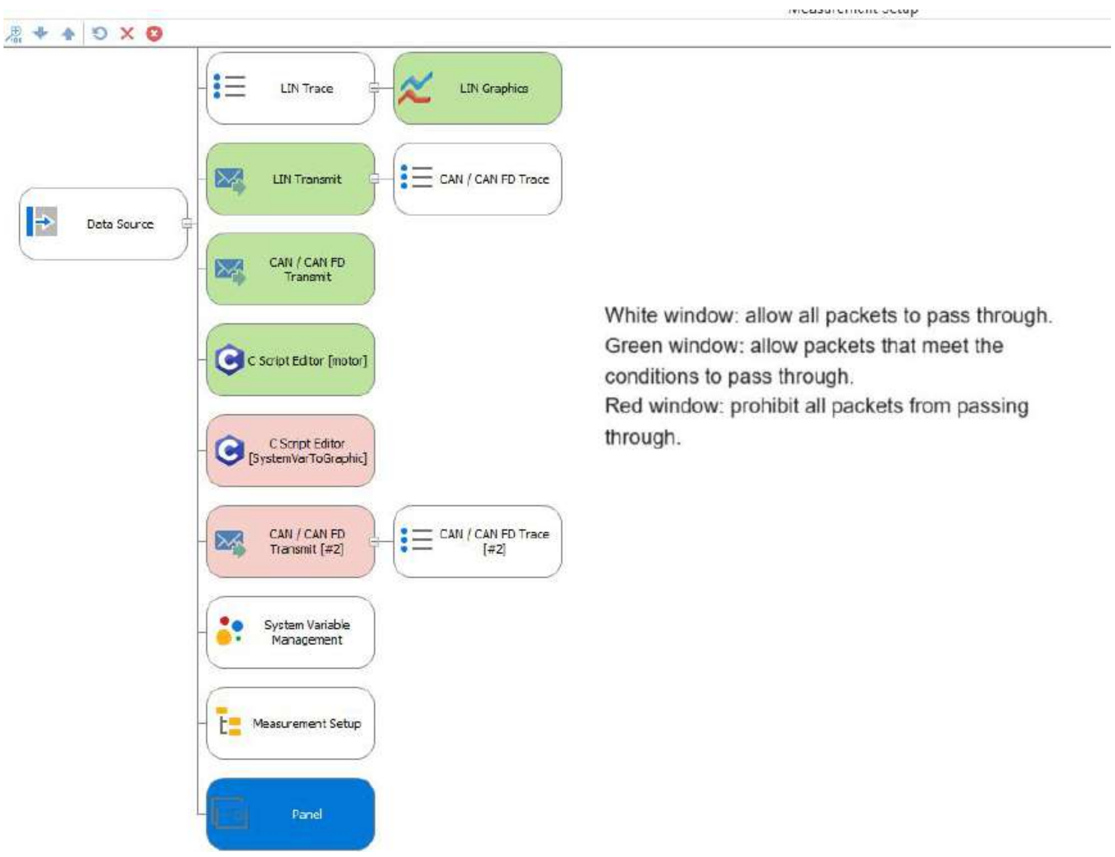
In the Measurement Setup panel, the filtering capabilities of the windows are categorized into three types, which are identified by colors:

1. White window: allow all data to pass through.

2. Green window: allow data that meets the criteria to pass through.

3. Red window: block all data to pass through.

As shown in the figure below:



White window: allow all packets to pass through.
Green window: allow packets that meet the conditions to pass through.
Red window: prohibit all packets from passing through.

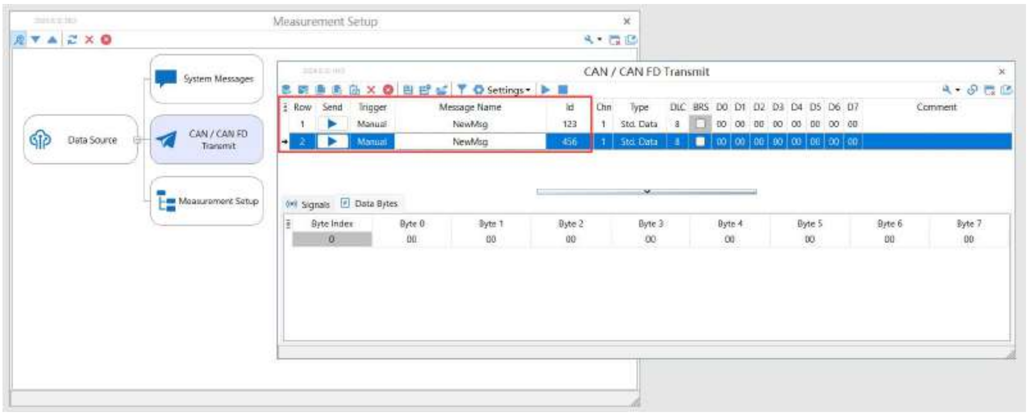
Windows to allow/block all data to pass through are relatively straightforward to understand. The mechanism for the green windows will be explained in details.

1.5.2.1. Allow Partial Data to Pass Through

Essence: Extract the information from the messages contained within the window (such as the number of messages, their respective IDs, and channel numbers), and then filter based on a combination of pass criteria and message IDs.

1.5.2.1.1. CAN/CAN FD Transmit Window as an Example

➢ When the window contains messages, it looks something like this:

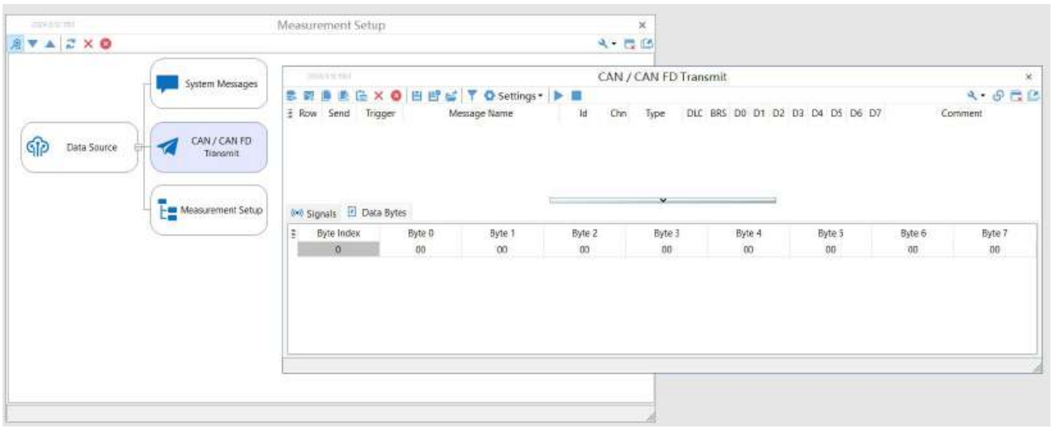


Row	Send	Trigger	Message Name	Id	Chn	Type	DLC	BRS	D0	D1	D2	D3	D4	D5	D6	D7	Comment
1	▶	Manual	NewMsg	123	1	Std. Data	8	☐	00	00	00	00	00	00	00	00	
2	▶	Manual	NewMsg	456	1	Std. Data	8	☐	00	00	00	00	00	00	00	00	

Byte Index	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0	00	00	00	00	00	00	00	00

At this point: The subsequent window is able to receive messages from channel 1 with a message ID equal to 0x123 or 0x456.

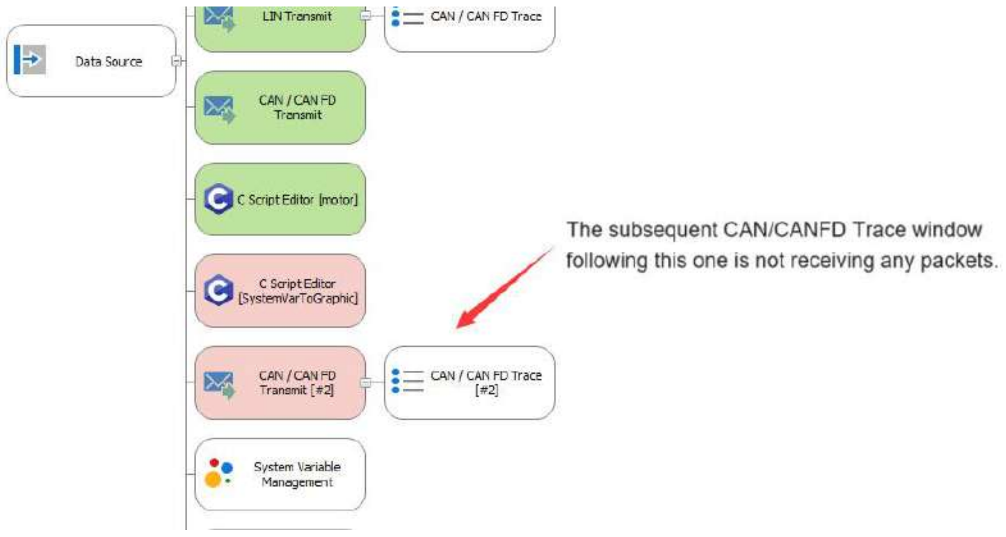
➢ After deleting all the messages in the send window, it looks like this:



Row	Send	Trigger	Message Name	Id	Chn	Type	DLC	BRS	D0	D1	D2	D3	D4	D5	D6	D7	Comment

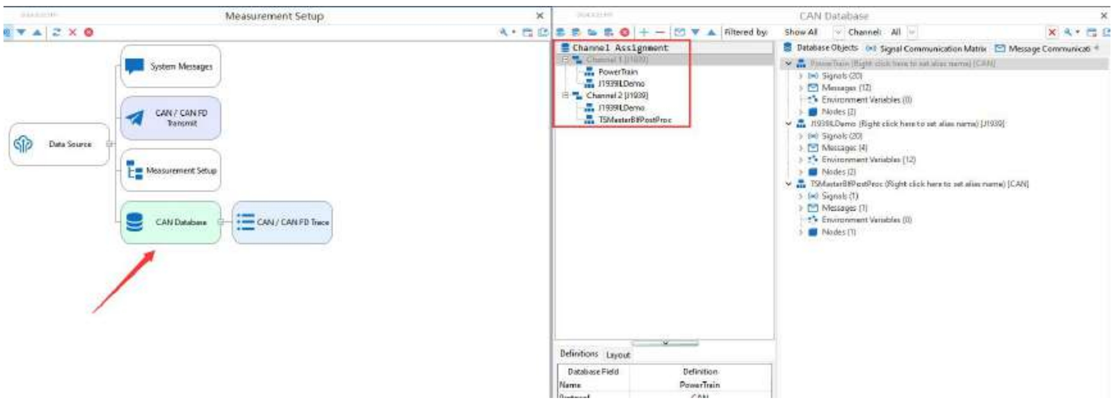
Byte Index	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0	00	00	00	00	00	00	00	00

➢ Modules following this window will not receive any messages, as shown below:


LIN Transmit
CAN / CAN FD Trace
Data Source
CAN / CAN FD Transmit
C Script Editor [motor]
The subsequent CAN/CANFD Trace window following this one is not receiving any packets.
C Script Editor [SystemVarToGraphic]
CAN / CAN FD Transmit [#2]
CAN / CAN FD Trace [#2]
System Variable Management

1.5.2.1.2. DBC Window as an Example

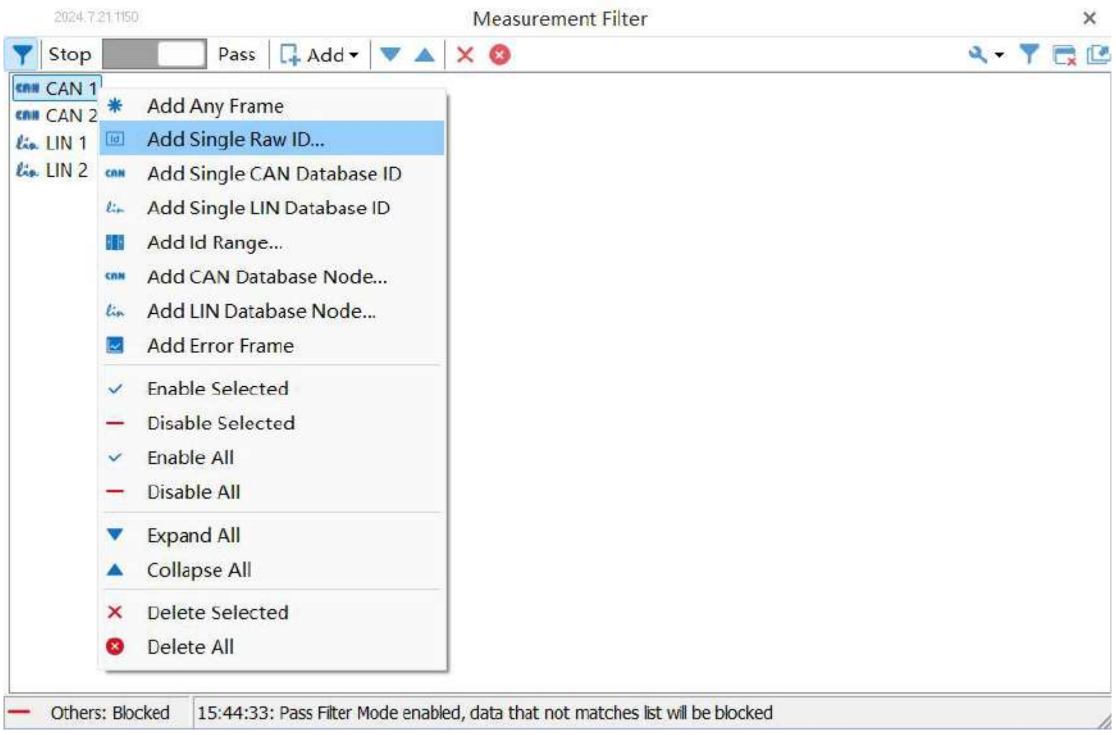
The CAN Database form uses the added DBC file as a filter condition. It filters based on the channel and the ID bound in the channel binding within the DBC file.



As shown above: If the channel and ID of a message are included in the database added in the CAN DataBase window, then that message will be allowed to pass through this window, and subsequent windows will be able to receive it. Messages that are not included in the database will be prohibited from passing through this window.

1.5.2.2.Measurement Filter Module

Measurement Filter, as a dedicated module for filtering, provides even more flexible window filtering capabilities. As shown below:



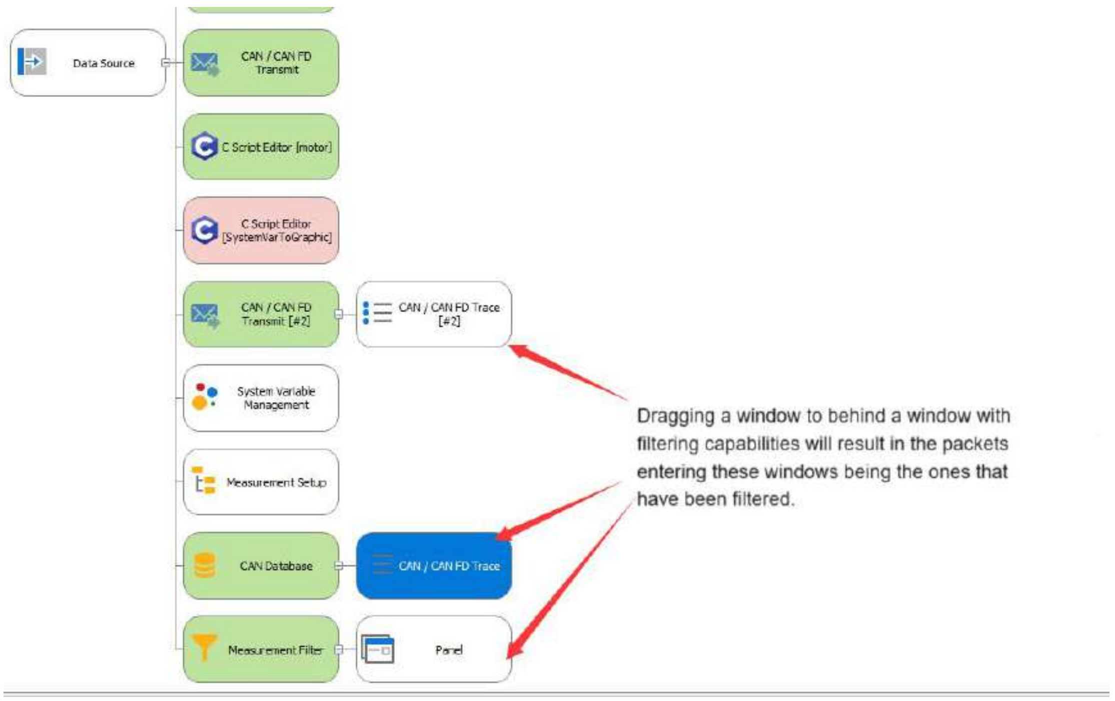
15:44:33: Pass Filter Mode enabled, data that not matches list will be blocked

1. Select the channel for filtering.
2. Add the filter conditions such as any frame/single raw ID/single CAN database ID, etc.
3. Select the Stop/Pass filter mode.

1.5.3. Enable/Disable Filtering Conditions

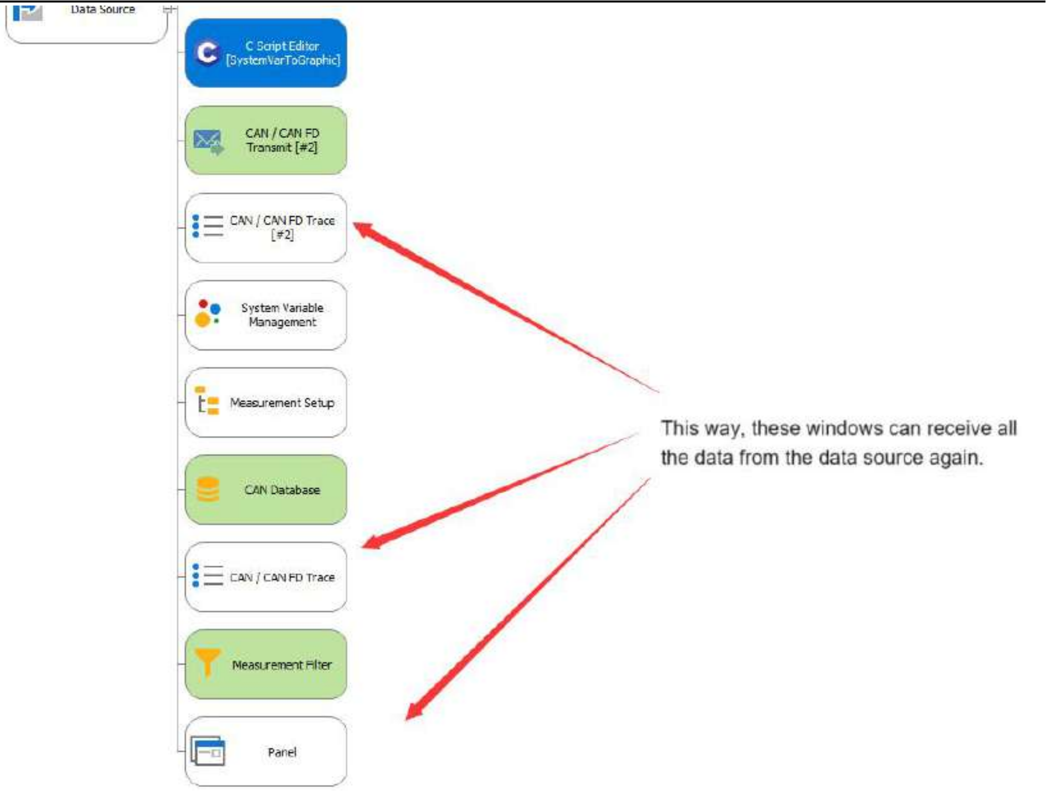
After configuring the filtering conditions in the Measurement Setup window, TSMaster also provides a flexible mechanism to enable/disable these filtering conditions. This can be achieved by adding associations between windows through drag-and-drop, thereby enabling or disabling these filtering conditions.

1.5.3.1.Enable Filtering Conditions


Data Source
CAN / CAN FD Transmit
C Script Editor [motor]
C Script Editor [SystemVarToGraphic]
CAN / CAN FD Transmit [#2]
CAN / CAN FD Trace [#2]
System Variable Management
Measurement Setup
CAN Database
CAN / CAN FD Trace
Measurement Filter
Panel
Dragging a window to behind a window with filtering capabilities will result in the packets entering these windows being the ones that have been filtered.

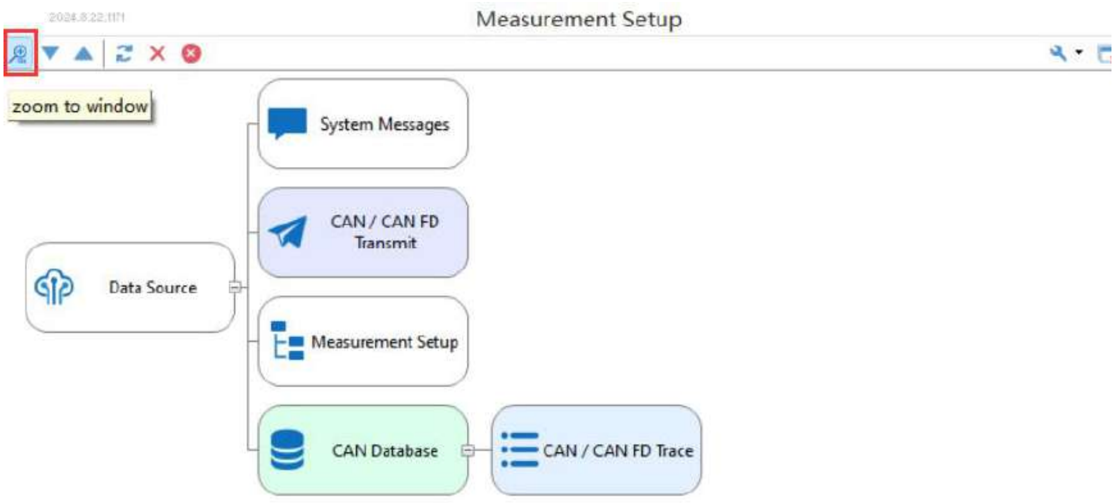
1.5.3.2.Disable Filtering Conditions

Instead of modifying the filtering condition window, simply dragging these windows away from behind the filtering window can disable the filtering conditions.


Data Source
C Script Editor [SystemVarToGraphic]
CAN / CAN FD Transmit [#2]
CAN / CAN FD Trace [#2]
System Variable Management
Measurement Setup
CAN Database
CAN / CAN FD Trace
Measurement Filter
Panel
This way, these windows can receive all the data from the data source again.

1.5.4. Window Scaling

The Measurement Setup window supports automatic scaling and normal display functions, as shown below:



➢ Auto-scaling mode (zoom to window button is pressed): The module automatically adjusts its size based on the dimensions of the parent window. Advantages: all modules can be displayed on the panel. **Note**: When too many modules are added, the size of each module becomes smaller.

- Normal display mode (zoom to window button is not pressed): The module maintains its original design size at all times. Advantages: The size of the module remains unchanged, preventing it from becoming too small due to the addition of too many modules, which can make operation difficult. **Note**: When too many modules are added, some of them may be obscured by the window edges and require scrolling or dragging a slider to view.

1.5.5. Application Examples

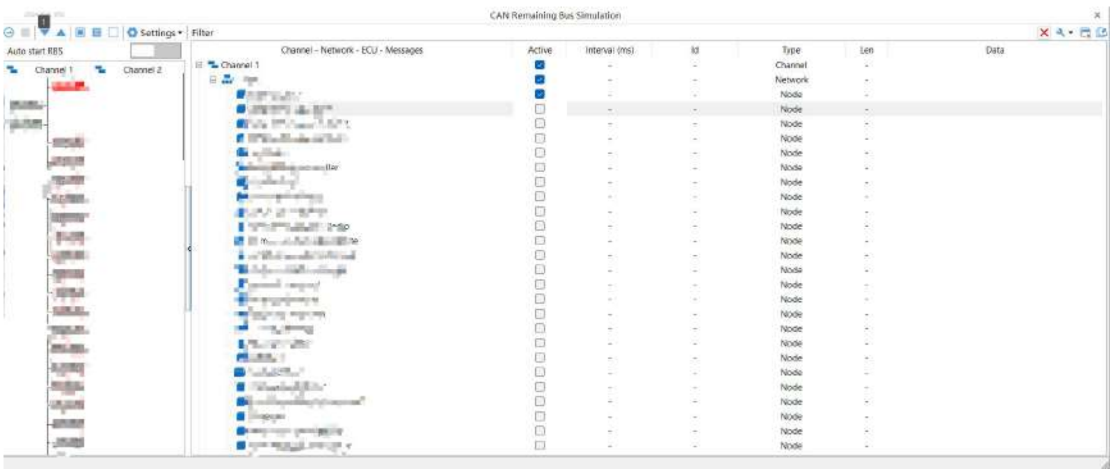
Only need to receive channel 1.

1.6. Residual Bus Simulation (RBS)

1.6.1. RBS Simulation

The full name of RBS is Residual Bus Simulation, which refers to the so-called residual bus simulation. It is primarily based on in-vehicle network databases such as CAN/LIN/FlexRay/Ethernet databases, simulating the communication behavior of each node within that network.

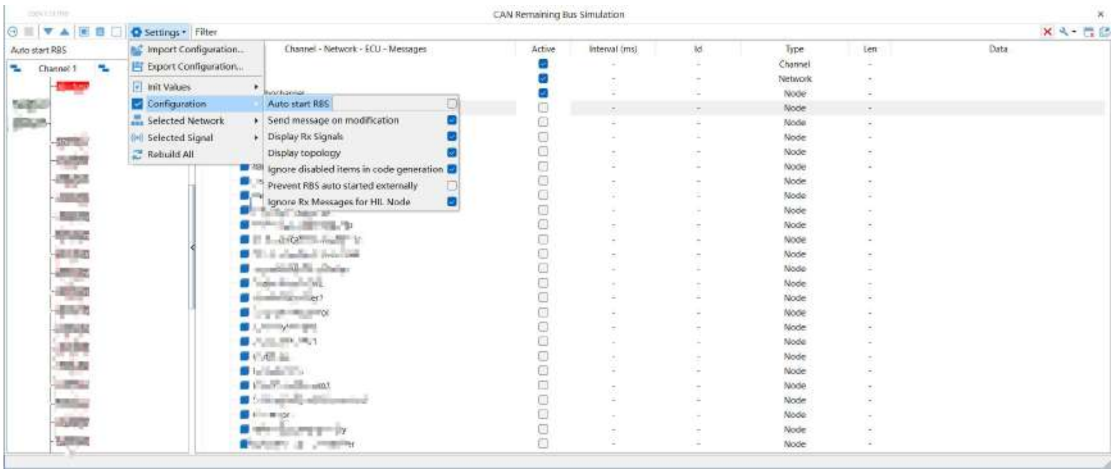
In TSMaster, if the user wants to start RBS, they can follow the path: Simulation -> select the corresponding RBS module, and then enter the corresponding RBS parameter configuration interface. As shown in the figure below, Channel 1 is “Active” by default, and the node under Channel 1 is the database node with an ECU node showing “Active”:



The RBS simulation engine is a standalone module. If the user selects the options of RBS Channel -> Database Node -> Controller Node, TSMaster will initiate the simulation for that node immediately upon clicking “Active”. If all of the aforementioned checkboxes are ticked, it indicates that the database node within channel x is selected for simulation. Therefore, as soon as the user clicks “Active”, the CAN RBS simulation will be initiated, and message data will appear on the bus.

1.6.2. Configuration Items

The RBS module in TSMaster primarily includes the following configuration items:



- Auto start RBS: If enabled, the residual bus simulation module will automatically start when connecting the application.

- Send message on modification: If enabled, the message containing the modified signal will be immediately sent upon modification of the signal.

- Display topology: Show a schematic diagram of the network topology on the interface.

- Ignore disabled items in code generation.

- Prevent RBS auto started externally: When the RBS module is called by Panel or other components, if this option is not selected, the RBS module will be started automatically by external modules that utilize RBS, even if the RBS module has not been explicitly opened. If this option is selected, the RBS module will only start when explicitly initiated by the user, even if it is called by external modules.

- Ignore Rx Messages for HIL nodes.

1.6.3. Clarification

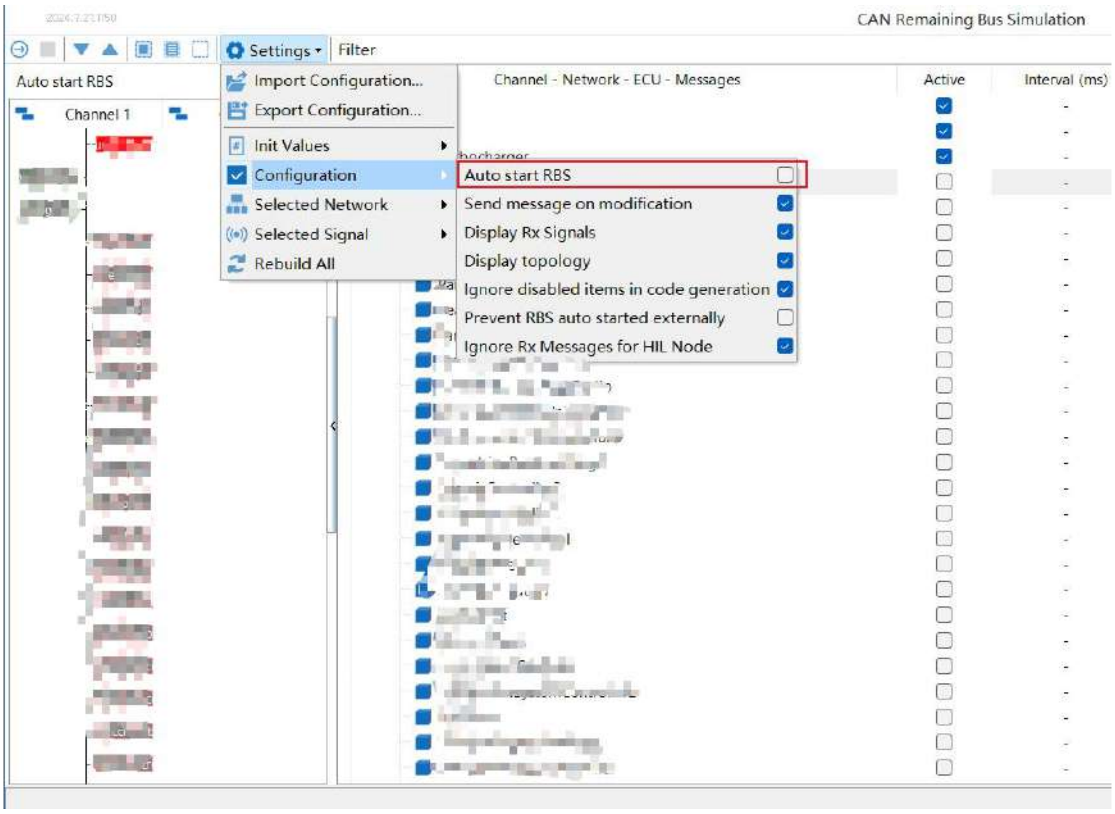
1.6.3.1. Why is the RBS Module Automatically Activated

1.6.3.1.1. User has Configured Automatic Startup

When a user selects the option to automatically start simulation nodes in the settings, and chooses the specific nodes to be simulated, the program will automatically initiate the corresponding network node simulation upon connection. This allows the user to observe the simulated messages in the Trace.

Solution:

In the configuration, do not select the option "Auto start RBS", as shown in the figure below:



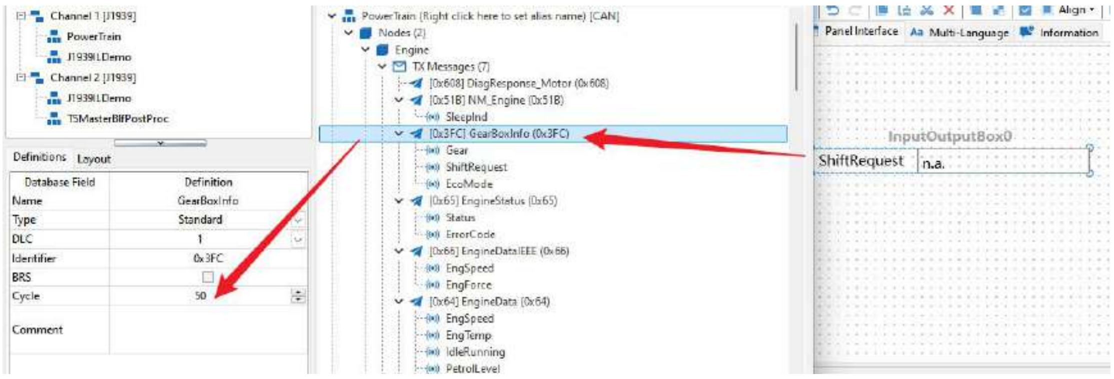
Auto start RBS ☐
Send message on modification ☑
Display Rx Signals ☑
Display topology ☑
Ignore disabled items in code generation ☑
Prevent RBS auto started externally ☐
Ignore Rx Messages for HIL Node ☑

1.6.3.1.2. CANRBS Simulation has been Activated through the Panel

This situation is the most covert and difficult for users to pinpoint the reason why messages are sent instantly upon clicking “Active”, there are several key premises to consider:

1. When the panel is associated with CAN messages, it operates based on the CAN RBS (Residual BUS Simulation) module. This allows for a realistic simulation of the bus environment.

2. Input controls have been added to the panel. If these input controls are associated with periodic messages in the DBC, CANRBS needs to promptly prepare this periodic message when the panel is launched. This is because any update by the user to this periodic signal on the panel requires that CANRBS promptly update and process the value of this message.



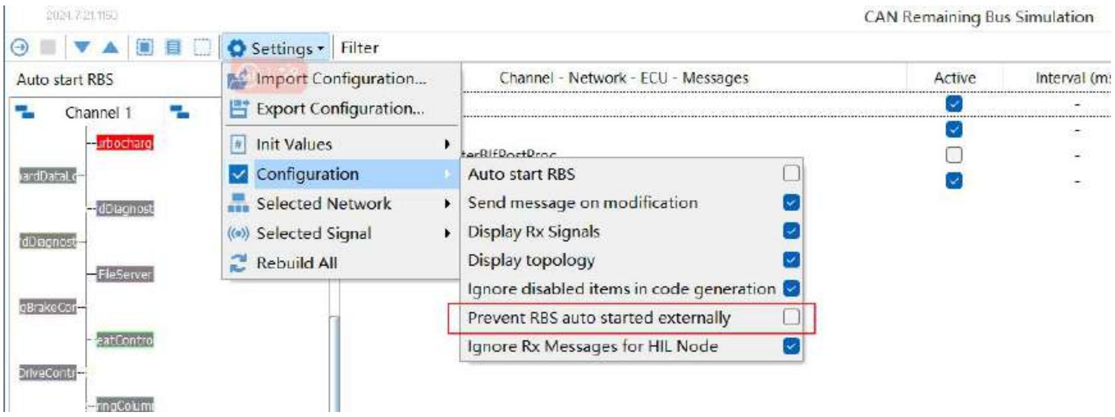
Database Field	Definition
Name	GearBoxInfo
Type	Standard
DLC	1
Identifier	0x3FC
BRS	☐
Cycle	50
Comment	

Why is the RBS module invoked by an external module even when the user has not set it to automatically start or manually started it ?

This is because the operation of the external module relies on the RBS module. If the RBS module is not automatically started alongside the external module, users may be unaware and suspect there is an issue with the software. Therefore, by default, in TSMaster, if an external module such as a Panel utilizes the RBS module, the RBS module will automatically start when the Panel is run.

Solution:

Prevent RBS auto started externally. As shown below:



Auto start RBS ☐
Send message on modification ☑
Display Rx Signals ☑
Display topology ☑
Ignore disabled items in code generation ☑
Prevent RBS auto started externally ☐
Ignore Rx Messages for HIL Node ☑

Users should cautiously disable this option. Otherwise, it would be unusual if the corresponding RBS module does not start when the Panel is opened.

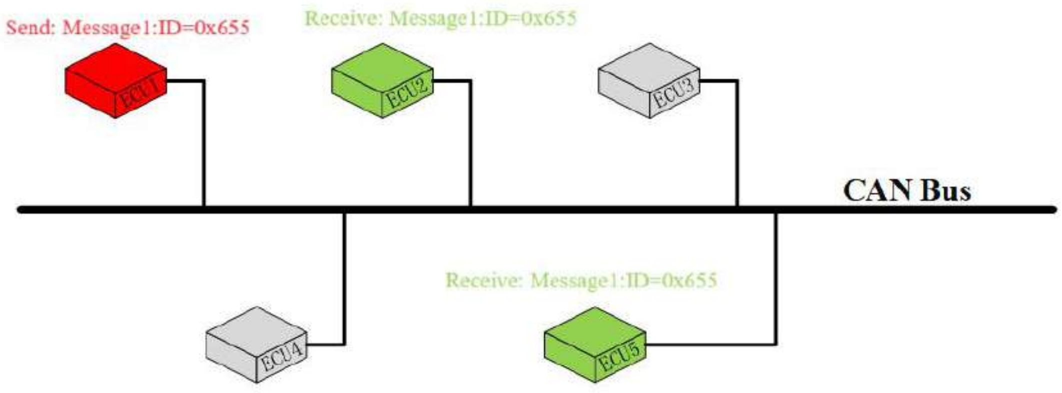
1.6.3.2. Why is it Ineffective to Read/Write RBS Node Signals

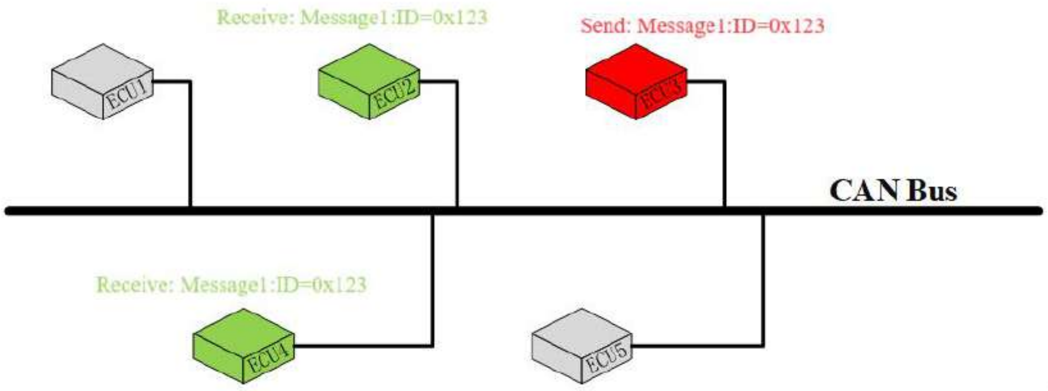
The CAN RBS (Residual BUS Simulation) simulation is rigorously conducted based on the network configuration defined in the network database. As shown in the figure below, this CAN network defines 5 ECU nodes, from ECU1 to ECU5. Each node has its own CAN messages for sending and receiving. Therefore, for a given message, there is a corresponding sending node and one or more receiving nodes. Normally, in a CAN network, a message has a single sending node and one or more receiving nodes. As shown below:

- For the message with ID = 0x655, ECU1 is the sending node, and ECU2 and ECU5 are the receiving nodes. For the other two nodes, this message is irrelevant and will be filtered out.

- For the message with ID = 0x123, ECU3 is the sending node, and ECU2 and ECU4 are the receiving nodes. The message is irrelevant to the other two nodes and will be filtered out.

Based on the network planning logic illustrated above. For instance, concerning signals within the message with ID=0x655, during the RBS simulation process, if a user modifies signals inside the message with ID = 0x655 located in ECU2 or ECU5, which act as receiving nodes, even after modifying the signal, there will be no opportunity for this signal to be transmitted onto the bus. Consequently, this leads to a scenario where changes to the signal become ineffective after modification.

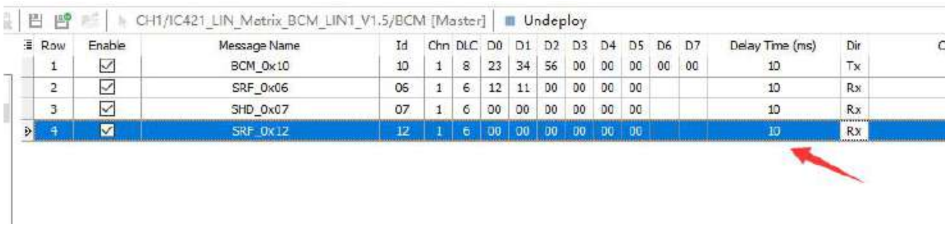




During RBS simulation, if a particular ECU node is not activated, the bus data internal to that ECU node will not be updated. Therefore, when reading signals from within this node, the data retrieved will not reflect the current valid data on the actual bus, resulting in invalid signal values being read. Therefore, when reading signals from RBS nodes, it is essential to confirm whether the node is indeed active on the RBS bus.

1.7. Transmission Window

1.7.1. LIN Transmission Window



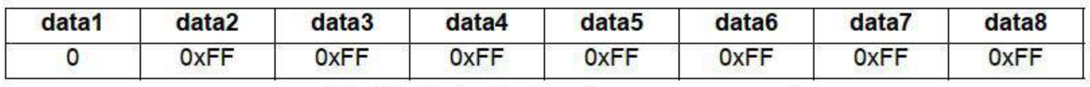
CH1/IC421_LIN_Matrix_BCM_LIN1_V1.5/BCM [Master] | Undeploy

Row	Enable	Message Name	Id	Chn	DLC	D0	D1	D2	D3	D4	D5	D6	D7	Delay Time (ms)	Dir
1	✓	BCM_0x10	10	1	8	23	34	56	00	00	00	00	00	10	Tx
2	✓	SRF_0x06	06	1	6	12	11	00	00	00	00			10	Rx
3	✓	SHD_0x07	07	1	6	00	00	00	00	00	00			10	Rx
4	✓	SRF_0x12	12	1	6	00	00	00	00	00	00			10	Rx

Delay Time: It refers to the time interval between LIN messages, which is measured after a message starts to be sent, rather than the interval after the message has finished being sent. Therefore, the interval cannot be set too small. If it is set too small, it may result in the situation where the previous frame of data has not been successfully sent before the next frame starts to be sent, causing the previous data to be overwritten. If all the intervals are set too small, it may lead to all messages being overwritten before they can be successfully sent. On the one hand, this would mean that no complete messages are transmitted on the bus; on the other hand, the LIN Trace window on the host computer would not display any messages.

1.7.1.1.Send Sleep Frame

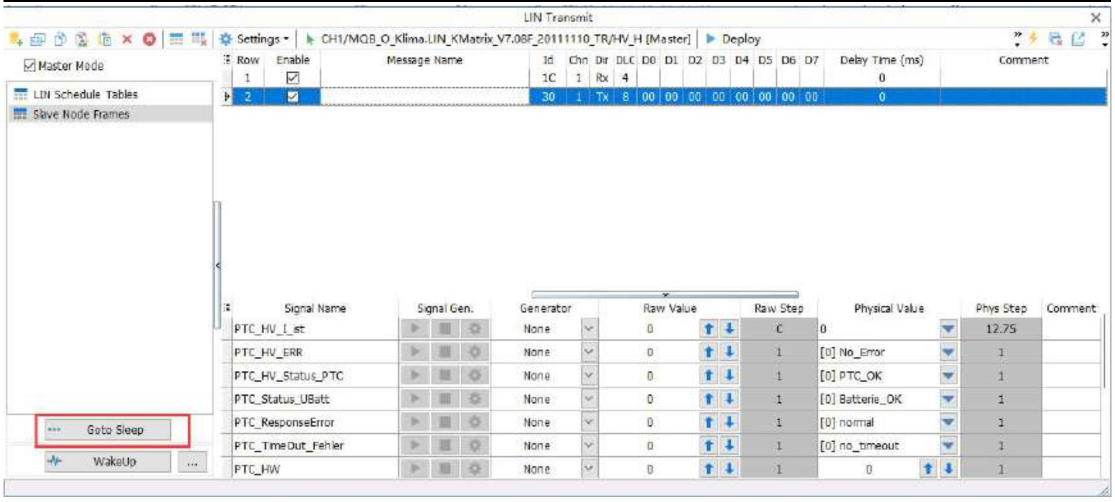
According to the LIN 2.1 protocol specification, the sleep frame must be initiated by the master node, and the content of the sleep frame is as follows:



data1	data2	data3	data4	data5	data6	data7	data8
0	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

Table 2.1: Go to sleep command

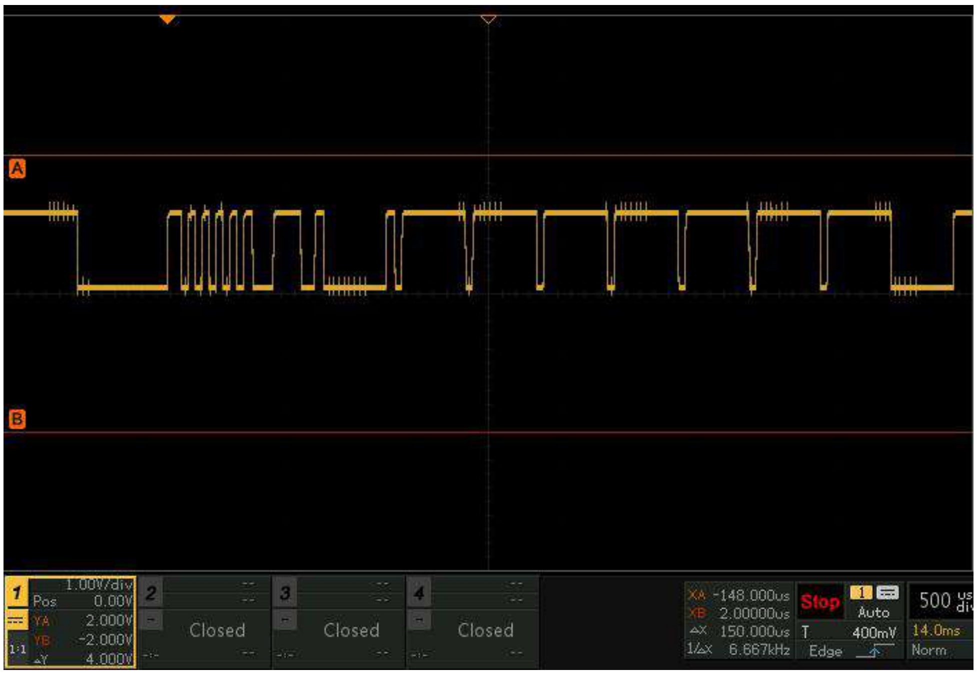
In the TSMaster LIN transmission interface, access the LIN transmit sending interface, which looks like the following:



Row	Enable	Message Name	Id	Chn	Dir	DLC	D0	D1	D2	D3	D4	D5	D6	D7	Delay Time (ms)	Comment
1	☑		1C	1	Rx	4									0	
2	☑		30	1	Tx	8	00	00	00	00	00	00	00	00	0	

Signal Name	Signal Gen.	Generator	Raw Value	Raw Step	Physical Value	Phys Step	Comment
PTC_HV_I_st		None	0	C	0	12.75	
PTC_HV_ERR		None	0	1	[0] No_Error	1	
PTC_HV_Status_PTC		None	0	1	[0] PTC_OK	1	
PTC_Status_UBatt		None	0	1	[0] Batterie_OK	1	
PTC_ResponseError		None	0	1	[0] normal	1	
PTC_TimeOut_Fehler		None	0	1	[0] no_timeout	1	
PTC_HW		None	0	1	0	1	

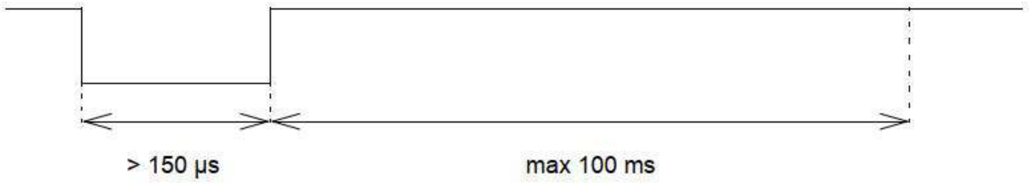
Select the node and channel, then click the "Goto Sleep" button. Observe the message through the oscilloscope as shown below:



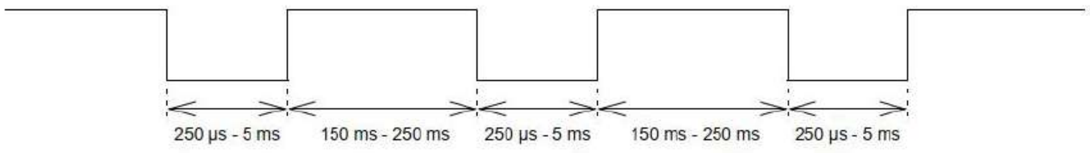
After receiving the above message from node, enters sleep mode.

1.7.1.2.Send Wake-up Frame

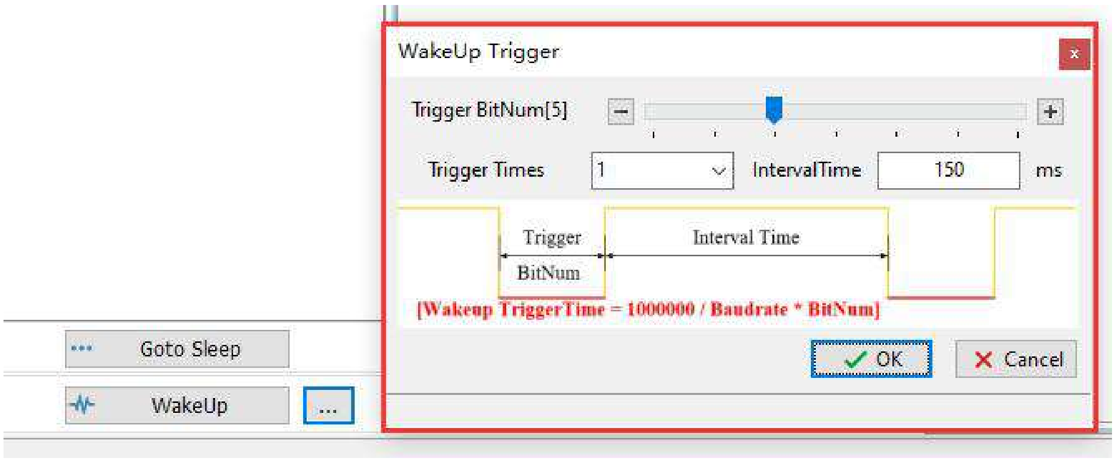
According to the LIN 2.1 specification, when a node is in the sleep state, it will be woken up by receiving a waveform with a duration exceeding 150 microseconds. After being woken up, it will enter the initialization state and then resume its working state. Taking a node with a baud rate of 20k as an example, when it receives a waveform exceeding 150 microseconds, it will be woken up and enter the working state.



When conducting actual wake-up tests, it is necessary to implement adjustable wake-up message lengths, and some nodes require multiple wake-up messages to be sent at intervals before they can be successfully woken up, as shown in the following figure:



To meet the above requirements, TSMaster provides a WakeUp signal configuration parameter module, as shown in the following figure:


WakeUp Trigger

Trigger BitNum[5]

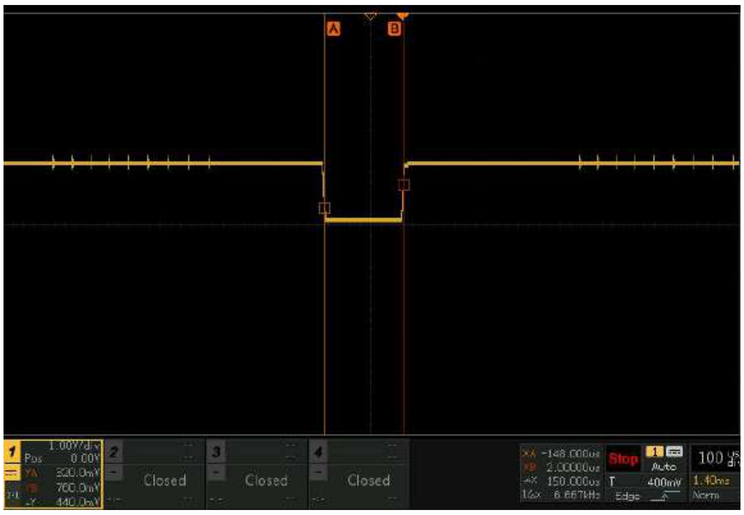
Trigger Times: 1

IntervalTime: 150 ms

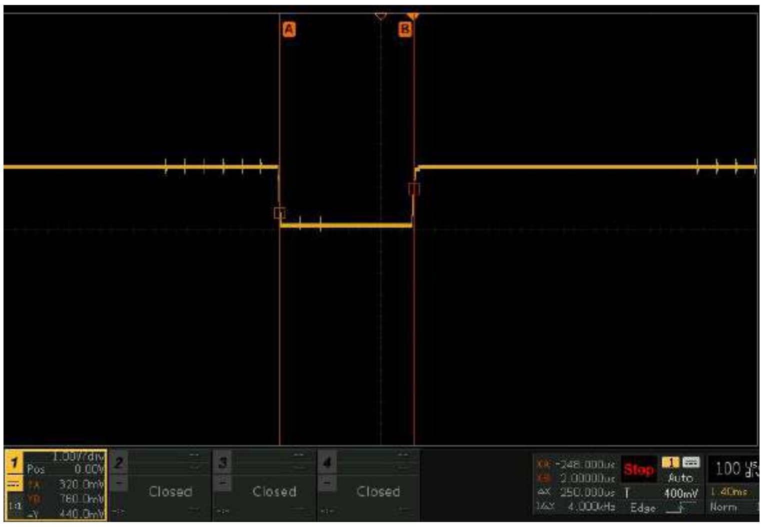
[Wakeup TriggerTime = 1000000 / Baudrate * BitNum]

It mainly includes the following configuration parameters:

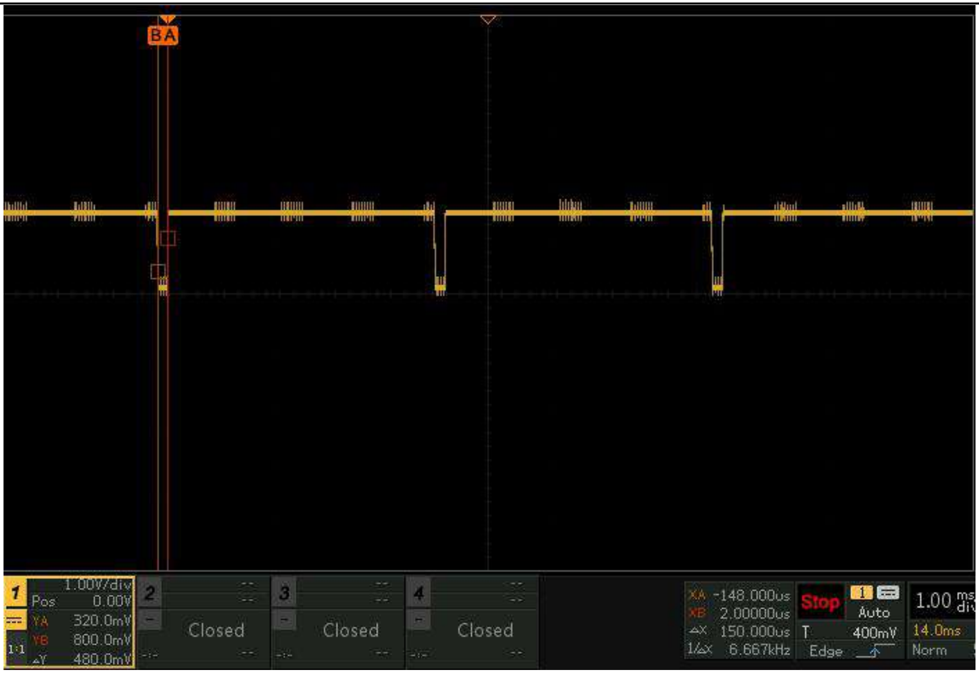
- Trigger BitNum: Wake-up Message Trigger Level Duration = 1000000/Baudrate * (BitNum). For example, if the baud rate is 20k and BitNum is set to 3, then the trigger level width would be 150 microseconds. As shown in the following figure



When setting BitNum = 5, the trigger level width would be 250 microseconds, as shown in the following figure:



When setting Trigger Times equal to 3 and the Interval Time to 3ms, TSMaster will transmit multiple wake-up message frames with an interval of 3ms between each message, as shown in the following figure. It initially sends a wake-up message (250us level), waits for 3ms, and then sends another wake-up message, repeating this process a total of three times in a loop:

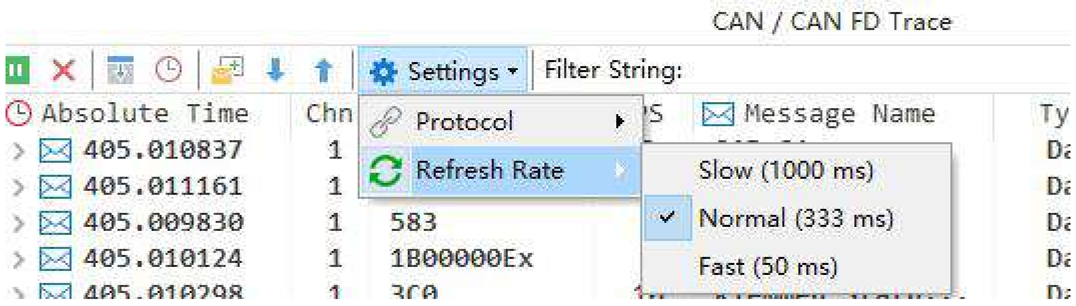


Send 3 Consecutive Wake-up Signals

1.8. Trace Window

1.8.1. Set Display Refresh Rate

To reduce the CPU usage by the software, the Trace window provides several display refresh rates for users to choose from. As shown below:

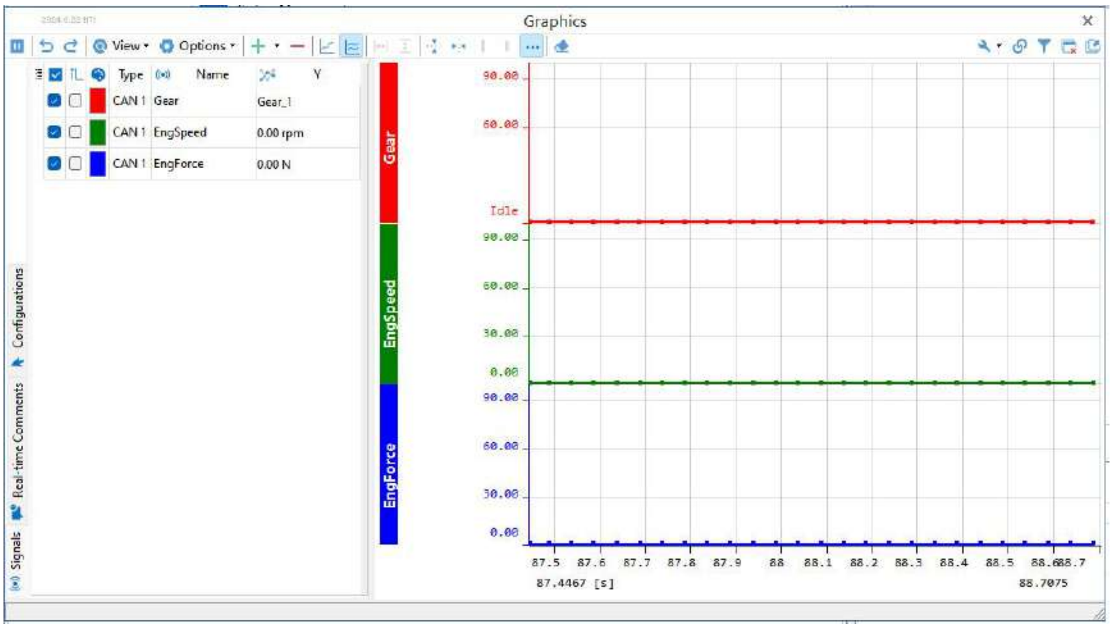


```
405.010837  1
405.011161  1
405.009830  1  583
405.010124  1  1B00000Ex
405.010298  1  3C0
```

For some older computers, you can choose a lower refresh rate to reduce the load on the computer.

1.8.2. Close Curve and Point Selection Function

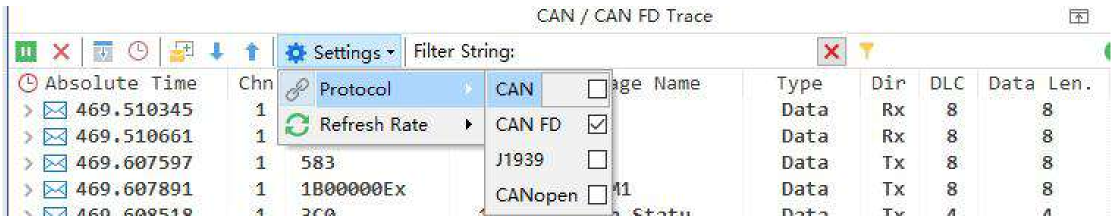
In TSMaster's Trace window, if several curves are very close together and it's difficult to select one with the mouse, TSMaster provides a function to switch between curves by simply use the mouse to click different curves.



1.8.3. Set Display Message Format

CANTrace supports displaying messages in formats such as CAN, CANFD, J1939, CANOpen, 15765-2, 15765-3, CCP, XCP, etc. Users simply need to switch to the corresponding display format as shown below:

For module settings, the operation path is always: Settings -> Protocol.



```
Absolute Time   Chn              ...age Name   Type  Dir  DLC  Data Len.
469.510345       1                             Data  Rx    8     8
469.510661       1                             Data  Rx    8     8
469.607597       1   583                       Data  Tx    8     8
469.607891       1   1B00000Ex        ...1     Data  Tx    8     8
```

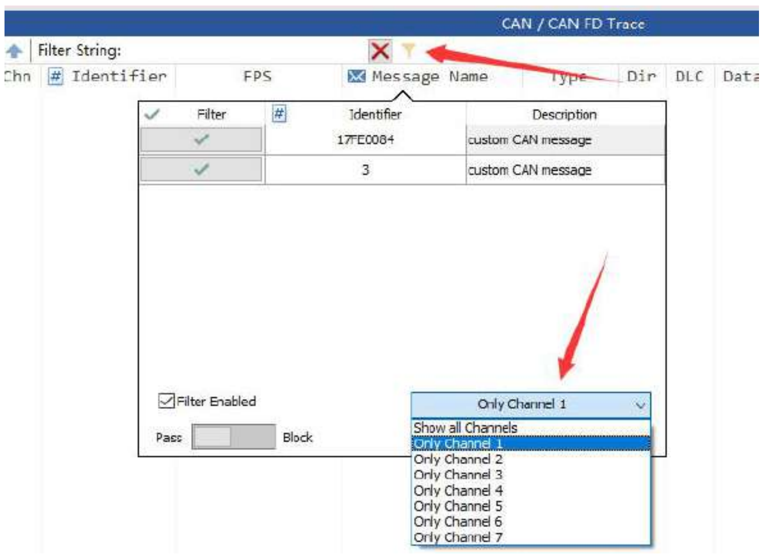
1.8.4. Message Filtering

In the Trace window, TSMaster includes three types of filtering: based on data channel, based on message ID, and based on signal value range.

1.8.4.1. Based on Message Channel

Users who use multi-channel CAN devices can choose to only view messages from the channels that they are interested in. The setup method is as follows:

Filter Enabled-> Select Channel, the channels can be selected are as shown below:



CAN / CAN FD Trace

Filter String:

Chn | Identifier | FPS | Message Name | Type | Dir | DLC | Data

Filter	Identifier	Description
✓	17FE0084	custom CAN message
✓	3	custom CAN message

Filter Enabled

Pass Block

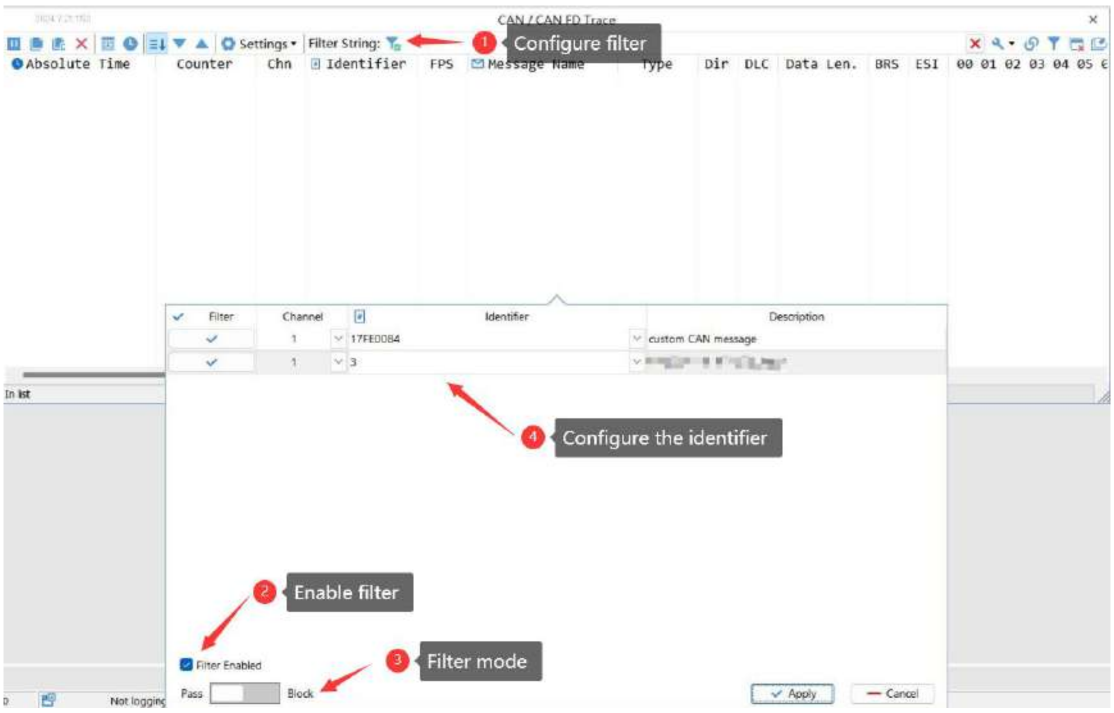
Only Channel 1

Show all Channels
Only Channel 1
Only Channel 2
Only Channel 3
Only Channel 4
Only Channel 5
Only Channel 6
Only Channel 7

If the user selects OnlyChannel1, only data from Channel 1 will be displayed in the Trace window.

1.8.4.2.Filter Based on Message ID

The configuration steps are as follows:



CAN / CAN FD Trace

Settings Filter String:

1 Configure filter

Absolute Time | Counter | Chn | Identifier | FPS | Message Name | Type | Dir | DLC | Data Len. | BRS | ESI | 00 01 02 03 04 05

Filter	Channel	Identifier	Description
✓	1	17FE0084	custom CAN message
✓	1	3	

4 Configure the identifier

2 Enable filter

Filter Enabled

3 Filter mode

Pass Block

Apply Cancel

Not logging

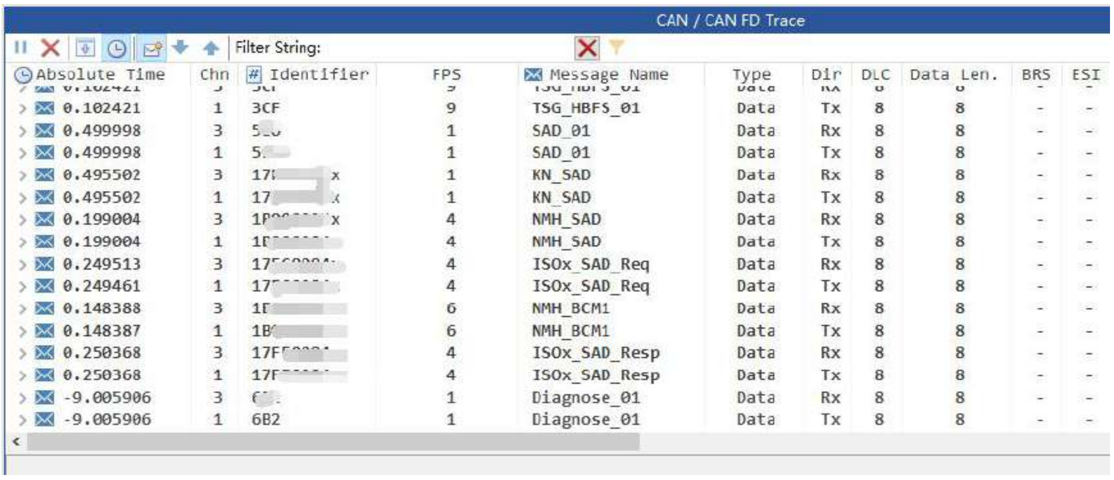
As shown in the example, “Filter Enabled” is checked and the filter mode is “Pass”, therefore, only message wit the configured identifier will be shown in the Trace window.

1.8.4.3.Filter Based on Filter String

In the Filter String, user can directly input an ID value, and then only the messages corresponding to that ID will be displayed in the Trace window. Alternatively, user can input a signal name, and only the messages associated with that signal will be displayed. Details are as follows:

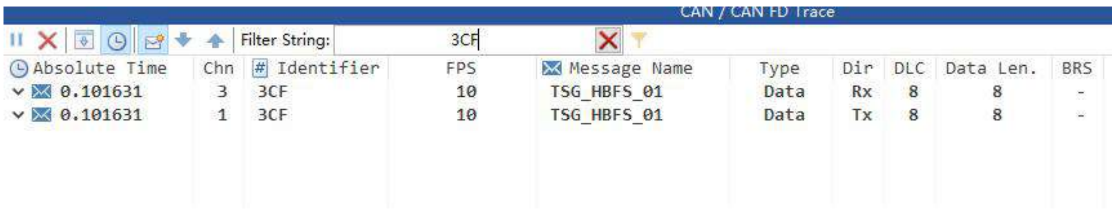
➢ FilterString Filtering by Inputting ID value:

The raw message window is shown below, containing multiple ECU communication messages, as shown in the following figure:



Absolute Time	Chn	Identifier	FPS	Message Name	Type	Dir	DLC	Data Len.	BRS	ESI
0.102421	1	3CF	9	TSG_HBFS_01	Data	Tx	8	8	-	-
0.499998	3	5[redacted]	1	SAD_01	Data	Rx	8	8	-	-
0.499998	1	5[redacted]	1	SAD_01	Data	Tx	8	8	-	-
0.495502	3	17[redacted]x	1	KN_SAD	Data	Rx	8	8	-	-
0.495502	1	17[redacted]x	1	KN_SAD	Data	Tx	8	8	-	-
0.199004	3	1[redacted]x	4	NMH_SAD	Data	Rx	8	8	-	-
0.199004	1	1[redacted]	4	NMH_SAD	Data	Tx	8	8	-	-
0.249513	3	17[redacted]	4	ISOx_SAD_Req	Data	Rx	8	8	-	-
0.249461	1	17[redacted]	4	ISOx_SAD_Req	Data	Tx	8	8	-	-
0.148388	3	1[redacted]	6	NMH_BCM1	Data	Rx	8	8	-	-
0.148387	1	1B[redacted]	6	NMH_BCM1	Data	Tx	8	8	-	-
0.250368	3	17F[redacted]	4	ISOx_SAD_Resp	Data	Rx	8	8	-	-
0.250368	1	17F[redacted]	4	ISOx_SAD_Resp	Data	Tx	8	8	-	-
-9.005906	3	[redacted]	1	Diagnose_01	Data	Rx	8	8	-	-
-9.005906	1	6B2	1	Diagnose_01	Data	Tx	8	8	-	-

In the Filter String window, by entering "3CF", the Trace window will only display messages with the ID "3CF", as shown below:



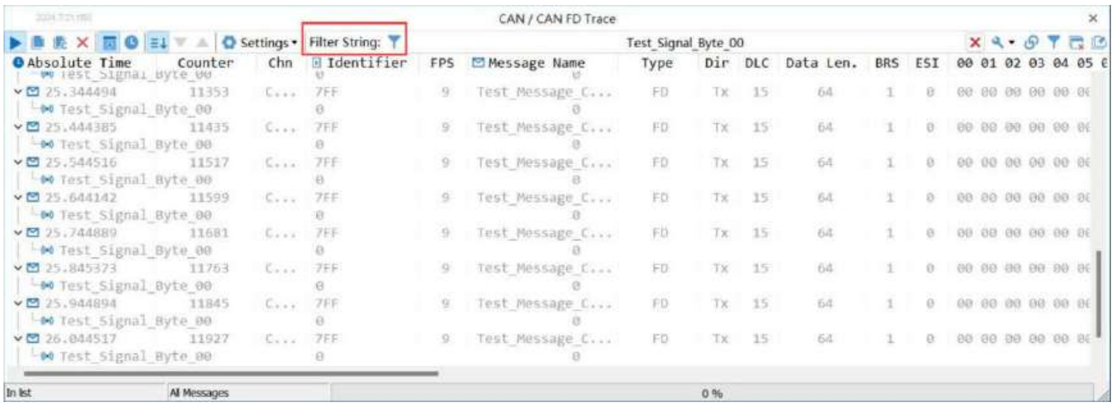
Absolute Time	Chn	Identifier	FPS	Message Name	Type	Dir	DLC	Data Len.	BRS
0.101631	3	3CF	10	TSG_HBFS_01	Data	Rx	8	8	-
0.101631	1	3CF	10	TSG_HBFS_01	Data	Tx	8	8	-

The Filter String window supports fuzzy search. If you enter the number "3" alone, the Trace window will only display messages starting with "3" as their ID.

Note: Please directly input the message ID value without any additional descriptions.

- Filter String by Inputting Signal Name:

The premise of filtering based on signals is that the database has been loaded, as the concept of signals exists only with a DBC database. This filtering method is convenient for research and development companies to view communication signals and messages in more detail. The filtering method is very simple, as shown below:


CAN / CAN FD Trace

Settings · Filter String: Test_Signal_Byte_00

Absolute Time	Counter	Chn	Identifier	FPS	Message Name	Type	Dir	DLC	Data Len.	BRS	ESI	00 01 02 03 04 05
25.344494	11353	C...	7FF	9	Test_Message_C...	FD	Tx	15	64	1	0	00 00 00 00 00
Test_Signal_Byte_00			0		0							
25.444385	11435	C...	7FF	9	Test_Message_C...	FD	Tx	15	64	1	0	00 00 00 00 00
Test_Signal_Byte_00			0		0							
25.544516	11517	C...	7FF	9	Test_Message_C...	FD	Tx	15	64	1	0	00 00 00 00 00
Test_Signal_Byte_00			0		0							
25.644142	11599	C...	7FF	9	Test_Message_C...	FD	Tx	15	64	1	0	00 00 00 00 00
Test_Signal_Byte_00			0		0							
25.744889	11681	C...	7FF	9	Test_Message_C...	FD	Tx	15	64	1	0	00 00 00 00 00
Test_Signal_Byte_00			0		0							
25.845373	11763	C...	7FF	9	Test_Message_C...	FD	Tx	15	64	1	0	00 00 00 00 00
Test_Signal_Byte_00			0		0							
25.944894	11845	C...	7FF	9	Test_Message_C...	FD	Tx	15	64	1	0	00 00 00 00 00
Test_Signal_Byte_00			0		0							
26.044517	11927	C...	7FF	9	Test_Message_C...	FD	Tx	15	64	1	0	00 00 00 00 00
Test_Signal_Byte_00			0		0							

In list · All Messages · 0 %

By clicking "Filter String" and entering the signal name (case-insensitive), Trace will automatically filter, leaving only the messages that contain the signal name, and the signal list will only show that signal.

Note:

1. Each Trace window exists independently, and corresponding filtering conditions can be set for different Trace windows without affecting each other.

2. The bus system is already operational, but the Trace window is not displaying or displaying abnormally. This may be due to previously set filtering conditions that are blocking relevant displays. Please check the settings.

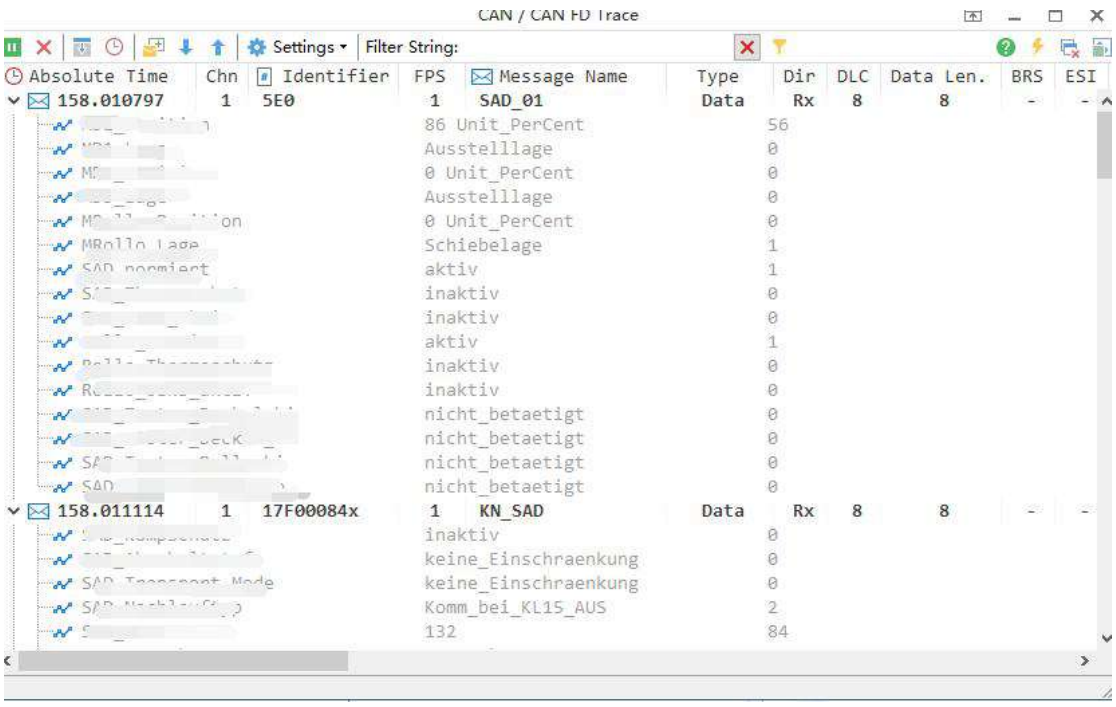
3. The filtering in the Trace window remains effective during the process of viewing log files.

1.8.4.4.Filtering Based on the Measurement Setup Window

Please check chapter: 1.5 Measurement Setup.

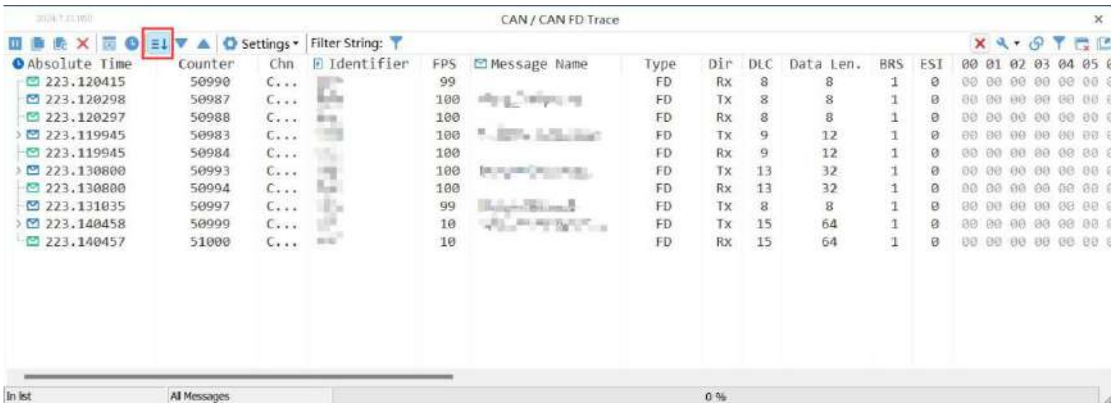
1.8.5. Clarification

1.8.5.1. When Messages are Fully Expanded by Signals, it can be Inconvenient to Observe, as Shown below:



```
CAN / CAN FD Trace
Settings ▾  Filter String:
Absolute Time   Chn  Identifier  FPS  Message Name  Type  Dir  DLC  Data Len.  BRS  ESI
158.010797      1    5E0         1    SAD_01        Data  Rx   8    8          -    -
  [redacted]          86 Unit_PerCent            56
  [redacted]          Ausstelllage               0
  [redacted]          0 Unit_PerCent             0
  [redacted]          Ausstelllage               0
  [redacted]          0 Unit_PerCent             0
  [redacted]          Schiebelage                1
  [redacted]          aktiv                      1
  [redacted]          inaktiv                    0
  [redacted]          inaktiv                    0
  [redacted]          aktiv                      1
  [redacted]          inaktiv                    0
  [redacted]          inaktiv                    0
  [redacted]          nicht_betaetigt            0
  [redacted]          nicht_betaetigt            0
  [redacted]          nicht_betaetigt            0
  [redacted]          nicht_betaetigt            0
158.011114      1    17F00084x   1    KN_SAD        Data  Rx   8    8          -    -
  [redacted]          inaktiv                    0
  [redacted]          keine_Einschraenkung       0
  [redacted]          keine_Einschraenkung       0
  [redacted]          Komm_bei_KL15_AUS          2
  [redacted]          132                        84
```

Solution: expand and drop down the signal display as follows:



```
CAN / CAN FD Trace
Settings ▾  Filter String:
Absolute Time  Counter  Chn   Identifier  FPS  Message Name  Type  Dir  DLC  Data Len.  BRS  ESI  00 01 02 03 04 05
223.120415     50990    C...  [redacted]  99                 FD    Rx   8    8          1    0    00 00 00 00 00 00
223.120298     50987    C...  [redacted]  100  [redacted]    FD    Tx   8    8          1    0    00 00 00 00 00 00
223.120297     50988    C...  [redacted]  100                FD    Rx   8    8          1    0    00 00 00 00 00 00
223.119945     50983    C...  [redacted]  100  [redacted]    FD    Tx   9    12         1    0    00 00 00 00 00 00
223.119945     50984    C...  [redacted]  100                FD    Rx   9    12         1    0    00 00 00 00 00 00
223.130800     50993    C...  [redacted]  100  [redacted]    FD    Tx   13   32         1    0    00 00 00 00 00 00
223.130800     50994    C...  [redacted]  100                FD    Rx   13   32         1    0    00 00 00 00 00 00
223.131035     50997    C...  [redacted]  99   [redacted]    FD    Tx   8    8          1    0    00 00 00 00 00 00
223.140458     50999    C...  [redacted]  10   [redacted]    FD    Tx   15   64         1    0    00 00 00 00 00 00
223.140457     51000    C...  [redacted]  10                 FD    Rx   15   64         1    0    00 00 00 00 00 00
In list   All Messages   0 %
```

1.8.5.2. Why Messages are Detected Instantly when Connecting to an Application

Problem Description: Upon clicking on "Connect Application", packets are always detected in the Trace window, even though there was no intention to send any packets. There are several possible causes for this.

1.8.5.2.1. "Start" is Associated with a Message Sending Module

The TSMaster program supports associating the startup (Start) process with functional modules. This detailed operation is outlined in the chapter titled "1.3.3 Start Application（Start）" For instance, if the Transmit window for message sending is checked, upon startup, the messages listed in the Transmit window will initiate sending automatically. Similarly, if the Bus Replay window is checked, upon startup, automatic bus message replay will commence. This principle applies to other functional modules in a similar fashion.

Solution: In the Tools -> System Information window, disassociate the Start process from the functional module. Alternatively, directly on the functional module, click the button in the top-right corner to disassociate the Start process.

1.8.5.2.2. The RBS Module is Automatically Activated.

For information on why the RBS module is automatically activated, refer to the subchapter "1.6.3.1.6.3.1 Why Is the RBS Module Automatically Activated" within the RBS chapter.

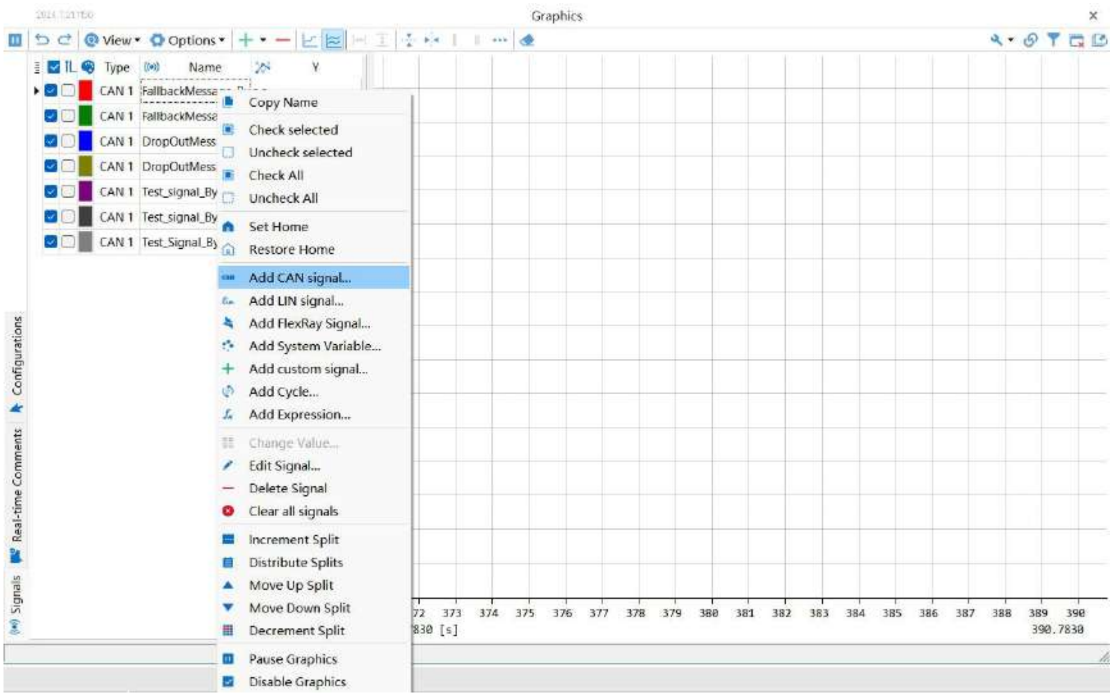
1.9. Curve Window（Graphic）

1.9.1. Basic Operation

1.9.1.1. Add Monitoring Variable

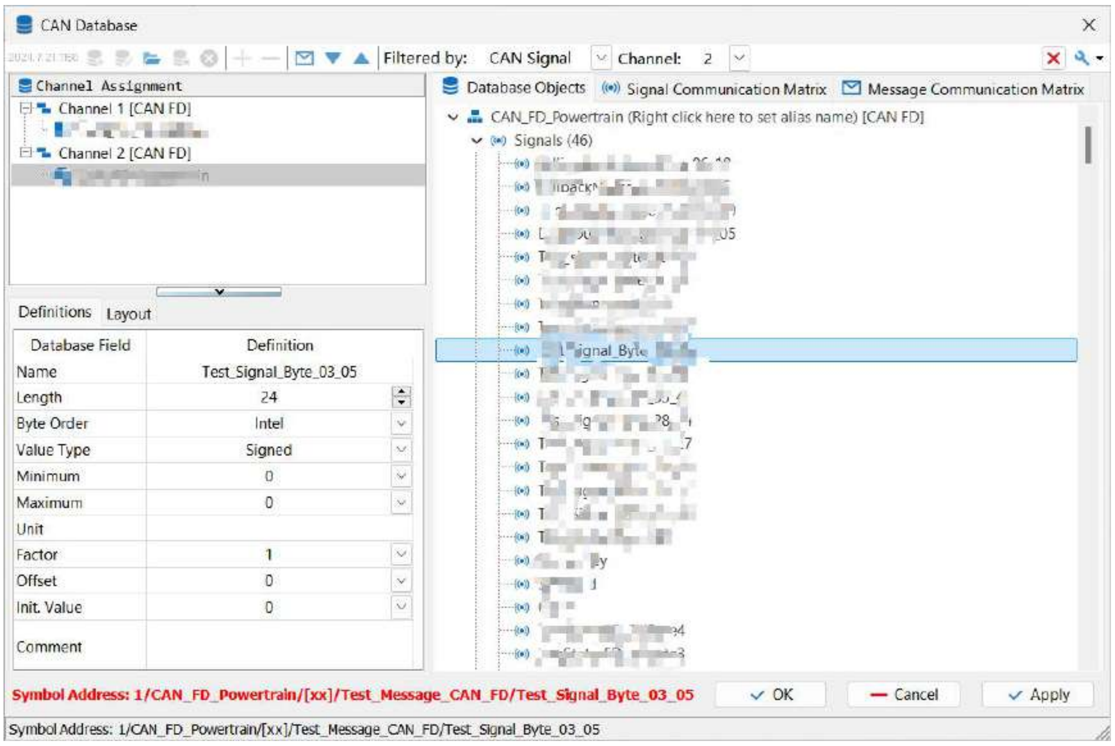
There are two main ways to add monitoring variables to the Graphic window: 1. Directly add them in the Curve window; 2. Add them from the Trace window.

➢ Directly add in the Graphic window:



In the Graphic window, right click to show the pop-up menu.

For example, by selecting "Add CAN Signal," the user will enter the database interface to choose the signal to monitor. It is important to emphasize here that the database is bound to a specific channel. If the data source is Channel 1, but the user select a signal from Channel 2, then it is certainly impossible to decode the corresponding signal value. As shown below:



CAN Database

Filtered by: CAN Signal Channel: 2

Channel Assignment
- Channel 1 [CAN FD]
- Channel 2 [CAN FD]

Database Objects | Signal Communication Matrix | Message Communication Matrix

CAN_FD_Powertrain (Right click here to set alias name) [CAN FD]
- Signals (46)

[redacted]

Definitions | Layout

Database Field	Definition
Name	Test_Signal_Byte_03_05
Length	24
Byte Order	Intel
Value Type	Signed
Minimum	0
Maximum	0
Unit	
Factor	1
Offset	0
Init. Value	0
Comment	

Symbol Address: 1/CAN_FD_Powertrain/[xx]/Test_Message_CAN_FD/Test_Signal_Byte_03_05

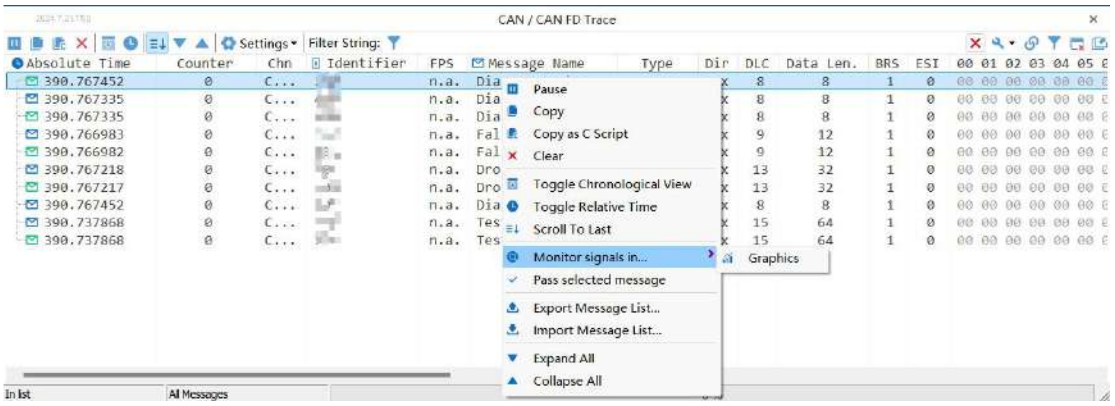
OK | Cancel | Apply

Symbol Address: 1/CAN_FD_Powertrain/[xx]/Test_Message_CAN_FD/Test_Signal_Byte_03_05

By selecting a database of Channel 2, the Channel will aromatically shows the value 2. Then, select the signal to be monitored and click OK, the signal will be added to the Graphic window.

➢ Add from the Trace window:

Select the Trace window -> Expand the message -> Select the signal that needs to be monitored -> Right-click to expand the menu -> Move the mouse to "Monitor signals in" -> Add the signal to the corresponding monitoring window.



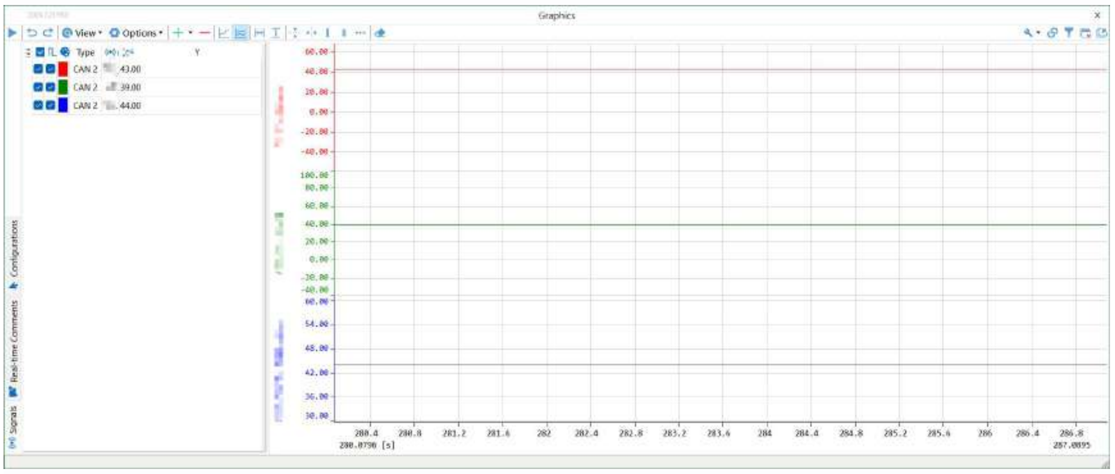
CAN / CAN FD Trace

Settings | Filter String:

Absolute Time	Counter	Chn	Identifier	FPS	Message Name	Type	Dir	DLC	Data Len.	BRS	ESI	00 01 02 03 04 05
390.767452	0	C...	[redacted]	n.a.	Dia		x	8	8	1	0	00 00 00 00 00 00
390.767335	0	C...	[redacted]	n.a.	Dia		x	8	8	1	0	00 00 00 00 00 00
390.767335	0	C...	[redacted]	n.a.	Dia		x	8	8	1	0	00 00 00 00 00 00
390.766983	0	C...	[redacted]	n.a.	Fal		x	9	12	1	0	00 00 00 00 00 00
390.766982	0	C...	[redacted]	n.a.	Fal		x	9	12	1	0	00 00 00 00 00 00
390.767218	0	C...	[redacted]	n.a.	Dro		x	13	32	1	0	00 00 00 00 00 00
390.767217	0	C...	[redacted]	n.a.	Dro		x	13	32	1	0	00 00 00 00 00 00
390.767452	0	C...	[redacted]	n.a.	Dia		x	8	8	1	0	00 00 00 00 00 00
390.737868	0	C...	[redacted]	n.a.	Tes		x	15	64	1	0	00 00 00 00 00 00
390.737868	0	C...	[redacted]	n.a.	Tes		x	15	64	1	0	00 00 00 00 00 00

1.9.1.2.Adjust X-axis

The X-axis represents the time axis, displaying the pattern of signal changes over time. The schematic diagram of the X-axis is shown below:



Descriptions for the X-axis in the above figure are as follows:

The starting point of the data on the current screen is 280.0796s, and the current scale size is: 0.4ms per division. There are a total of nearly 18 divisions, so the time width of the current screen is: 0.4*18 = 7.2ms. The end time of the screen data is: 280.0796 + 7.2 ≈ 287.0895s. During data analysis, users often need to adjust the observation fineness (X-axis scale) and the observation point position (horizontal position on the X-axis). The following sections explain how to operate these adjustments.

1

1.9.1.2.1. Adjust the X-axis Scale

To adjust the X-axis scale, simply hover the mouse over the X-axis without clicking (neither single-click nor double-click), and then use the scroll wheel to adjust the X-axis scale value directly.

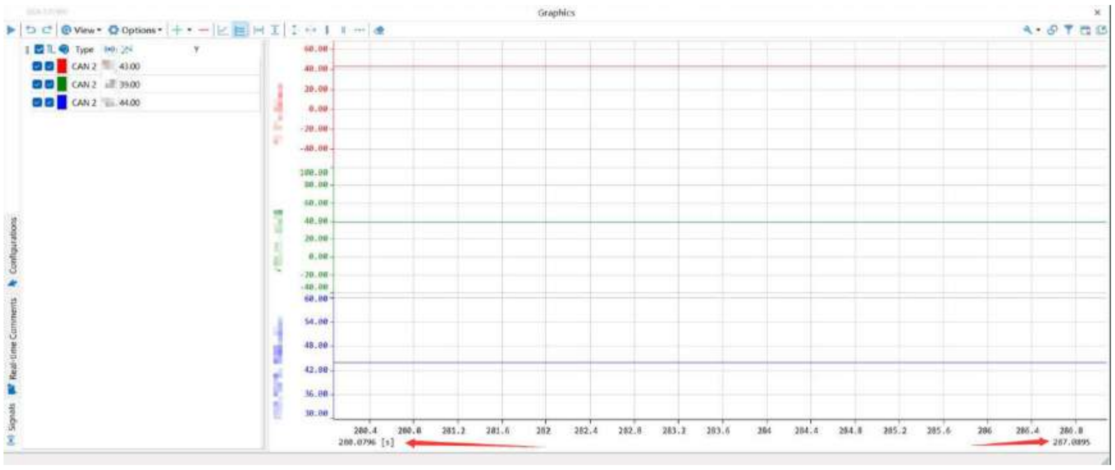
- Scrolling the wheel upwards to adjust the scale value to a smaller one. For example, from representing 1 second per division to 1 millisecond per division.

- Scrolling the wheel downwards to adjust the scale value to a larger one. For example, from representing 1 millisecond per division to 1 second per division.

1.9.1.2.2. Move the X-axis Position（Display Range of the X-axis）

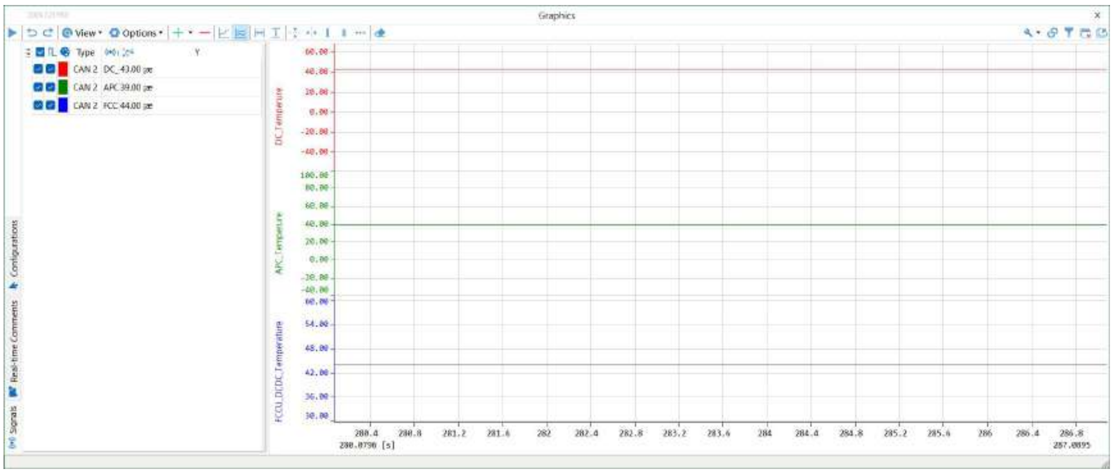
Method 1:

Placing the mouse cursor on the X-axis, pressing and holding the left mouse button, and then moving left or right is a very intuitive way to adjust and is highly recommended.



Method 2:

By placing the mouse cursor on the X-axis, pressing and holding the Ctrl key, scrolling the wheel upwards achieves the effect of moving left, while scrolling the wheel downwards achieves the effect of moving right.



1.9.1.2.3. Usage Recommendations

Combining the adjustment of the X-axis position with the calibration of the scale values will result in a more efficient and accurate movement. When a significant positional shift is required, it is recommended to first adjust the scale value to a larger setting to quickly move the cursor to a vicinity near the desired observation point. Then, reduce the scale value to a smaller setting to precisely navigate to the exact observation position.

1.9.1.3. Adjust Y-axis

1.9.1.3.1. Adjust the Y-axis Scale

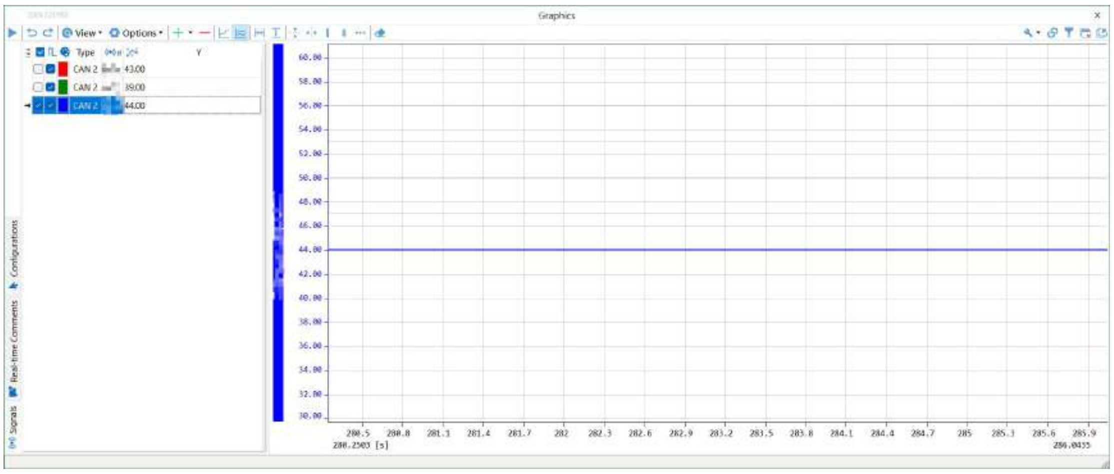
To adjust the Y-axis scale, simply hover your mouse over the Y-axis without clicking (neither single-click nor double-click), and then directly use the scroll wheel to adjust the Y-axis scale value. This operation is the same as adjusting the X-axis scale.

- Scrolling the wheel upwards to adjust the scale value to a smaller one. For example, from representing 1 second per division to 1 millisecond per division.

- Scrolling the wheel downwards to adjust the scale value to a larger one. For example, from representing 1 millisecond per division to 1 second per division.

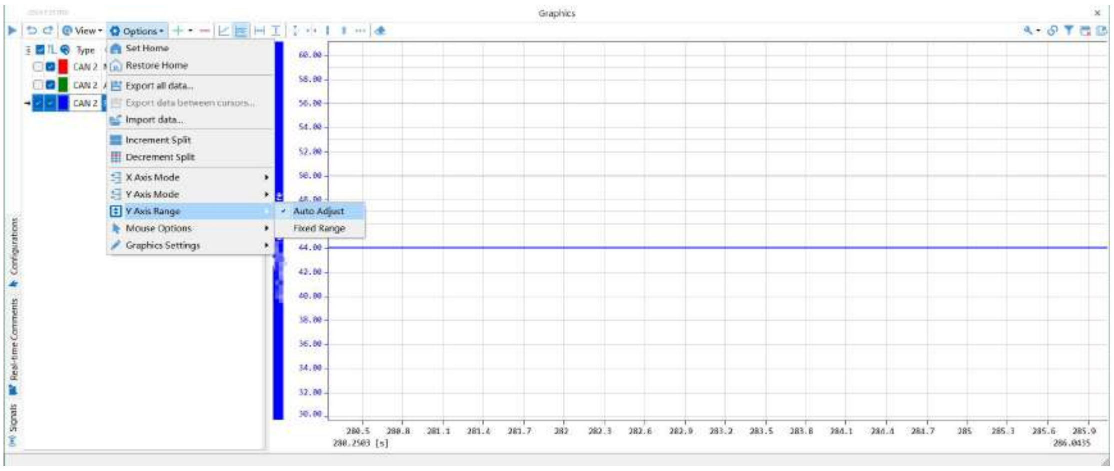
1.9.1.3.2. Move the Y-axis Position（Display Range of the Y-axis）

The Y-axis is associated with signal values. In order to establish a correlation with the database signals and to prevent users from accidentally dragging the Y-axis to an unreasonable position that causes the data curve to become invisible, the Y-axis does not support direct modification of the display range through mouse dragging. The default range of the Y-axis is set to the maximum (Max) and minimum (Min) values of the signal values in the database, as shown below:

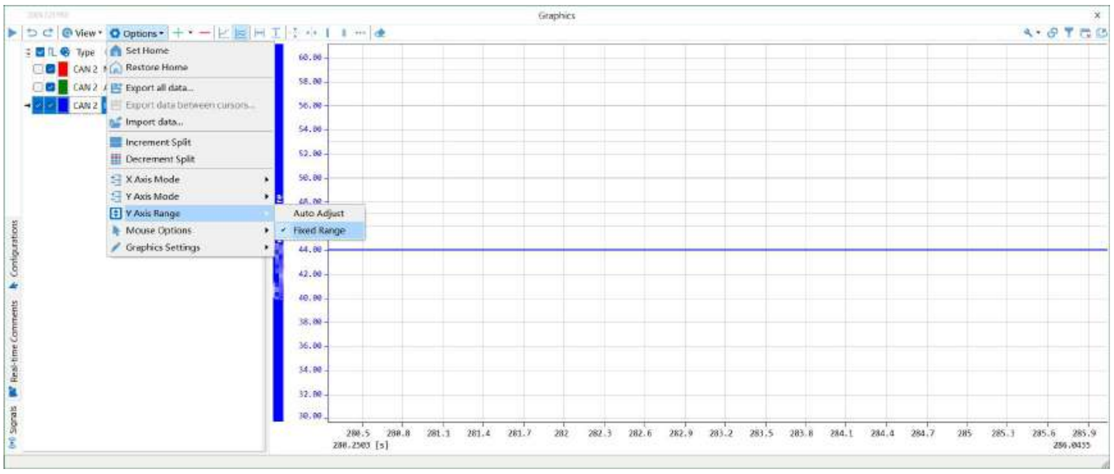


To adjust the display range of the Y-axis, there are three methods as follows:

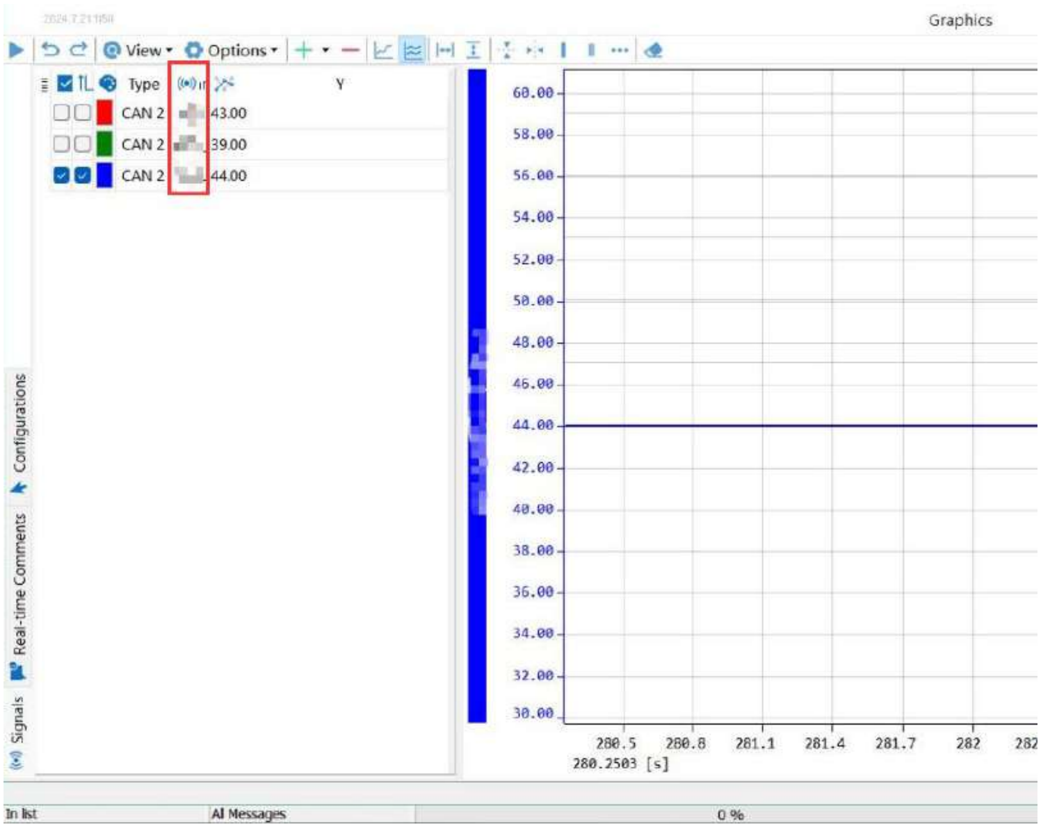
➢ Set the Y-axis display range to be adaptive. The Y-axis will dynamically adjust its display range based on the size of the data values. This method can ensure that all display value ranges on the Y-axis are covered. The setting method is shown in the figure below:



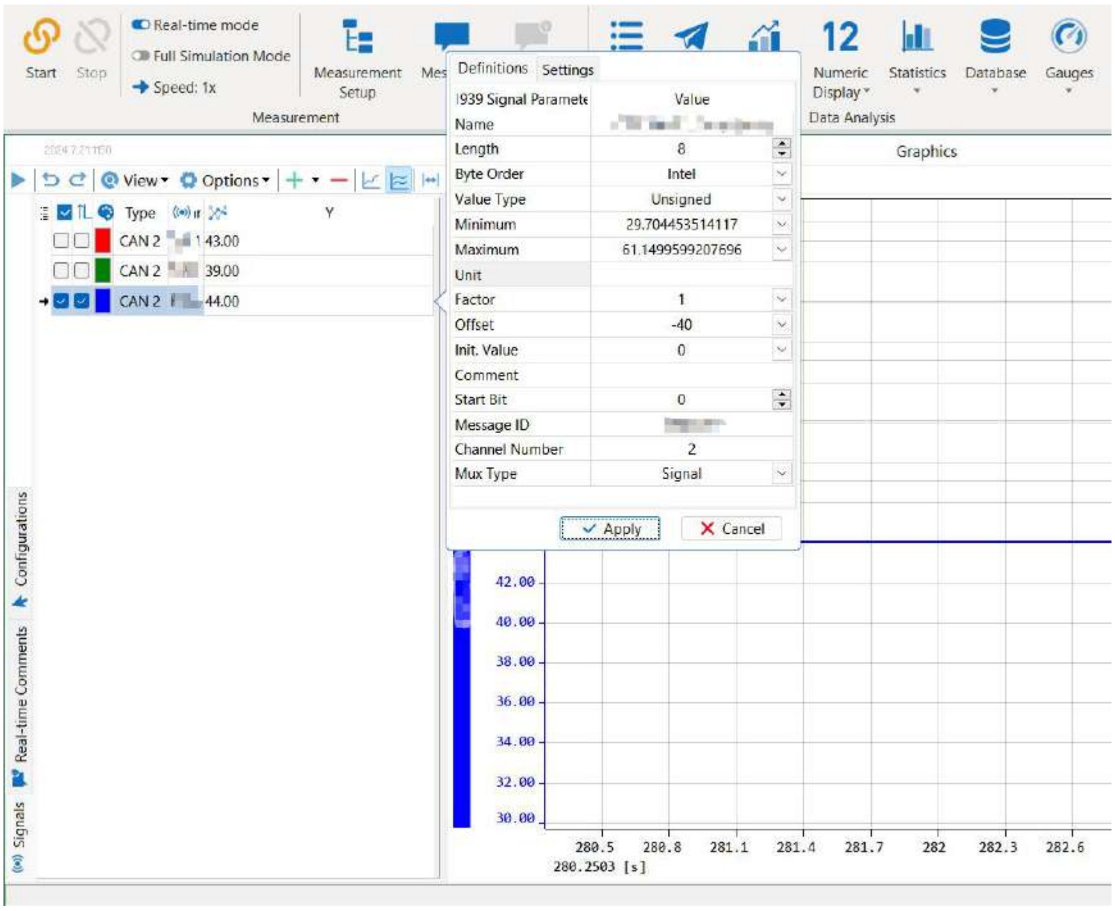
➢ Set the Y-axis display range to a fixed value, as follows:



Double-click on the signal name value (or directly double-click on the Y-axis corresponding to that signal), as shown below:



Expand the signal properties window as shown below, and set the minimum and maximum values for the signal, which will define the display range of the Y-axis:



	Type		Y
☐ ☐ 🟥	CAN 2	[redacted]	43.00
☐ ☐ 🟩	CAN 2	[redacted]	39.00
☑ ☑ 🟦	CAN 2	[redacted]	44.00

Definitions | Settings

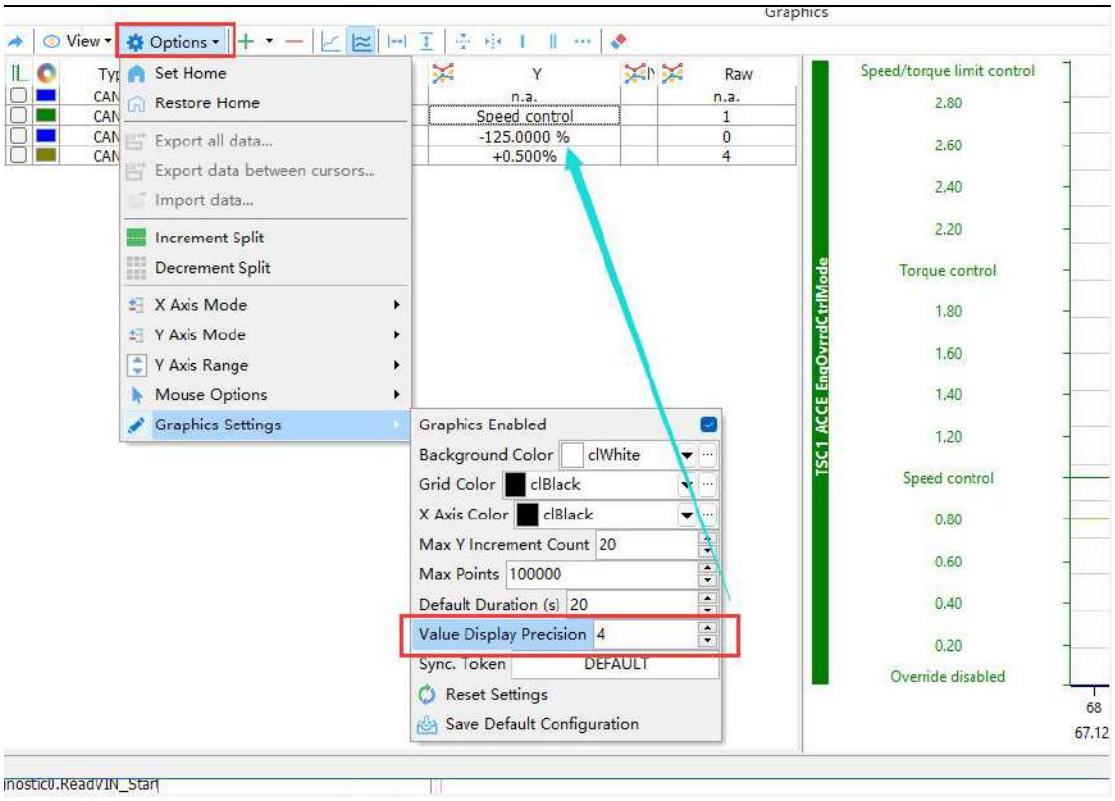
J939 Signal Paramete	Value
Name	[redacted]
Length	8
Byte Order	Intel
Value Type	Unsigned
Minimum	29.704453514117
Maximum	61.1499599207696
Unit	
Factor	1
Offset	-40
Init. Value	0
Comment	
Start Bit	0
Message ID	[redacted]
Channel Number	2
Mux Type	Signal

Apply | Cancel

➢ Adjust using the mouse scroll wheel.

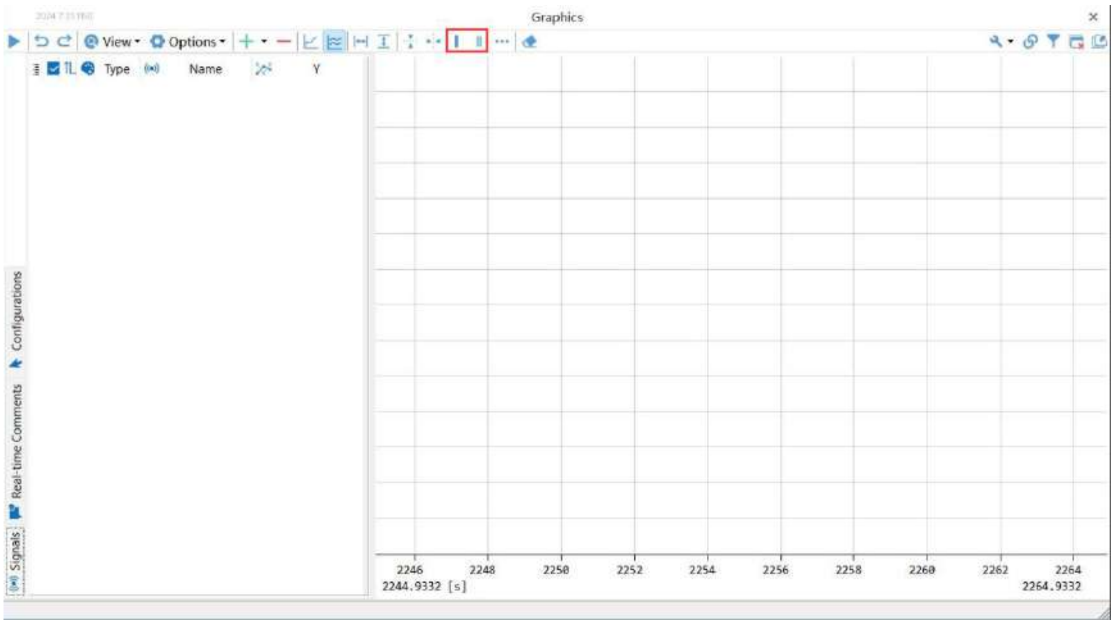
1.9.2. Floating Point Precision

The location of the floating point precision in the Graphics is as shown below:



1.9.3. Move the Cursor

1.9.3.1. Add the Cursor



The Graphic window in TSMaster supports adding a single cursor or two cursors to observe data. The adding method is as shown in the figure above. To cancel the cursor display, simply click the added button again, and the cursor will disappear.

1.9.3.2.Move the Cursor

1.9.3.2.1. Move Single Cursor

1) Right-click to trigger cursor following, allowing the cursor to move with the mouse.

2) Left-click to position the cursor at the desired location for observation.

1.9.3.2.2. Dual Cursor Movement

1) Click on the left to select the observation position of the blue cursor.

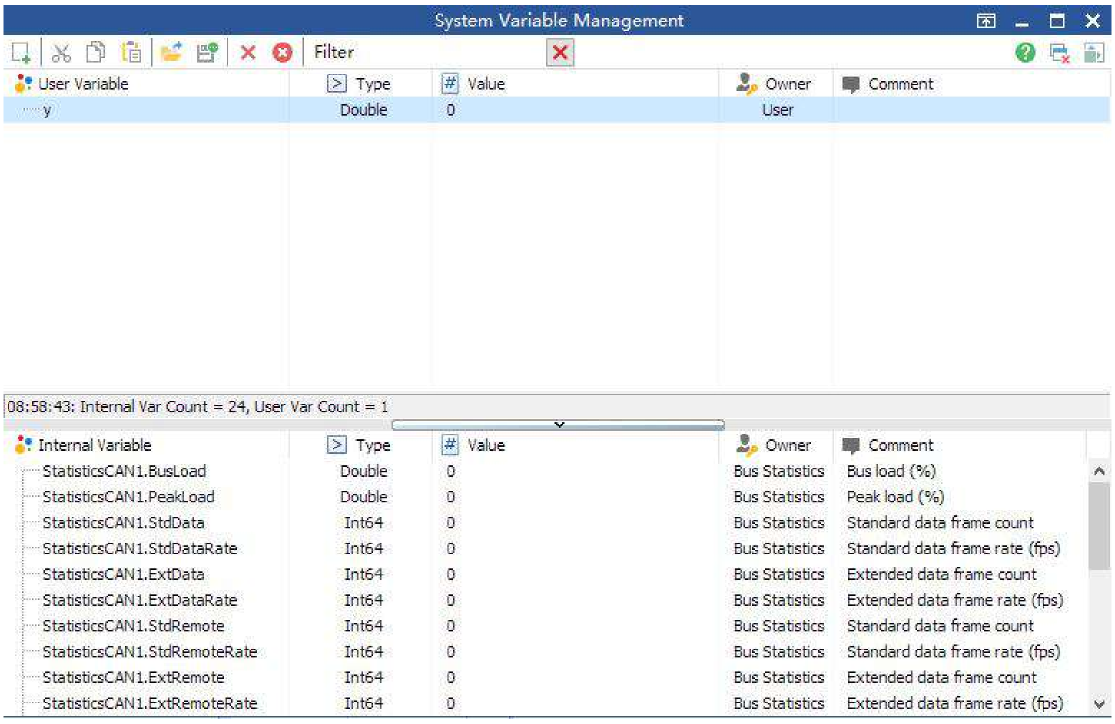
2) Click on the right to select the observation position of the red cursor.

1.9.4. Application Cases

1.9.4.1.Use TSMaster to Draw any ECU Variable Curve

【1】 Define system variable y.

For a detailed explanation of system variables, please refer to the chapter 1.13 System Variables.



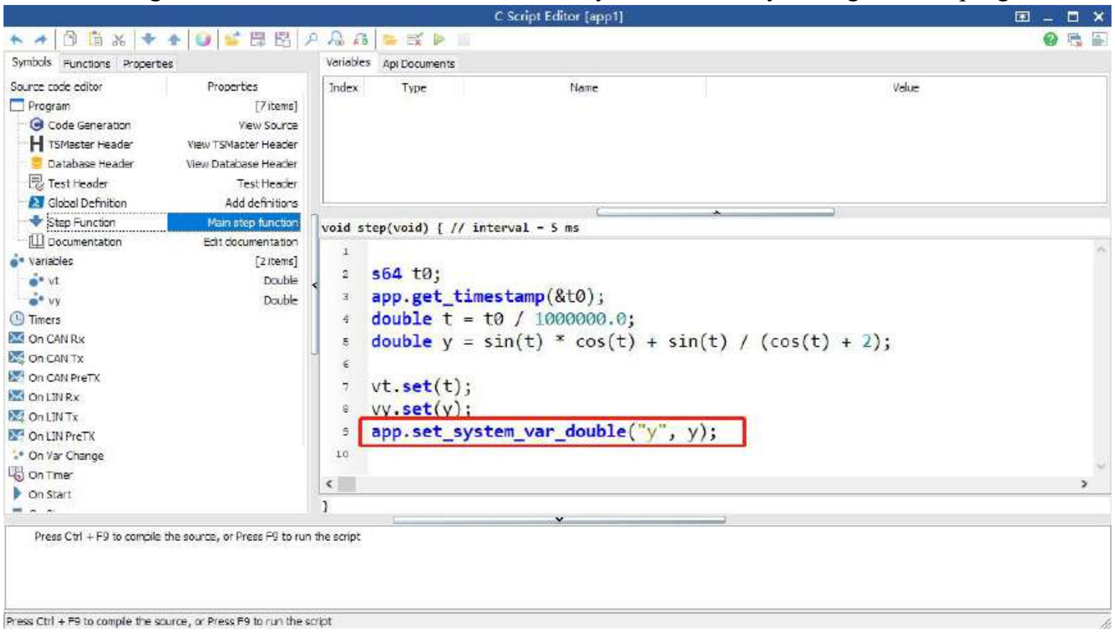
System Variable Management

User Variable	Type	Value	Owner	Comment
y	Double	0	User	

08:58:43: Internal Var Count = 24, User Var Count = 1

Internal Variable	Type	Value	Owner	Comment
StatisticsCAN1.BusLoad	Double	0	Bus Statistics	Bus load (%)
StatisticsCAN1.PeakLoad	Double	0	Bus Statistics	Peak load (%)
StatisticsCAN1.StdData	Int64	0	Bus Statistics	Standard data frame count
StatisticsCAN1.StdDataRate	Int64	0	Bus Statistics	Standard data frame rate (fps)
StatisticsCAN1.ExtData	Int64	0	Bus Statistics	Extended data frame count
StatisticsCAN1.ExtDataRate	Int64	0	Bus Statistics	Extended data frame rate (fps)
StatisticsCAN1.StdRemote	Int64	0	Bus Statistics	Standard data frame count
StatisticsCAN1.StdRemoteRate	Int64	0	Bus Statistics	Standard data frame rate (fps)
StatisticsCAN1.ExtRemote	Int64	0	Bus Statistics	Extended data frame count
StatisticsCAN1.ExtRemoteRate	Int64	0	Bus Statistics	Extended data frame rate (fps)

【2】 Assign the internal variable of ECU to this system variable y through a mini program



C Script Editor [app1]

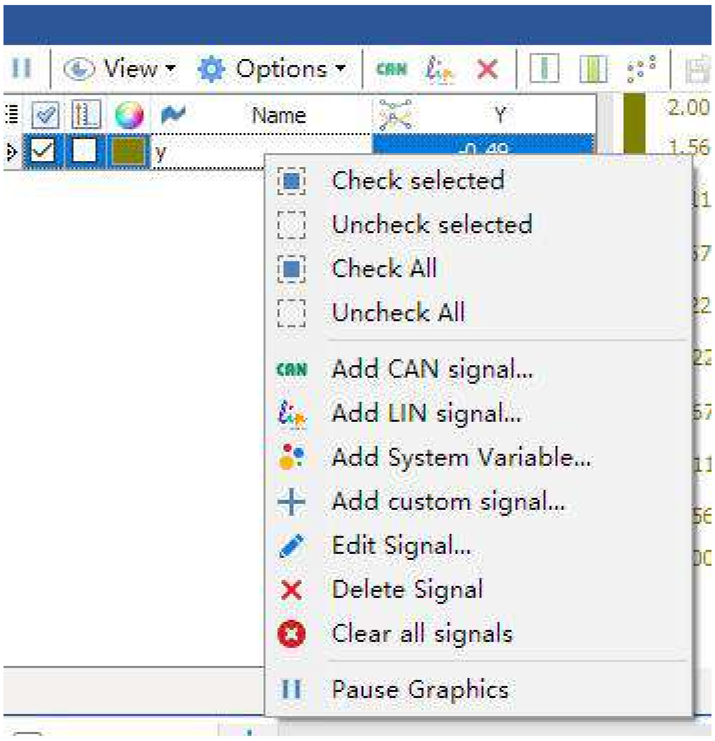
Source code editor	Properties
Program	[7 items]
Code Generation	View Source
TSMaster Header	View TSMaster Header
Database Header	View Database Header
Test Header	Test Header
Global Definition	Add definitions
Step Function	Main step function
Documentation	Edit documentation
Variables	[2 items]
vt	Double
vy	Double
Timers	
On CAN Rx	
On CAN Tx	
On CAN PreTX	
On LIN Rx	
On LIN Tx	
On LIN PreTX	
On Var Change	
On Timer	
On Start	

```
void step(void) { // interval = 5 ms

s64 t0;
app.get_timestamp(&t0);
double t = t0 / 1000000.0;
double y = sin(t) * cos(t) + sin(t) / (cos(t) + 2);

vt.set(t);
vy.set(y);
app.set_system_var_double("y", y);

}
```

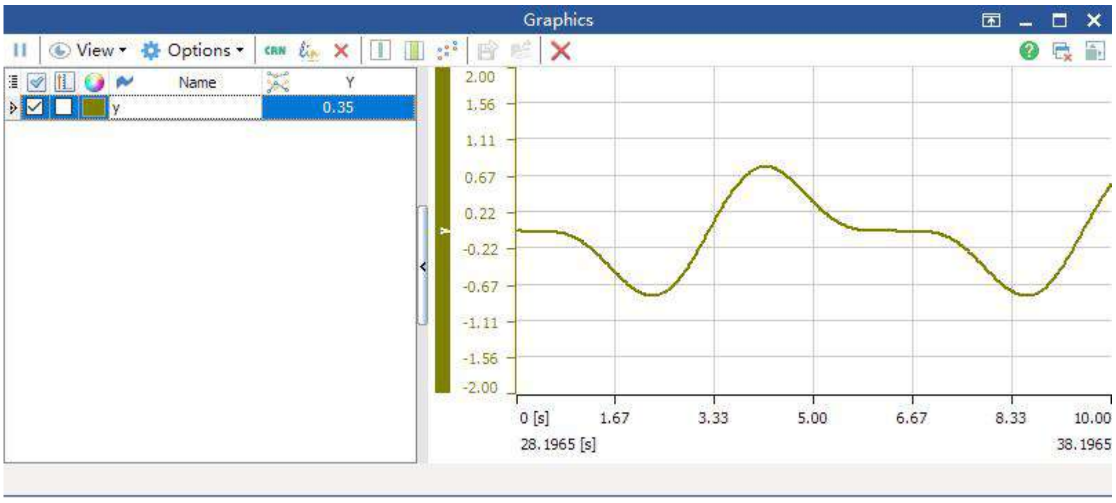
Press Ctrl + F9 to compile the source, or Press F9 to run the script

Press Ctrl + F9 to compile the source, or Press F9 to run the script

【3】 In graphics, add system variable y.



【4】 In graphics, check system variable y



1.9.5. Clarification

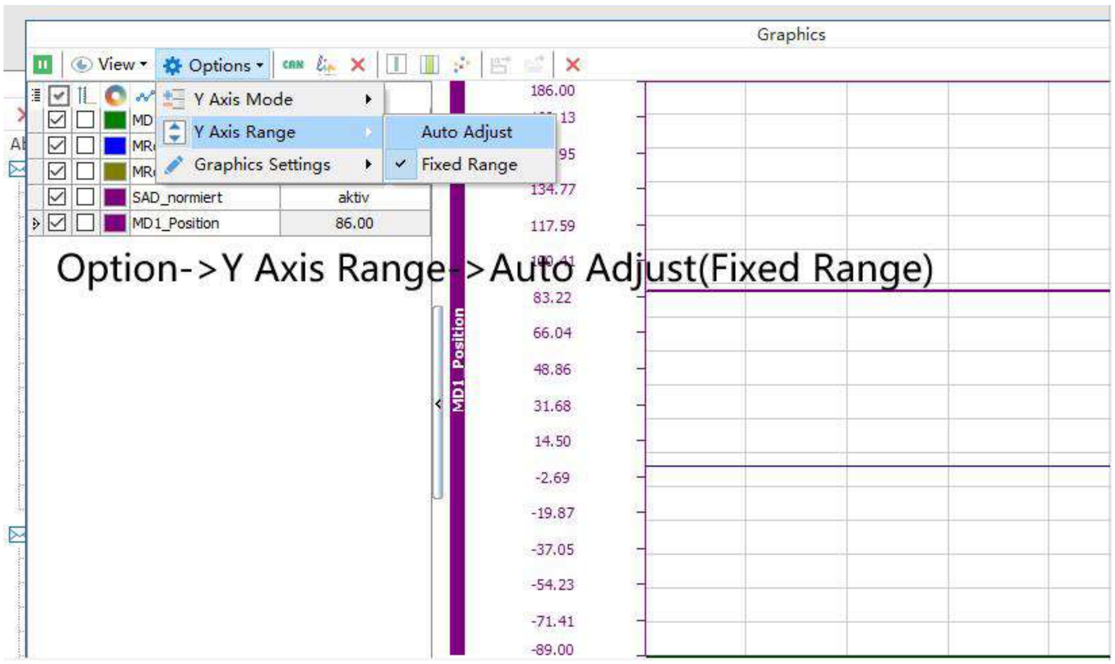
1.9.5.1. Why does the Y-axis Display Values far beyond the Set Signal Value Range in the Database

Problem Description: The signal value range is set in the database, for example, from 0 to 100.

However, why does the Y-axis range display far beyond this, for example, reaching 0 to 10000?

This is because, the display mode of the Y-axis in the graphic has been set to automatic mode. In this mode, the Y-axis range automatically adjusts based on the maximum and minimum values that the signal has reached. If the signal value ever reached 10000 during display, the Y-axis range would be adjusted to 0-10000.

To set the display mode of the Y-axis, follow these steps:



Auto Adjust: The Y-axis automatically adjusts its range based on actual values.

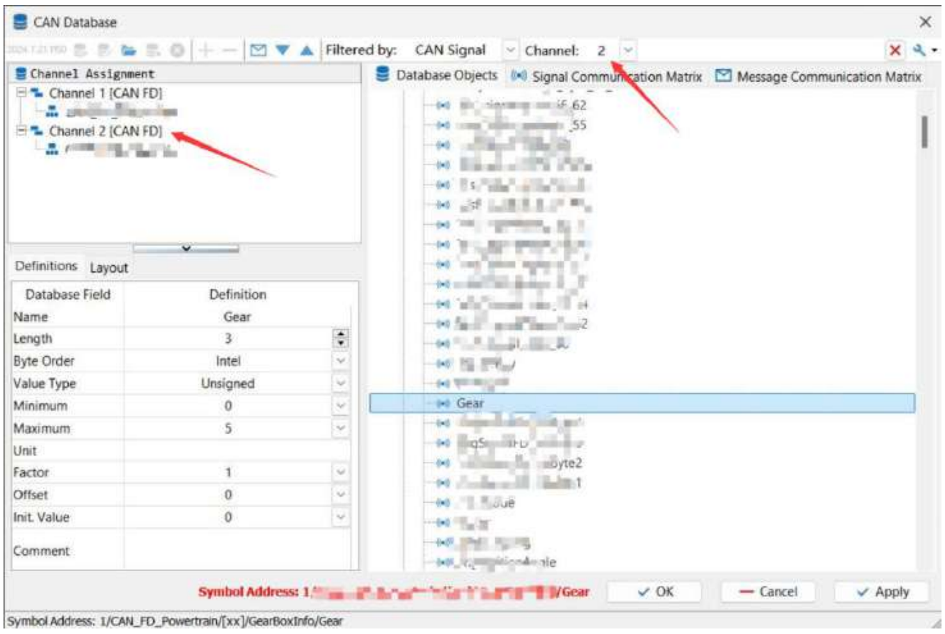
Fixed Range: Y-axis has fixed range value, in this mode, for the operation to adjust the display range (Y-axis position), see section 1.9.1.3.2 Move the Y-axis Position（Display Range of the Y-axis）.

1.9.5.2. Why Can't I See Signal Changes in Windows like Graphic when Signals are Added during Message Replay or Monitoring

Possible reason: The channel where the signal is added does not match the channel of the current message (whether it's during replay or real-time monitoring).

For example:

- The channel selected for adding the signal is 2, as shown below: The added signal is from the signal in the database that belongs to channel 2.

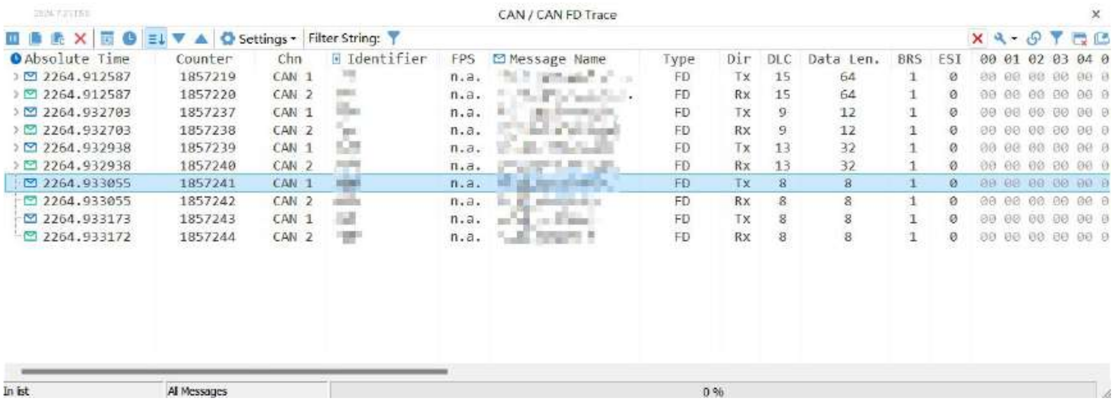


Database Field	Definition
Name	Gear
Length	3
Byte Order	Intel
Value Type	Unsigned
Minimum	0
Maximum	5
Unit	
Factor	1
Offset	0
Init. Value	0
Comment	

Symbol Address: 1/[redacted]/Gear

Symbol Address: 1/CAN_FD_Powertrain/[xx]/GearBoxInfo/Gear

- However, the message is from channel 1, as follows:



Absolute Time	Counter	Chn	Identifier	FPS	Message Name	Type	Dir	DLC	Data Len.	BRS	ESI	00 01 02 03 04 0
2264.912587	1857219	CAN 1	[redacted]	n.a.	[redacted]	FD	Tx	15	64	1	0	00 00 00 00 00 0
2264.912587	1857220	CAN 2	[redacted]	n.a.	[redacted]	FD	Rx	15	64	1	0	00 00 00 00 00 0
2264.932703	1857237	CAN 1	[redacted]	n.a.	[redacted]	FD	Tx	9	12	1	0	00 00 00 00 00 0
2264.932703	1857238	CAN 2	[redacted]	n.a.	[redacted]	FD	Rx	9	12	1	0	00 00 00 00 00 0
2264.932938	1857239	CAN 1	[redacted]	n.a.	[redacted]	FD	Tx	13	32	1	0	00 00 00 00 00 0
2264.932938	1857240	CAN 2	[redacted]	n.a.	[redacted]	FD	Rx	13	32	1	0	00 00 00 00 00 0
2264.933055	1857241	CAN 1	[redacted]	n.a.	[redacted]	FD	Tx	8	8	1	0	00 00 00 00 00 0
2264.933055	1857242	CAN 2	[redacted]	n.a.	[redacted]	FD	Rx	8	8	1	0	00 00 00 00 00 0
2264.933173	1857243	CAN 1	[redacted]	n.a.	[redacted]	FD	Tx	8	8	1	0	00 00 00 00 00 0
2264.933172	1857244	CAN 2	[redacted]	n.a.	[redacted]	FD	Rx	8	8	1	0	00 00 00 00 00 0

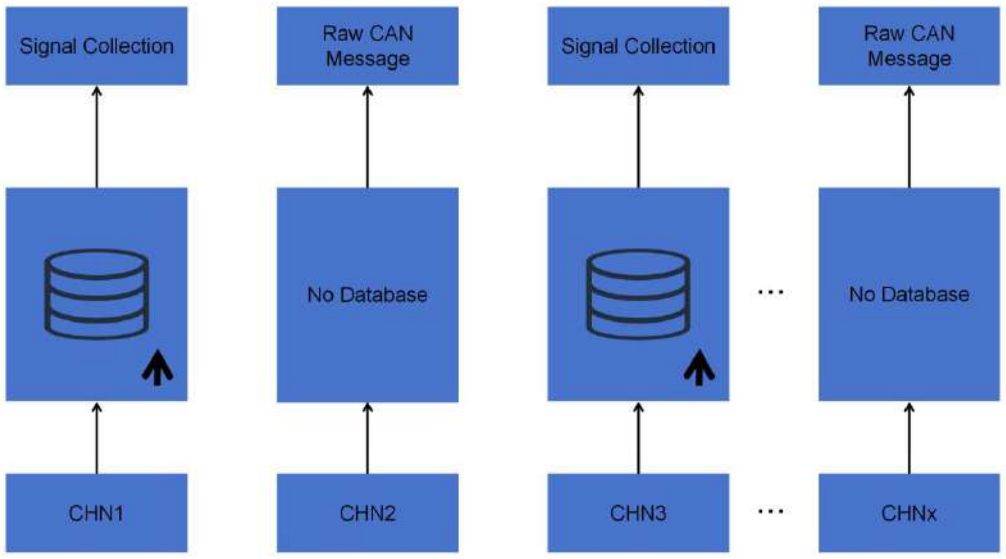
In this case, the parsed signal value will definitely have no numerical value.

1.10. CAN DBC/ LIN LDF

1.10.1. Basic Concepts

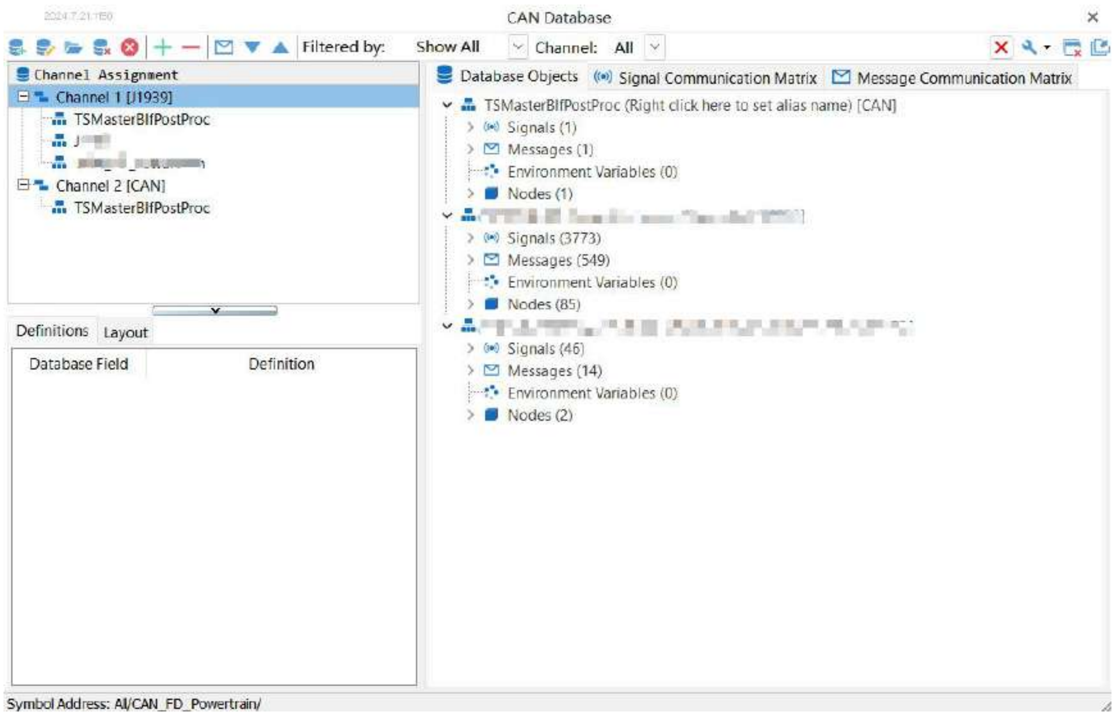
Before using CAN/CANFD/LIN bus databases, it is necessary to clarify a few basic concepts:

- Databases are bound to channels. When each channel receives CAN/CANFD/LIN bus data, it parses the data according to the database bound to that channel, as shown in the following diagram:



- A single channel can be bound to multiple databases. Likewise, a single database can be bound to different channels.

As shown in below figure, multiple databases are bound to channel 1, and the same database TSMasterBlfPostProc is bound to different channels.


CAN Database

Filtered by: Show All Channel: All

Channel Assignment
- Channel 1 [J1939]
  - TSMasterBlfPostProc
  - J[redacted]
  - [redacted]
- Channel 2 [CAN]
  - TSMasterBlfPostProc

Definitions Layout

Database Field Definition

Database Objects Signal Communication Matrix Message Communication Matrix

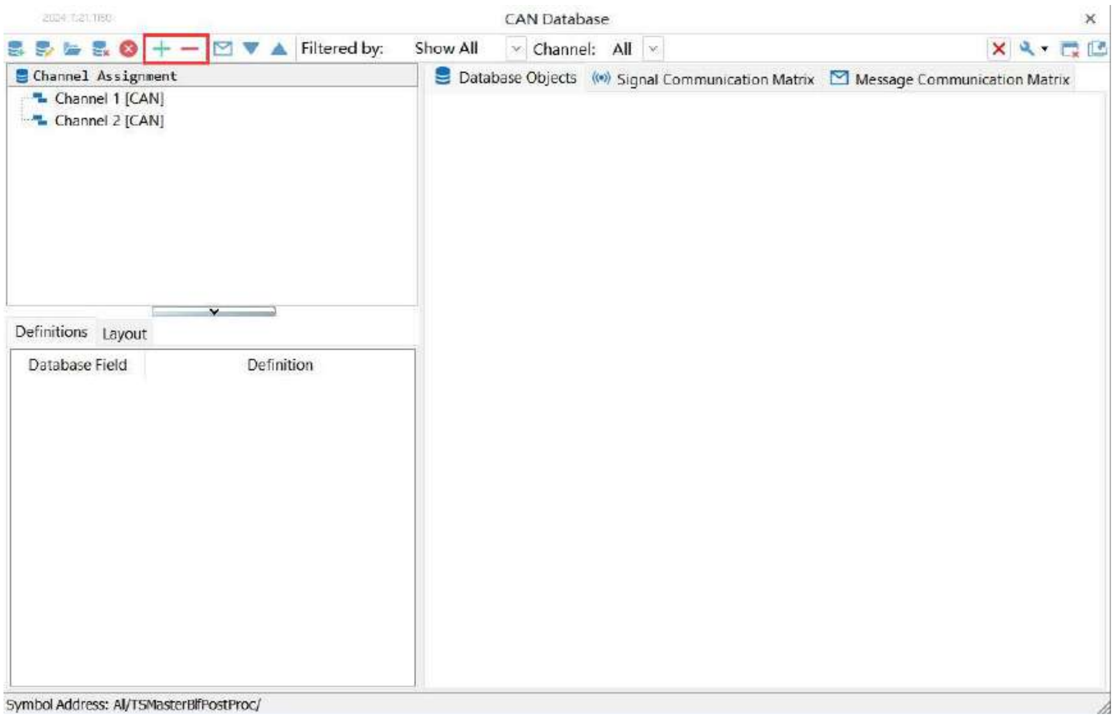
- TSMasterBlfPostProc (Right click here to set alias name) [CAN]
  - Signals (1)
  - Messages (1)
  - Environment Variables (0)
  - Nodes (1)
- [redacted]
  - Signals (3773)
  - Messages (549)
  - Environment Variables (0)
  - Nodes (85)
- [redacted]
  - Signals (46)
  - Messages (14)
  - Environment Variables (0)
  - Nodes (2)

Symbol Address: AI/CAN_FD_Powertrain/

When multiple databases are bound to the same channel, these databases are in a parallel relationship. After the channel receives CAN message data, the TSMaster database engine will query these databases in turn. Once the message ID is found in one of the databases, the parsing of the message data will be completed.

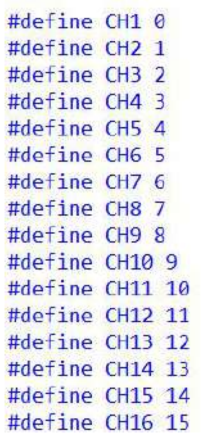
1.10.2. Database Channel Management

The database channel corresponds one-to-one with the hardware channel. As shown below, Channel1 corresponds to Application Channel 1 in the TSMaster hardware configuration interface. The same principle applies to other channels. In the database management interface, users can add and delete database channels according to their needs.



1.10.2.1. About Channel Numbering

Regarding the actual numerical values of channel numbers in software, here's an example:



```
#define CH1 0
#define CH2 1
#define CH3 2
#define CH4 3
#define CH5 4
#define CH6 5
#define CH7 6
#define CH8 7
#define CH9 8
#define CH10 9
#define CH11 10
#define CH12 11
#define CH13 12
#define CH14 13
#define CH15 14
#define CH16 15
```

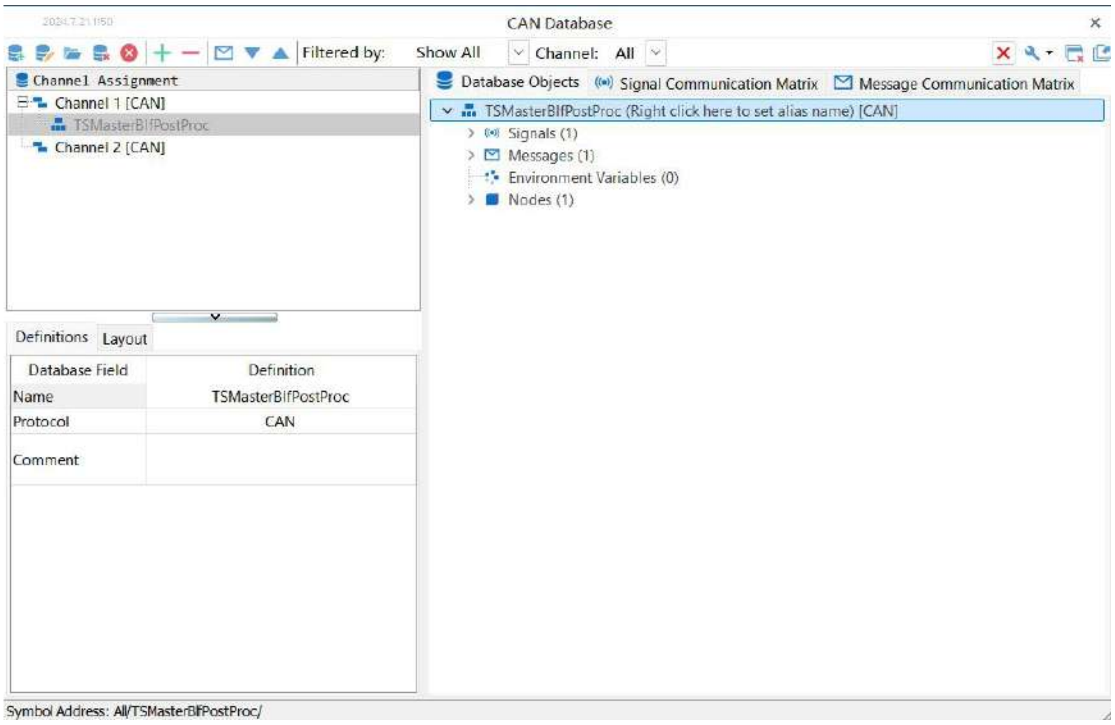
In other words, the corresponding index for CHANNEL1 in the application is 0, and the index for CHANNEL2 is 1, and so on. This is because, in programming development, indices often start from 0 (for example, with arrays and similar variables); whereas, in real-world descriptive conventions, numbering typically begins from 1 (such as channel 1, and so forth).

1.10.3. Add CAN/LIN Database File

TSMaster supports two methods for adding CAN/LIN database files:

1.10.3.1. Add by Drag-and-drop Directly

When a user directly drags a CAN/LIN database file into the TSMaster window area, the program automatically associates the database with the first available channel. As shown below:



Database Field	Definition
Name	TSMasterBlfPostProc
Protocol	CAN
Comment	

Symbol Address: All/TSMasterBlfPostProc/

Advantages:

Easy to operate, users can even add the database by simply dragging the DBC/LDF file directly into the TSMaster form without needing to open the database management window.

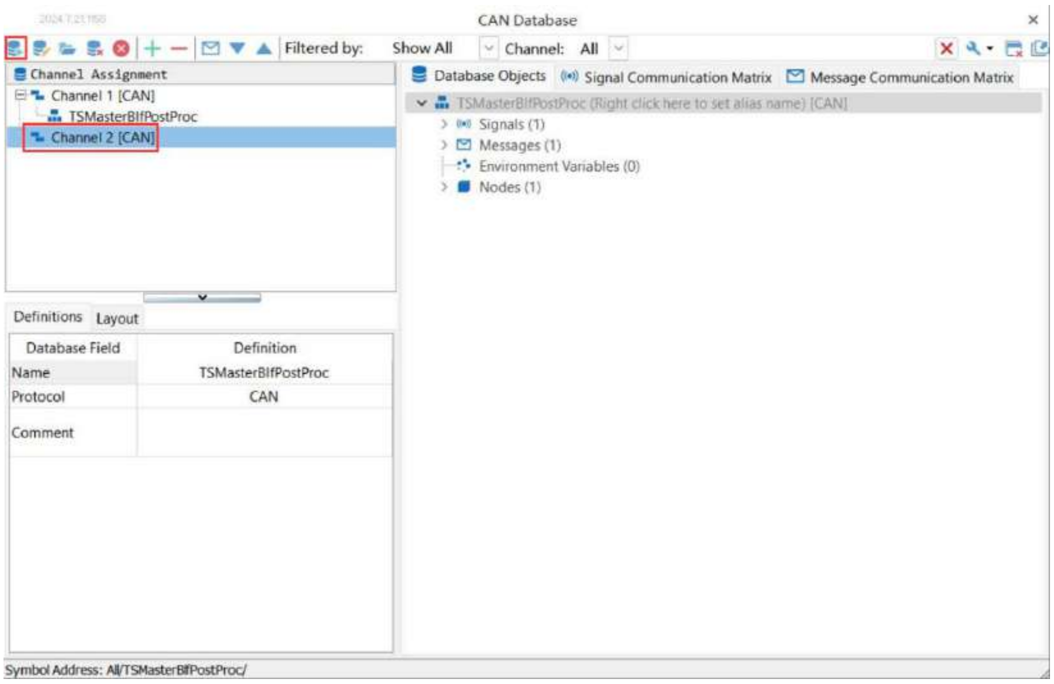
Disadvantages:

This method of adding the database by default only associates it with the first channel. If you need to associate it with other channels, you will need to manually add it in the database management interface.

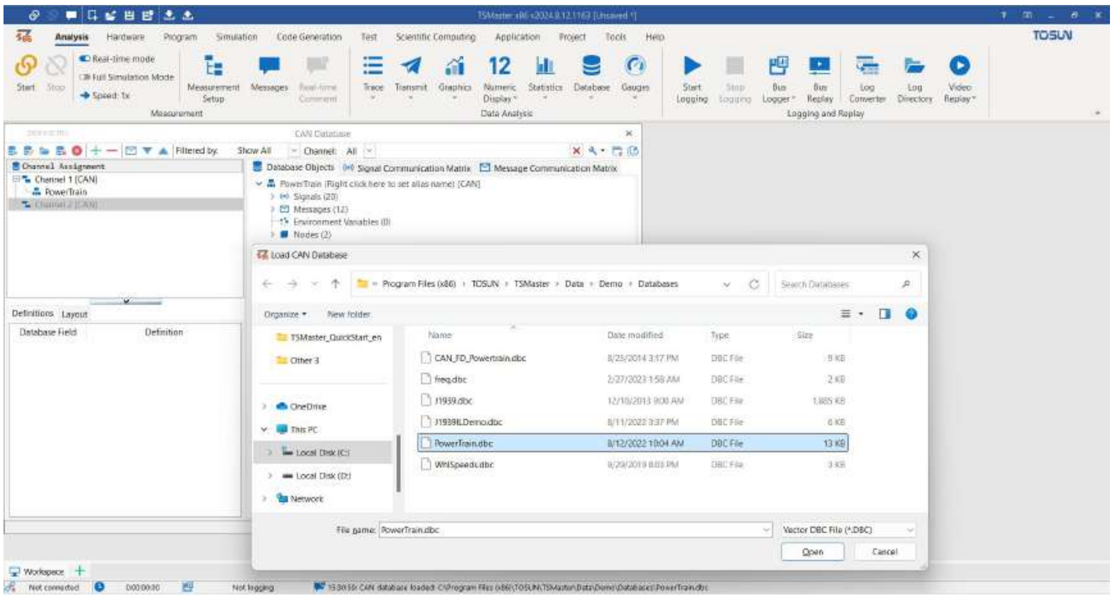
1.10.3.2. Add in the Database Window

The steps to add a database management channel in the database form are as follows:

➢ Select the channel and click the add database button, as shown below:

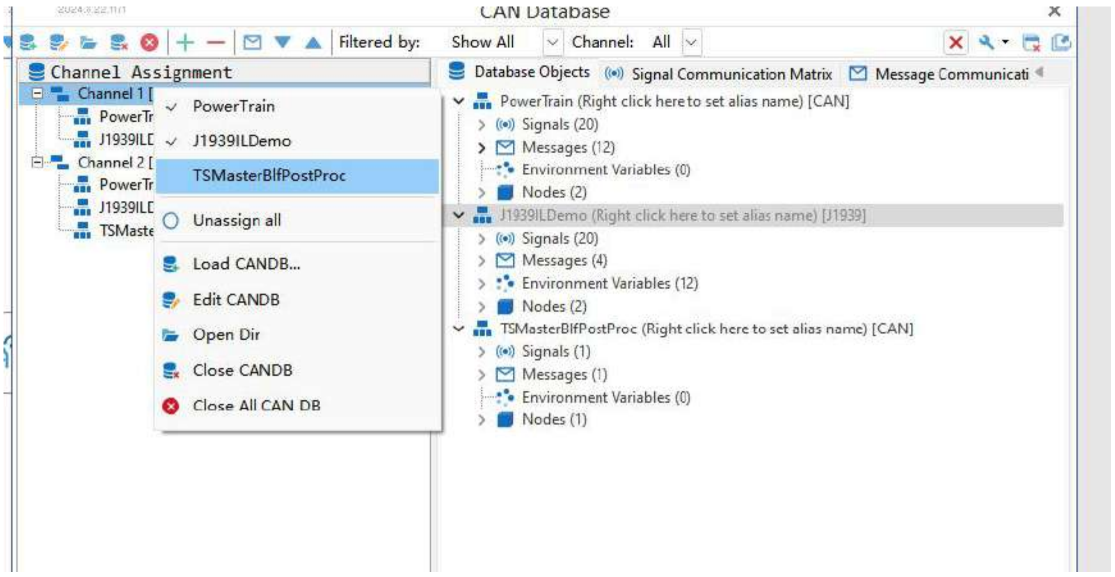


➢ By selecting the dbc file in the path, it can add the database to the target channel under "Channel Assignment", as shown below:

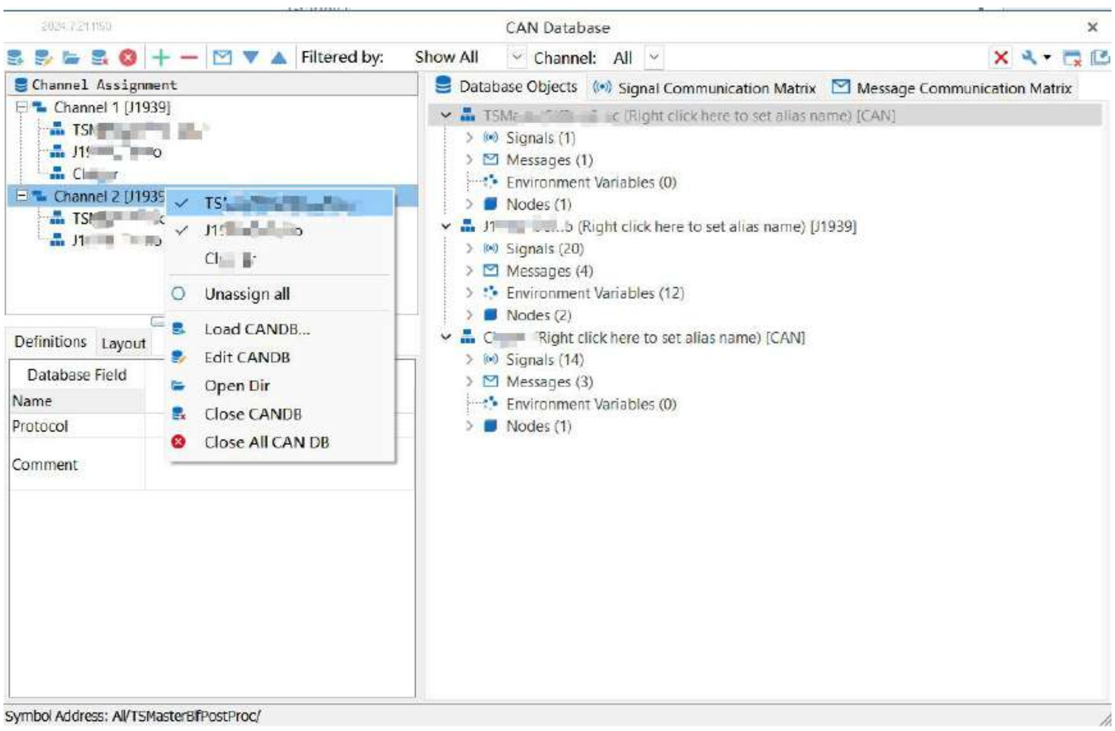


1.10.3.3. Associate the Database with an ECU Channel

Through the Database Manager, TSMaster provides a convenient way for users to switch database files between different channels, as shown in the figure below:



After associating the database with the channel, right-click on the channel and you will see a ✓ in front of the already associated database, as shown below:



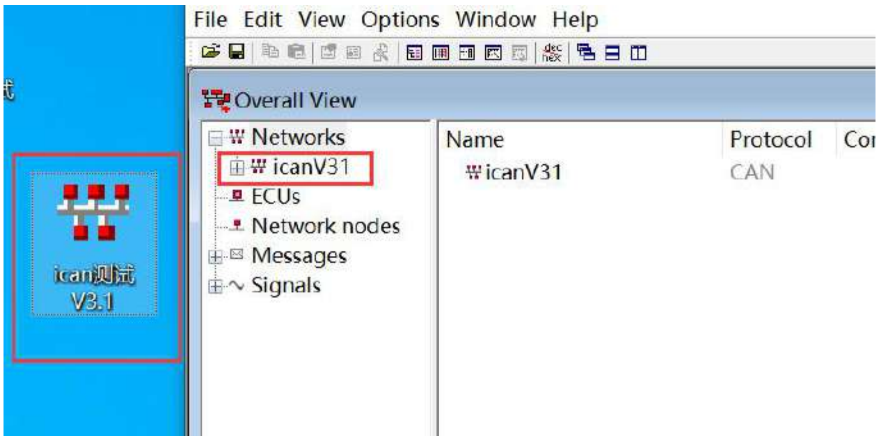
As mentioned in the previous section: In TSMaster, directly dragging and dropping a database file will by default associate it with the first channel. After the drag-and-drop operation, you can re-associate the database and ECU channel in the Database Management interface. Therefore, it is recommended to follow the steps below to complete the database adding and binding in the simplest way possible:

- Firstly, use the drag-and-drop method to quickly add all the required databases into TSMaster, as explained in 1.10.3.1 Add by Drag-and-drop Directly.

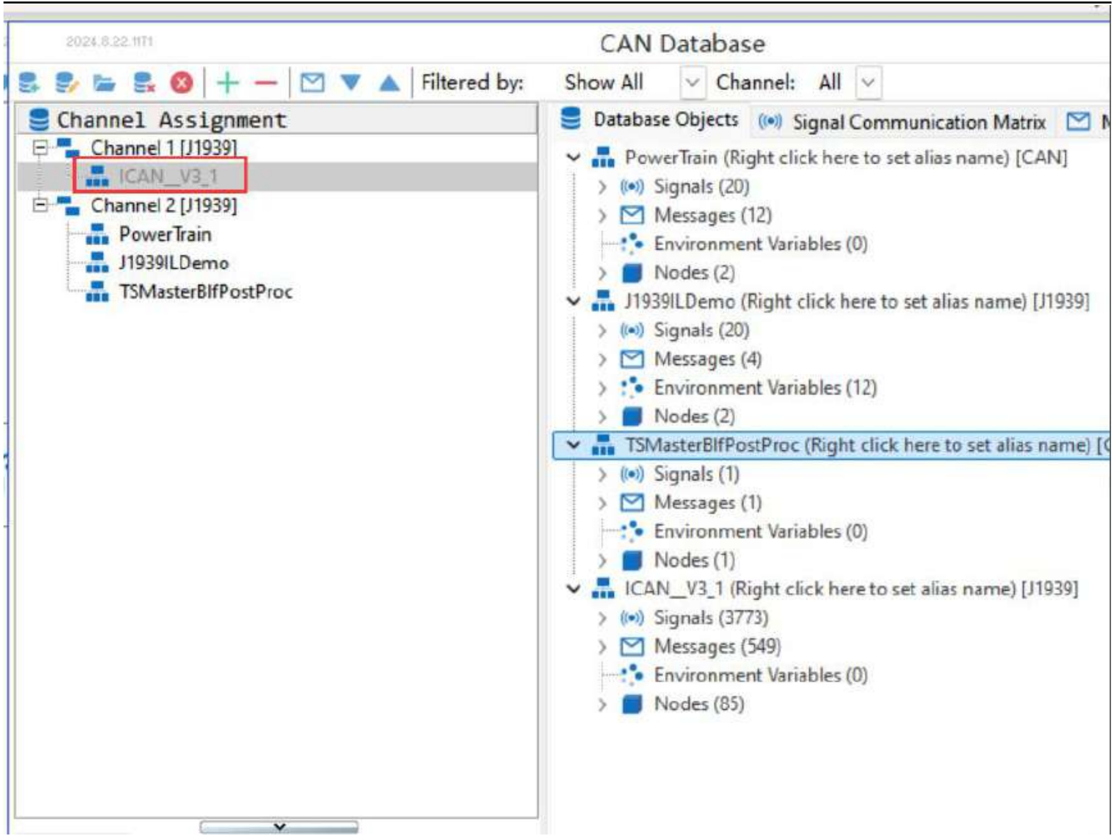
- In the Database Management interface of TSMaster, bind all the newly added databases to their corresponding channels according to your requirements, as described in 1.10.3.2 Add in The Database Window.

1.10.4. Database Name

According to the requirements of the DBC file format, characters such as Chinese and periods are prohibited in DBC network definitions. For example, a file with the name "ICAN 测试 V3.1.dbc", when loaded into VECTOR's CANDB++, will display its name as "ICANV31", directly discarding the Chinese characters and the period, as shown in the figure below:



TSMaster also strictly adheres to this principle. Taking the DBC file shown in the previous image as an example, when loaded into TSMaster, the network name will be displayed as shown in the following figure:


CAN Database

Filtered by: Show All Channel: All

Channel Assignment
- Channel 1 [J1939]
  - ICAN_V3_1
- Channel 2 [J1939]
  - PowerTrain
  - J1939ILDemo
  - TSMasterBlfPostProc

Database Objects | Signal Communication Matrix

- PowerTrain (Right click here to set alias name) [CAN]
  - Signals (20)
  - Messages (12)
  - Environment Variables (0)
  - Nodes (2)
- J1939ILDemo (Right click here to set alias name) [J1939]
  - Signals (20)
  - Messages (4)
  - Environment Variables (12)
  - Nodes (2)
- TSMasterBlfPostProc (Right click here to set alias name) [
  - Signals (1)
  - Messages (1)
  - Environment Variables (0)
  - Nodes (1)
- ICAN_V3_1 (Right click here to set alias name) [J1939]
  - Signals (3773)
  - Messages (549)
  - Environment Variables (0)
  - Nodes (85)

All Chinese characters, periods, or other non-compliant characters are replaced with underscores.

Users must pay attention to the symbol definitions in DBC files during use, as they are directly related to subsequent signal associations and other functionalities.

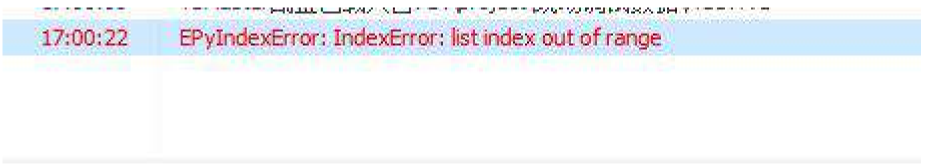
1.10.5. Check Message Definition

1.10.6. Check Signal Definition

1.10.7. Clarification

1.10.7.1. Loading DBC Failed and Prompting 'Index out of range'

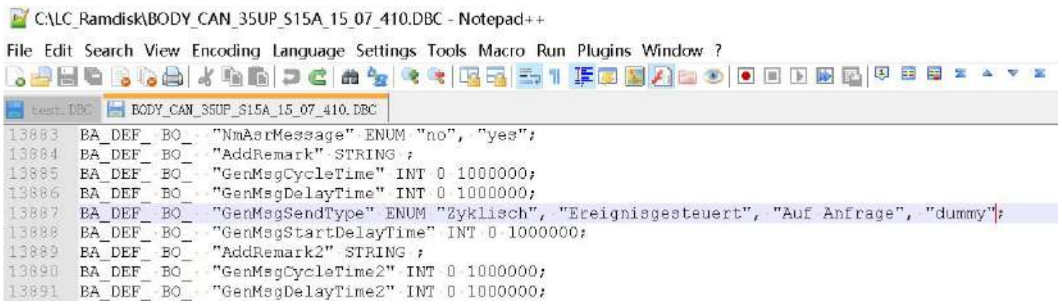
When loading a database, encounter the following error:



```
17:00:22    EPyIndexError: IndexError: list index out of range
```

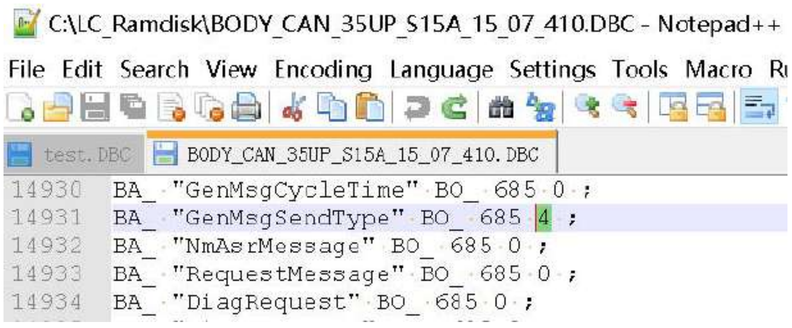
The property values defined in the DBC exceed the actual defined range, as shown in the following example.

When using the DBC editor, some attributes were under-defined, leading to an out-of-bounds subscript for internal signals in the file, causing inconsistencies within itself. As illustrated by the following DBC file, the property 'GenMsgSendType' was only defined with four elements:



```
BA_DEF_ BO_  "NmAsrMessage" ENUM "no", "yes";
BA_DEF_ BO_  "AddRemark" STRING ;
BA_DEF_ BO_  "GenMsgCycleTime" INT 0 1000000;
BA_DEF_ BO_  "GenMsgDelayTime" INT 0 1000000;
BA_DEF_ BO_  "GenMsgSendType" ENUM "Zyklisch", "Ereignisgesteuert", "Auf Anfrage", "dummy";
BA_DEF_ BO_  "GenMsgStartDelayTime" INT 0 1000000;
BA_DEF_ BO_  "AddRemark2" STRING ;
BA_DEF_ BO_  "GenMsgCycleTime2" INT 0 1000000;
BA_DEF_ BO_  "GenMsgDelayTime2" INT 0 1000000;
```

The subscript, however, utilized the fifth element.



```
BA_ "GenMsgCycleTime" BO_ 685 0 ;
BA_ "GenMsgSendType" BO_ 685 4 ;
BA_ "NmAsrMessage" BO_ 685 0 ;
BA_ "RequestMessage" BO_ 685 0 ;
BA_ "DiagRequest" BO_ 685 0 ;
```

Solution: By adding a dummy field to all related attribute fields, they can now be loaded into the DBC normally.

1.10.7.2. The Loaded DBC File is Missing

Problem Description: The DBC file that was loaded and saved with the project was lost after reopening the project, and it requires reloading the DBC file again.

This is because TSMaster does not support Chinese file paths by default. If the DBC file is located under a Chinese path, the above issue will occur.

Solution: Storing the DBC file in a path that contains only English characters can resolve the issue.

1.10.7.2.1. Loading DBC File Fails, Prompting CANDBParser.ini not Existing

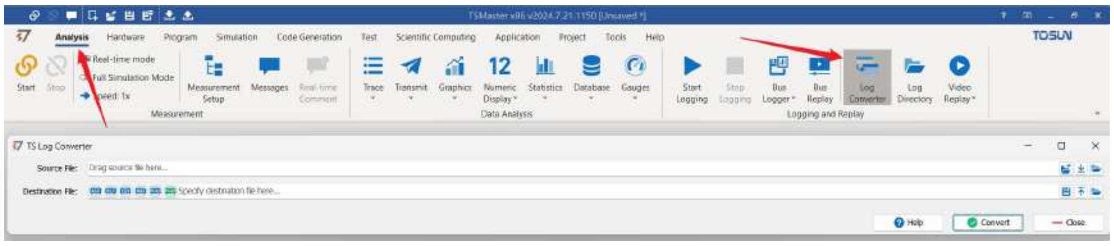
1.11. Message Format Conversion

The TSMaster format converter supports: mutual conversion between blf and asc files, as well as conversion from blf to mat files and from asc to mat files. Note: Avoid using spaces in file names for both message format conversion and message playback.

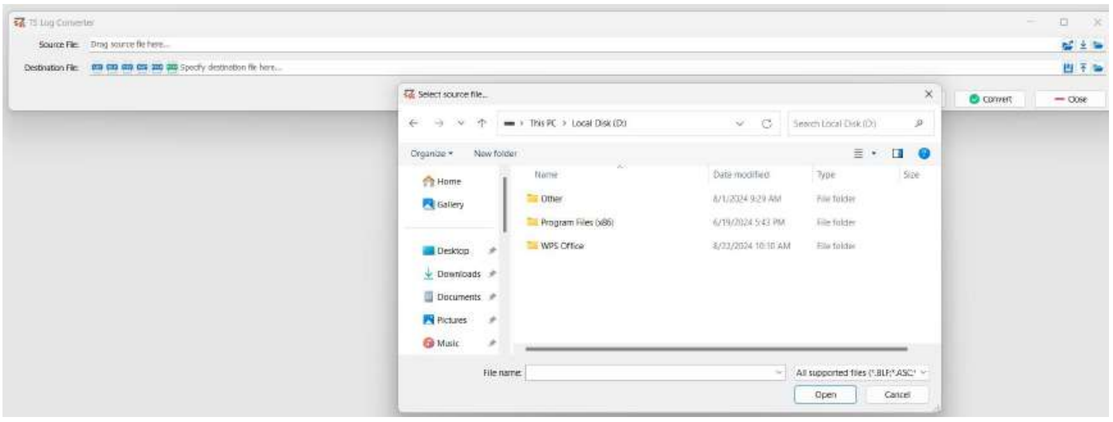
1.11.1. Mutual Conversion between ASC and blf

The process for converting between ASC and blf file formats is as follows:

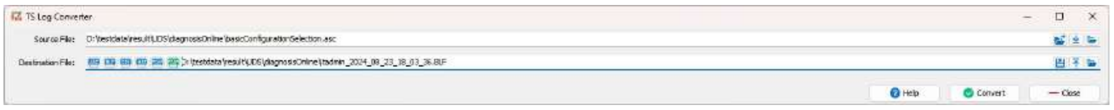
➢ Open the Log Converter:



➢ Add the source file for conversion by selecting the file or drag-and-drop:



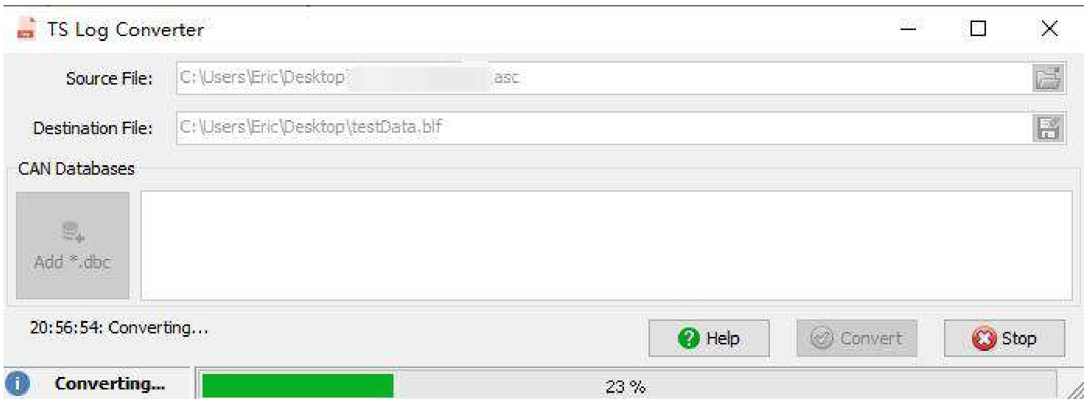
➢ Set the output file path and name.



Source File: D:\testdata\result\UDS\diagnosisOnline\basicConfigurationSelection.asc

Destination File: D:\testdata\result\UDS\diagnosisOnline\tadmin_2024_08_23_18_03_36.BLF

➢ Click Convert to output the target data file, as shown below:

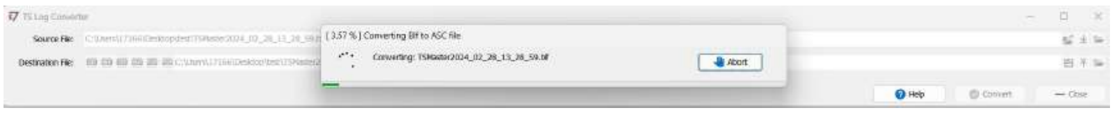


C:\Users\Eric\Desktop asc

C:\Users\Eric\Desktop\testData.blf

20:56:54: Converting...

Converting... 23 %



[3.57 %] Converting Blf to ASC file

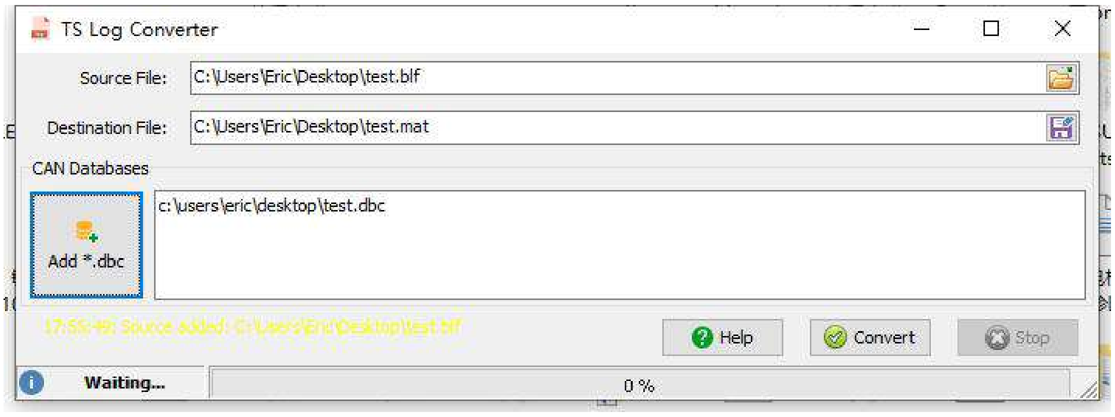
Converting: TSMaster2024_02_28_13_28_59.blf

1.11.1.1. Header Time Format Supported by ASC

Including the following ten formats:

date 21-11-2020 9:46:39.851
date 21-11-2020 9:46:39
date 21/11/2020 8:20:28.851
date 21/11/2020 8:20:28
date 2020-11-21 9:46:39.851
date 2020-11-21 9:46:39
date 2020/11/21 8:20:28.851
date 2020/11/21 8:20:28
date Sun Sep 27 02:03:58.450 pm 2020
date Wed Nov 25 10:36:29 2020

1.11.2. blf to mat File


TS Log Converter

Source File: C:\Users\Eric\Desktop\test.blf

Destination File: C:\Users\Eric\Desktop\test.mat

CAN Databases

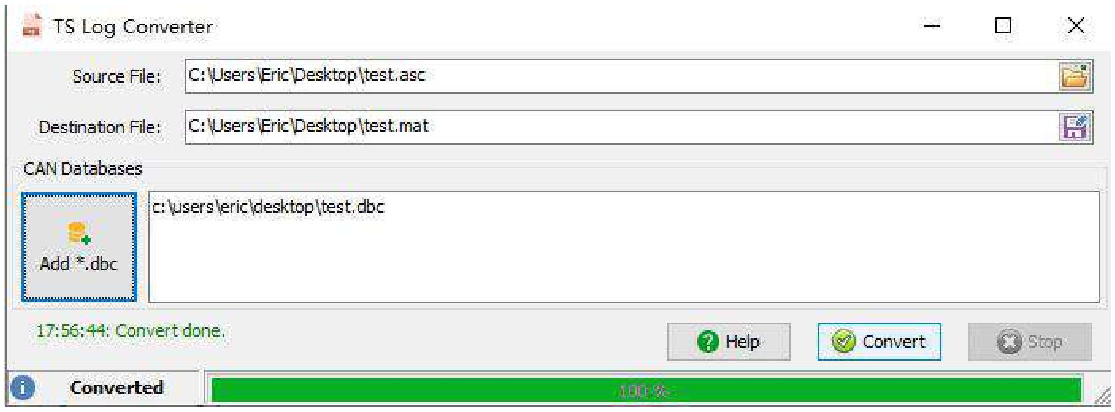
c:\users\eric\desktop\test.dbc

17:55:49: Source added: C:\Users\Eric\Desktop\test.blf

Waiting... 0 %

The conversion to mat format requires adding a .dbc file for parsing.

1.11.3. asc to mat File


TS Log Converter

Source File: C:\Users\Eric\Desktop\test.asc

Destination File: C:\Users\Eric\Desktop\test.mat

CAN Databases

c:\users\eric\desktop\test.dbc

17:56:44: Convert done.

Converted 100 %

The conversion to mat format requires adding a .dbc file for parsing.

1.11.4. Clarification

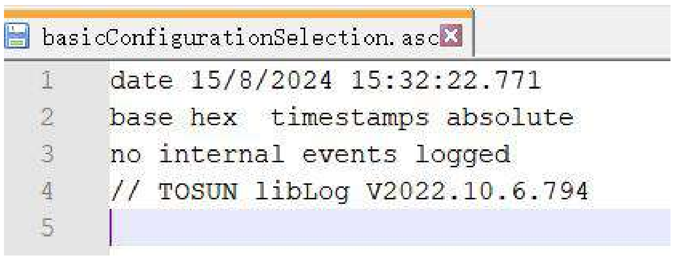
1.11.4.1. File Conversion Failed

Please note that there should be no spaces in the file names of both the source file and the converted file. If there are spaces, it will cause the file conversion to fail with an error type of 6.

1.11.4.2. Why when Some ASC Files Converted to BLF, the Starting Timestamps do not Match

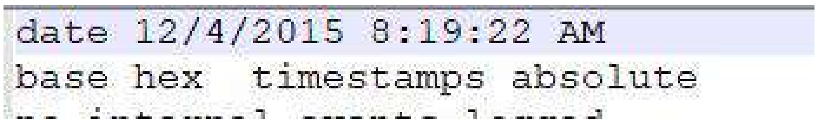
When converting an Asc file to a Blf file, user may encounter a situation where the relative timestamps of the messages are correct, but the absolute starting point of the entire data segment is incorrect. This occurs because the description of the starting point in the ASC file format does not adhere to the format set by TSMaster. Why does this happen?

Because ASC files are in plaintext format, there are no explicit formatting specifications to constrain the header descriptions of ASC files. Each manufacturer can write them in a format that they can interpret, including directly inserting Chinese comments. As shown below:



basicConfigurationSelection.asc

```
date 15/8/2024 15:32:22.771
base hex  timestamps absolute
no internal events logged
// TOSUN libLog V2022.10.6.794

```



```
date 12/4/2015 8:19:22 AM
base hex  timestamps absolute
```

It can be seen that in Image 1, the format description is in the first line, and the timestamp data is in the second line, with the data arrangement completely based on the developer's personal preference. For other images, the timestamp might be in the first line, and the format description in the second line. Each manufacturer defines their own format, with differences in capitalization, delimiters, expressions, and so on, which can reach dozens or even hundreds of variations. It is unrealistic to support all of them. Therefore, TSMaster has built-in support for multiple timestamp formats. For details, please refer to the section: ASC Support for Header Time Formats.

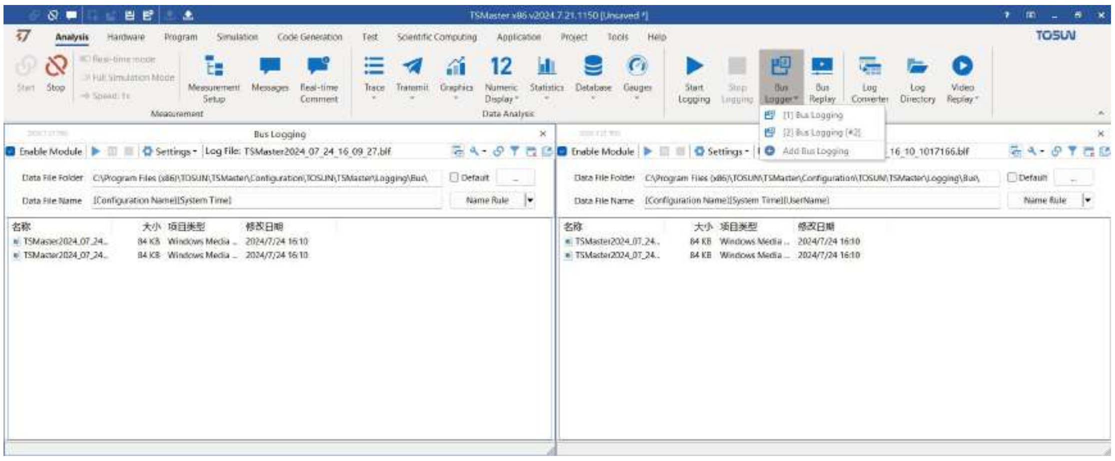
When users encounter other ASC files, before conversion, please first confirm the data format of the starting time. If it is not in the supported format, please manually adjust it to the format mentioned above so that the system can correctly identify the starting time point of the data block.

1.12. Message Recording

1.12.1. Bus Recording Module

The path to the TSMaster Bus Recording Module is as follows: Analysis -> Logging and Replay. This module has the following characteristics:

- Multiple recording files can be added to record simultaneously.
- Each recording file can be individually configured with parameters such as storage path, file size limit, naming convention, and so on.
- The storage file names are flexible and configurable, allowing users to set the naming rules according to their requirements.



Log File: TSMaster2024_07_24_16_09_27.blf

Data File Folder: C:\Program Files (x86)\TOSUN\TSMaster\Configuration\TOSUN\TSMaster\Logging\Bus\ ☐ Default

Data File Name: [Configuration Name][System Time]

```
名称                   大小   项目类型           修改日期
TSMaster2024_07_24...  84 KB  Windows Media ...  2024/7/24 16:10
TSMaster2024_07_24...  84 KB  Windows Media ...  2024/7/24 16:10
```

Log File: ..._16_10_1017166.blf

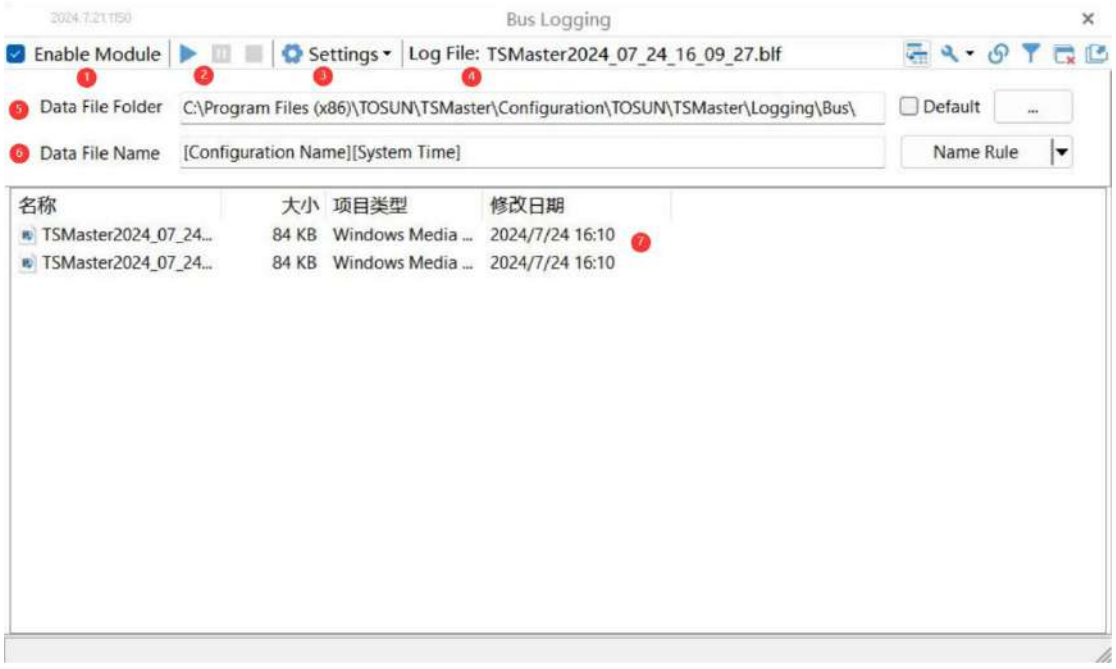
Data File Folder: C:\Program Files (x86)\TOSUN\TSMaster\Configuration\TOSUN\TSMaster\Logging\Bus\ ☐ Default

Data File Name: [Configuration Name][System Time][UserName]

```
名称                   大小   项目类型           修改日期
TSMaster2024_07_24...  84 KB  Windows Media ...  2024/7/24 16:10
TSMaster2024_07_24...  84 KB  Windows Media ...  2024/7/24 16:10
```

1.12.2. Log File Parameters

A logging module, which primarily includes the following configurable parameters:

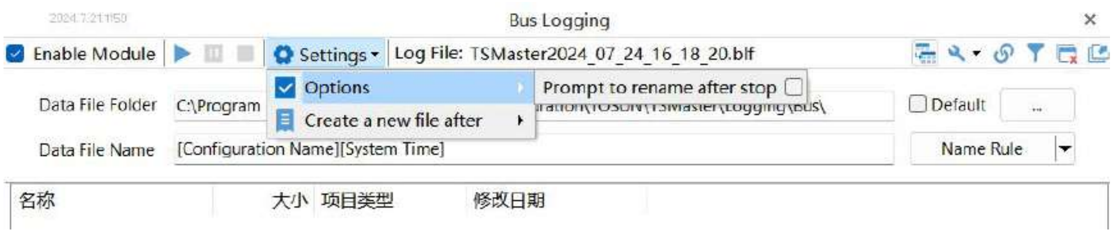


➢ Area 1: Whether to enable this logging module. If you choose not to enable it, this module will not be activated when logging is initiated.

➢ Area 2: The operation area, which includes buttons for starting, pausing, and stopping the recording functions.

➢ Area 3: Settings. Expand this button as follows, which mainly includes the following configurations:

1. Whether to pop up a prompt to rename the window when stopping the recording.
2. Is the size of the recorded file unlimited or a specified data size.



➢ Area 4: Logging file name. This area is read-only, and when the naming rules in area 6 change, this area will synchronously update the current file name.

➢ Area 5: Data file folder. Used to set the folder path for storing data files. If you choose the default configuration, select the default directory for storing data in the project.

➢ Area 6: Data file name and naming rules. Users can flexibly set the names of data files by combining naming rules. The main naming rules include the following:

1. UserName is the username of the current computer;
2. Computer Name is the device name of the current computer;
3. System Time is the current system time;
4. Measurement Index is the index of the record, which defaults to start from 000. When the user adds multiple bus records, the index of the record automatically increments;
5. Measurement Start Time is the begin time for the recording;
6. Configuration Name is the name for current configuration.

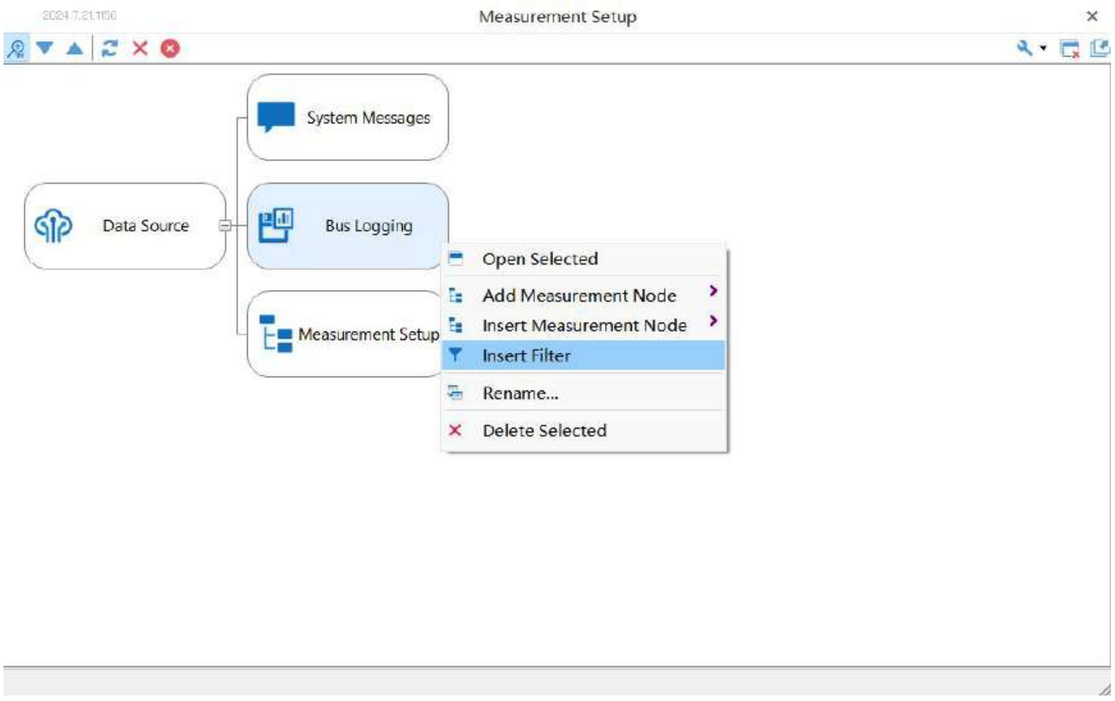
For example, if the configuration is set as [UserName][Measurement Index], then the corresponding data file name would be: XYY001.blf. If the rule is modified to [UserName]_[Measurement Index], then the corresponding data file name would be: XYY_001.blf.

➢ Area 7: Display the logging files contained in the currently selected data folder.

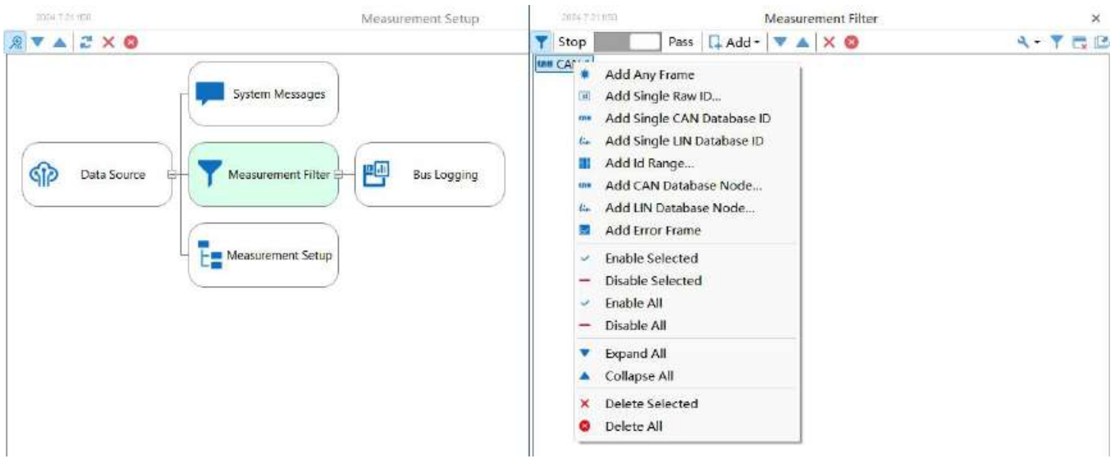
Note: When multiple modules use the same file path to store data, using the same data file name will cause file conflicts and prevent the starting of data recording.

1.12.3. Add Logging Filter

In the Measurement Settings module, the user can right-click on a selected bus logging to insert a filter for that bus logging.

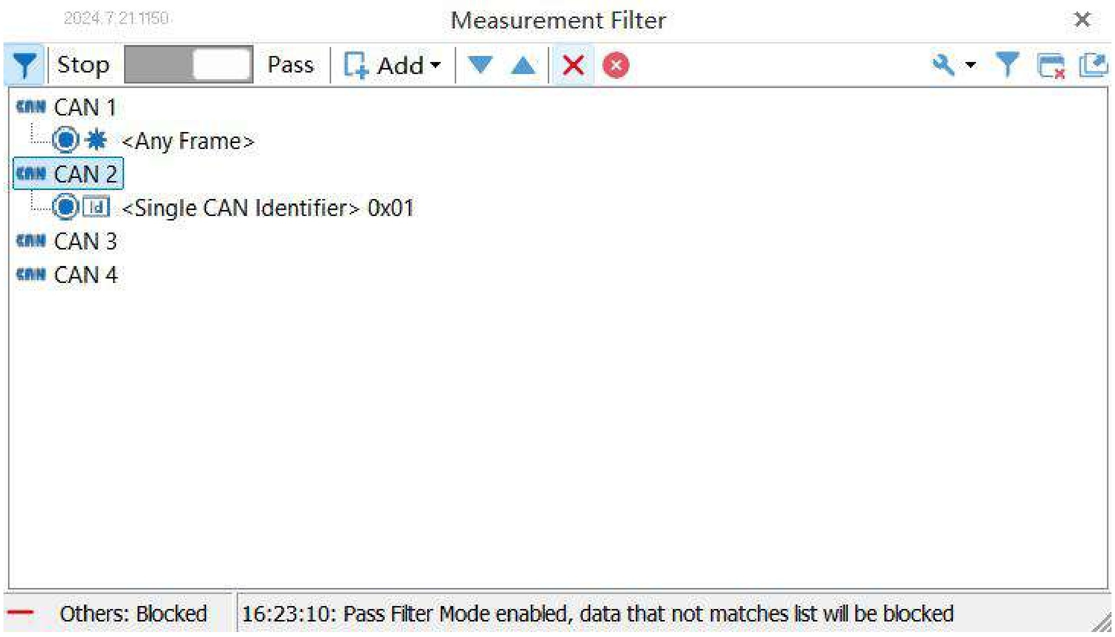


The user can double-click on a test filter with the left mouse button to open the configuration page of the test filter. On this page, a list of channels used in the current project is displayed. The user can select a specific channel, right-click on it, and add the message information that needs to be filtered.



When the user wishes to log message information from multiple channels in the bus logging, they simply need to add a message filter for each channel. After starting the program, only channels with a message filter added will be logged, while channels without a message filter will not be logged, as shown in the following figure:

When a user wishes to record message information from multiple channels in this bus logging, all that is needed is to add message filters for the channels. After starting the program, only the channels with message filters will be logged, while those without will not be logged, as shown in the figure below:

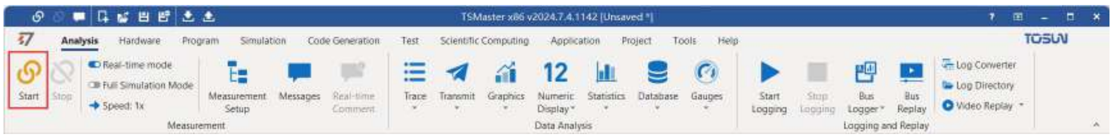


Others: Blocked

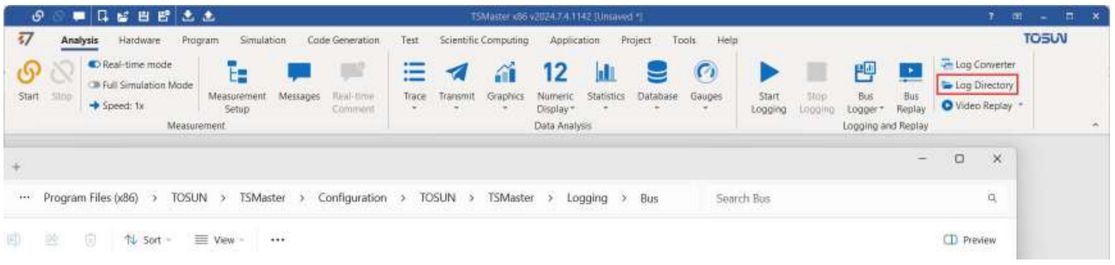
16:23:10: Pass Filter Mode enabled, data that not matches list will be blocked

In this example, only CAN 1 and CAN 2 will be logged.

After configuring the filters, you can start the program and send messages for bus logging.

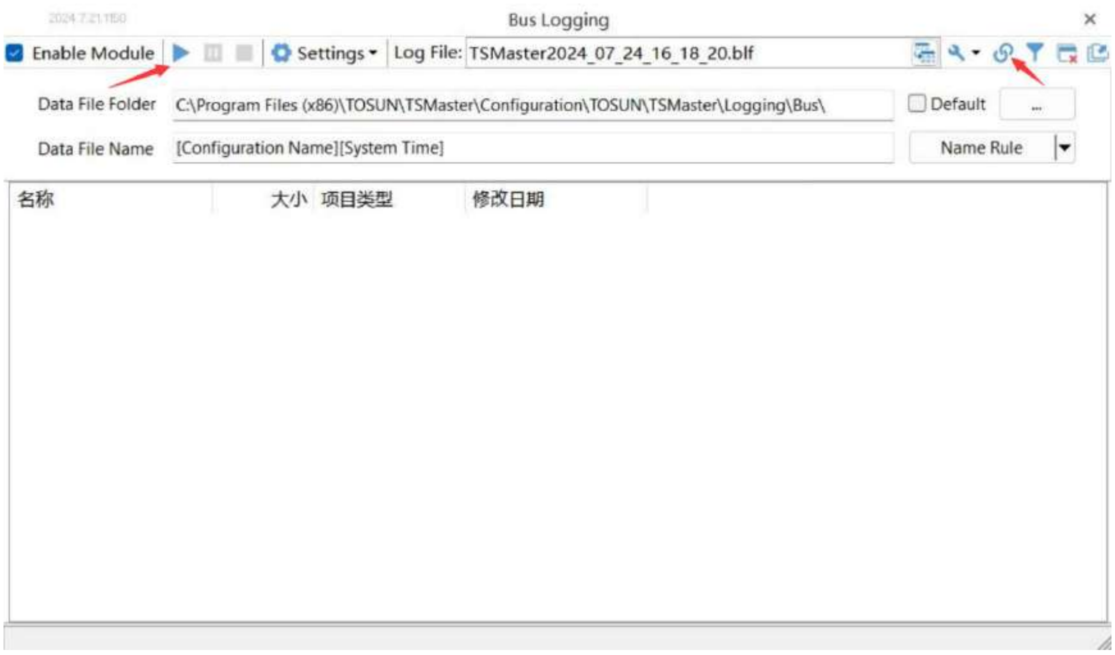


After the program stops, the logged files will be automatically saved to the pre-set path. The user can directly open the folder at the pre-set path by clicking the "Log Directory" button. If the user has pre-set multiple storage paths, clicking the "Log Directory" button will open all folders where the logs are saved.


TSMaster x86 v2024.7.4.1142 [Unsaved *]
Analysis
Start
Stop
Log Converter
Log Directory
Video Replay
Logging and Replay
Program Files (x86) › TOSUN › TSMaster › Configuration › TOSUN › TSMaster › Logging › Bus
Search Bus
Sort
View
Preview

1.12.4. Start Automatic Logging

Data logging can be started manually or automatically. The logging module can be launched independently or simultaneously. Clicking the ▶ button can manually start the logging. Users can also click the button at the position shown in the previous figure to set the logging to start automatically, which means the logging will automatically start when the program starts. When the user needs to start multiple logging simultaneously, they can simply enable the automatic start for multiple bus logging, and when the program starts, multiple logging will begin simultaneously, as shown below:



Bus Logging

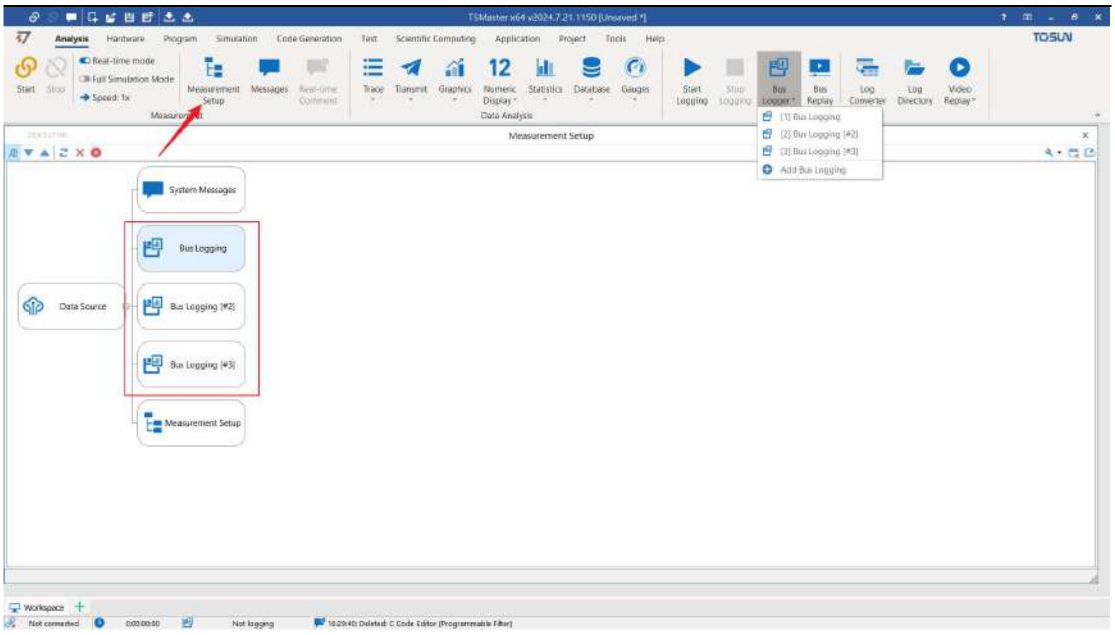
Enable Module | Settings | Log File: TSMaster2024_07_24_16_18_20.blf

Data File Folder: C:\Program Files (x86)\TOSUN\TSMaster\Configuration\TOSUN\TSMaster\Logging\Bus\ | Default

Data File Name: [Configuration Name][System Time] | Name Rule

名称	大小	项目类型	修改日期

After the user adds multiple bus logging, the bus logs added by the user will be displayed in the test settings, as shown in the figure below:

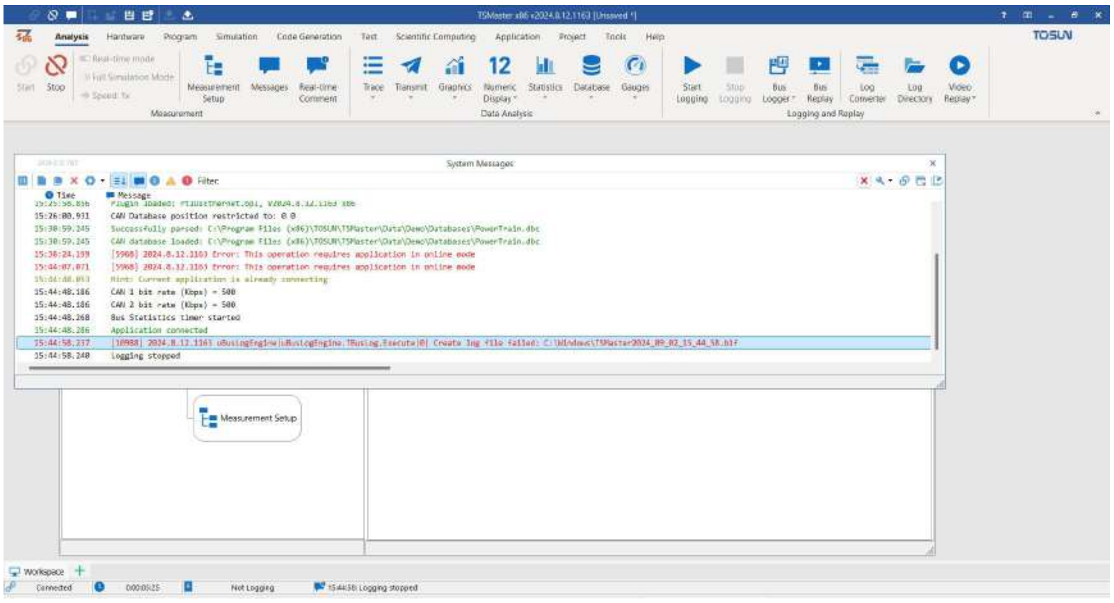


1.12.5. Common Errors

First of all, in case of an error, users should get into the habit of checking the system information window, where they can usually find the answer.

1.12.5.1. Path Permission Management, Failed to Start Logging - Error 1

If the user selects a data storage folder that has been configured with management permissions that do not allow the creation of new data files within it, the data logging will fail to start when the user attempts to do so. The prompt is as follows:



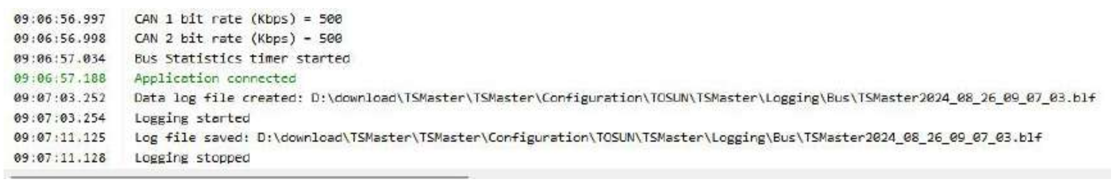
```
Time            Message
15:25:58.856    Plugin loaded: PluginEthernet.dll, v2024.8.12.1163 x86
15:26:00.931    CAN Database position restricted to: 0 0
15:30:59.245    Successfully parsed: C:\Program Files (x86)\TOSUN\TSMaster\Data\Demo\Databases\PowerTrain.dbc
15:30:59.245    CAN database loaded: C:\Program Files (x86)\TOSUN\TSMaster\Data\Demo\Databases\PowerTrain.dbc
15:36:24.159    [5968] 2024.8.12.1163 Error: This operation requires application in online mode
15:44:07.071    [5968] 2024.8.12.1163 Error: This operation requires application in online mode
15:44:48.053    Hint: Current application is already connecting
15:44:48.186    CAN 1 bit rate (Kbps) = 500
15:44:48.186    CAN 2 bit rate (Kbps) = 500
15:44:48.268    Bus Statistics timer started
15:44:48.286    Application connected
15:44:58.237    [10988] 2024.8.12.1163 uBusLogEngine|uBusLogEngine.TBusLog.Execute|0| Create log file failed: C:\Windows\TSMaster2024_09_02_15_44_58.blf
15:44:58.240    Logging stopped
```

<span style="color:red">**Solution:**</span>

Reset the data directory and select a directory that allows modifications.

1.12.5.2. Failed to Start Logging for a Project Copied from Another User - Error 2

When a user uses a configuration project from another user, and if the data logging folder configured in that project selects some special directories, such as the user's desktop location, copying it to the current computer may result in failure to create the data directory and data folder due to differences in user names and other factors. This is shown as follows:



```
09:06:56.997    CAN 1 bit rate (Kbps) = 500
09:06:56.998    CAN 2 bit rate (Kbps) = 500
09:06:57.034    Bus Statistics timer started
09:06:57.188    Application connected
09:07:03.252    Data log file created: D:\download\TSMaster\TSMaster\Configuration\TOSUN\TSMaster\Logging\Bus\TSMaster2024_08_26_09_07_03.blf
09:07:03.254    Logging started
09:07:11.125    Log file saved: D:\download\TSMaster\TSMaster\Configuration\TOSUN\TSMaster\Logging\Bus\TSMaster2024_08_26_09_07_03.blf
09:07:11.128    Logging stopped
```

Because the project was copied from a user named "seven", under the current user, the path "seven\Desktop" does not exist. Additionally, because it is located in the C drive directory, the creation of new directories is not allowed, resulting in a failure to create the data file.

Solution:

Reset the data directory. In order to prevent issues with path invalidation when sharing the project with other users, it is recommended to set a directory related to the project folder instead of using a directory tied to a specific computer or user.

1.13. Message Replay

1.13.1. Supported Format

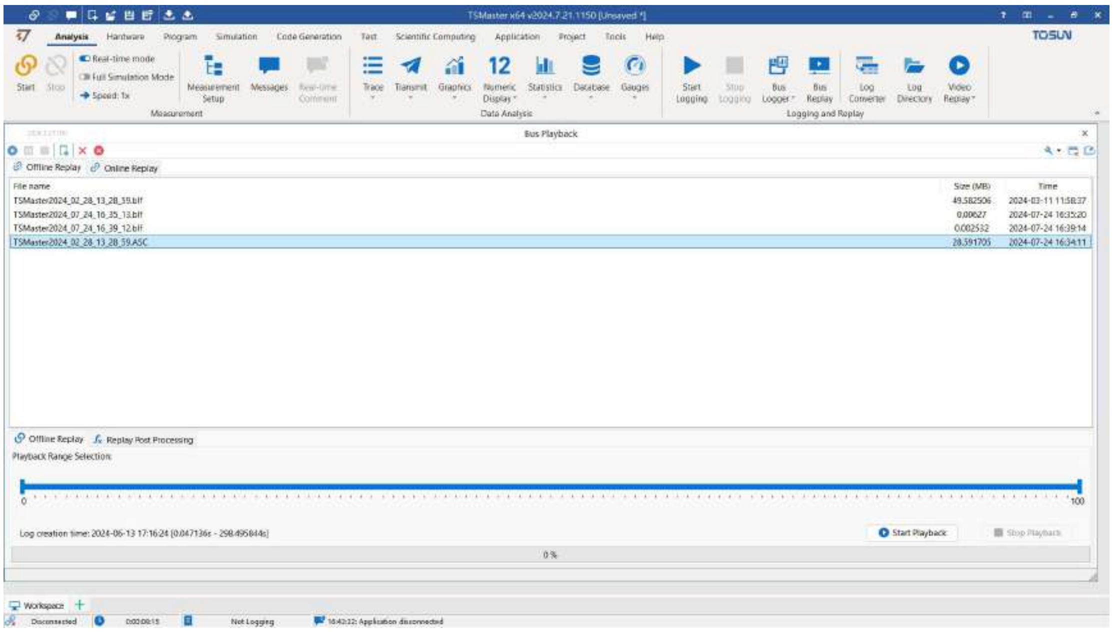
TSMaster's message replay functionality by default supports the BLF format (with support for other formats to be added in the future). If you need to analyze log files in other data formats, you will need to convert them from those formats to the BLF format using a file converter. The steps for format conversion can be found in the previous chapter.

1.13.2. Offline Replay

Offline replay, also commonly referred to as viewing recorded messages, fully simulates the process of receiving messages. Users can directly view the message records in the Trace window. Similarly, all the properties of the Trace window, such as filtering, are also effective.

1.13.2.1. Basic Steps for Offline Replay

In Analysis, select Bus Replay->Offline Replay, and select the replay file. Then select the playback range and click “Start Playback”.

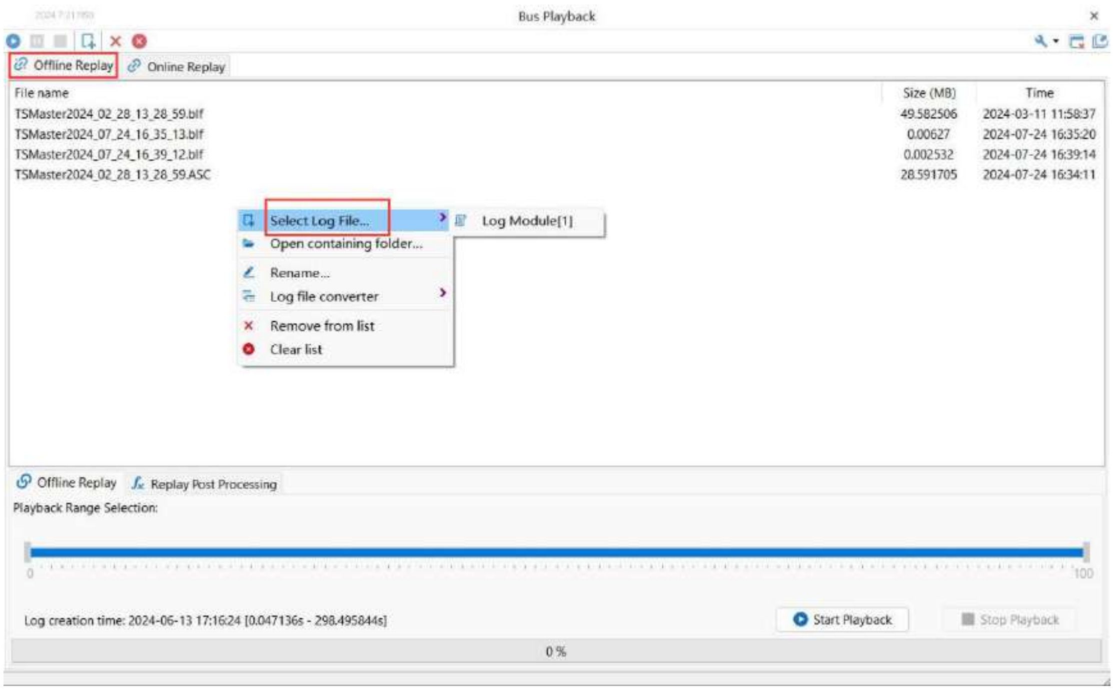


File name	Size (MB)	Time
TSMaster2024_02_28_13_28_35.blf	49.582506	2024-03-11 11:58:37
TSMaster2024_07_24_16_35_13.blf	0.00627	2024-07-24 16:35:20
TSMaster2024_07_24_16_39_12.blf	0.002532	2024-07-24 16:39:14
TSMaster2024_02_28_13_28_59.ASC	28.591705	2024-07-24 16:34:11

Log creation time: 2024-06-13 17:16:24 [0.047136s - 298.495844s]

1.13.2.2. Add Replay File

1.13.2.2.1. Add from Replay Window



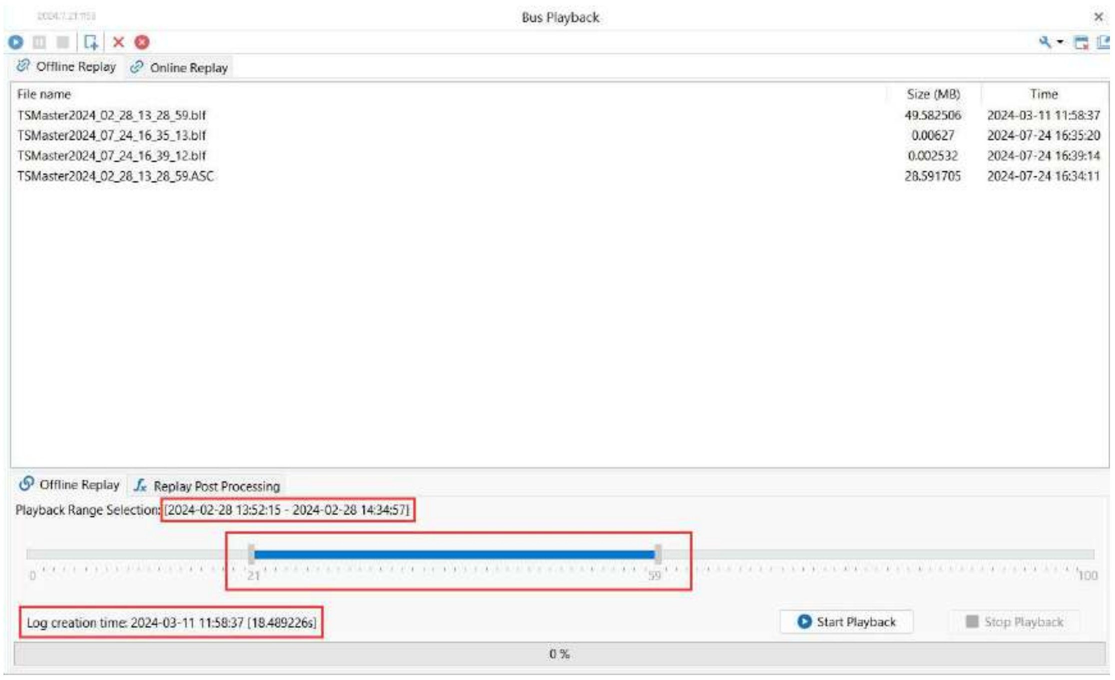
File name	Size (MB)	Time
TSMaster2024_02_28_13_28_59.blf	49.582506	2024-03-11 11:58:37
TSMaster2024_07_24_16_35_13.blf	0.00627	2024-07-24 16:35:20
TSMaster2024_07_24_16_39_12.blf	0.002532	2024-07-24 16:39:14
TSMaster2024_02_28_13_28_59.ASC	28.591705	2024-07-24 16:34:11

Log creation time: 2024-06-13 17:16:24 [0.047136s - 298.495844s]

1.13.2.2.2. Drag and Drop to Add

Select a log file on your desktop, drag it into the TSMaster software area, release the mouse button, and TSMaster will automatically replay the message while also adding it to the message replay management window.

1.13.2.3. Select Message Range



File name	Size (MB)	Time
TSMaster2024_02_28_13_28_59.blf	49.582506	2024-03-11 11:58:37
TSMaster2024_07_24_16_35_13.blf	0.00627	2024-07-24 16:35:20
TSMaster2024_07_24_16_39_12.blf	0.002532	2024-07-24 16:39:14
TSMaster2024_02_28_13_28_59.ASC	28.591705	2024-07-24 16:34:11

Playback Range Selection: [2024-02-28 13:52:15 - 2024-02-28 14:34:57]

Log creation time: 2024-03-11 11:58:37 [18.489226s]

As shown in the figure, the log creation time is displayed. By selecting the playback range, the time range will be automatically calculated.

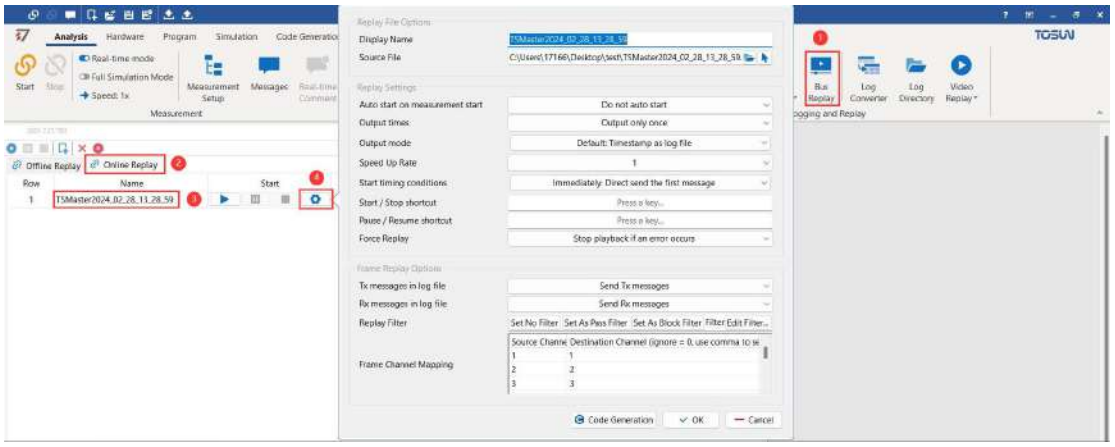
Because the Trace window can only display a maximum of 9999 frames of messages at a time on one screen, it is necessary to select the message range appropriately when analyzing the log file.

In the new version, a script module will be added to the message replay module to provide greater flexibility for message replay analysis.

1.13.3. Online Replay

1.13.3.1. Online Replay Configuration

Online replay is also colloquially referred to as "data re-injection into the bus." To provide users with the greatest possible flexibility, the following configuration interface is provided:



Replay File Options

Display Name	TSMaster2024_02_28_13_28_59
Source File	C:\Users\17166\Desktop\test\TSMaster2024_02_28_13_28_59

Replay Settings

Auto start on measurement start	Do not auto start
Output times	Output only once
Output mode	Default: Timestamp as log file
Speed Up Rate	1
Start timing conditions	Immediately: Direct send the first message
Start / Stop shortcut	Press a key...
Pause / Resume shortcut	Press a key...
Force Replay	Stop playback if an error occurs

Frame Replay Options

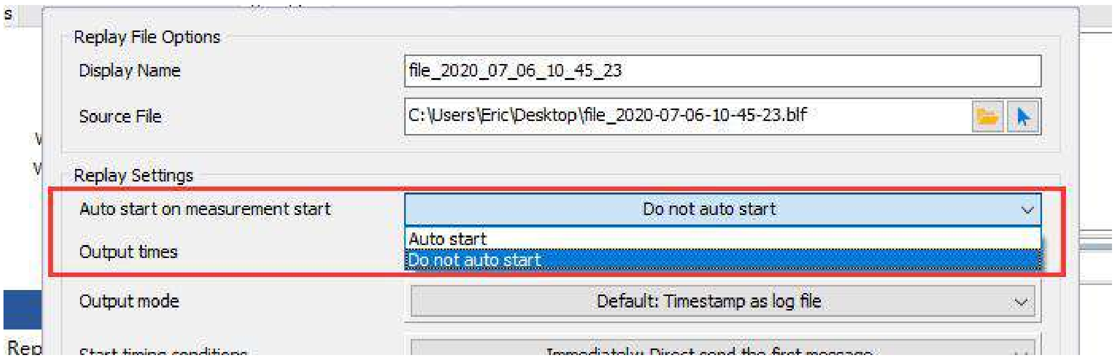
Tx messages in log file	Send Tx messages
Rx messages in log file	Send Rx messages
Replay Filter	Set No Filter · Set As Pass Filter · Set As Block Filter · Filter Edit Filter...

Frame Channel Mapping

Source Channe	Destination Channel (ignore = 0, use comma to se
1	1
2	2
3	3

As shown in the configuration interface, it mainly includes the following configuration parameters:

1.13.3.1.1. Whether to Automatically Start Online Replay



Replay File Options

Display Name	file_2020_07_06_10_45_23
Source File	C:\Users\Eric\Desktop\file_2020-07-06-10-45-23.blf

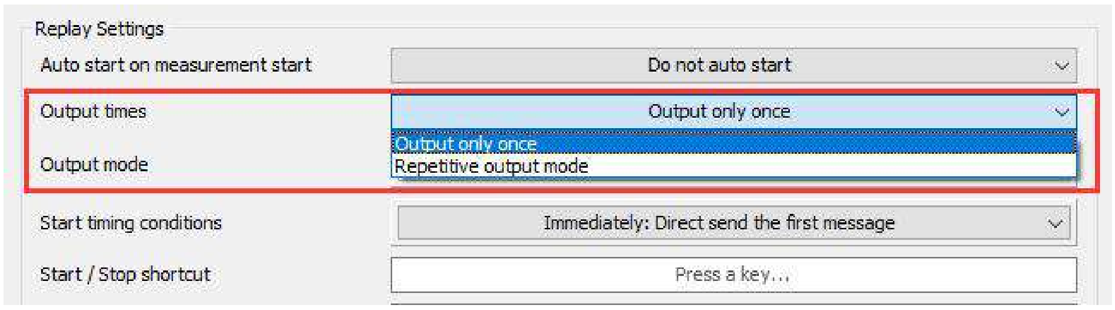
Replay Settings

Auto start on measurement start	Do not auto start (Auto start / Do not auto start)
Output times	
Output mode	Default: Timestamp as log file
Start timing conditions	Immediately: Direct send the first message

Auto start: automatically start message replay after device connection.

Do not auto start: do not immediately start message replay after device connection, but allow the user to manually start it in the replay interface.

1.13.3.1.2. Select the Number of Outputs

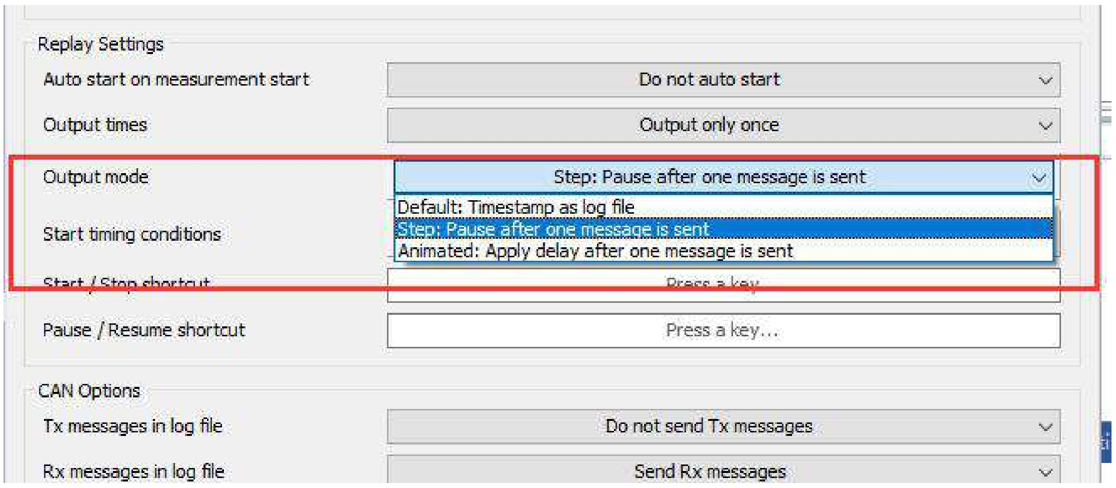

Replay Settings

Auto start on measurement start	Do not auto start
Output times	Output only once
Output mode	
Start timing conditions	Immediately: Direct send the first message
Start / Stop shortcut	Press a key...

Output only once: only replay for one time.

Repetitive output mode: replay message records in a loop.

1.13.3.1.3. Replay Message Records in a Loop


Replay Settings

Auto start on measurement start	Do not auto start
Output times	Output only once
Output mode	Step: Pause after one message is sent
Start timing conditions	
Start / Stop shortcut	Press a key...
Pause / Resume shortcut	Press a key...

CAN Options

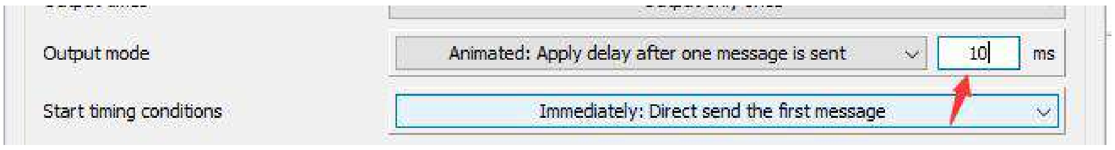
Tx messages in log file	Do not send Tx messages
Rx messages in log file	Send Rx messages

Default: Timestamp as log file
Step: Pause after one message is sent
Animated: Apply delay after one message is sent

Default: TimeStamp as log file: replay messages based on timestamps in the log file.

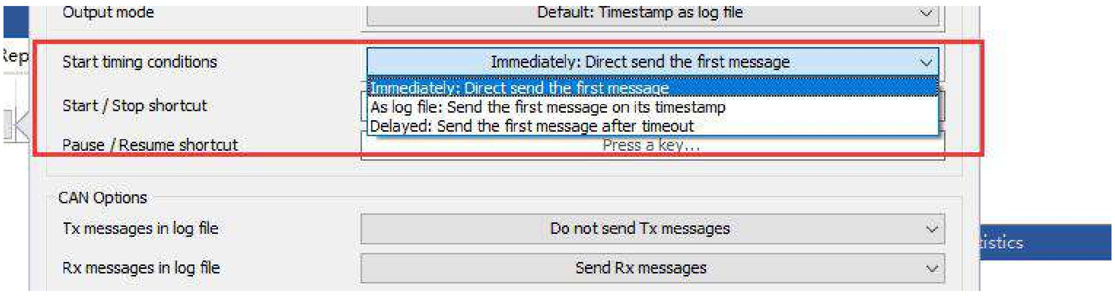
Step: Pause after one message is sent: require user to click, and each time only sends one frame of the message in the log file .

Animated: Apply delay after one message is sent: instead of basing on the timestamps in the log file, the user sets a message event interval, and the messages are replayed according to this interval. As shown below:


Output mode: Animated: Apply delay after one message is sent, 10 ms
Start timing conditions: Immediately: Direct send the first message

The above figure indicates that the messages are replayed with a time interval of 10ms between them.

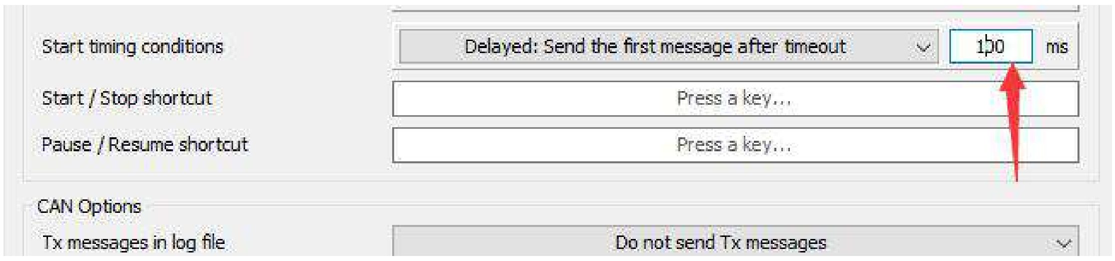
1.13.3.1.4. Select Start Time Mode


Output mode: Default: Timestamp as log file
Start timing conditions: Immediately: Direct send the first message
Immediately: Direct send the first message
As log file: Send the first message on its timestamp
Delayed: Send the first message after timeout
Start / Stop shortcut
Pause / Resume shortcut: Press a key...
CAN Options
Tx messages in log file: Do not send Tx messages
Rx messages in log file: Send Rx messages

Immediately: Direct send the first message: after the user selects to replay messages, the replay starts immediately.

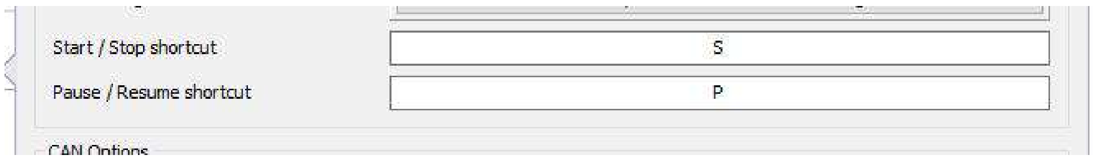
As log file: Send the first message on its timestamp: Determine the start time for replaying messages based on the timestamps in the log file. For example, if the timestamp of the first message frame in the log file is 15s, then after connecting to the device, the replay of messages will start after 15 seconds.

Delayed: Send the first message after timeout: the user manually configures a delay time. After connecting to the device, the replay of messages will start after the set delay interval has passed. As shown below:


Start timing conditions: Delayed: Send the first message after timeout, 100 ms
Start / Stop shortcut: Press a key...
Pause / Resume shortcut: Press a key...
CAN Options
Tx messages in log file: Do not send Tx messages

The above figure indicates that after the user initiates the playback of messages, there is a delay of 100ms before the actual replay of messages begins.

1.13.3.1.5. Start/Pause Shortcut



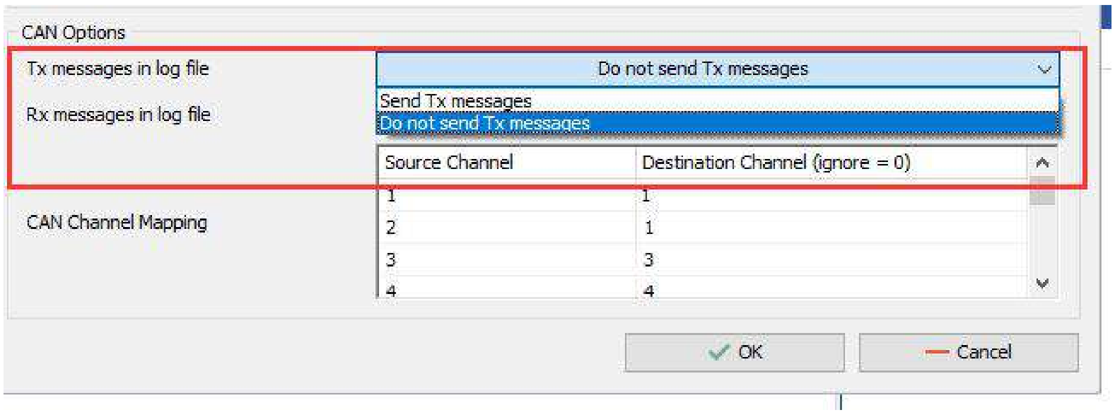
Start / Stop shortcut: S

Pause / Resume shortcut: P

CAN Options

Allow the user to set a shortcut key for starting/pausing replay, as shown above: when the user presses the 'S' key on the keyboard, the replay starts; when the user presses the 'P' key on the keyboard, the replay pauses.

1.13.3.1.6. Replay TX/RX Messages Selection



CAN Options

Tx messages in log file: Do not send Tx messages

Send Tx messages

Do not send Tx messages

Rx messages in log file

CAN Channel Mapping

Source Channel	Destination Channel (ignore = 0)
1	1
2	1
3	3
4	4

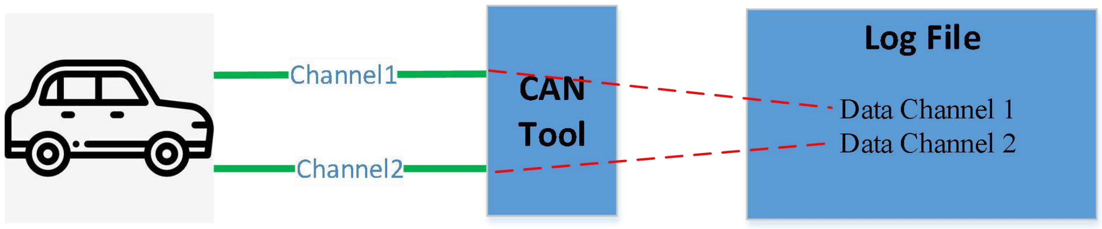
Send Tx messages: the replay includes messages with a direction of "send".

Do not send Tx messages: the replay does not include messages with a direction of "send".

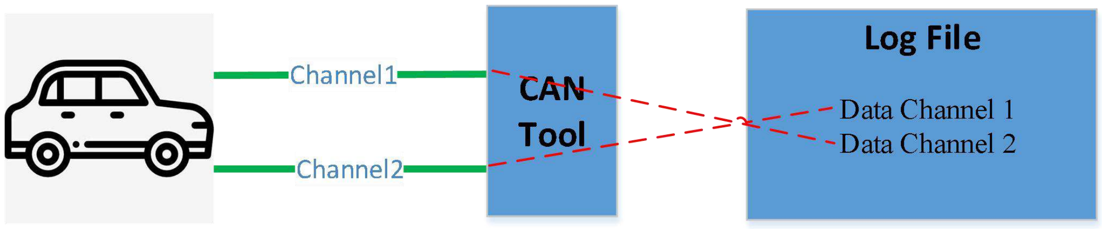
1.13.3.1.7. Replay Channel Selection

To provide users with maximum flexibility, TSMaster's replay module offers replay channel mapping. This is primarily designed to address issues in the following application scenarios:

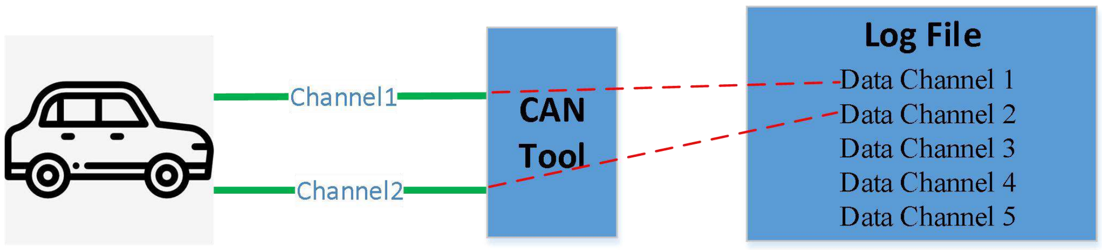
1. The physical channels are already connected, but there is a need to flexibly switch channels, as shown in the following diagram:



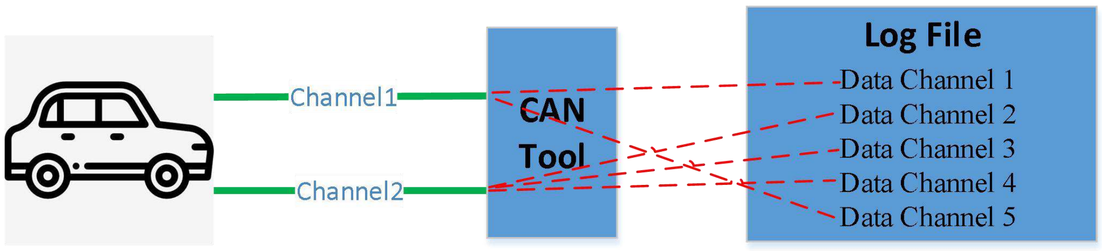
The log messages contain data from both Channel 1 and Channel 2. Ideally, the data from Channel 1 should be played back on the corresponding channel on the tool side. However, if the data from Channel 2 needs to be played back on Channel 1 of the CAN tool, and the data from Channel 1 needs to be played back on Channel 2 of the CAN tool, then mapping is required, as shown below:



2. Insufficient actual channels: the original log messages utilize five channels (1, 2, 3, 4, 5), but currently, there are only two channels available, as shown below:

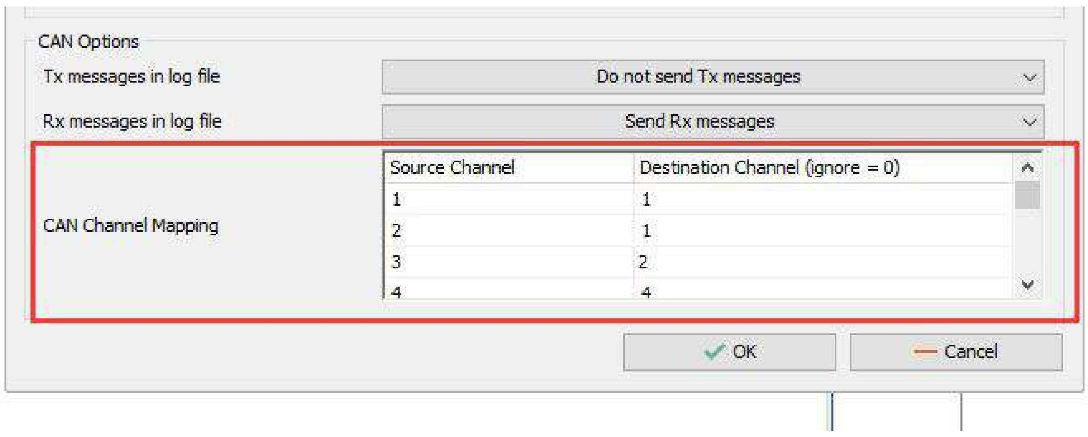


In this scenario, the data from channels 3, 4, and 5 have no channel to play on. However, by using channel mapping, the user can choose to play the data from any of these channels on any channel of the CAN tool, as shown below:



With the above channel mapping, the data from channels 2, 3, and 4 are mapped to play on channel 2 of the CAN tool, while the data from channels 1 and 5 are mapped to play on channel 1 of the CAN tool.

The replay channel mapping configuration interface is shown below:



CAN Options

Tx messages in log file: Do not send Tx messages

Rx messages in log file: Send Rx messages

CAN Channel Mapping

Source Channel	Destination Channel (ignore = 0)
1	1
2	1
3	2
4	4

Source Channel: the data channel in the log file.

Destination Channel: the data channel in the CAN tool of TSMaster.

The configuration shown above indicates that: data from data channels 1 and 2 in the log file are replayed on channel 1 of the CAN tool, and data from data channel 3 is replayed on channel 2 of the CAN tool.

1.13.4. Application Cases

1.13.4.1. Automatically Filter Certain Messages during Replay

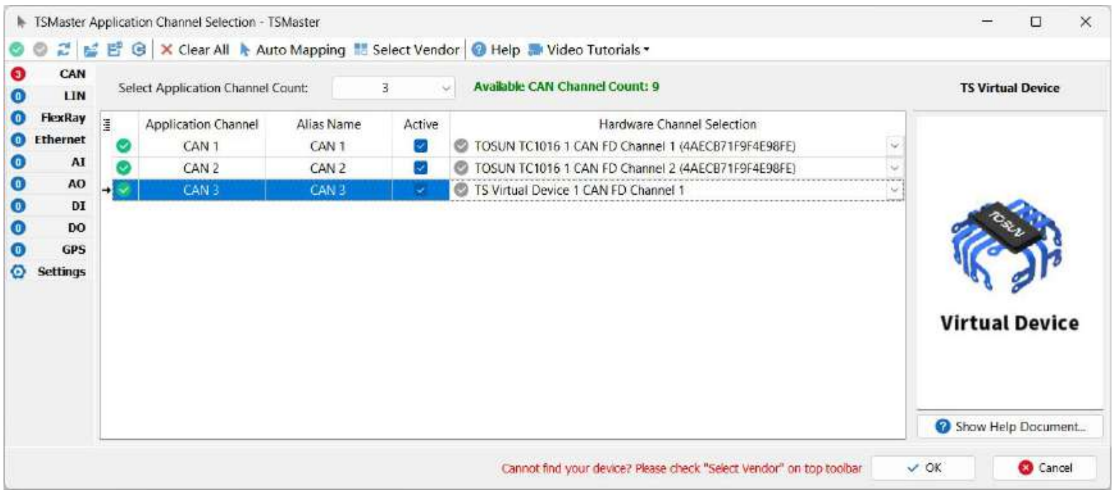
Based on the previous introduction, automatic replay of messages provides mechanisms such as channel selection, and the choice to replay RX, TX, etc. However, during user operation, there is often a need to selectively replay only a portion of the messages or selectively filter some messages. Because the number of replayed messages can be quite large, ID-based replay filtering has not been integrated into the online replay module. However, through the use of TSMaster's C script tool, users can still achieve the functionality of online replay filtering based on IDs, effectively filtering unwanted messages during replay.

➢ **Fundamental approach:**

Send messages with IDs that need to be filtered to a virtual channel, so that these messages will not be replayed to the physical channel in a real sense.

➢ **Detailed steps:**

1. In the TSMaster hardware configuration, add an additional channel and select this channel as a virtual channel, as shown below:



TSMaster Application Channel Selection - TSMaster

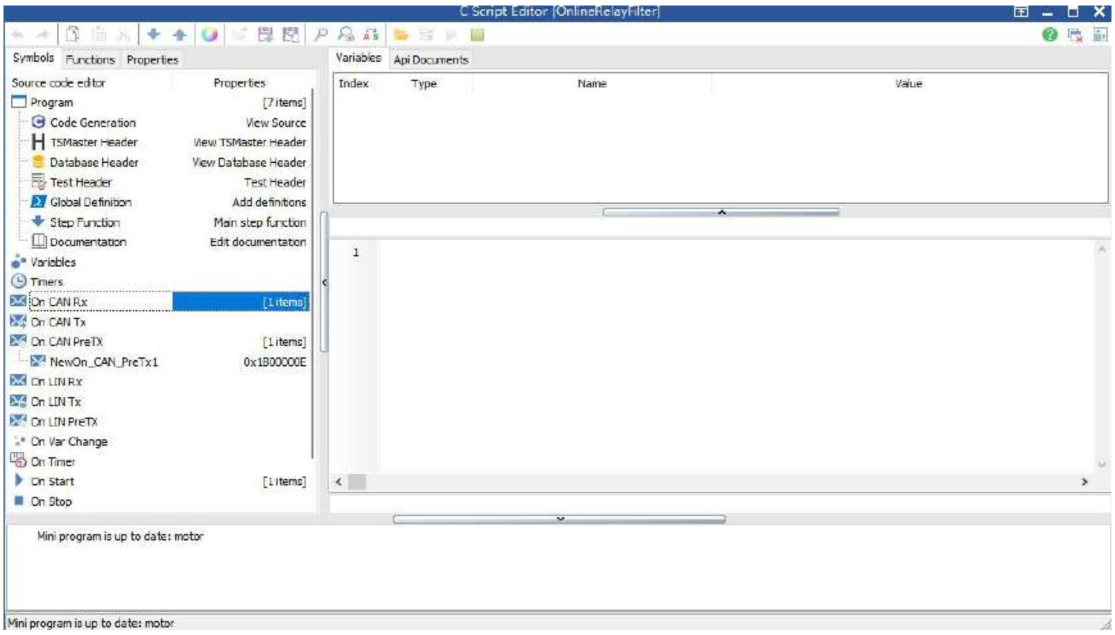
Select Application Channel Count: 3 — Available CAN Channel Count: 9

Application Channel	Alias Name	Active	Hardware Channel Selection
CAN 1	CAN 1	✓	TOSUN TC1016 1 CAN FD Channel 1 (4AECB71F9F4E98FE)
CAN 2	CAN 2	✓	TOSUN TC1016 1 CAN FD Channel 2 (4AECB71F9F4E98FE)
CAN 3	CAN 3	✓	TS Virtual Device 1 CAN FD Channel 1

Cannot find your device? Please check "Select Vendor" on top toolbar

In this example, there are three application channels. CAN 1 and CAN 2 corresponds to the TC1016 hardware channels, while CAN 3 corresponds to a virtual device, and CAN 3 is used for replay filtering.

2. To create a new script module named "OnlineRelayFilter", follow the general instructions for creating a script module as outlined in the "C Script" chapter. After creating the module, the script should look like this:

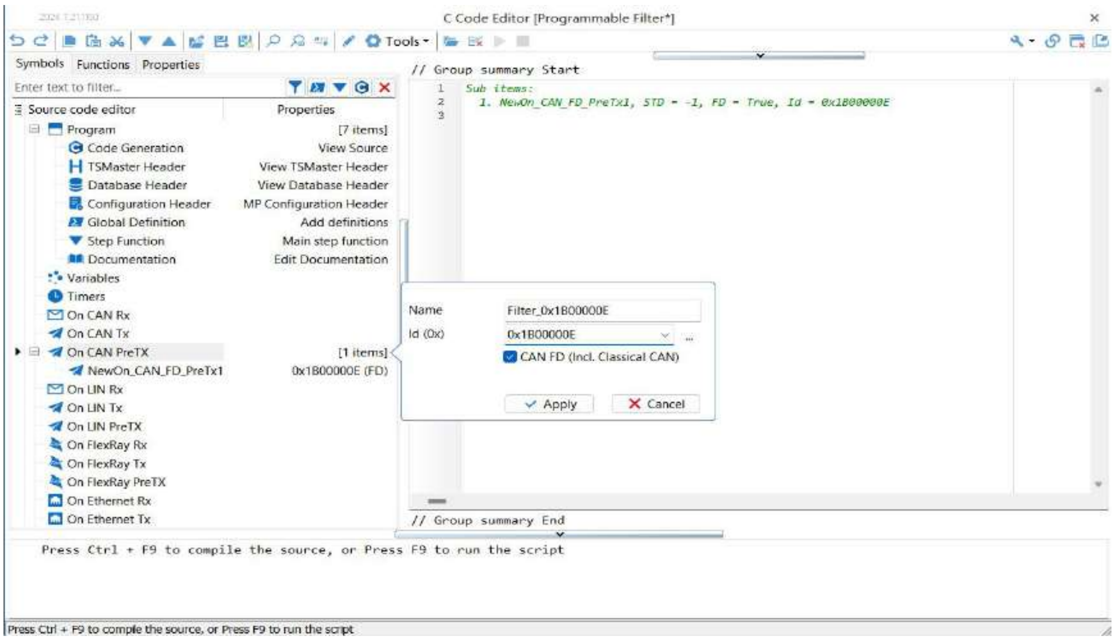


C Script Editor [OnlineRelayFilter]

Source code editor	Properties
Program	[7 items]
Code Generation	View Source
TSMaster Header	View TSMaster Header
Database Header	View Database Header
Test Header	Test Header
Global Definition	Add definitions
Step Function	Main step function
Documentation	Edit documentation
Variables	
Timers	
On CAN Rx	[1 items]
On CAN Tx	
On CAN PreTX	[1 items]
NewOn_CAN_PreTx1	0x1B00000E
On LIN Rx	
On LIN Tx	
On LIN PreTX	
On Var Change	
On Timer	
On Start	[1 items]
On Stop	

Mini program is up to date: motor

3. To create a CAN message PreTX event (which is invoked before a message is sent out on the bus), for example, if the user wants to filter out messages with the ID 0x1B00000E, right click to set up the PreTX event for that specific message.

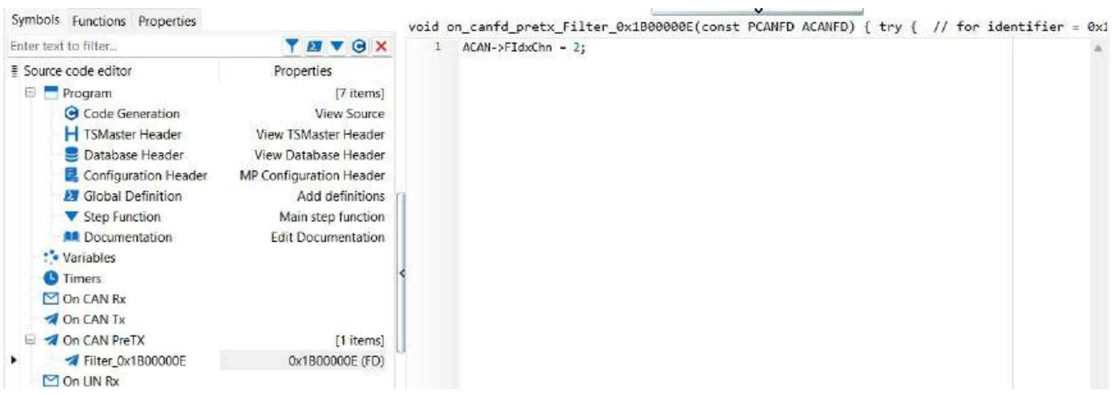


```
// Group summary Start
Sub items:
  1. NewOn_CAN_FD_PreTx1, STD = -1, FD = True, Id = 0x1B00000E
// Group summary End
```

```
Press Ctrl + F9 to compile the source, or Press F9 to run the script
```

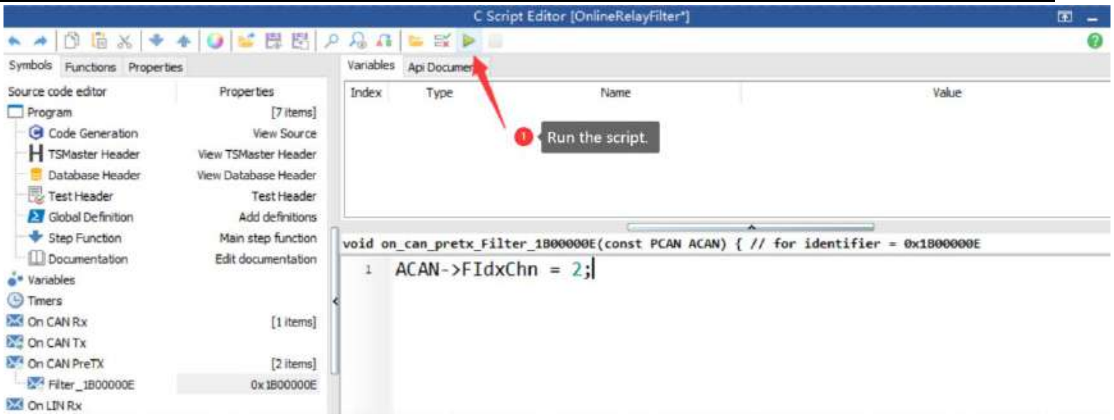
Press Ctrl + F9 to compile the source, or Press F9 to run the script

4. Select the event and change the sending channel of the message data to channel number 2 (which is actually channel 3 due to 0-indexing, i.e., channels 0, 1, 2 correspond to numbered 1, 2, 3).

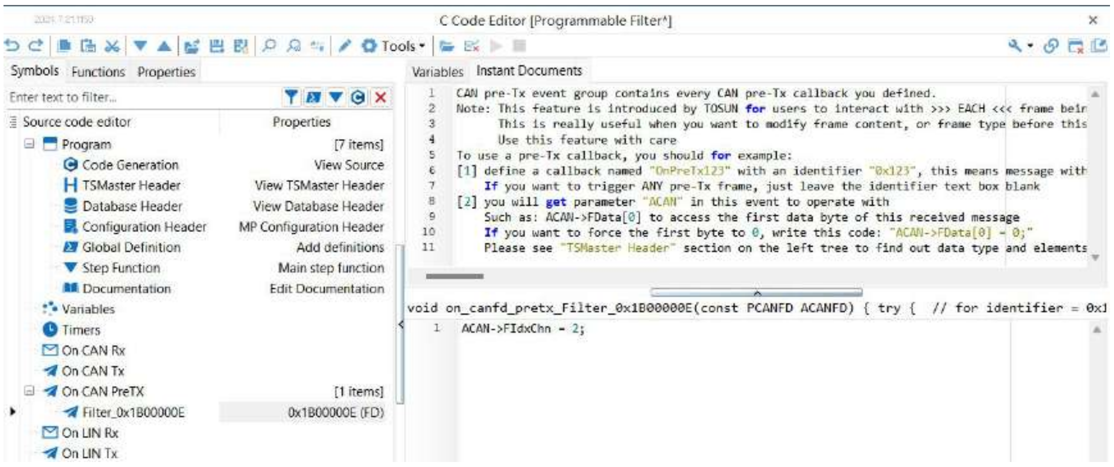


```
void on_canfd_pretx_Filter_0x1B00000E(const PCANFD ACANFD) { try {  // for identifier = 0x1
ACAN->FIdxChn = 2;
```

5. Click to run the script as follows:



```
void on_can_pretx_Filter_1B00000E(const PCAN ACAN) { // for identifier = 0x1B00000E
ACAN->FIdxChn = 2;
```



```
CAN pre-Tx event group contains every CAN pre-Tx callback you defined.
Note: This feature is introduced by TOSUN for users to interact with >>> EACH <<< frame bein
      This is really useful when you want to modify frame content, or frame type before this
      Use this feature with care
To use a pre-Tx callback, you should for example:
[1] define a callback named "OnPreTx123" with an identifier "0x123", this means message with
    If you want to trigger ANY pre-Tx frame, just leave the identifier text box blank
[2] you will get parameter "ACAN" in this event to operate with
    Such as: ACAN->FData[0] to access the first data byte of this received message
    If you want to force the first byte to 0, write this code: "ACAN->FData[0] = 0;"
    Please see "TSMaster Header" section on the left tree to find out data type and elements
```

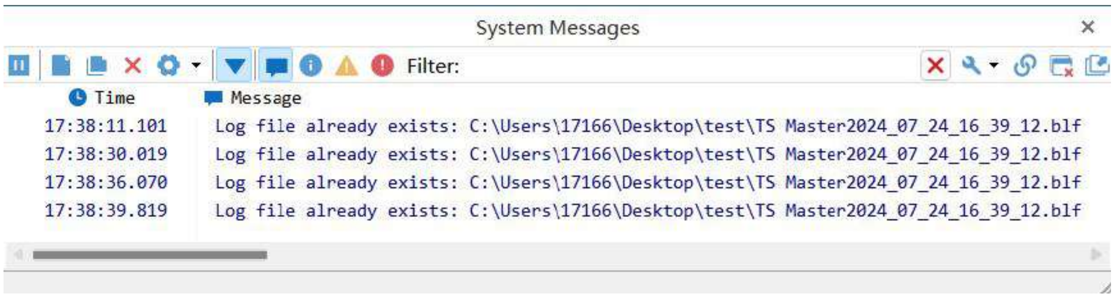
```
void on_canfd_pretx_Filter_0x1B00000E(const PCANFD ACANFD) { try {  // for identifier = 0x1
ACAN->FIdxChn = 2;
```

6. At this point, by clicking "Online Playback", the user will be able to see that the message with ID 0x1B00000E is being sent to the virtual channel 3, while other messages are transmitted on the physical channels 1 and 2. This approach achieves the desired filtering of online messages. Similarly, for any other messages that require filtering, you can achieve the intended effect by adding a Pre_TX event in the script for those specific messages.

1.13.5. Clarification

1.13.5.1. Can not Load Blf File (Filename Contains Spaces)

When loading a blf file into the TSMaster playback (offline/online) module, a loading error occurs, and the system message are as follows:



```
17:38:11.101   Log file already exists: C:\Users\17166\Desktop\test\TS Master2024_07_24_16_39_12.blf
17:38:30.019   Log file already exists: C:\Users\17166\Desktop\test\TS Master2024_07_24_16_39_12.blf
17:38:36.070   Log file already exists: C:\Users\17166\Desktop\test\TS Master2024_07_24_16_39_12.blf
17:38:39.819   Log file already exists: C:\Users\17166\Desktop\test\TS Master2024_07_24_16_39_12.blf
```

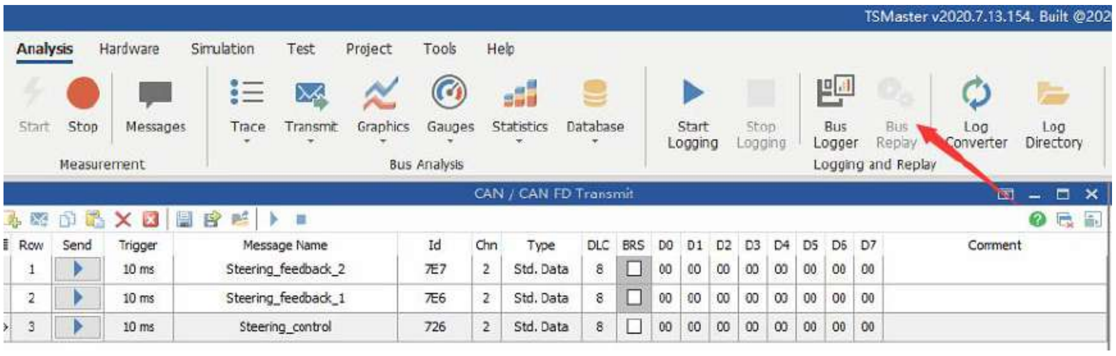
Cause: The blf file name cannot contain spaces in the middle, by modifying the file name to "TSMaster.blf", the loading is successful.

1.13.5.2. Replay Messages According to the Capture Time

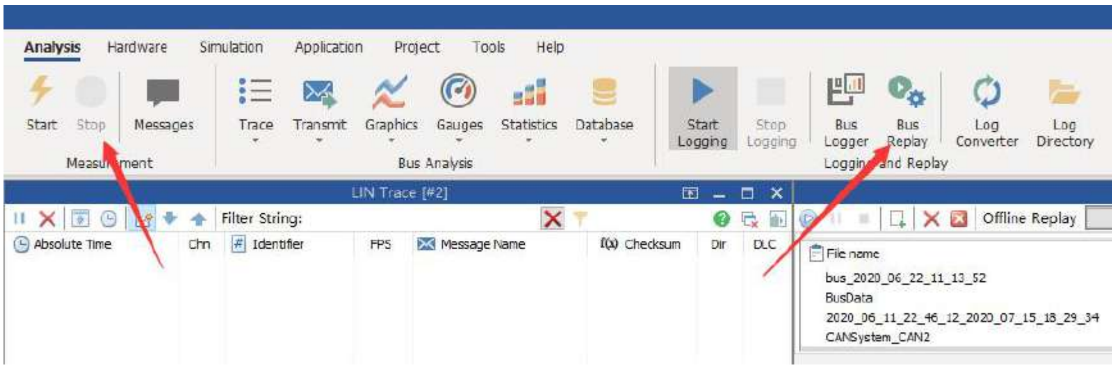
If user wishes to replay messages according to the captured timestamps, please select the online playback option. Offline playback is primarily used for viewing messages, where the focus is on achieving the fastest replay speed possible. However, if user intends to replay messages in the exact order and timing they were captured, use the online playback method. Specific setup instructions for online playback can be found in 1.13.3 Online Replay.

1.13.5.3. Why the Bus Replay Button is Grayed out (Disabled State)

When the bus connection is in working state, message recording playback cannot be performed. User needs to click the Stop button to stop the working state, then the “Bus Replay” button will become enabled and allow the user to add messages for playback.



Row	Send	Trigger	Message Name	Id	Chn	Type	DLC	BRS	D0	D1	D2	D3	D4	D5	D6	D7	Comment
1	▶	10 ms	Steering_feedback_2	7E7	2	Std. Data	8	☐	00	00	00	00	00	00	00	00	
2	▶	10 ms	Steering_feedback_1	7E6	2	Std. Data	8	☐	00	00	00	00	00	00	00	00	
3	▶	10 ms	Steering_control	726	2	Std. Data	8	☐	00	00	00	00	00	00	00	00	



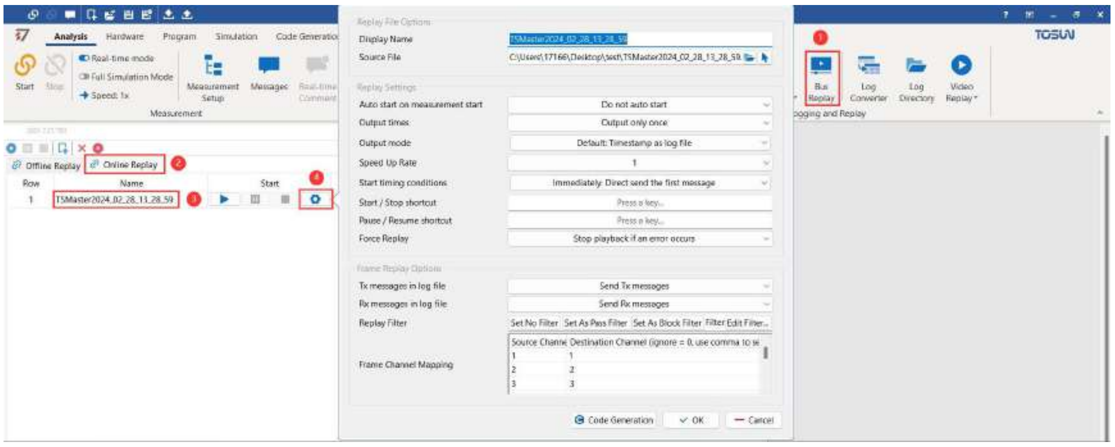
File name
bus_2020_06_22_11_13_52
BusData
2020_06_11_22_46_12_2020_07_15_18_29_34
CANSystem_CAN2

Stop button->Bus Replay Button Becomes Available

1.13.5.4. Why does TSMaster Immediately Send Messages to the Bus after Connecting to the Device

TSMaster provides the capability of online data replay. To support users' requirement of starting replay immediately after connecting to the bus, a function has been added to the replay settings that allows automatic replay upon bus connection, as shown in the figure below:

First, follow these steps to enter the online playback configuration interface:



Replay File Options

Display Name	TSMaster2024_02_28_13_28_59
Source File	C:\Users\17166\Desktop\test\TSMaster2024_02_28_13_28_59

Replay Settings

Auto start on measurement start	Do not auto start
Output times	Output only once
Output mode	Default: Timestamp as log file
Speed Up Rate	1
Start timing conditions	Immediately: Direct send the first message
Start / Stop shortcut	Press a key...
Pause / Resume shortcut	Press a key...
Force Replay	Stop playback if an error occurs

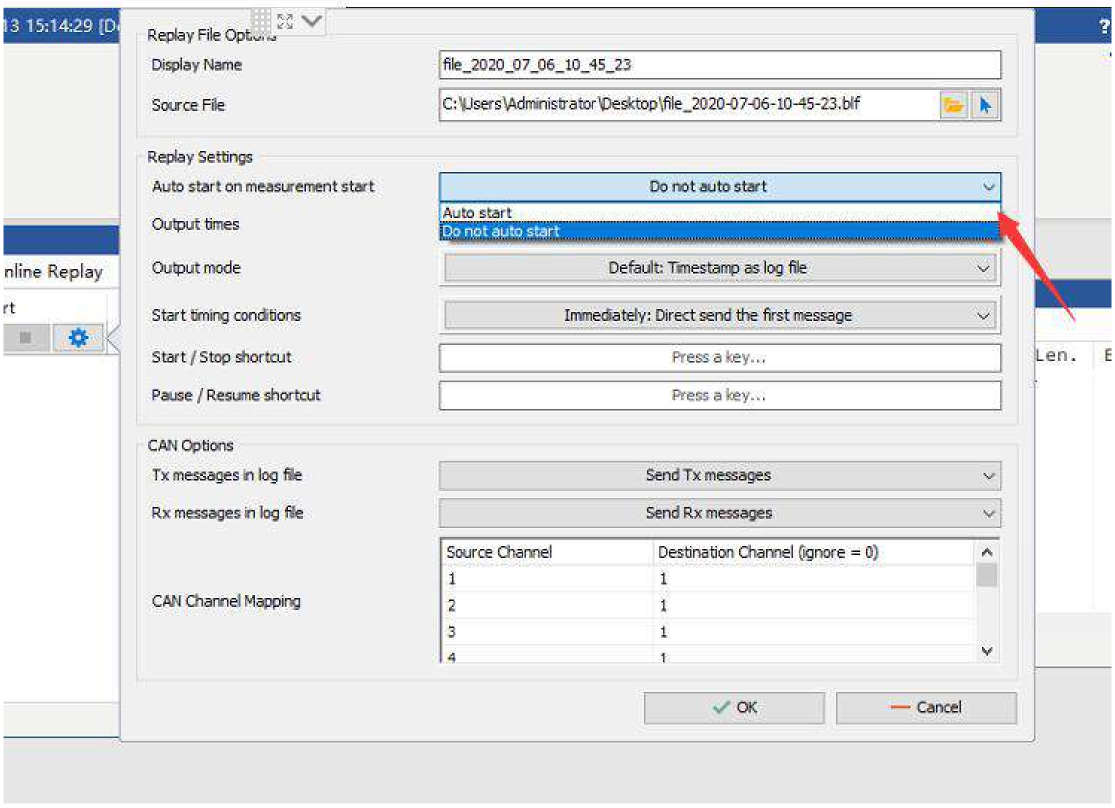
Frame Replay Options

Tx messages in log file	Send Tx messages
Rx messages in log file	Send Rx messages
Replay Filter	Set No Filter · Set As Pass Filter · Set As Block Filter · Filter Edit Filter...

Frame Channel Mapping

Source Channel	Destination Channel (ignore = 0, use comma to se
1	1
2	2
3	3

In the configuration interface, select whether to automatically start message replay instantly upon starting the device, as shown below:



Replay File Options

Display Name	file_2020_07_06_10_45_23
Source File	C:\Users\Administrator\Desktop\file_2020-07-06-10-45-23.blf

Replay Settings

Auto start on measurement start	Do not auto start (options: Auto start / Do not auto start)
Output times	
Output mode	Default: Timestamp as log file
Start timing conditions	Immediately: Direct send the first message
Start / Stop shortcut	Press a key...
Pause / Resume shortcut	Press a key...

CAN Options

Tx messages in log file	Send Tx messages
Rx messages in log file	Send Rx messages

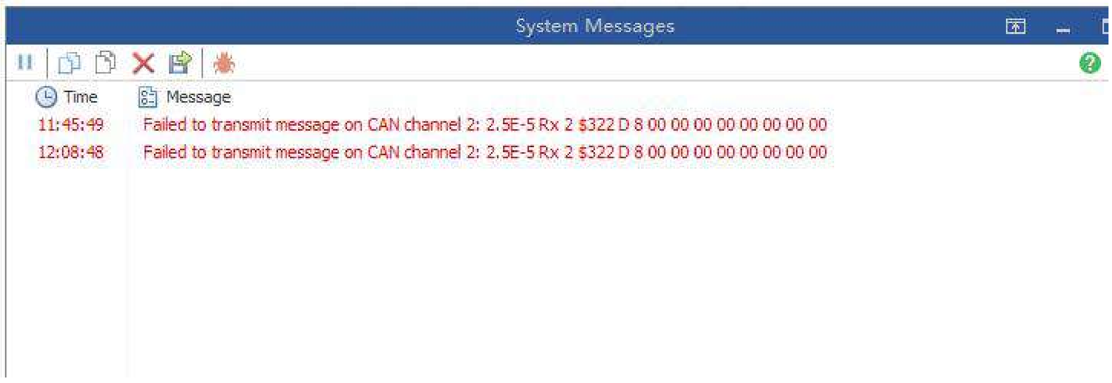
CAN Channel Mapping

Source Channel	Destination Channel (ignore = 0)
1	1
2	1
3	1
4	1

If "Do not auto start" is selected, then messages will not be automatically played back when connecting to the device.

1.13.5.5. Playback Failed due to Channel Error

During online playback, the message playback failed after a period of time with an error message indicating a channel error. As shown below:



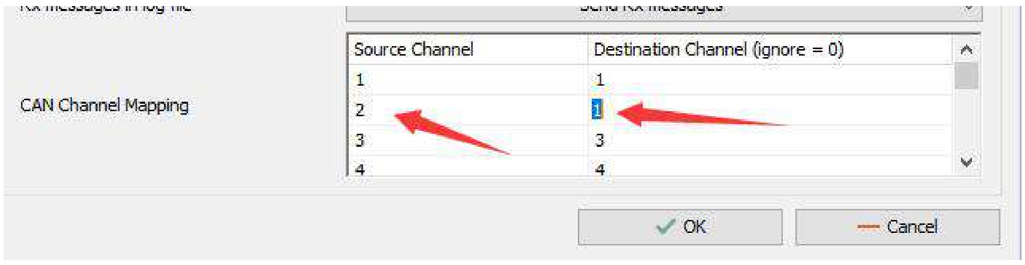
```
11:45:49  Failed to transmit message on CAN channel 2: 2.5E-5 Rx 2 $322 D 8 00 00 00 00 00 00 00 00
12:08:48  Failed to transmit message on CAN channel 2: 2.5E-5 Rx 2 $322 D 8 00 00 00 00 00 00 00 00
```

Trigger reason: channel mapping error.

The data channel in the Log message has not been correctly mapped to the CAN tool's channel. For example, in the error above, data channel 2 is used in the Log file, and this channel is mapped to CAN tool's channel 2 in TSMaster. However, CAN channel 2 is not actually configured in TSMaster, resulting in a transmission failure. For specific instructions on channel mapping, please refer to the previous chapter 1.12.3.1.7 Replay Channel Selection.

Solution:

When entering the online playback configuration interface, configure the channel mapping as shown below:



CAN Channel Mapping

Source Channel	Destination Channel (ignore = 0)
1	1
2	1
3	3
4	4

As shown above, channel 2 does not actually exist in the Destination Channel, therefore, the channel with Source Channel = 2 should also be configured to Destination Channel = 1. By restarting the playback, this error should no longer occur.

1.13.5.6. Online Replay Directly Displaying Error Frames

If during playback, the bus replay directly displays error frames, it is likely that in the Log file, messages with the same ID exist on different data channels. When played back, these messages are transmitted on the same CAN bus through different channels, causing a frame ID conflict, which in turn results in error frames and prevents normal data playback.

1.13.5.7. Online Replay Frozen

If online replay becomes frozen, please check if virtual channels are being used. Some computer platforms may have issues supporting virtual channels, and in such cases, it is recommended for users to avoid using virtual channels if possible. Alternatively, user can connect physical channels and perform online replay of messages using those physical channels.

1.13.5.8. Inconsistent between the Replayed Data and the Data from Third-party Tools

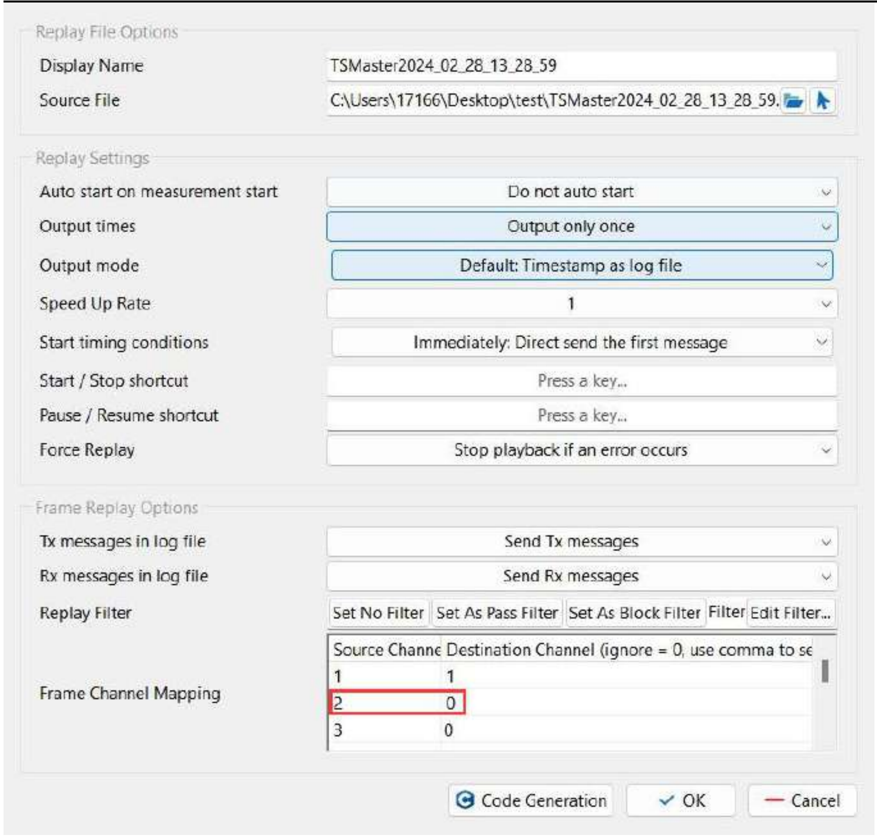
Problem Description:

Through online data replay and signal value monitoring, it was discovered that the values of certain signals, such as signal A, on the bus messages do not align with those parsed by third-party monitoring tools (such as OE). For example, while signal A displays a value of 0.0 on TSMaster, the same signal is interpreted as having a value of 0.3 by OE.

Cause Analysis:

After analysis, it was discovered that the replayed BLF file contained data from two channels (Channel 1 and Channel 2). Within these two channels, there were messages with the same ID (for example, messages that both contain signal A). Additionally, all the message values originating from Channel 2 were found to be 0. Therefore, during replay, the signal A values extracted from Channel 2 were all interpreted as 0. This led to the user mistakenly comparing the values from Channel 2 with those parsed by the third-party tool.

Solution: because the signal values in Channel 2 are not valid data, disabling the replay of Channel 2 during the replay process can resolve the issue. This can be done as follows:



Replay File Options

Display Name	TSMaster2024_02_28_13_28_59
Source File	C:\Users\17166\Desktop\test\TSMaster2024_02_28_13_28_59.

Replay Settings

Auto start on measurement start	Do not auto start
Output times	Output only once
Output mode	Default: Timestamp as log file
Speed Up Rate	1
Start timing conditions	Immediately: Direct send the first message
Start / Stop shortcut	Press a key...
Pause / Resume shortcut	Press a key...
Force Replay	Stop playback if an error occurs

Frame Replay Options

Tx messages in log file	Send Tx messages
Rx messages in log file	Send Rx messages
Replay Filter	Set No Filter / Set As Pass Filter / Set As Block Filter / Filter / Edit Filter...
Frame Channel Mapping	

Source Channe	Destination Channel (ignore = 0, use comma to se
1	1
2	0
3	0

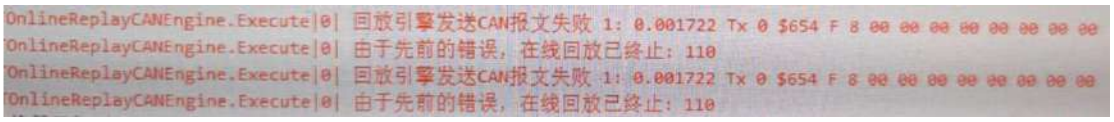
Tips:

If users encounter any doubts while analyzing data signals, please pay attention to the channel and timestamp of the messages, otherwise the comparison with third-party tools is not meaningful.

1.13.5.9. CAN Channel is Ready but Software Online Replay Fails

Problem Description:

When loading a blf file, the messages only contain reports from Channel 1, but the replay still fails with the following error message:



```
OnlineReplayCANEngine.Execute|0| 回放引擎发送CAN报文失败 1: 0.001722 Tx 0 $654 F 8 00 00 00 00 00 00 00 00
OnlineReplayCANEngine.Execute|0| 由于先前的错误, 在线回放已终止: 110
OnlineReplayCANEngine.Execute|0| 回放引擎发送CAN报文失败 1: 0.001722 Tx 0 $654 F 8 00 00 00 00 00 00 00 00
OnlineReplayCANEngine.Execute|0| 由于先前的错误, 在线回放已终止: 110
```

Cause Analysis:

Online replay involves reversing the recorded messages and re-injecting them back onto the physical bus. In this specific case, the messages recorded in the Blf file are fd messages, but the hardware is a classic CAN system, this is the reason for the failure of the online replay.

Solution:

Replace the hardware with one that supports FDCAN. If the hardware already supports FDCAN, then set the hardware's operating mode to FDCAN mode. After that, retry the replay process.

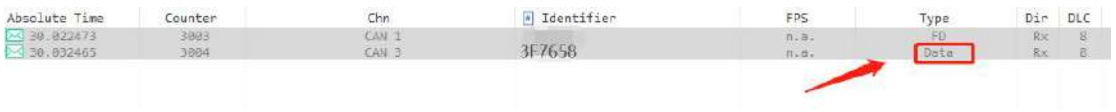
1.13.5.10. When Conducting Data Analysis, there is a Specific ID that can be Seen during Offline Replay but becomes Invisible during Online Replay

Problem Description:

When loading a blf file for analysis, it is observed that the message with ID = 0x3F2758 can be seen during offline replay, but this particular frame cannot be found during online analysis.

Cause Analysis:

The message properties are incorrect. The message ID = 0x3F7658 should belong to an extended frame message, but the message's own attributes indicate that it is a standard frame message. During offline replay, it appears as shown in the figure below:



Absolute Time	Counter	Chn	Identifier	FPS	Type	Dir	DLC
30.022473	3003	CAN 1		n.a.	FD	Rx	8
30.032465	3004	CAN 3	3F7658	n.a.	Data	Rx	8

As shown in the figure, the Type should be “Ext Data” instead of “Data”.

During offline replay, the blf data is presented exactly as it is, without any correction, making it easier to identify issues.

However, during online replay, TSMaster will correct the ID based on the message's attributes. For example, in the case mentioned above, if the ID property indicates it is a standard ID, the actual sent ID will be corrected to:

ID = SRCID & 0x7FF = 0x3F7658 & 0x7FF = 0x658

Therefore, during actual online replay, the user will not see the original message 0x3F7658, but instead will see an additional standard frame message with the ID 0x658.

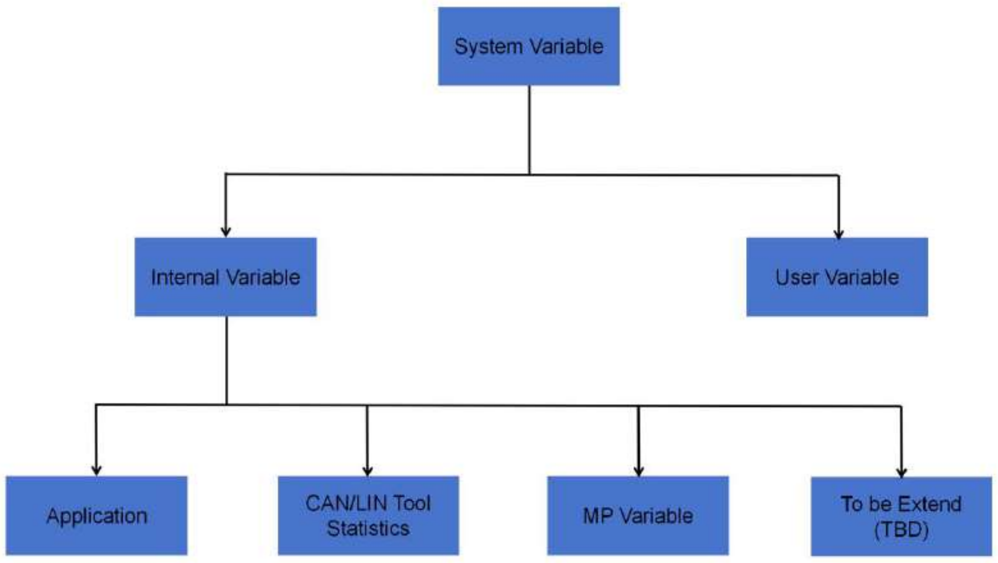
1.14. System Variables

1.14.1. Overview

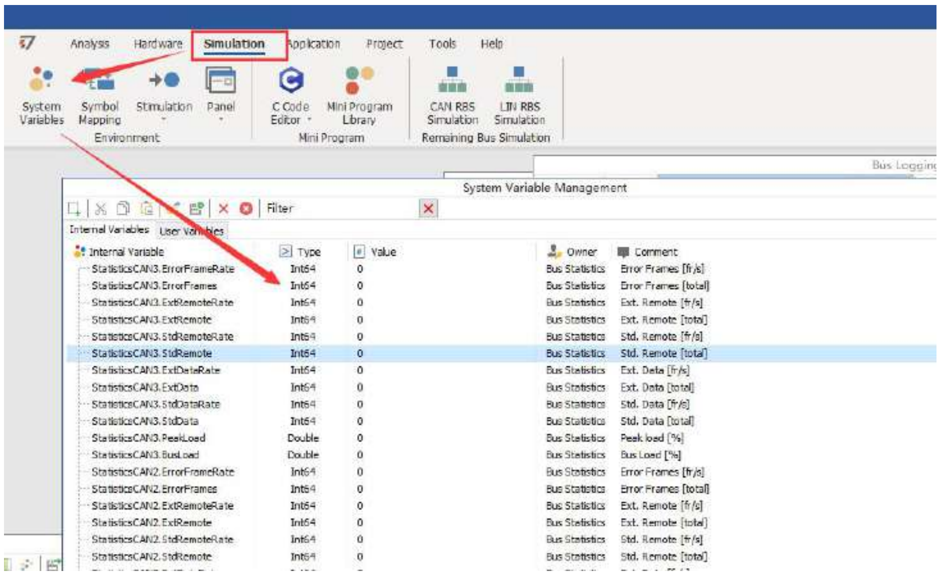
When users are utilizing the Panel, scripting, calibration, diagnostics, or other extended functionalities, the variables they can directly access are defined as system variables. Based on the principles behind their generation, system variables are primarily classified into two types: Internal Variables (also known as intrinsic system variables) and User Variables (user-defined system variables). The main differences between them are:

- Intrinsic system variables are automatically generated and released by the system, and users cannot directly add, delete, or modify them;
- User variables are created and managed by the users themselves.

The system architecture for system variables are as shown in the figure below:



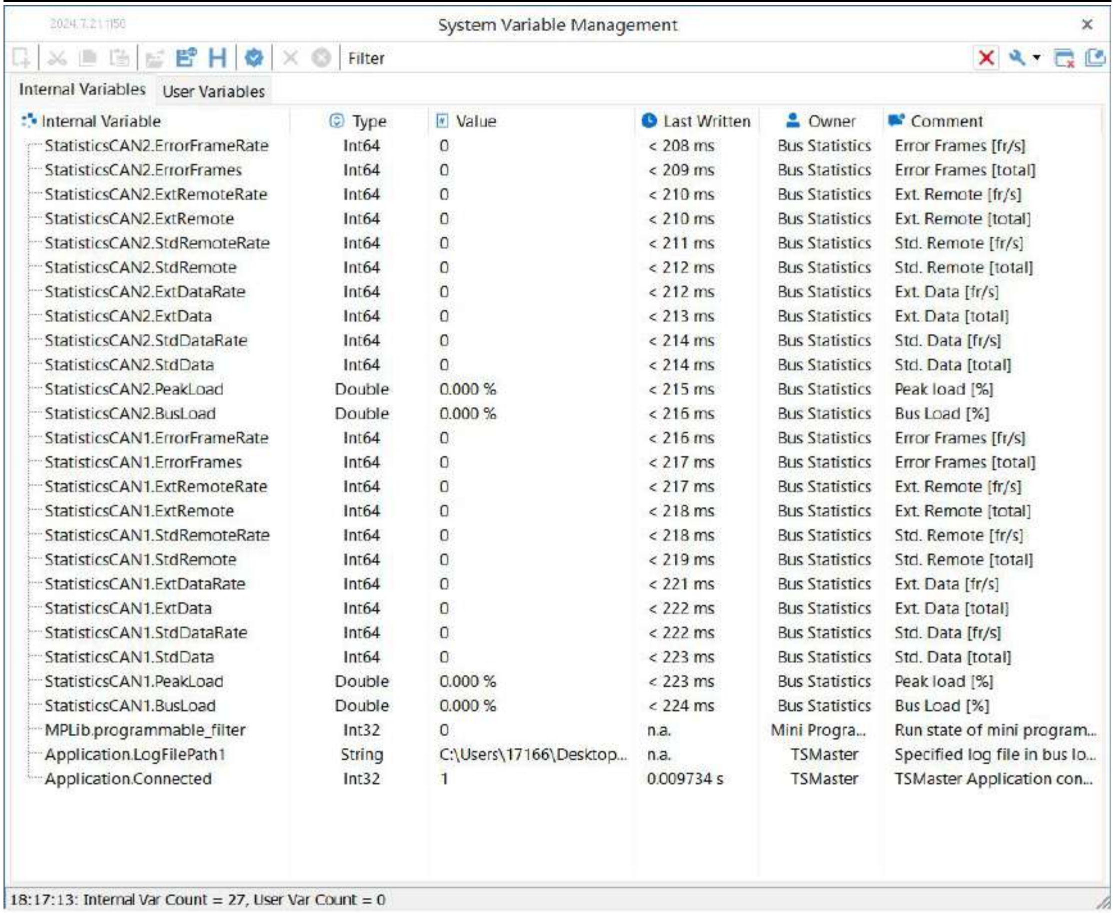
To view the currently available system variables, the user needs to navigate to the system variable management interface by following the path: Simulation -> System Variables, as shown in the figure below:



Internal Variable	Type	Value	Owner	Comment
StatisticsCAN3.ErrorFrameRate	Int64	0	Bus Statistics	Error Frames [fr/s]
StatisticsCAN3.ErrorFrames	Int64	0	Bus Statistics	Error Frames [total]
StatisticsCAN3.ExtRemoteRate	Int64	0	Bus Statistics	Ext. Remote [fr/s]
StatisticsCAN3.ExtRemote	Int64	0	Bus Statistics	Ext. Remote [total]
StatisticsCAN3.StdRemoteRate	Int64	0	Bus Statistics	Std. Remote [fr/s]
StatisticsCAN3.StdRemote	Int64	0	Bus Statistics	Std. Remote [total]
StatisticsCAN3.ExtDataRate	Int64	0	Bus Statistics	Ext. Data [fr/s]
StatisticsCAN3.ExtData	Int64	0	Bus Statistics	Ext. Data [total]
StatisticsCAN3.StdDataRate	Int64	0	Bus Statistics	Std. Data [fr/s]
StatisticsCAN3.StdData	Int64	0	Bus Statistics	Std. Data [total]
StatisticsCAN3.PeakLoad	Double	0	Bus Statistics	Peak load [%]
StatisticsCAN3.BusLoad	Double	0	Bus Statistics	Bus Load [%]
StatisticsCAN2.ErrorFrameRate	Int64	0	Bus Statistics	Error Frames [fr/s]
StatisticsCAN2.ErrorFrames	Int64	0	Bus Statistics	Error Frames [total]
StatisticsCAN2.ExtRemoteRate	Int64	0	Bus Statistics	Ext. Remote [fr/s]
StatisticsCAN2.ExtRemote	Int64	0	Bus Statistics	Ext. Remote [total]
StatisticsCAN2.StdRemoteRate	Int64	0	Bus Statistics	Std. Remote [fr/s]
StatisticsCAN2.StdRemote	Int64	0	Bus Statistics	Std. Remote [total]

1.14.2. Internal Variables (Intrinsic System Variables)

Intrinsic System Variables are automatically generated and released along with the system's operation. Common examples of Intrinsic System Variables are as shown in the figure below: 1. system information 2. device statistics 3. mini-program variables.


2024.7.21.1150 System Variable Management

Filter

Internal Variables | User Variables

Internal Variable	Type	Value	Last Written	Owner	Comment
StatisticsCAN2.ErrorFrameRate	Int64	0	< 208 ms	Bus Statistics	Error Frames [fr/s]
StatisticsCAN2.ErrorFrames	Int64	0	< 209 ms	Bus Statistics	Error Frames [total]
StatisticsCAN2.ExtRemoteRate	Int64	0	< 210 ms	Bus Statistics	Ext. Remote [fr/s]
StatisticsCAN2.ExtRemote	Int64	0	< 210 ms	Bus Statistics	Ext. Remote [total]
StatisticsCAN2.StdRemoteRate	Int64	0	< 211 ms	Bus Statistics	Std. Remote [fr/s]
StatisticsCAN2.StdRemote	Int64	0	< 212 ms	Bus Statistics	Std. Remote [total]
StatisticsCAN2.ExtDataRate	Int64	0	< 212 ms	Bus Statistics	Ext. Data [fr/s]
StatisticsCAN2.ExtData	Int64	0	< 213 ms	Bus Statistics	Ext. Data [total]
StatisticsCAN2.StdDataRate	Int64	0	< 214 ms	Bus Statistics	Std. Data [fr/s]
StatisticsCAN2.StdData	Int64	0	< 214 ms	Bus Statistics	Std. Data [total]
StatisticsCAN2.PeakLoad	Double	0.000 %	< 215 ms	Bus Statistics	Peak load [%]
StatisticsCAN2.BusLoad	Double	0.000 %	< 216 ms	Bus Statistics	Bus Load [%]
StatisticsCAN1.ErrorFrameRate	Int64	0	< 216 ms	Bus Statistics	Error Frames [fr/s]
StatisticsCAN1.ErrorFrames	Int64	0	< 217 ms	Bus Statistics	Error Frames [total]
StatisticsCAN1.ExtRemoteRate	Int64	0	< 217 ms	Bus Statistics	Ext. Remote [fr/s]
StatisticsCAN1.ExtRemote	Int64	0	< 218 ms	Bus Statistics	Ext. Remote [total]
StatisticsCAN1.StdRemoteRate	Int64	0	< 218 ms	Bus Statistics	Std. Remote [fr/s]
StatisticsCAN1.StdRemote	Int64	0	< 219 ms	Bus Statistics	Std. Remote [total]
StatisticsCAN1.ExtDataRate	Int64	0	< 221 ms	Bus Statistics	Ext. Data [fr/s]
StatisticsCAN1.ExtData	Int64	0	< 222 ms	Bus Statistics	Ext. Data [total]
StatisticsCAN1.StdDataRate	Int64	0	< 222 ms	Bus Statistics	Std. Data [fr/s]
StatisticsCAN1.StdData	Int64	0	< 223 ms	Bus Statistics	Std. Data [total]
StatisticsCAN1.PeakLoad	Double	0.000 %	< 223 ms	Bus Statistics	Peak load [%]
StatisticsCAN1.BusLoad	Double	0.000 %	< 224 ms	Bus Statistics	Bus Load [%]
MPLib.programmable_filter	Int32	0	n.a.	Mini Progra...	Run state of mini program...
Application.LogFilePath1	String	C:\Users\17166\Desktop...	n.a.	TSMaster	Specified log file in bus lo...
Application.Connected	Int32	1	0.009734 s	TSMaster	TSMaster Application con...

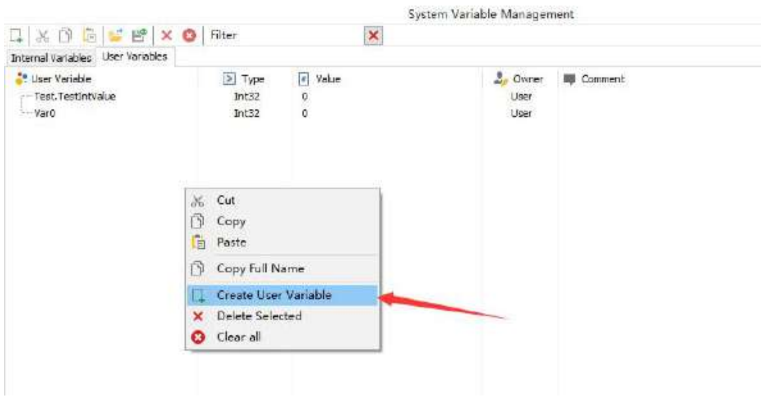
18:17:13: Internal Var Count = 27, User Var Count = 0

Taking device statistics as an example, if a CAN1 device is added, the related statistics information such as StaticsCAN1 will be dynamically generated. Conversely, if the CAN1 device is removed, these statistics information will disappear. As the TSMaster software upgrades, more and more dynamically generated and loaded data types will be introduced in the future. For detailed information regarding mini-program variables, please refer to the section 1.15.1.2 Mini-Program Variable (Variable In Mini-program).

1.14.3. User Variables (User-defined System Variables)

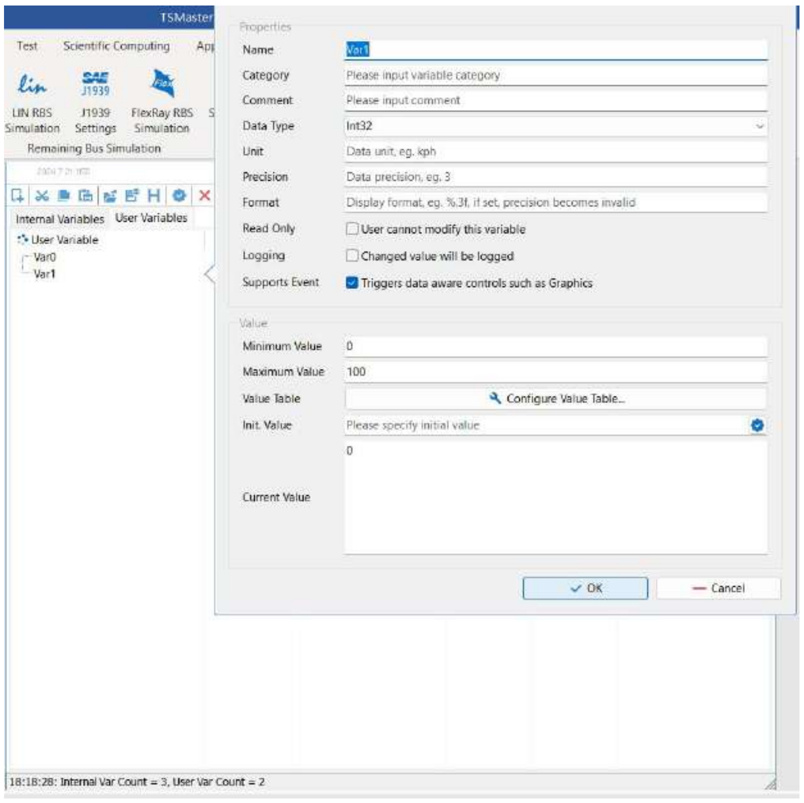
This type of variable is user-defined, allowing users to perform operations such as adding and deleting them. The process of adding a new user variable is outlined below:

- Step 1: Right-click in the System Variables management interface and click: "Create User Variable".



User Variable	Type	Value	Owner	Comment
Test.TestIntValue	Int32	0	User	
Var0	Int32	0	User	

➢ Step 2: Set the properties of the user variable.



Properties

Name	Var1
Category	Please input variable category
Comment	Please input comment
Data Type	Int32
Unit	Data unit, eg. kph
Precision	Data precision, eg. 3
Format	Display format, eg. %.3f, if set, precision becomes invalid
Read Only	☐ User cannot modify this variable
Logging	☐ Changed value will be logged
Supports Event	☑ Triggers data aware controls such as Graphics

Value

Minimum Value	0
Maximum Value	100
Value Table	Configure Value Table...
Init. Value	Please specify initial value
Current Value	0

18:18:28: Internal Var Count = 3, User Var Count = 2

Name: The name of the variable.

Category: The group the variable belongs to, making it easier for users to manage variables. Variables with the same name may be used in different applications.

Comment: Add comments and descriptions to the variable for clarification.

Data Type: The type of the variable, including integer, float, and other types. Details can be expanded for viewing.

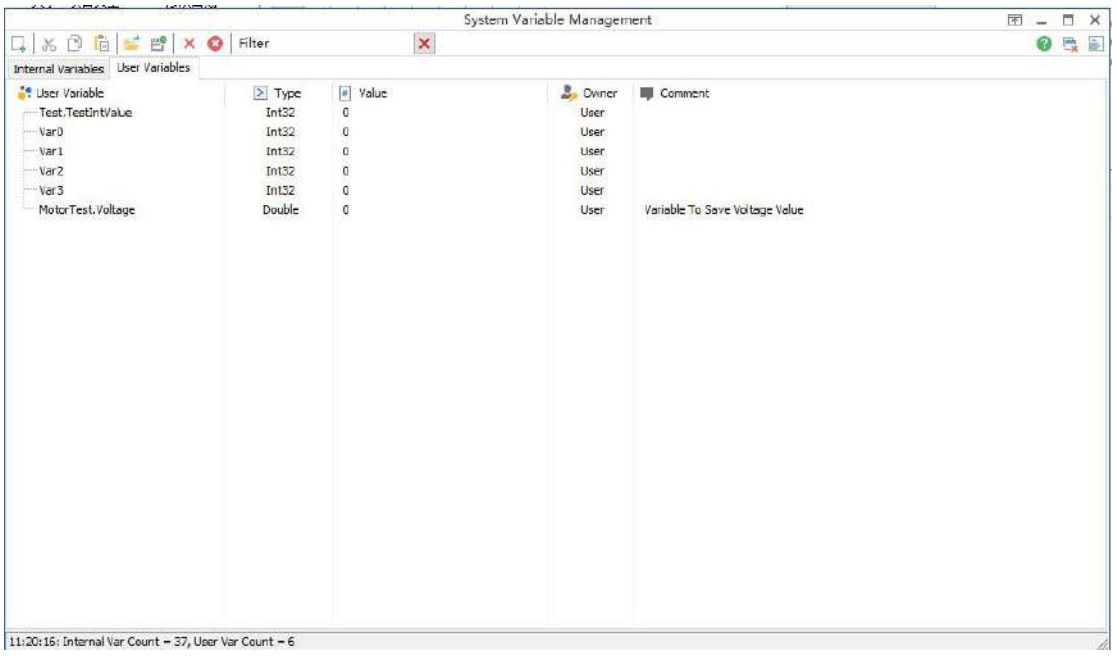
Read Only: Whether the variable is read-only. If it is read-only, users cannot modify it.

Minimum Value: The minimum value allowed for the variable.

Maximum Value: The maximum value allowed for the variable.

Current Value: The current value of the variable.

➢ After adding to the system, it will look like the following figure:

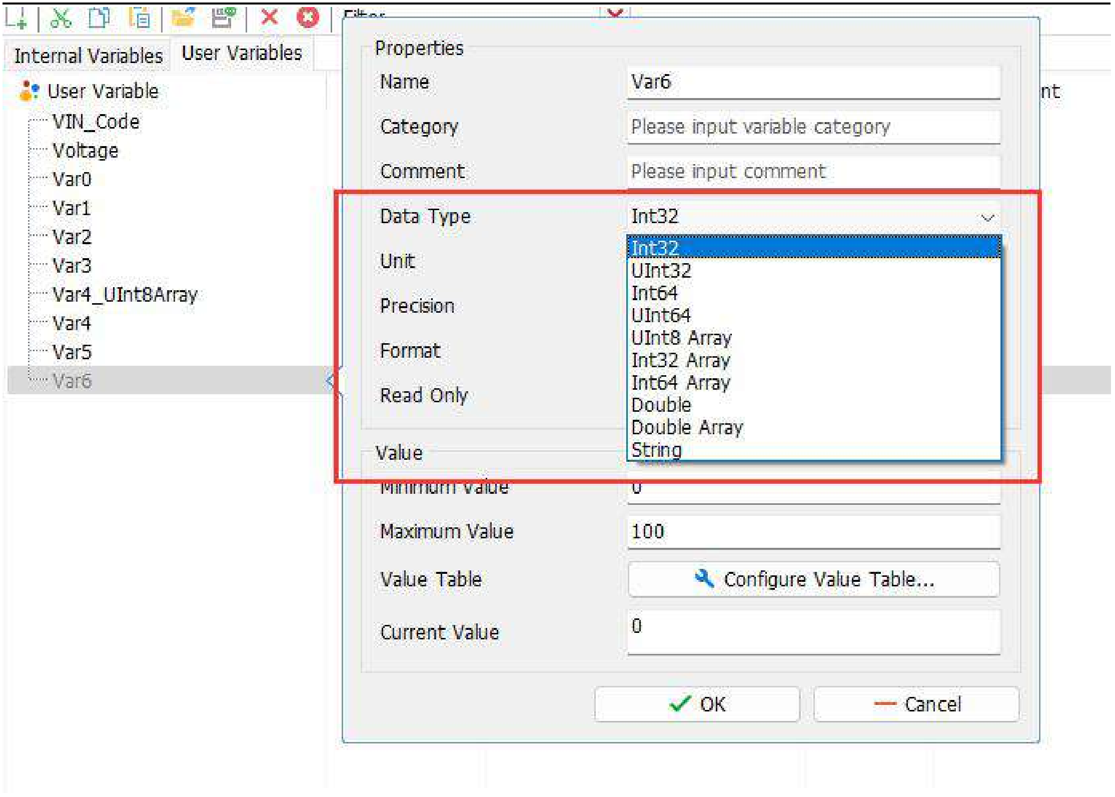


User Variable	Type	Value	Owner	Comment
Test.TestIntValue	Int32	0	User	
Var0	Int32	0	User	
Var1	Int32	0	User	
Var2	Int32	0	User	
Var3	Int32	0	User	
MotorTest.Voltage	Double	0	User	Variable To Save Voltage Value

11:20:16: Internal Var Count = 37, User Var Count = 6

1.14.4. System Variable Data Types

TSMaster system variable mainly include the following data types: Int32，UInt32，Int64，UInt64，UInt8 Array，Int32 Array, Int64 Array, Double, Double Array, String, as show below:


Internal Variables | User Variables

User Variable
- VIN_Code
- Voltage
- Var0
- Var1
- Var2
- Var3
- Var4_UInt8Array
- Var4
- Var5
- Var6

Properties

Name	Var6
Category	Please input variable category
Comment	Please input comment
Data Type	Int32
Unit	
Precision	
Format	
Read Only	

Int32
UInt32
Int64
UInt64
UInt8 Array
Int32 Array
Int64 Array
Double
Double Array
String

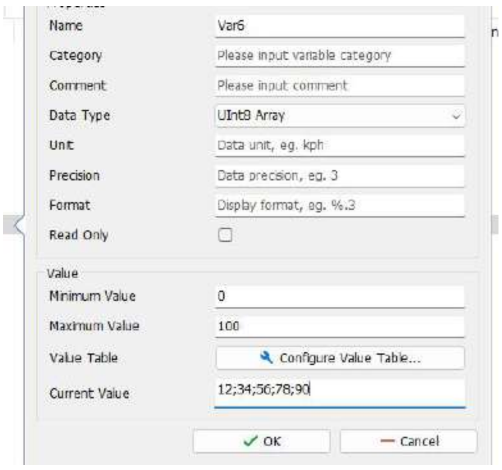
Value

Minimum Value	0
Maximum Value	100
Value Table	Configure Value Table...
Current Value	0

The meanings of these data types are as follows:

- Int32: signed 32-bit type, which can encompass types like Int8, Int16, and others.
- UInt32: unsigned 32-bit type, which can encompass data types like UInt8, UInt16, and others.
- Int64: signed 64-bit data type.
- UInt64: unsigned 64-bit data type.

UInt8 Array: Unsigned 8-bit array, also commonly known as a Byte array. When assigning values to an array of this data type, the array elements are separated by a semicolon ';', as shown below. This is equivalent to defining the array as: UInt8 Var6[5] = {12, 34, 56, 78, 90};



Name	Var6
Category	Please input variable category
Comment	Please input comment
Data Type	UInt8 Array
Unit	Data unit, eg. kph
Precision	Data precision, eg. 3
Format	Display format, eg. %.3
Read Only	☐

Value

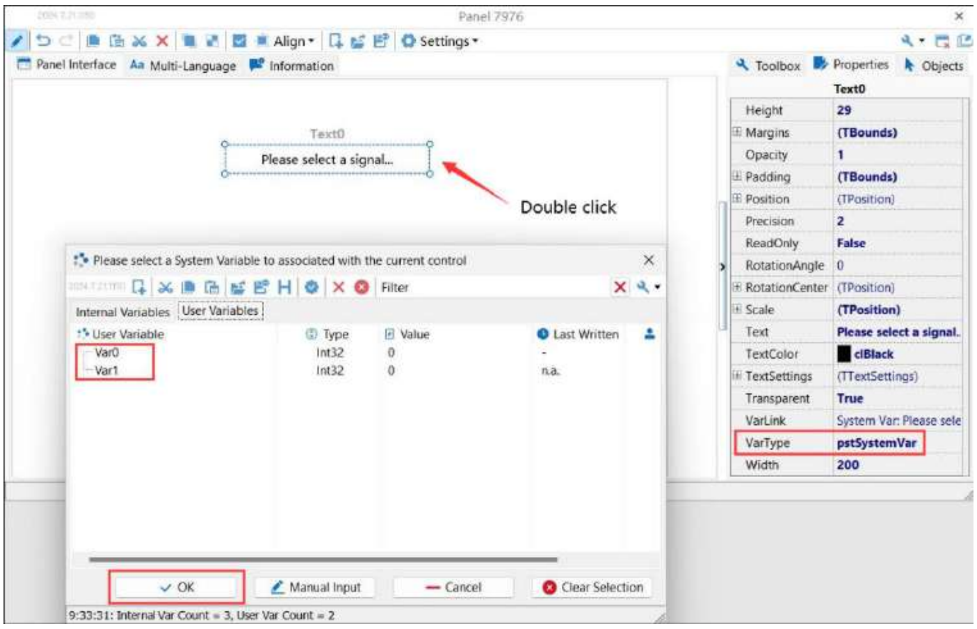
Minimum Value	0
Maximum Value	100
Value Table	Configure Value Table...
Current Value	12;34;56;78;90

- Int32 Array:
- Int64 Array:
- Double:
- Double Array:
- String:

1.14.5. Access System Variables

Whether it's an intrinsic system variables or a user-defined variable, the way to access them is exactly the same. Here are two main application scenarios for introduction: 1. Panel associated system variables 2. reading and writing system variables through scripts.

1.14.5.1. Panel Associated System Variables



User Variable	Type	Value	Last Written
Var0	Int32	0	-
Var1	Int32	0	n.a.

9:33:31: Internal Var Count = 3, User Var Count = 2

Text0	
Height	29
Margins	(TBounds)
Opacity	1
Padding	(TBounds)
Position	(TPosition)
Precision	2
ReadOnly	False
RotationAngle	0
RotationCenter	(TPosition)
Scale	(TPosition)
Text	Please select a signal.
TextColor	clBlack
TextSettings	(TTextSettings)
Transparent	True
VarLink	System Var: Please sele
VarType	pstSystemVar
Width	200

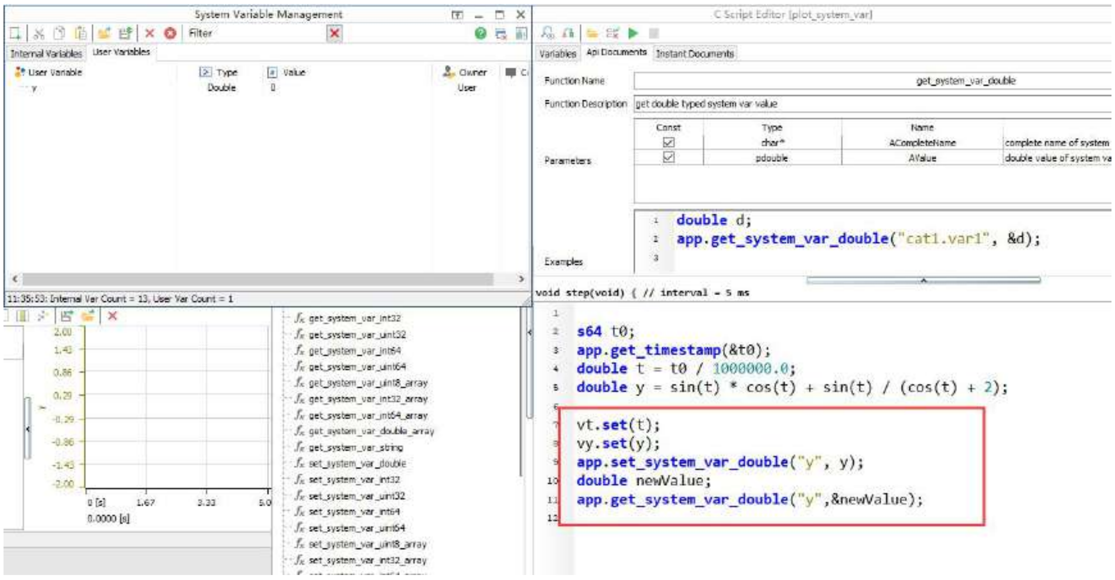
The process of associating panel controls with system variables is illustrated in the figure above:

1. Select the variable type to associate with the control as system variable (SystemVar).

2. Double-click to expand the variable selection panel.

3. On the panel, the user can select both built-in system variables and user-defined system variables.

1.14.5.2. Reading and Writing System Variables through Scripts


System Variable Management
Filter
Internal Variables
User Variables
11:35:53: Internal Var Count = 13, User Var Count = 1
C Script Editor [plot_system_var]
Variables
Api Documents
Instant Documents

User Variable	Type	Value	Owner
y	Double	0	User

Function Name: get_system_var_double

Function Description: get double typed system var value

Parameters:

Const	Type	Name	
✓	char*	ACompleteName	complete name of system
✓	pdouble	AValue	double value of system va

Examples:

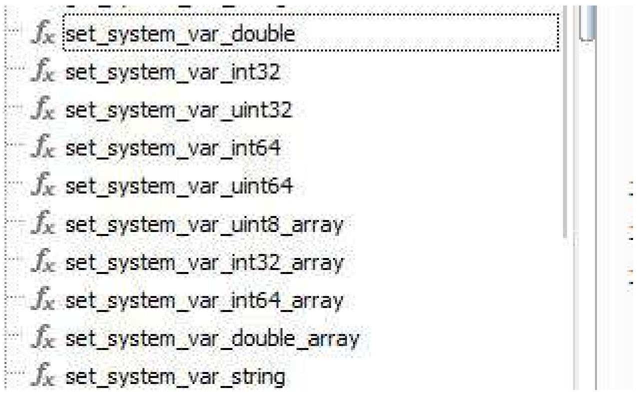
```
double d;
app.get_system_var_double("cat1.var1", &d);
```

```
void step(void) { // interval = 5 ms

s64 t0;
app.get_timestamp(&t0);
double t = t0 / 1000000.0;
double y = sin(t) * cos(t) + sin(t) / (cos(t) + 2);

vt.set(t);
vy.set(y);
app.set_system_var_double("y", y);
double newValue;
app.get_system_var_double("y",&newValue);
```

get_system_var_int32
get_system_var_uint32
get_system_var_int64
get_system_var_uint64
get_system_var_uint8_array
get_system_var_int32_array
get_system_var_int64_array
get_system_var_double_array
get_system_var_string
set_system_var_double
set_system_var_int32
set_system_var_uint32
set_system_var_int64
set_system_var_uint64
set_system_var_uint8_array
set_system_var_int32_array

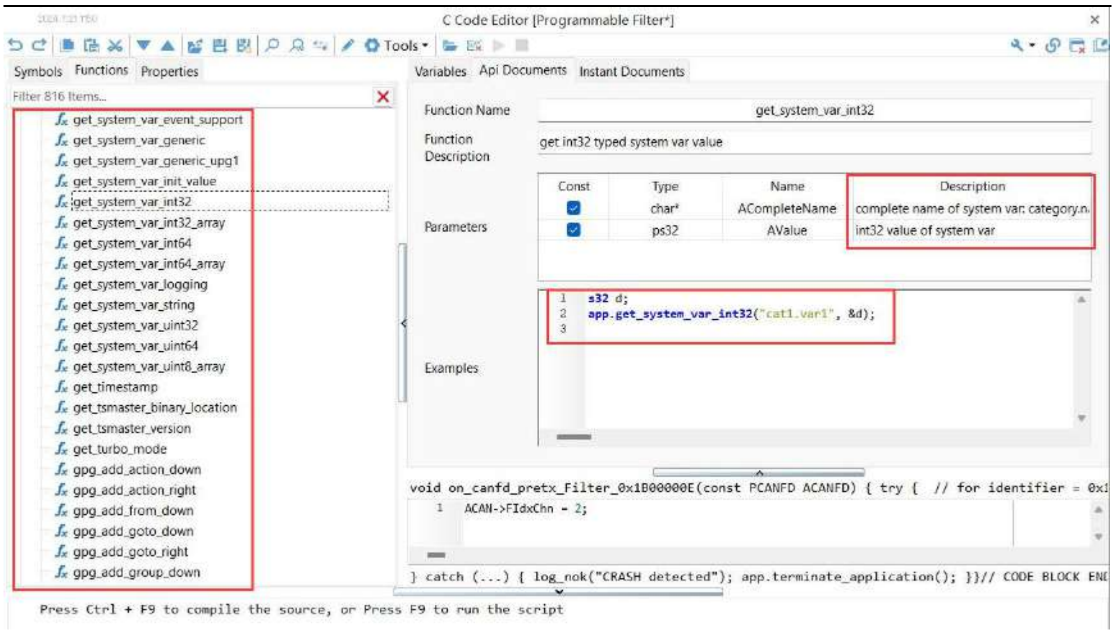
As shown in the diagram above, in simple terms, accessing system variables primarily involves using two types of functions, which are located under the APP directory within the Functions section of the mini-program.

➢ **Writing the variables:**



set_system_var_double
set_system_var_int32
set_system_var_uint32
set_system_var_int64
set_system_var_uint64
set_system_var_uint8_array
set_system_var_int32_array
set_system_var_int64_array
set_system_var_double_array
set_system_var_string

Regarding function parameters and other details, after clicking to select the function, the usage instructions for that function will be displayed in the top-right corner, as shown in the following figure:


C Code Editor [Programmable Filter*]

Tools

Symbols | Functions | Properties

Filter 816 Items...

- get_system_var_event_support
- get_system_var_generic
- get_system_var_generic_upg1
- get_system_var_init_value
- get_system_var_int32
- get_system_var_int32_array
- get_system_var_int64
- get_system_var_int64_array
- get_system_var_logging
- get_system_var_string
- get_system_var_uint32
- get_system_var_uint64
- get_system_var_uint8_array
- get_timestamp
- get_tsmaster_binary_location
- get_tsmaster_version
- get_turbo_mode
- gpg_add_action_down
- gpg_add_action_right
- gpg_add_from_down
- gpg_add_goto_down
- gpg_add_goto_right
- gpg_add_group_down

Variables | Api Documents | Instant Documents

Function Name: get_system_var_int32

Function Description: get int32 typed system var value

Parameters:

Const	Type	Name	Description
✓	char*	ACompleteName	complete name of system var: category.n
✓	ps32	AValue	int32 value of system var

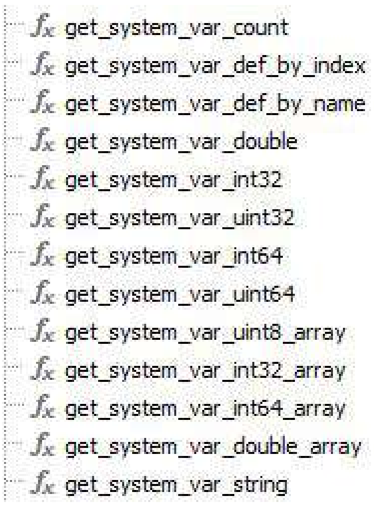
Examples:

```
s32 d;
app.get_system_var_int32("cat1.var1", &d);
```

```
void on_canfd_pretx_Filter_0x1B00000E(const PCANFD ACANFD) { try {  // for identifier = 0x1
    ACAN->FIdxChn = 2;
} catch (...) { log_nok("CRASH detected"); app.terminate_application(); }}// CODE BLOCK END
```

Press Ctrl + F9 to compile the source, or Press F9 to run the script

➢ **Reading the variables:**



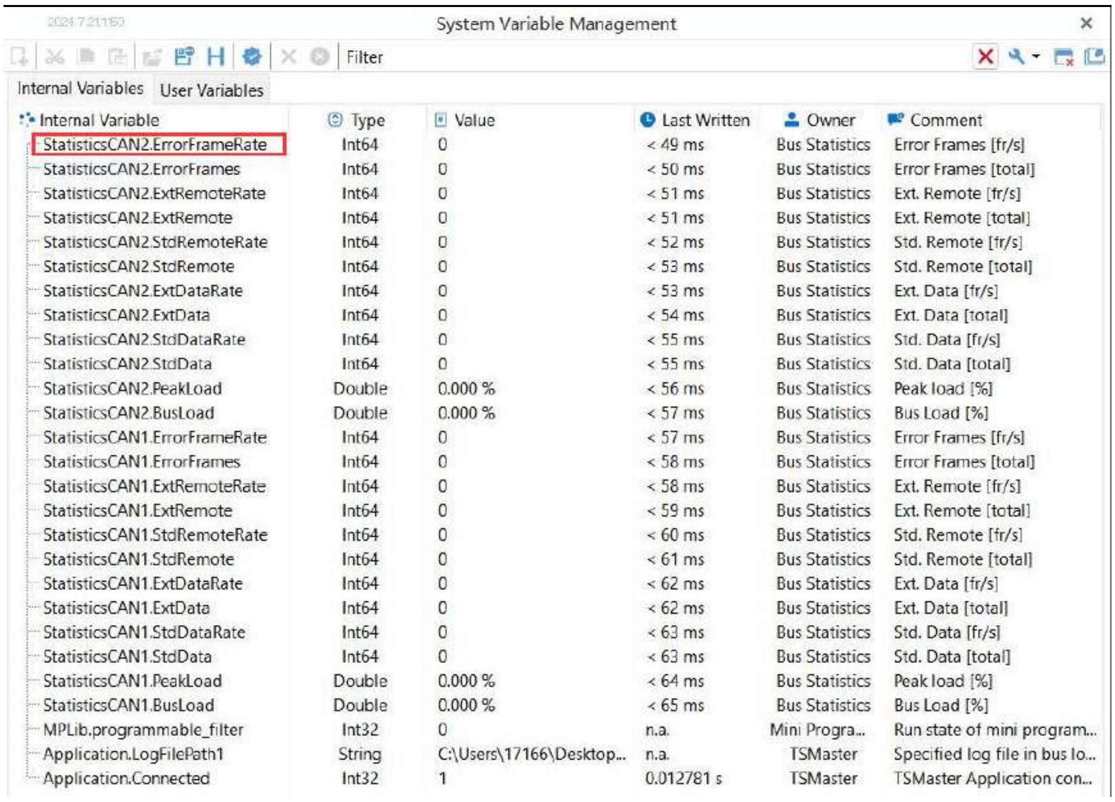
- get_system_var_count
- get_system_var_def_by_index
- get_system_var_def_by_name
- get_system_var_double
- get_system_var_int32
- get_system_var_uint32
- get_system_var_int64
- get_system_var_uint64
- get_system_var_uint8_array
- get_system_var_int32_array
- get_system_var_int64_array
- get_system_var_double_array
- get_system_var_string

Based on the set_system and get_system functions, the script system is endowed with the ability to access internal script variables across different scripts.

1.14.6. Clarification

1.14.6.1. Unable to See the Mini-program Variables

When a user tries to access a mini-program variable, for example, when attempting to associate a mini-program variable in the Panel, they may find that the variable is not present in the panel, as shown in the following figure, this variable may suddenly disappear :

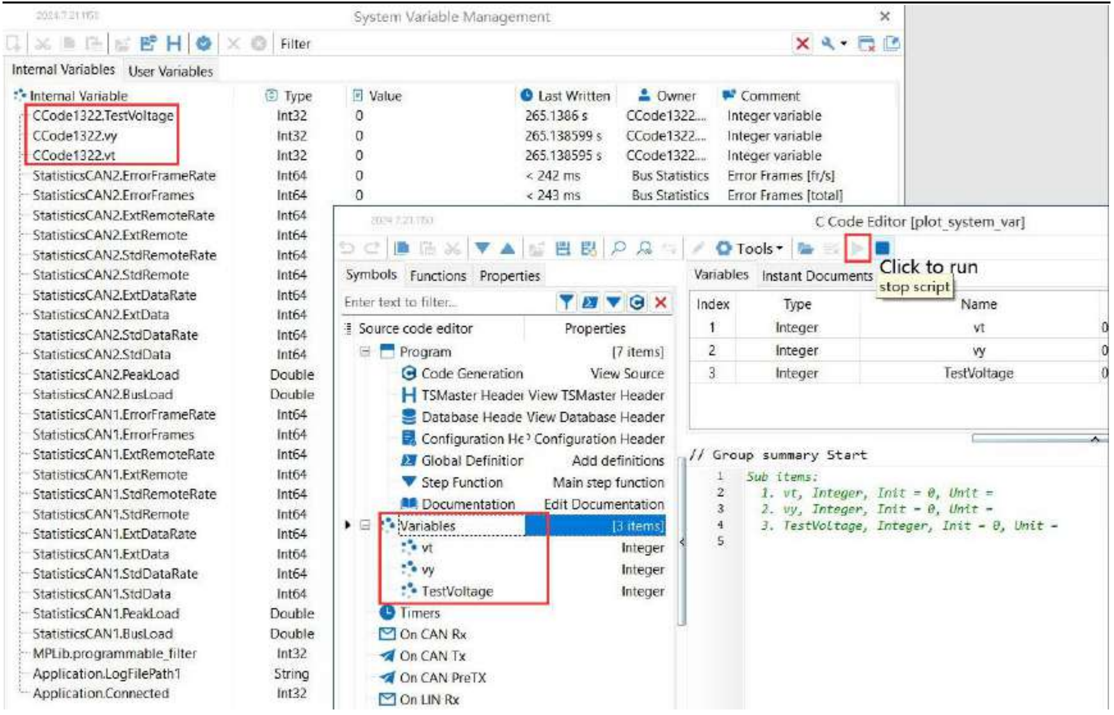

System Variable Management

Internal Variables | User Variables

Internal Variable	Type	Value	Last Written	Owner	Comment
StatisticsCAN2.ErrorFrameRate	Int64	0	< 49 ms	Bus Statistics	Error Frames [fr/s]
StatisticsCAN2.ErrorFrames	Int64	0	< 50 ms	Bus Statistics	Error Frames [total]
StatisticsCAN2.ExtRemoteRate	Int64	0	< 51 ms	Bus Statistics	Ext. Remote [fr/s]
StatisticsCAN2.ExtRemote	Int64	0	< 51 ms	Bus Statistics	Ext. Remote [total]
StatisticsCAN2.StdRemoteRate	Int64	0	< 52 ms	Bus Statistics	Std. Remote [fr/s]
StatisticsCAN2.StdRemote	Int64	0	< 53 ms	Bus Statistics	Std. Remote [total]
StatisticsCAN2.ExtDataRate	Int64	0	< 53 ms	Bus Statistics	Ext. Data [fr/s]
StatisticsCAN2.ExtData	Int64	0	< 54 ms	Bus Statistics	Ext. Data [total]
StatisticsCAN2.StdDataRate	Int64	0	< 55 ms	Bus Statistics	Std. Data [fr/s]
StatisticsCAN2.StdData	Int64	0	< 55 ms	Bus Statistics	Std. Data [total]
StatisticsCAN2.PeakLoad	Double	0.000 %	< 56 ms	Bus Statistics	Peak load [%]
StatisticsCAN2.BusLoad	Double	0.000 %	< 57 ms	Bus Statistics	Bus Load [%]
StatisticsCAN1.ErrorFrameRate	Int64	0	< 57 ms	Bus Statistics	Error Frames [fr/s]
StatisticsCAN1.ErrorFrames	Int64	0	< 58 ms	Bus Statistics	Error Frames [total]
StatisticsCAN1.ExtRemoteRate	Int64	0	< 58 ms	Bus Statistics	Ext. Remote [fr/s]
StatisticsCAN1.ExtRemote	Int64	0	< 59 ms	Bus Statistics	Ext. Remote [total]
StatisticsCAN1.StdRemoteRate	Int64	0	< 60 ms	Bus Statistics	Std. Remote [fr/s]
StatisticsCAN1.StdRemote	Int64	0	< 61 ms	Bus Statistics	Std. Remote [total]
StatisticsCAN1.ExtDataRate	Int64	0	< 62 ms	Bus Statistics	Ext. Data [fr/s]
StatisticsCAN1.ExtData	Int64	0	< 62 ms	Bus Statistics	Ext. Data [total]
StatisticsCAN1.StdDataRate	Int64	0	< 63 ms	Bus Statistics	Std. Data [fr/s]
StatisticsCAN1.StdData	Int64	0	< 63 ms	Bus Statistics	Std. Data [total]
StatisticsCAN1.PeakLoad	Double	0.000 %	< 64 ms	Bus Statistics	Peak load [%]
StatisticsCAN1.BusLoad	Double	0.000 %	< 65 ms	Bus Statistics	Bus Load [%]
MPLib.programmable_filter	Int32	0	n.a.	Mini Progra...	Run state of mini program...
Application.LogFilePath1	String	C:\Users\17166\Desktop...	n.a.	TSMaster	Specified log file in bus lo...
Application.Connected	Int32	1	0.012781 s	TSMaster	TSMaster Application con...

This is because mini-program variables are loaded dynamically. TSMaster supports the execution of any number of mini-programs, and each mini-program may have its own set of variables. Therefore, only after clicking to run a script, the system variables built into that mini-program will be loaded into the system, making them accessible to other modules.

Therefore, by finding and clicking to run the script, the mini-program variables will reappear in the system variables, as shown in the following figure:



System Variable Management

Internal Variables | User Variables

Internal Variable	Type	Value	Last Written	Owner	Comment
CCode1322.TestVoltage	Int32	0	265.1386 s	CCode1322....	Integer variable
CCode1322.vy	Int32	0	265.138599 s	CCode1322....	Integer variable
CCode1322.vt	Int32	0	265.138595 s	CCode1322....	Integer variable
StatisticsCAN2.ErrorFrameRate	Int64	0	< 242 ms	Bus Statistics	Error Frames [fr/s]
StatisticsCAN2.ErrorFrames	Int64	0	< 243 ms	Bus Statistics	Error Frames [total]
StatisticsCAN2.ExtRemoteRate	Int64				
StatisticsCAN2.ExtRemote	Int64				
StatisticsCAN2.StdRemoteRate	Int64				
StatisticsCAN2.StdRemote	Int64				
StatisticsCAN2.ExtDataRate	Int64				
StatisticsCAN2.ExtData	Int64				
StatisticsCAN2.StdDataRate	Int64				
StatisticsCAN2.StdData	Int64				
StatisticsCAN2.PeakLoad	Double				
StatisticsCAN2.BusLoad	Double				
StatisticsCAN1.ErrorFrameRate	Int64				
StatisticsCAN1.ErrorFrames	Int64				
StatisticsCAN1.ExtRemoteRate	Int64				
StatisticsCAN1.ExtRemote	Int64				
StatisticsCAN1.StdRemoteRate	Int64				
StatisticsCAN1.StdRemote	Int64				
StatisticsCAN1.ExtDataRate	Int64				
StatisticsCAN1.ExtData	Int64				
StatisticsCAN1.StdDataRate	Int64				
StatisticsCAN1.StdData	Int64				
StatisticsCAN1.PeakLoad	Double				
StatisticsCAN1.BusLoad	Double				
MPLib.programmable_filter	Int32				
Application.LogFilePath1	String				
Application.Connected	Int32				

C Code Editor [plot_system_var]

Source code editor	Properties
Program	[7 items]
Code Generation	View Source
TSMaster Header	View TSMaster Header
Database Header	View Database Header
Configuration He	Configuration Header
Global Definition	Add definitions
Step Function	Main step function
Documentation	Edit Documentation
Variables	[3 items]
vt	Integer
vy	Integer
TestVoltage	Integer
Timers	
On CAN Rx	
On CAN Tx	
On CAN PreTX	
On LIN Rx	

Index	Type	Name
1	Integer	vt
2	Integer	vy
3	Integer	TestVoltage

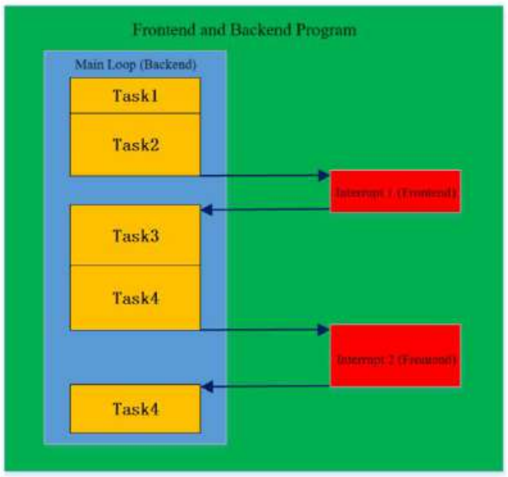
```
// Group summary Start
Sub items:
  1. vt, Integer, Init = 0, Unit =
  2. vy, Integer, Init = 0, Unit =
  3. TestVoltage, Integer, Init = 0, Unit =
```

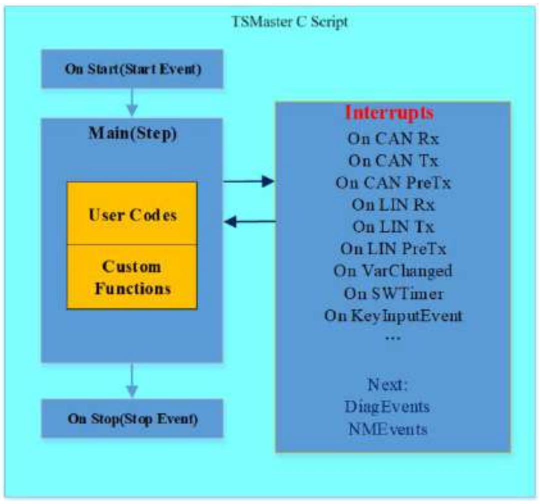
1.15. C Script

1.15.1. The Mechanism of C Script Execution

1.15.1.1. Software Architecture

In TSMaster, each test script can be metaphorically understood as an MCU, and its internal code framework adopts a frontend-backend model, as shown in the figure below:





TSMaster C Script Software Architecture

The frontend and backend program architecture mainly consists of a large loop, which is what we call the backend. This backend process runs continuously by default; then there's the frontend, based on event-driven (interrupt mechanism). In TSMaster's C script, the main loop is the step function. Frontend interrupts are various events, such as the CAN receive completion event On CAN Rx, etc.

After completing the configuration and code editing in the script editor, the final dynamically generated complete codes for the entire project can be viewed in the "Code Generation" section. Therefore, the "Code Generation" section is read-only.

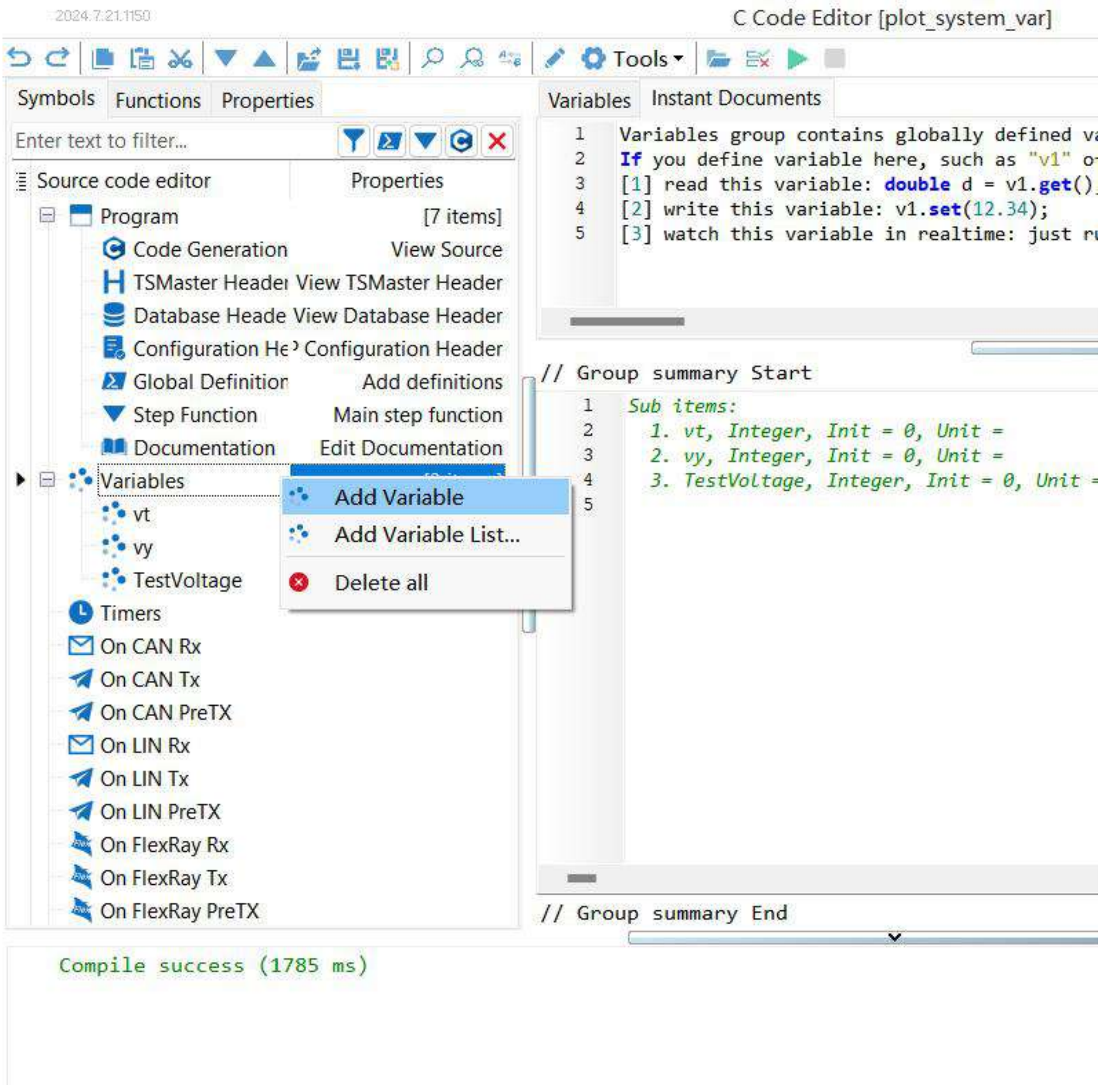
1.15.1.2. Mini-program Variable (Variable in Mini-program)

In the C language, variables typically include global variables and local variables. In TSMaster's C scripts, there is an additional data type called mini-program variables, which have the following characteristics:

- ➢ Mini-program variables are essentially a wrapper around global variables, thus inheriting the same scope of access as global variables.
- ➢ During the execution of the script program, users can directly view the values of mini-program variables, modify them directly, and trigger value change events when these variables change. These three features are also the main purposes behind the creation of mini-program variables.
- ➢ Mini-program variables cannot be directly read or written to. Instead, their values are set through a set function and retrieved through a get function.

1.15.1.2.1. Create Mini-Program Variable

1. Add mini-program variable by selecting a system variable, right click and select “Add Variable”.

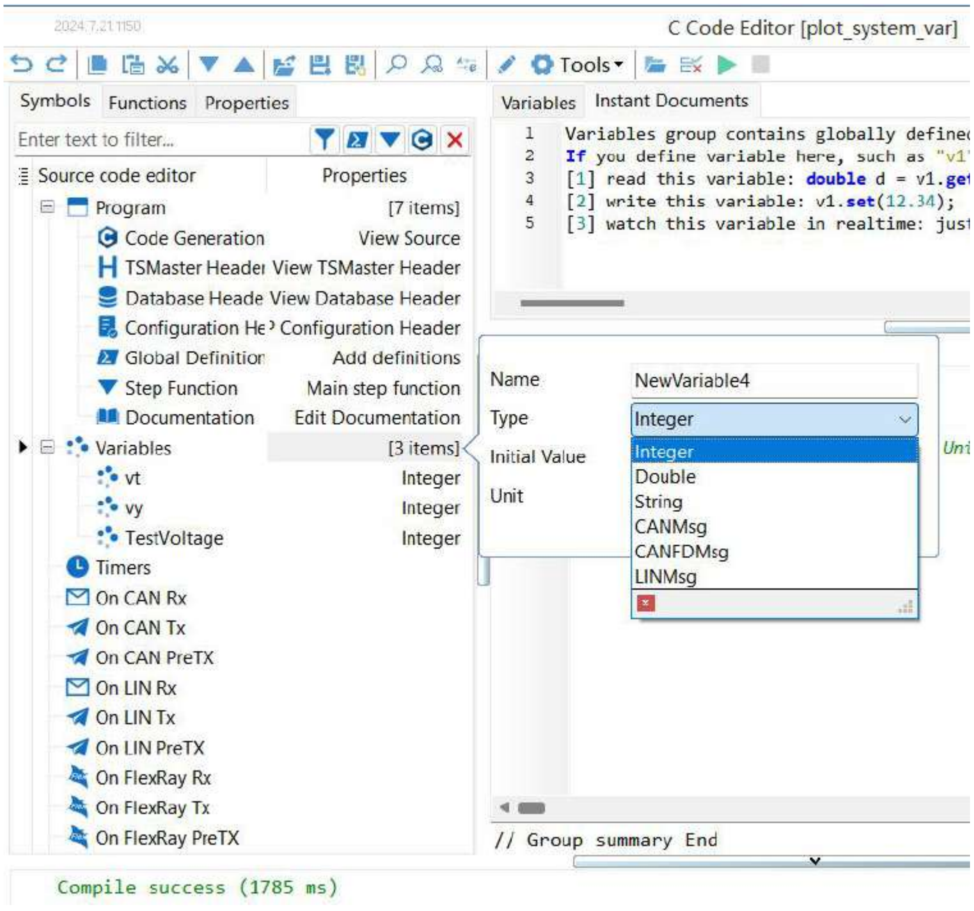


```
Variables group contains globally defined va
If you define variable here, such as "v1" o
[1] read this variable: double d = v1.get()
[2] write this variable: v1.set(12.34);
[3] watch this variable in realtime: just ru
```

```
// Group summary Start
Sub items:
  1. vt, Integer, Init = 0, Unit =
  2. vy, Integer, Init = 0, Unit =
  3. TestVoltage, Integer, Init = 0, Unit =

// Group summary End
```

```
Compile success (1785 ms)
```

2. Setting mini-program variable properties: Name + Type.



```
Variables group contains globally define
If you define variable here, such as "v1
[1] read this variable: double d = v1.get
[2] write this variable: v1.set(12.34);
[3] watch this variable in realtime: just
```

```
// Group summary End
```

```
Compile success (1785 ms)
```

1.15.1.2.2. Access MP Variable

Associate MP Variable in Panel

MP Variable is essentially an internally generated system variable. Therefore, the operation of associating a Panel with an MP Variable is described in 1.13.5.1 Panel Associated System Variables.

Cross-script Reading and Writing of MP Variable

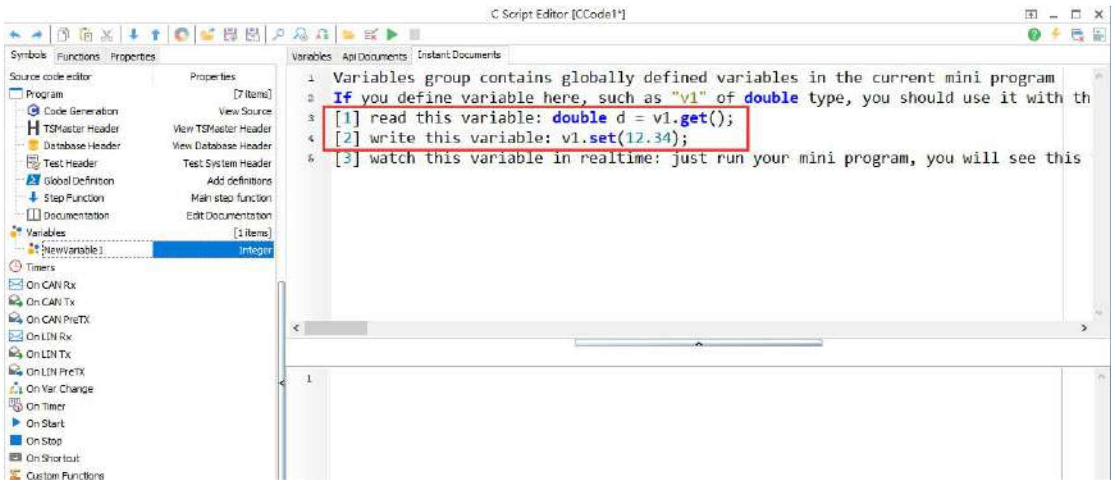
MP variable is essentially an internally generated system variable. Therefore, the operation of associating a panel with a MP variable is described in 1.13.5.2 Reading and Writing System Variables through Scripts.

Reading and Writing MP Variable within A Script

Accessing mini-program variables in local scripts differs from accessing ordinary variables. In simple terms, reading and writing to them involves:

- Reading: var.get();
- Writing: var.set(Value).

Within the script editor, after selecting the variable, there will be detailed parameter descriptions and usage instructions for that variable located in the upper-right corner.

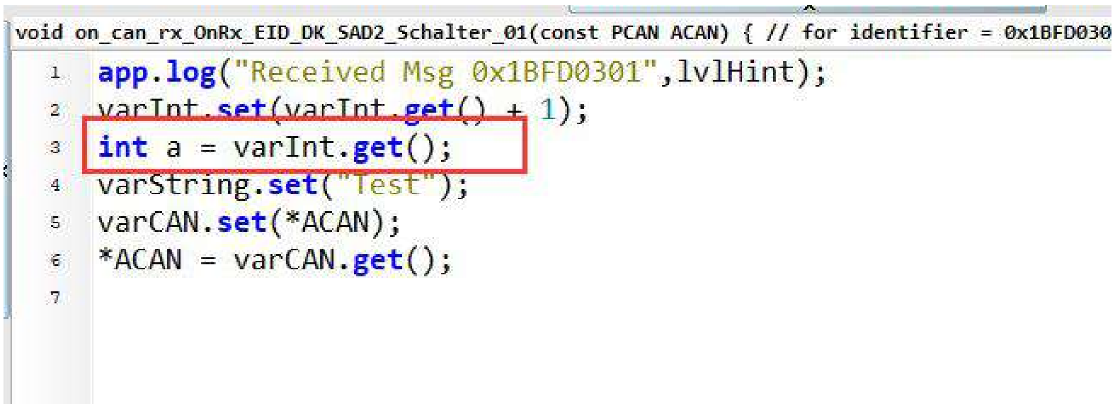


```
Variables group contains globally defined variables in the current mini program
If you define variable here, such as "v1" of double type, you should use it with th
[1] read this variable: double d = v1.get();
[2] write this variable: v1.set(12.34);
[3] watch this variable in realtime: just run your mini program, you will see this
```

The data type returned by get and the data type written by set correspond to the data type of the mini-program variable. For example, if there is a variable named varInt, which corresponds to the Integer type, then the data written by set and the data returned by get must be of the integer type. A detailed example is shown below:

- Reading the variable:

Use variable varInt as an example:

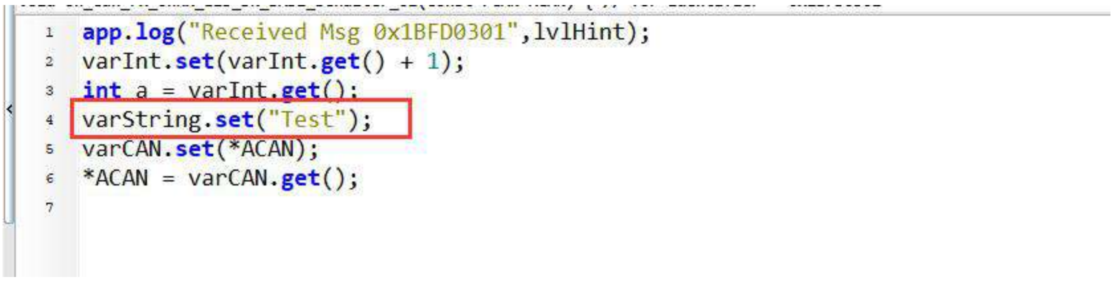


```
void on_can_rx_OnRx_EID_DK_SAD2_Schalter_01(const PCAN ACAN) { // for identifier = 0x1BFD030
app.log("Received Msg 0x1BFD0301",lvlHint);
varInt.set(varInt.get() + 1);
int a = varInt.get();
varString.set("Test");
varCAN.set(*ACAN);
*ACAN = varCAN.get();
```

It indicates reading the value of varInt and assign to variable 'a';

- Writing the variable:

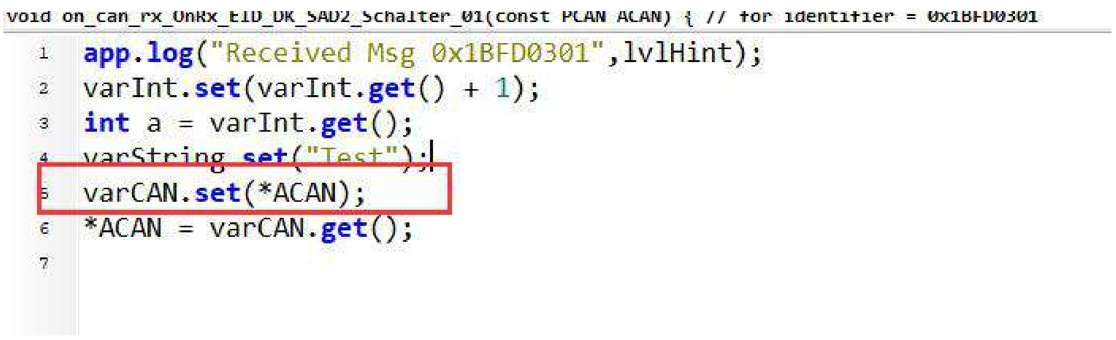
Use variable vatString as an example:



```
app.log("Received Msg 0x1BFD0301",lvlHint);
varInt.set(varInt.get() + 1);
int a = varInt.get();
varString.set("Test");
varCAN.set(*ACAN);
*ACAN = varCAN.get();
```

It indicates assigning the string "Test" to the variable varString.

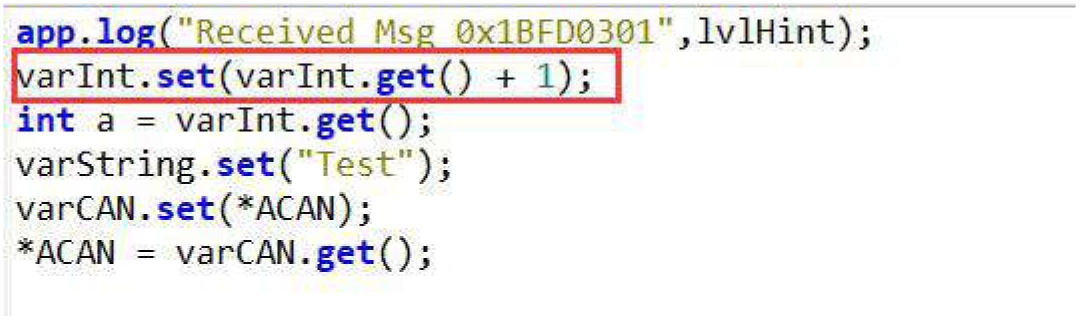
Use variable varCAN as an example:



```
void on_can_rx_OnRx_EID_DK_SAD2_Schalter_01(const PCAN ACAN) { // for identifier = 0x1BFD0301
app.log("Received Msg 0x1BFD0301",lvlHint);
varInt.set(varInt.get() + 1);
int a = varInt.get();
varString.set("Test");
varCAN.set(*ACAN);
*ACAN = varCAN.get();
```

Assign the received ACAN message to the variable varCAN.

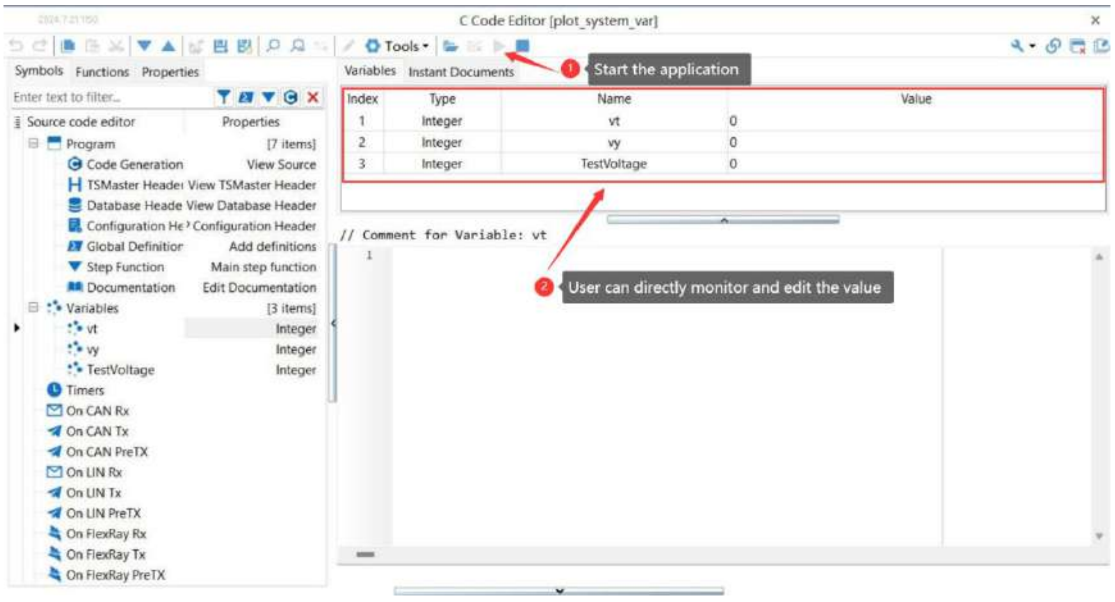
Use variable varInt as an example:



```
app.log("Received Msg 0x1BFD0301",lvlHint);
varInt.set(varInt.get() + 1);
int a = varInt.get();
varString.set("Test");
varCAN.set(*ACAN);
*ACAN = varCAN.get();
```

Indicates that the vatInt value has been incremented by 1.

Debugging Mini-program Variable



Index	Type	Name	Value
1	Integer	vt	0
2	Integer	vy	0
3	Integer	TestVoltage	0

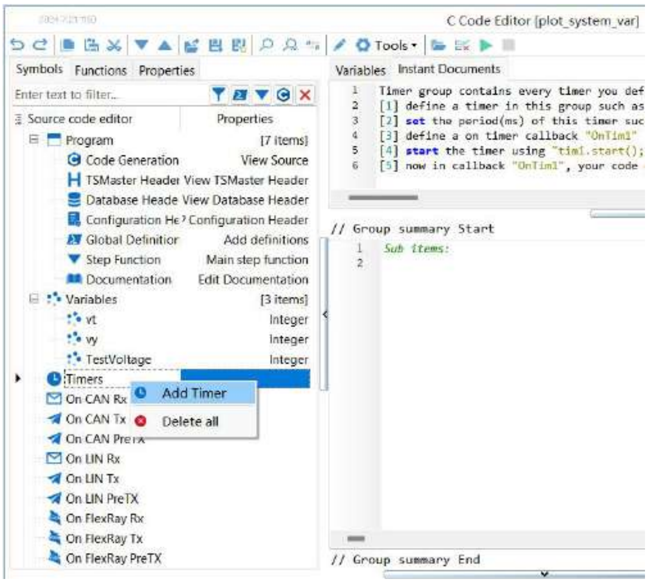
```
// Comment for Variable: vt
```

Essentially, this feature is currently unique to TSMaster, making it extremely convenient for users to monitor and manually modify variable values during the debugging process. It is recommended that users declare the variables that need to be dynamically monitored and modified during the testing process as mini-program variables.

1.15.1.3. Timer Software Timer Variable

In order to introduce software timer events into the event mechanism, it is necessary to first create a software timer variable, which serves as the trigger source for driving software timer events. This can be done as follows:

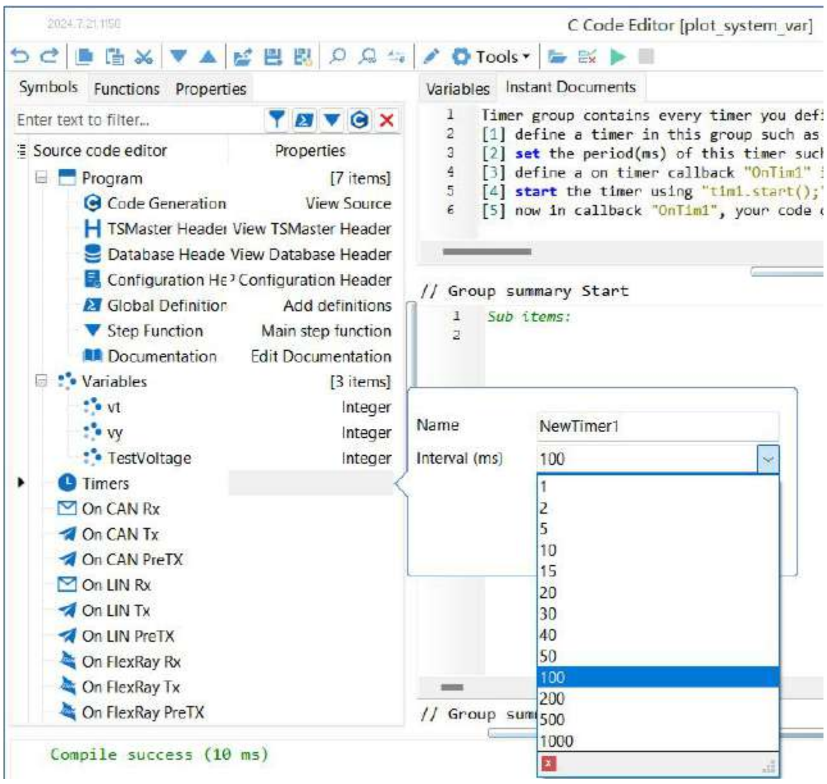
1. Add timer by right clicking “Timers” and select “Add Timer”.



```
Timer group contains every timer you def
[1] define a timer in this group such as
[2] set the period(ms) of this timer suc
[3] define a on timer callback "OnTim1"
[4] start the timer using "tim1.start();
[5] now in callback "OnTim1", your code
```

```
// Group summary Start
Sub items:

// Group summary End
```

2. Edit timer properties: name and interval.



```
Timer group contains every timer you def
[1] define a timer in this group such as
[2] set the period(ms) of this timer suc
[3] define a on timer callback "OnTim1"
[4] start the timer using "tim1.start();
[5] now in callback "OnTim1", your code
```

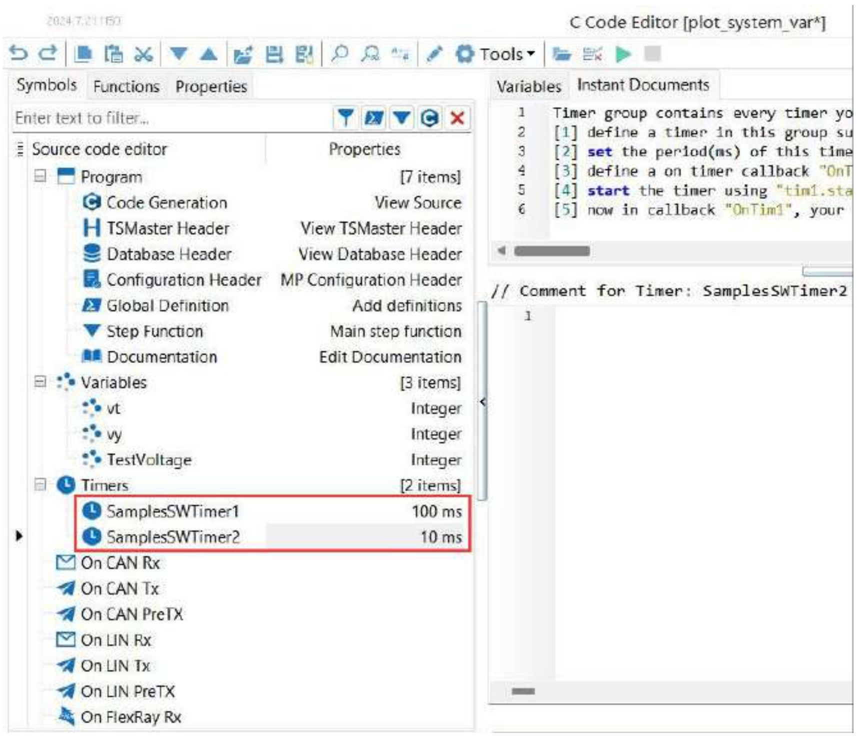
```
// Group summary Start
Sub items:
```

Name	NewTimer1
Interval (ms)	100

```
Compile success (10 ms)
```

After creation, several software timer objects are added to the system, serving as the trigger sources for the subsequent OnTimer event.

As shown below, the two timers have different intervals:



Source code editor	Properties
Program	[7 items]
Code Generation	View Source
TSMaster Header	View TSMaster Header
Database Header	View Database Header
Configuration Header	MP Configuration Header
Global Definition	Add definitions
Step Function	Main step function
Documentation	Edit Documentation
Variables	[3 items]
vt	Integer
vy	Integer
TestVoltage	Integer
Timers	[2 items]
SamplesSWTimer1	100 ms
SamplesSWTimer2	10 ms
On CAN Rx	
On CAN Tx	
On CAN PreTX	
On LIN Rx	
On LIN Tx	
On LIN PreTX	
On FlexRay Rx	

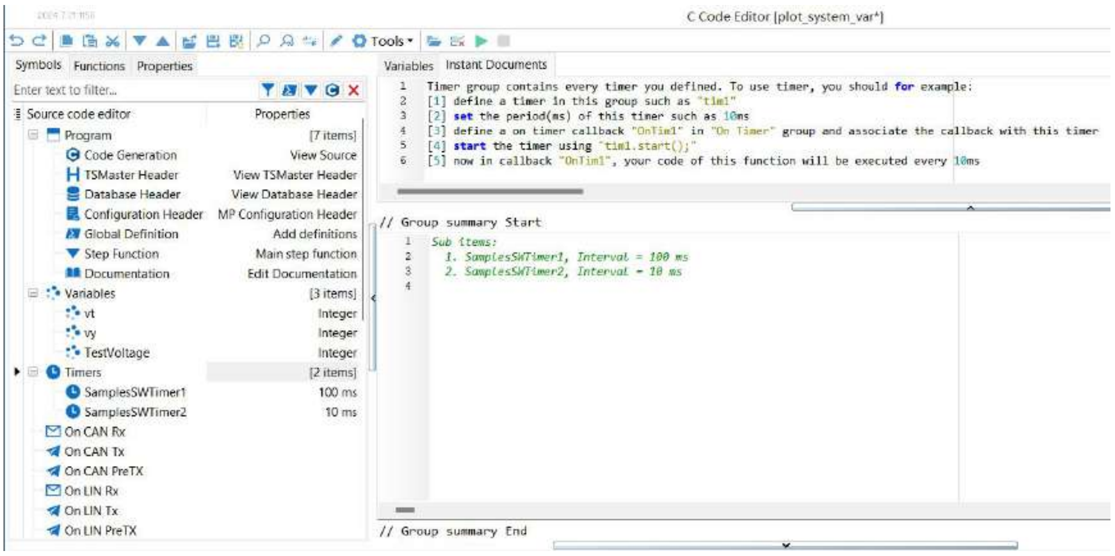
```
Timer group contains every timer yo
[1] define a timer in this group su
[2] set the period(ms) of this time
[3] define a on timer callback "OnT
[4] start the timer using "tim1.sta
[5] now in callback "OnTim1", your
```

```
// Comment for Timer: SamplesSWTimer2
```

1.15.1.3.1. Start/Stop Timer

After creating the software Timer variable, user can directly start and stop the Timer by calling its property functions. By selecting the Timer, the instructions will be automatically shown.

For example, [4] describe how to start the timer.



Source code editor	Properties
Program	[7 items]
Code Generation	View Source
TSMaster Header	View TSMaster Header
Database Header	View Database Header
Configuration Header	MP Configuration Header
Global Definition	Add definitions
Step Function	Main step function
Documentation	Edit Documentation
Variables	[3 items]
vt	Integer
vy	Integer
TestVoltage	Integer
Timers	[2 items]
SamplesSWTimer1	100 ms
SamplesSWTimer2	10 ms
On CAN Rx	
On CAN Tx	
On CAN PreTX	
On LIN Rx	
On LIN Tx	
On LIN PreTX	

```
Timer group contains every timer you defined. To use timer, you should for example:
[1] define a timer in this group such as "tim1"
[2] set the period(ms) of this timer such as 10ms
[3] define a on timer callback "OnTim1" in "On Timer" group and associate the callback with this timer
[4] start the timer using "tim1.start();"
[5] now in callback "OnTim1", your code of this function will be executed every 10ms
```

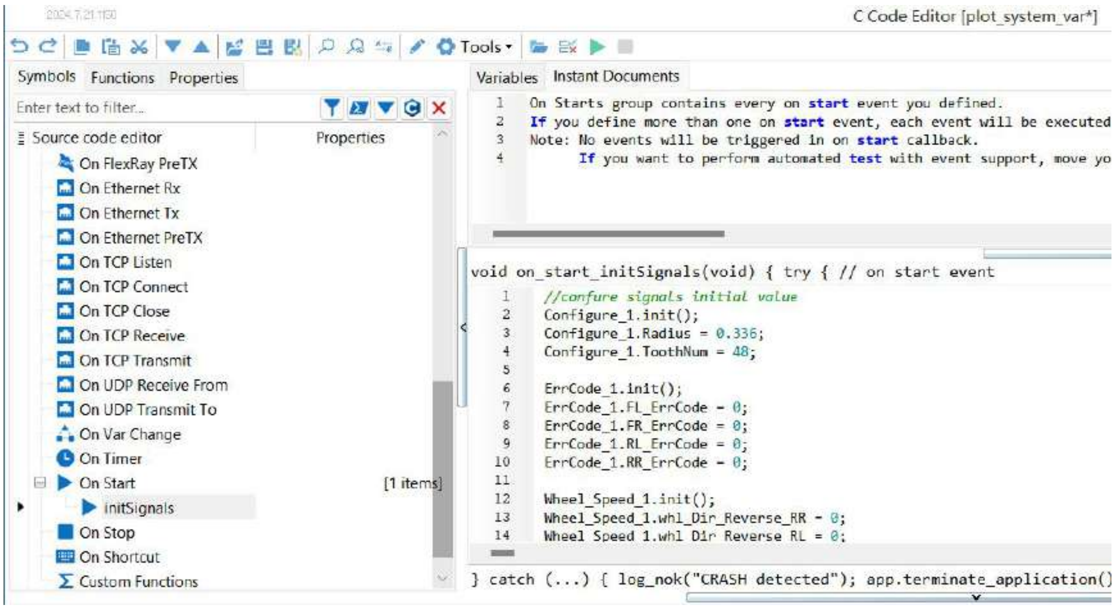
```
// Group summary Start
Sub items:
  1. SamplesSWTimer1, Interval = 100 ms
  2. SamplesSWTimer2, Interval = 10 ms

// Group summary End
```

According to the instructions, to start the timer event for the Timer, user can execute the code: SampleSWTimer1.start(); To stop the Timer, user can execute the code: SampleSWTimer1.stop();

1.15.1.4. Start/Stop

1.15.1.4.1. OnStart

The function event that is executed when the entire C script program is launched. The addition process is the same as above.



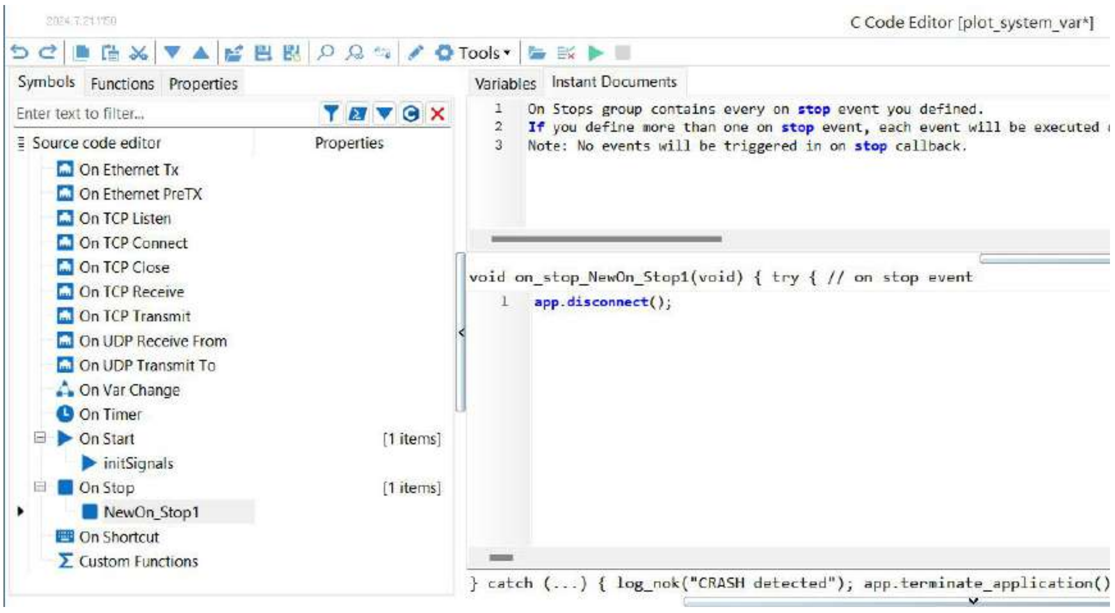
```
On Starts group contains every on start event you defined.
If you define more than one on start event, each event will be executed
Note: No events will be triggered in on start callback.
      If you want to perform automated test with event support, move yo
```

```
void on_start_initSignals(void) { try { // on start event
    //confure signals initial value
    Configure_1.init();
    Configure_1.Radius = 0.336;
    Configure_1.ToothNum = 48;

    ErrCode_1.init();
    ErrCode_1.FL_ErrCode = 0;
    ErrCode_1.FR_ErrCode = 0;
    ErrCode_1.RL_ErrCode = 0;
    ErrCode_1.RR_ErrCode = 0;

    Wheel_Speed_1.init();
    Wheel_Speed_1.whl_Dir_Reverse_RR = 0;
    Wheel_Speed_1.whl_Dir_Reverse_RL = 0;
} catch (...) { log_nok("CRASH detected"); app.terminate_application()
```

For example, the “initSignals” event is executed on start, and the right side are the logics to initiate the variables, configure the hardware parameters and perform the connection.

1.15.1.4.2. OnStop

The function event that is executed when the entire C script program is terminated. The addition process is the same as above.



```
On Stops group contains every on stop event you defined.
If you define more than one on stop event, each event will be executed
Note: No events will be triggered in on stop callback.
```

```
void on_stop_NewOn_Stop1(void) { try { // on stop event
    app.disconnect();
} catch (...) { log_nok("CRASH detected"); app.terminate_application()
```

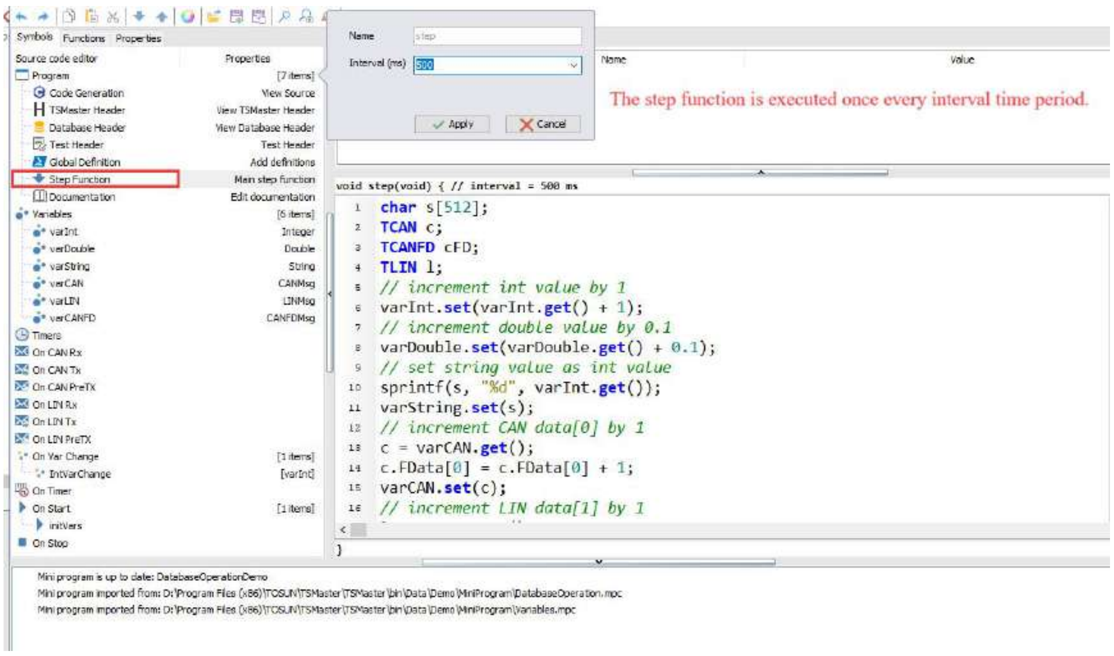
For example, the “NewOn_Stop1” event is executed on stop. This event performs the disconnect action.

1.15.1.4.3. Precautions (Warning):

Do not include mechanisms like While(1) in the OnStart/OnStop events, as this can cause the script to get stuck in the startup/shutdown steps, preventing it from entering the main loop and thus being unable to respond to event mechanisms.

1.15.1.5. Step Mechanism

This mechanism provides a main loop scheduling interface for users' program logic. Users can run their algorithms based on this periodic scheduling interface. Taking an example as an illustration, it is shown as follows:



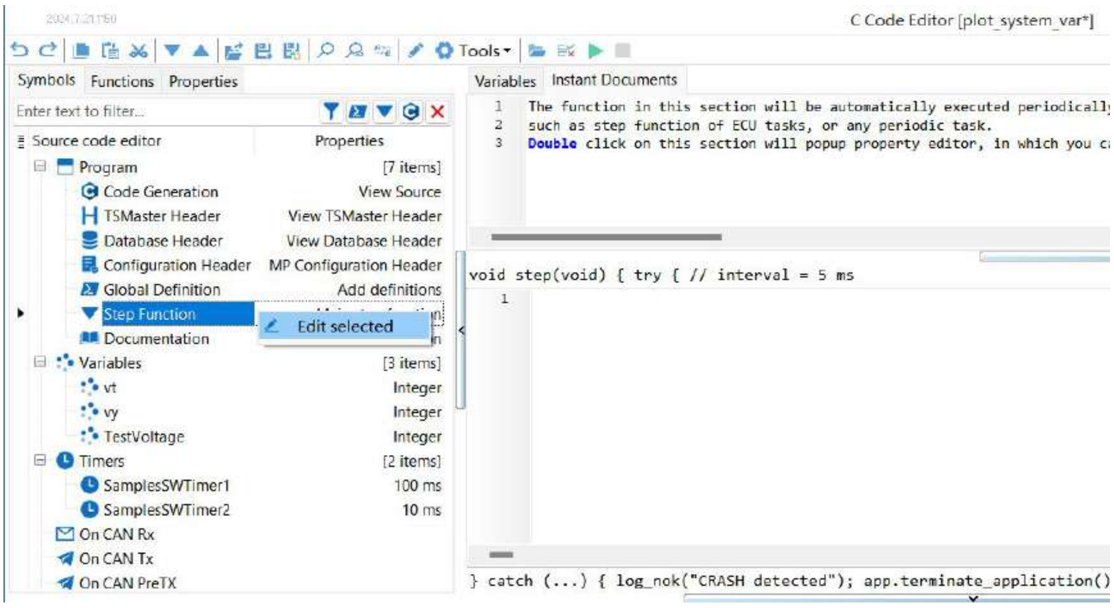
Name: step
Interval (ms): 500

The step function is executed once every interval time period.

```
void step(void) { // interval = 500 ms
char s[512];
TCAN c;
TCANFD cFD;
TLIN l;
// increment int value by 1
varInt.set(varInt.get() + 1);
// increment double value by 0.1
varDouble.set(varDouble.get() + 0.1);
// set string value as int value
sprintf(s, "%d", varInt.get());
varString.set(s);
// increment CAN data[0] by 1
c = varCAN.get();
c.FData[0] = c.FData[0] + 1;
varCAN.set(c);
// increment LIN data[1] by 1
}
```

```
Mini program is up to date: DatabaseOperationDemo
Mini program imported from: D:\Program Files (x86)\TOSUN\TSMaster\TSMaster\bin\Data\Demo\MiniProgram\DatabaseOperation.mpc
Mini program imported from: D:\Program Files (x86)\TOSUN\TSMaster\TSMaster\bin\Data\Demo\MiniProgram\Variables.mpc
```

User adds code to the step function, where this logic is scheduled once every intervaltime time interval. To modify the interval time, right click and select “Edit selected”, then edit the interval time:

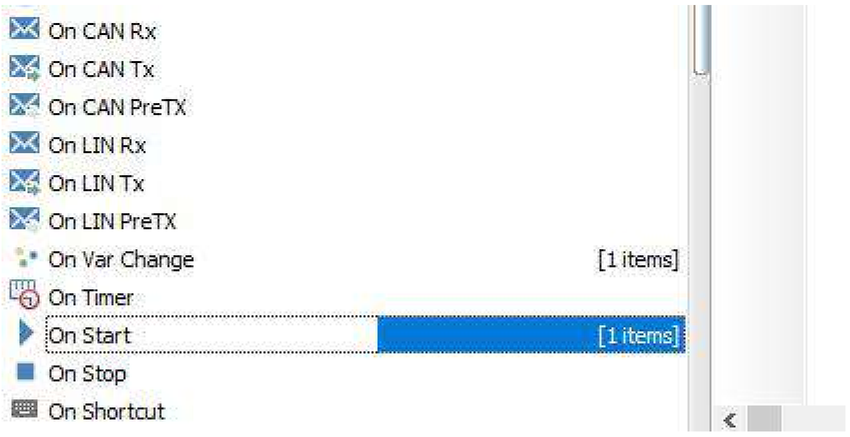


```
The function in this section will be automatically executed periodicall
such as step function of ECU tasks, or any periodic task.
Double click on this section will popup property editor, in which you c
```

```
void step(void) { try { // interval = 5 ms

} catch (...) { log_nok("CRASH detected"); app.terminate_application()
```

1.15.1.6. Event Mechanism

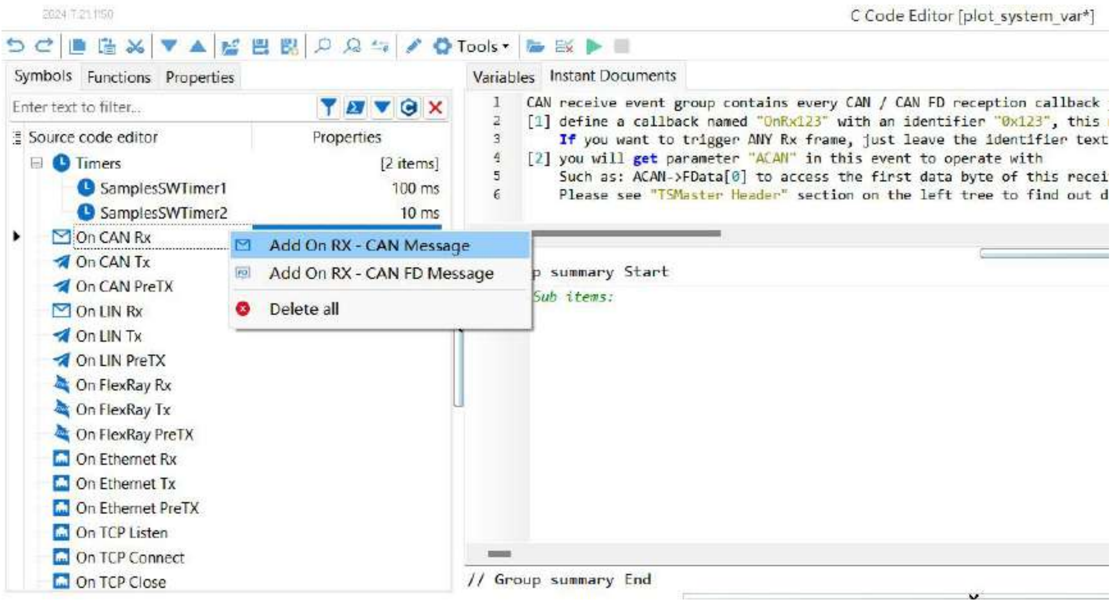
As the name suggests, the event-driven mechanism refers to a mechanism that triggers function calls immediately after the preset events occur. In the C script of TSMaster, the main event mechanisms include the following:



1.15.1.6.1. OnCANRx

The function event that is triggered when a CAN message with a specified ID is received. An example is as follows:

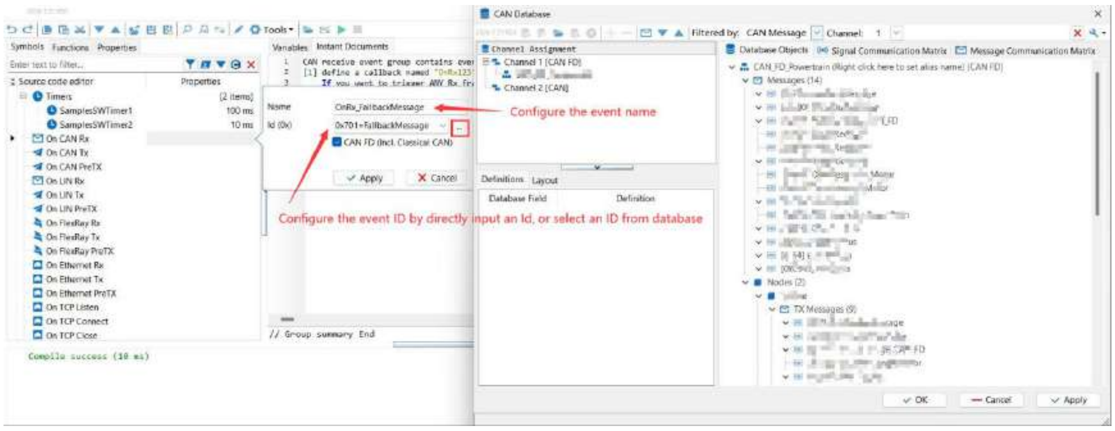
1. Right click and select "Add On RX - CAN Message" to add a on RX event.



```
CAN receive event group contains every CAN / CAN FD reception callback
[1] define a callback named "OnRx123" with an identifier "0x123", this
    If you want to trigger ANY Rx frame, just leave the identifier text
[2] you will get parameter "ACAN" in this event to operate with
    Such as: ACAN->FData[0] to access the first data byte of this recei
    Please see "TSMaster Header" section on the left tree to find out d
```

```
p summary Start
Sub items:
// Group summary End
```

2. Edit the CANID for the event.

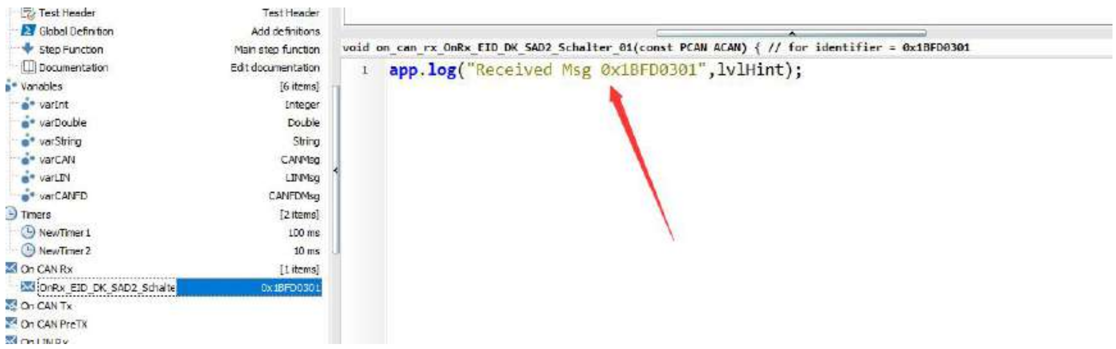

Configure the event name
Configure the event ID by directly input an Id, or select an ID from database
Name
OnRx_FallbackMessage
Id (0x)
0x701=FallbackMessage
CAN FD (Incl. Classical CAN)
Apply
Cancel
CAN Database
Channel Assignment
Database Objects
Signal Communication Matrix
Message Communication Matrix
OK
Cancel
Apply

```
CAN receive event group contains eve
[1] define a callback named "OnRx123
    If you want to trigger ANY Rx fr
```

```
// Group summary End
```

```
Compile success (10 ms)
```

3. Add logic to the event.

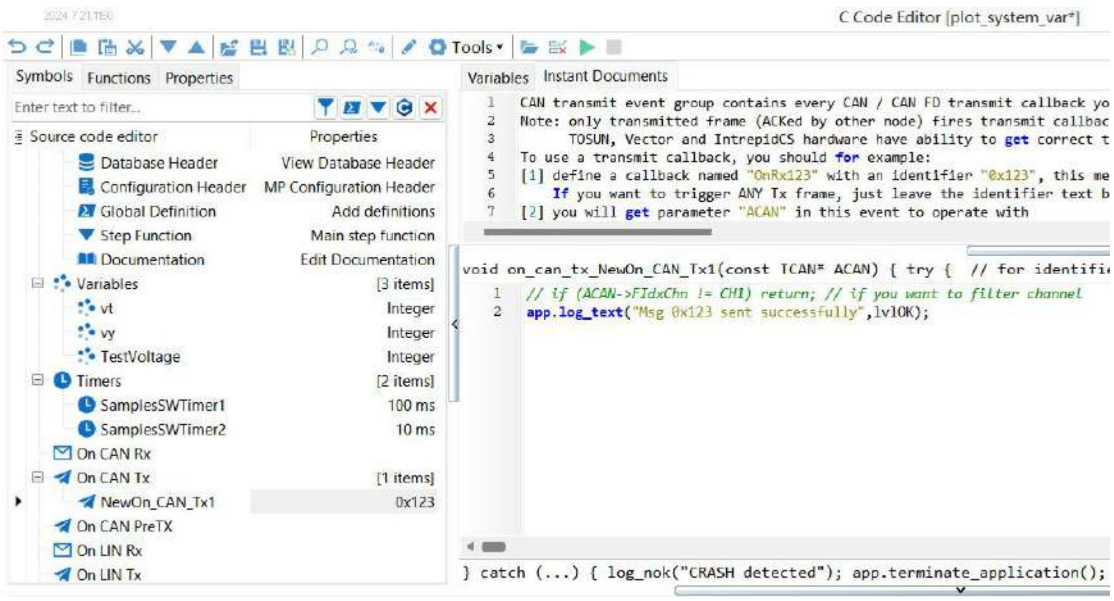


```
void on_can_rx_OnRx_EID_DK_SAD2_Schalter_01(const PCAN ACAN) { // for identifier = 0x1BFD0301
app.log("Received Msg 0x1BFD0301",lvlHint);
```

As shown, the logic is when a message with the identifier 0x1BFD0301 is received, then print the information.

1.15.1.6.2. OnCANTx

The function event that is triggered after a CAN message with a specified ID is successfully sent. The process of adding this event is the same as adding the OnCANRx message.



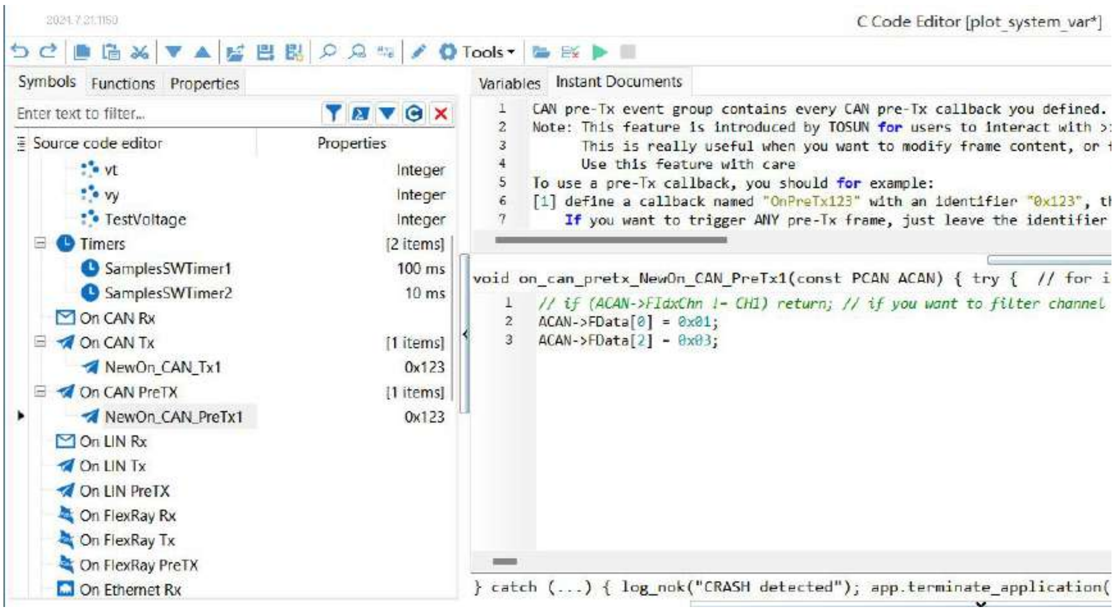
```
CAN transmit event group contains every CAN / CAN FD transmit callback yo
Note: only transmitted frame (ACKed by other node) fires transmit callbac
      TOSUN, Vector and IntrepidCS hardware have ability to get correct t
To use a transmit callback, you should for example:
[1] define a callback named "OnRx123" with an identifier "0x123", this me
    If you want to trigger ANY Tx frame, just leave the identifier text b
[2] you will get parameter "ACAN" in this event to operate with
```

```
void on_can_tx_NewOn_CAN_Tx1(const TCAN* ACAN) { try {  // for identifi
 // if (ACAN->FIdxChn != CH1) return; // if you want to filter channel
 app.log_text("Msg 0x123 sent successfully",lvlOK);
} catch (...) { log_nok("CRASH detected"); app.terminate_application();
```

As shown, the logic is when a 0x123 CAN message is sent, then print the successful information.

1.15.1.6.3. OnCANPreTx

The function event that is triggered before sending a CAN message with a specified ID. The process of adding this event is the same as before.



```
CAN pre-Tx event group contains every CAN pre-Tx callback you defined.
Note: This feature is introduced by TOSUN for users to interact with >
      This is really useful when you want to modify frame content, or 
      Use this feature with care
To use a pre-Tx callback, you should for example:
[1] define a callback named "OnPreTx123" with an identifier "0x123", t
    If you want to trigger ANY pre-Tx frame, just leave the identifier
```

```
void on_can_pretx_NewOn_CAN_PreTx1(const PCAN ACAN) { try {  // for i
 // if (ACAN->FIdxChn != CH1) return; // if you want to filter channel
 ACAN->FData[0] = 0x01;
 ACAN->FData[2] - 0x03;
} catch (...) { log_nok("CRASH detected"); app.terminate_application(
```

As shown, the logic is before sending a CAN message with a specified ID, modify the FData[0] and FData[2] values.

1.15.1.6.4. OnLINRx

The function event that is triggered when a LIN message with a specified ID is received. The process of adding this event is the same as before.

1.15.1.6.5. OnLINTx

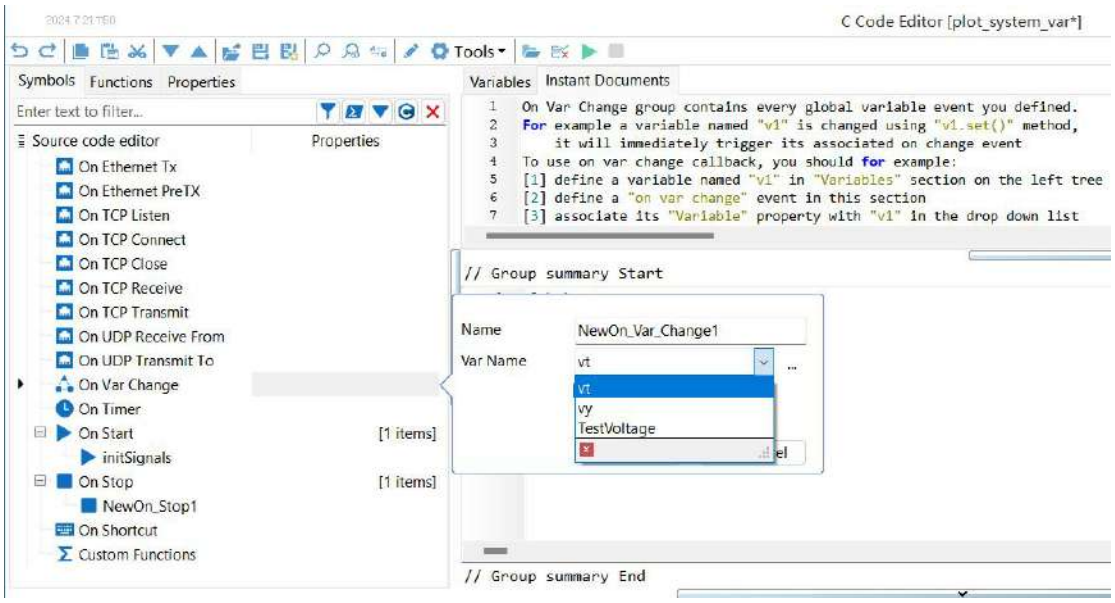
The function event that is triggered after a LIN message with a specified ID is successfully sent. The process of adding this event is the same as before.

1.15.1.6.6. OnLINPreTx

The function event that is triggered before sending a LIN message with a specified ID. The process of adding this event is the same as before.

1.15.1.6.7. OnVarChange

The function event that is triggered when the value of a mini-program variable changes. The process of adding this event is right click “On Var Change”, and select a system variable which created before from the list:



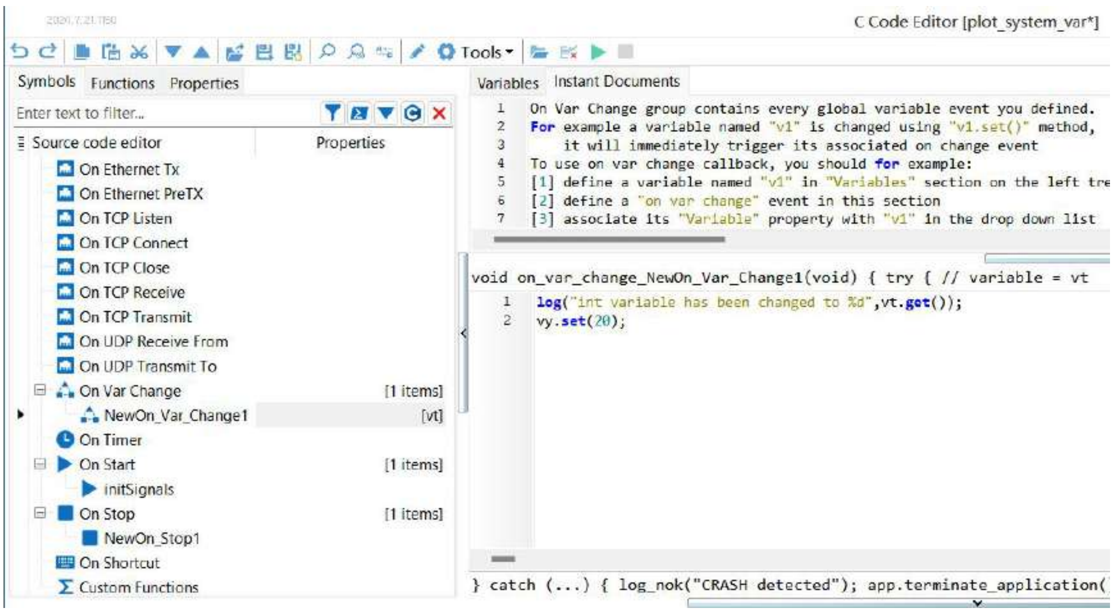
```
On Var Change group contains every global variable event you defined.
For example a variable named "v1" is changed using "v1.set()" method,
    it will immediately trigger its associated on change event
To use on var change callback, you should for example:
[1] define a variable named "v1" in "Variables" section on the left tree
[2] define a "on var change" event in this section
[3] associate its "Variable" property with "v1" in the drop down list
```

```
// Group summary Start
```

Name: NewOn_Var_Change1
Var Name: vt

```
// Group summary End
```

Add logic to the variable change event, as shown below:



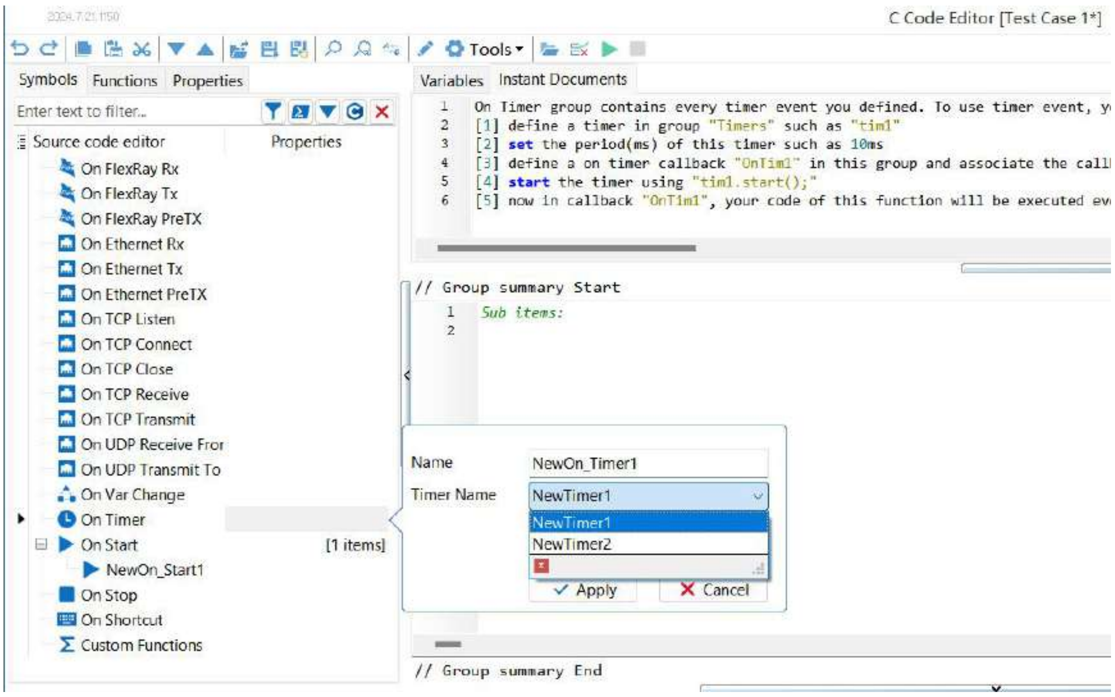
```
On Var Change group contains every global variable event you defined.
For example a variable named "v1" is changed using "v1.set()" method,
  it will immediately trigger its associated on change event
To use on var change callback, you should for example:
[1] define a variable named "v1" in "Variables" section on the left tre
[2] define a "on var change" event in this section
[3] associate its "Variable" property with "v1" in the drop down list
```

```
void on_var_change_NewOn_Var_Change1(void) { try { // variable = vt
log("int variable has been changed to %d",vt.get());
vy.set(20);
} catch (...) { log_nok("CRASH detected"); app.terminate_application(
```

As shown, when the int variable value is changed, the new value is printed out, and the vy variable value is set to 20.

1.15.1.6.8. OnTimer

Software Timer event, a function event that is triggered once the software timer expires. The process of adding this event is the same as above.



```
On Timer group contains every timer event you defined. To use timer event, y
[1] define a timer in group "Timers" such as "tim1"
[2] set the period(ms) of this timer such as 10ms
[3] define a on timer callback "OnTim1" in this group and associate the call
[4] start the timer using "tim1.start();"
[5] now in callback "OnTim1", your code of this function will be executed ev
```

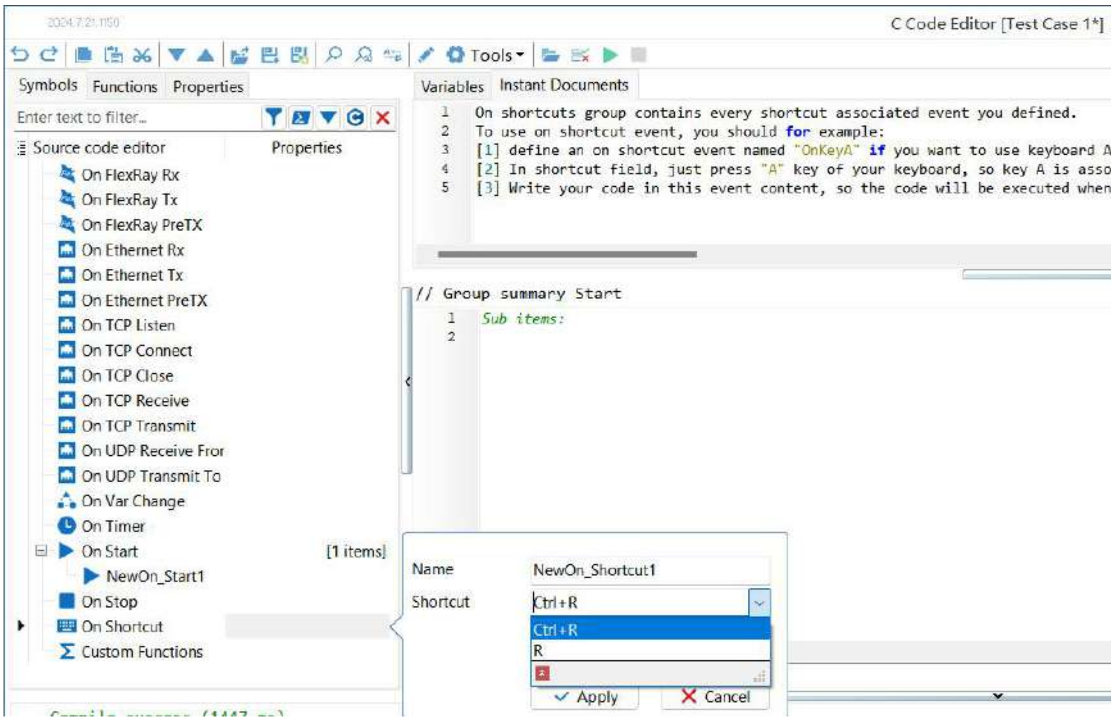
```
// Group summary Start
Sub items:

// Group summary End
```

Name: NewOn_Timer1
Timer Name: NewTimer1
NewTimer1
NewTimer2

Edit the event name and select the timer. The interval is depends on the selected timer.

1.15.1.6.9. OnShortCut

Function event triggered by a shortcut key. The process of adding this event is as follows:



```
On shortcuts group contains every shortcut associated event you defined.
To use on shortcut event, you should for example:
[1] define an on shortcut event named "OnKeyA" if you want to use keyboard A
[2] In shortcut field, just press "A" key of your keyboard, so key A is asso
[3] Write your code in this event content, so the code will be executed when
```

```
// Group summary Start
Sub items:
```

Name: NewOn_Shortcut1
Shortcut: Ctrl+R

Edit the event name and define the shortcut by pressing the key/keys on the keyboard.

1.15.1.6.10. Precautions (Warning)

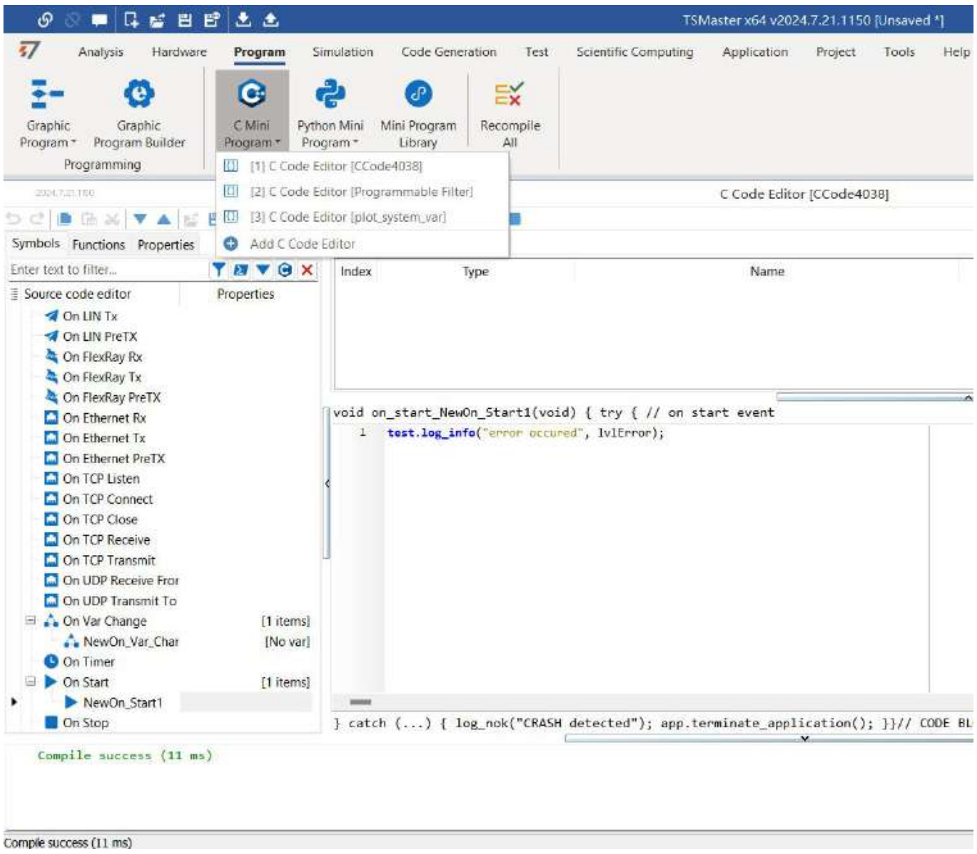
The event mechanism is similar to an interrupt in microcontrollers, where it is advised to avoid performing very time-consuming tasks within them, and it is strictly forbidden to execute tasks such as a While(1) loop that would directly cause the system to get stuck in that interrupt.

1.15.1.7. CAN Message Forwarding Example

1.15.2. Basic Usage Process

1.15.2.1. Create (or Open) Script

The process of creating (or opening) a script is shown in the figure below: Simulation->C Code Editor->C Script Editor (open existing)/Add C Script Editor (create new).



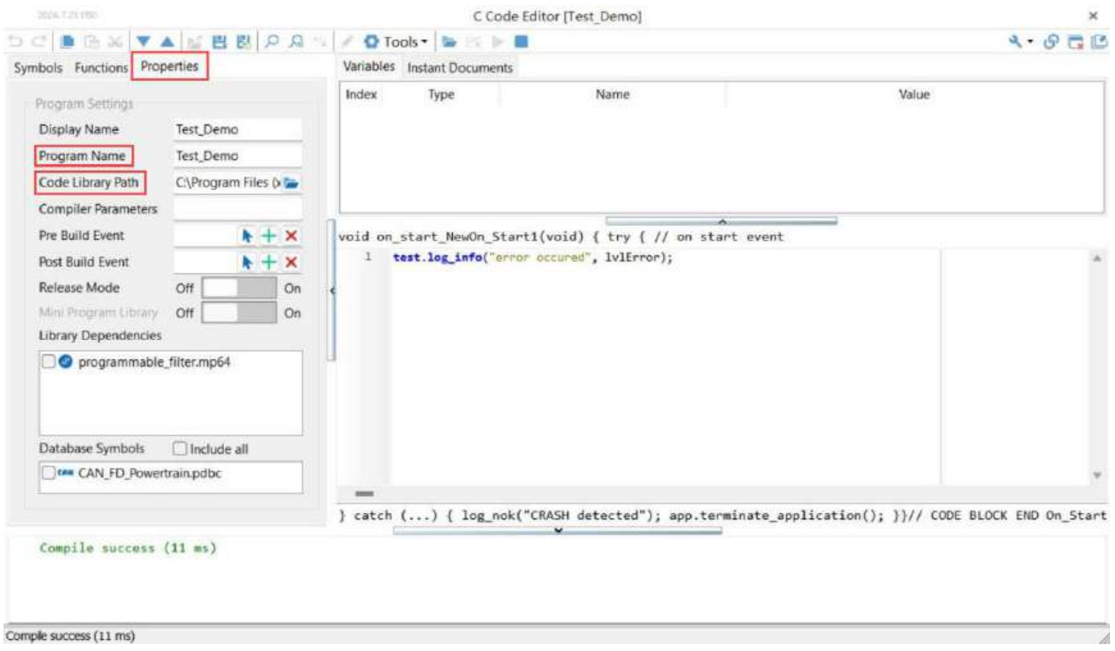
```
void on_start_NewOn_Start1(void) { try { // on start event
  test.log_info("error occured", lvlError);
} catch (...) { log_nok("CRASH detected"); app.terminate_application(); }}// CODE BL
```

```
Compile success (11 ms)
```

Compile success (11 ms)

1.15.2.2. Set Script Properties

Before editing and running the script, first set the script properties, which mainly include: program name, code library path, and library dependencies. As shown in the figure below:



Program Settings

Display Name	Test_Demo
Program Name	Test_Demo
Code Library Path	C:\Program Files (
Compiler Parameters	
Pre Build Event	
Post Build Event	
Release Mode	Off On
Mini Program Library	Off On

Library Dependencies

- programmable_filter.mp64

Database Symbols Include all

- CAN_FD_Powertrain.pdbc

```
void on_start_NewOn_Start1(void) { try { // on start event
test.log_info("error occured", lvlError);
} catch (...) { log_nok("CRASH detected"); app.terminate_application(); }}// CODE BLOCK END On_Start
```

```
Compile success (11 ms)
```

Compile success (11 ms)

Note: A C script is essentially a standalone application, just like any program based on the TSMaster API (including TSMaster itself), hardware mapping is performed through the application name. Therefore, users need to clearly know the name of their own program and map it to the actual hardware device based on this name. As shown in the figure above: Set the name of this C script to Test_Demo.

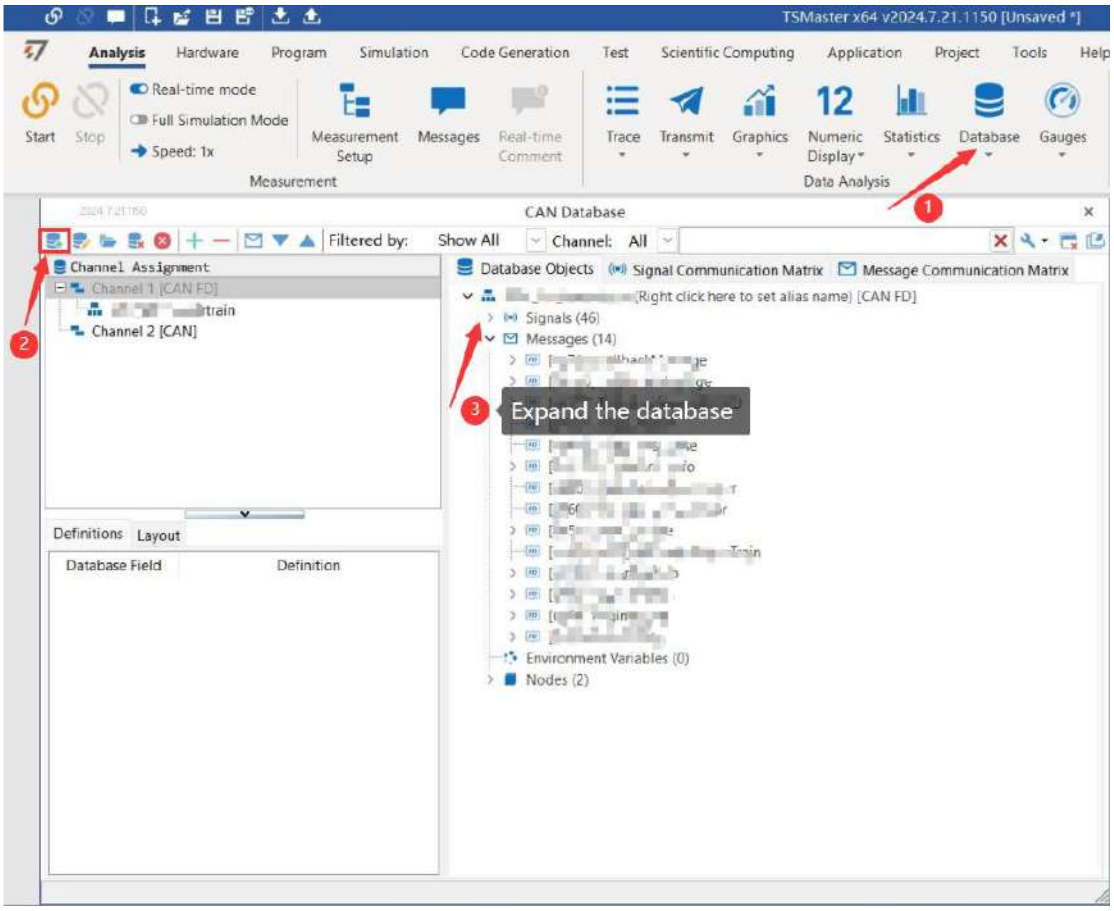
1.15.3. Database Signal Operations (Based on RBS)

Advantages and Disadvantages:

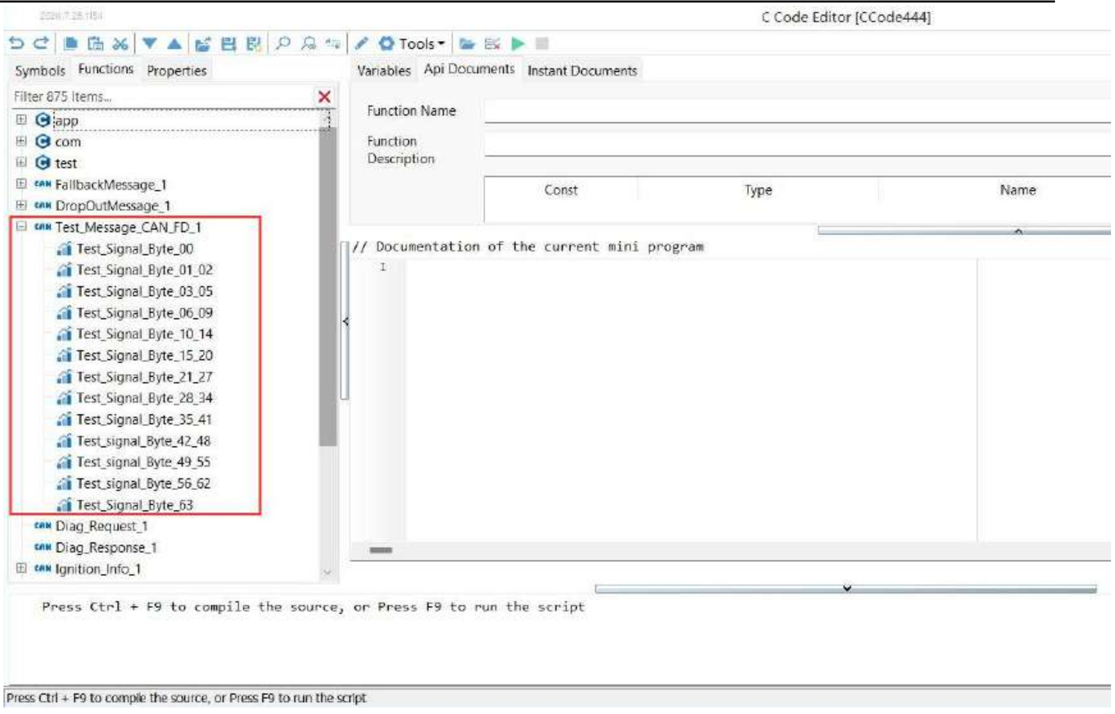
Advantages: Directly read and set signal values based on the signal path, which is simple and convenient. This method is recommended by default for reading and writing database signals.

Disadvantages: This operation mode is tightly coupled with the database, and in some test scenarios deliberately designed to create exceptions, this operation mode may not be fully satisfactory. The prerequisite for using this method is to start the CANRBS simulation; if the user simply wants to test the reception and transmission of a few isolated message frames, this method will not meet the requirements.

➢ First step: add database



After the database is loaded, you can see all the database messages and signals in the quick suggestion window of the script editor, as shown below:



```
// Documentation of the current mini program
```

```
Press Ctrl + F9 to compile the source, or Press F9 to run the script
```

Press Ctrl + F9 to compile the source, or Press F9 to run the script

Explanation: The purpose of adding a database is to dynamically generate C language data type definitions related to the messages and signals in the database, so users do not have to construct the numerous data structures for various CAN messages and signals themselves.

➢ Second step: Read/Write CAN Signal

Read CAN signal:

```
double d;
com.can_rbs_get_signal_value_by_address("0/CAN_FD_Powertrain/Engine/EngineData/EngSpeed", &d);
log("EngSpeed = %f", d);
```

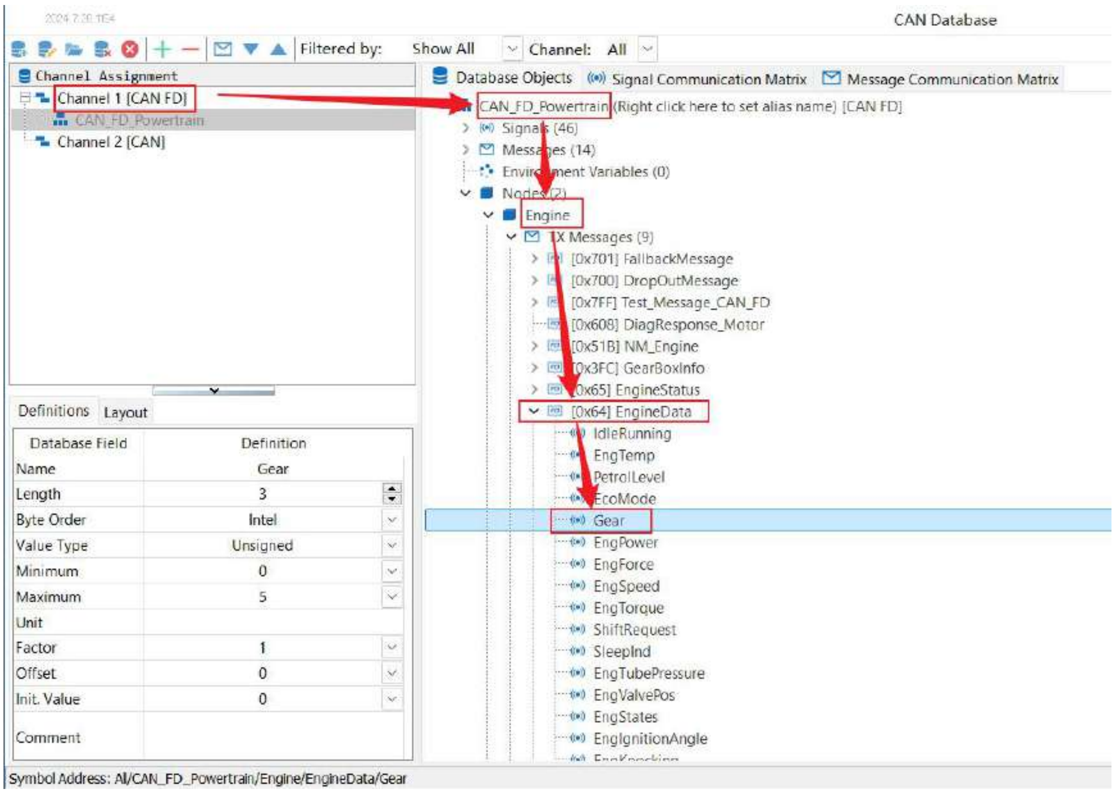
Write CAN signal:

```
double d;
com.can_rbs_set_signal_value_by_address("0/CAN_FD_Powertrain/Engine/EngineData/Gear", &d);
log("Gear = %f", d);
```

The function of these two functions is: to directly read and write CAN signals based on the path of the CAN signal in the database. Taking the Gear signal as an example, the composition of the CAN signal string is parsed as follows:

Channel (0)	Database Name (CAN_FD_Powertrain)	ECU Node (Engine)	Message (EngineData)	Signal (Gear)

Corresponding to the database structure, it is shown as follows:



CAN Database

Filtered by: Show All Channel: All

Channel Assignment
- Channel 1 [CAN FD]
  - CAN_FD_Powertrain
- Channel 2 [CAN]

Database Objects | Signal Communication Matrix | Message Communication Matrix

CAN_FD_Powertrain (Right click here to set alias name) [CAN FD]
- Signals (46)
- Messages (14)
- Environment Variables (0)
- Nodes (2)
  - Engine
    - TX Messages (9)
      - [0x701] FallbackMessage
      - [0x700] DropOutMessage
      - [0x7FF] Test_Message_CAN_FD
      - [0x608] DiagResponse_Motor
      - [0x51B] NM_Engine
      - [0x3FC] GearBoxInfo
      - [0x65] EngineStatus
      - [0x64] EngineData
        - IdleRunning
        - EngTemp
        - PetrolLevel
        - EcoMode
        - Gear
        - EngPower
        - EngForce
        - EngSpeed
        - EngTorque
        - ShiftRequest
        - SleepInd
        - EngTubePressure
        - EngValvePos
        - EngStates
        - EngIgnitionAngle

Definitions | Layout

Database Field	Definition
Name	Gear
Length	3
Byte Order	Intel
Value Type	Unsigned
Minimum	0
Maximum	5
Unit	
Factor	1
Offset	0
Init. Value	0
Comment	

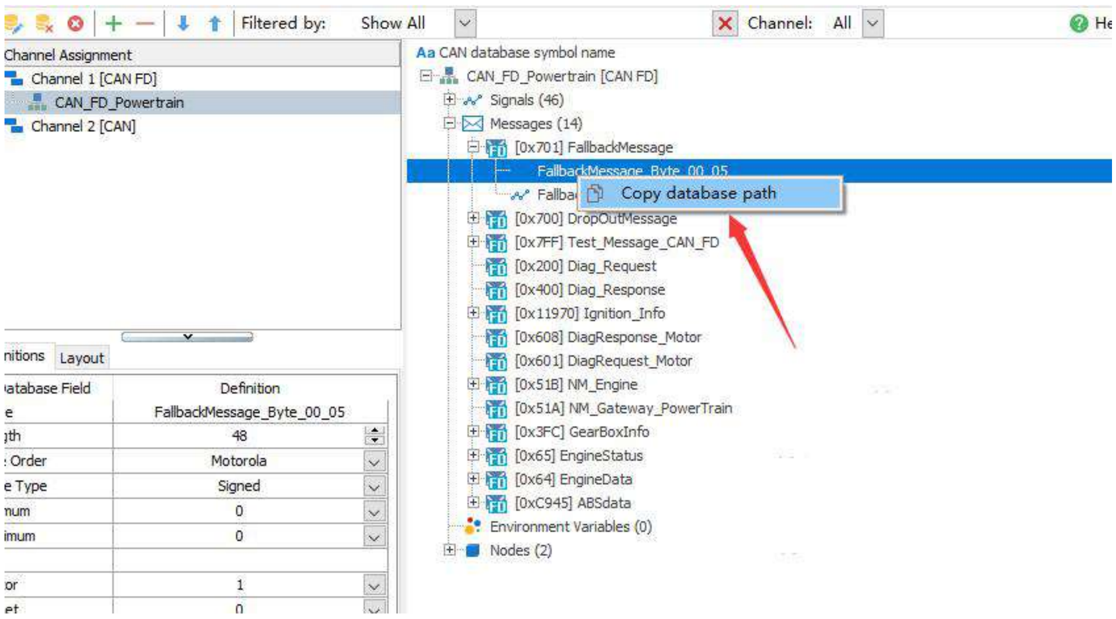
Symbol Address: AI/CAN_FD_Powertrain/Engine/EngineData/Gear

Regarding the channel number, its corresponding relationship is as follows:

```
#define CH1 0
#define CH2 1
#define CH3 2
#define CH4 3
#define CH5 4
#define CH6 5
#define CH7 6
#define CH8 7
#define CH9 8
#define CH10 9
#define CH11 10
#define CH12 11
#define CH13 12
#define CH14 13
#define CH15 14
#define CH16 15
```

That is to say, in the application, CHANNEL1 corresponds to the number 0, CHANNEL2 corresponds to the number 1, and so on. This is because for program development, numbering usually starts from 0 (such as arrays and other variables); whereas in real-life descriptions, it generally starts from 1 (such as Channel 1, etc.).

TSMaster provides a graphical interface for users to directly extract the signal path. As shown in the figure below:



atabase Field	Definition
e	FallbackMessage_Byte_00_05
th	48
Order	Motorola
e Type	Signed
num	0
imum	0
or	1
et	0

In the database interface, select a signal and right-click to choose "Copy database path", the path string for that signal will be copied to the clipboard.

Conclusion:

As shown above, with this method, reading and writing CAN signals is very simple. There is no need for operations such as pre-declaring signal variables. Signals can be directly read from and written to simulated variables based on the RBS system.

1.15.4. Supported Data Types

TSMaster's built-in C script supports common data types and custom data types, which mainly include the following data types:

1.15.4.1. Common Data Types

Symbol	Name	Range	Description
s8	signed 8-bit integer		
u8	unsigned 8-bit integer		
s16	signed 16-bit integer		
u16	unsigned 16-bit integer		

s32	signed 32-bit integer		
u32	unsigned 32-bit integer		
s64	signed 64-bit integer		
u64	unsigned 64-bit integer		
bool	Boolean		
float	single-precision floating-point number		
double	double-precision floating-point number		
Pointer			
pvoid	void pointer		
ps8	signed 8-bit integer pointer		
pu8	unsigned 8-bit integer pointer		
ps16	signed 16-bit integer pointer		
pu16	unsigned 16-bit integer pointer		
ps32	signed 32-bit integer pointer		
s32*	signed 32-bit integer pointer		
pu32	unsigned 32-bit integer pointer		
u32*	unsigned 32-bit integer pointer		
ps64	signed 64-bit integer pointer		
s64*	signed 64-bit integer pointer		
pu64	unsigned 64-bit integer pointer		
u64*	unsigned 64-bit integer pointer		
pbool	boolean pointer		
bool*	boolean pointer		
int*	integer pointer		
uint*	unsigned integer pointer		
pfloat	single-precision floating-point pointer		
float*	single-precision floating-point pointer		
pdouble	double-precision floating-point pointer		
double*	double-precision floating-point pointer		
char*	character pointer		
pchar	character pointer		
Pointer to a pointer			
ppvoid	pointer to a void pointer		
ppchar	pointer to a character pointer		
char**	pointer to a character pointer		
ppdouble	pointer to a double-precision floating-point pointer		
double**	pointer to a double-precision floating-point number pointer		
float**	pointer to a single-precision floating-point number pointer		
pps32	pointer to a signed 32-bit integer pointer		

1.15.4.2. Internally Defined Variable Types

PCANSignal	CAN signal pointer		
PCAN	CAN message pointer		
PCANFD	CANFD message pointer		
PLIN	LIN message pointer		
TSystemVar	system variable types		
PLIBTSMapping	hardware-mapped variable pointer		
TCANFDControllerType	FD controller type		
TCANFDControllerMode	FD controller mode		
TOnIoIpData	IP data type		
PLIBSystemVarDef	system variable pointer		
Prealtime_comment_t	real-time comment variable pointer		
TLogLevel	level for print comment		
TCheckResultCallback	callback function for test result		

1.15.4.3. User-defined Data Types

If the user has self-defined data types, such as structures, structure pointers, etc., the TSMaster script does not support these by default. However, there is a workaround. This can be achieved by using a void pointer to pass the data address, thereby achieving the effect of data transmission. As shown below:

1.15.5. Database Signal Operations

Advantages and disadvantages:

Advantages:

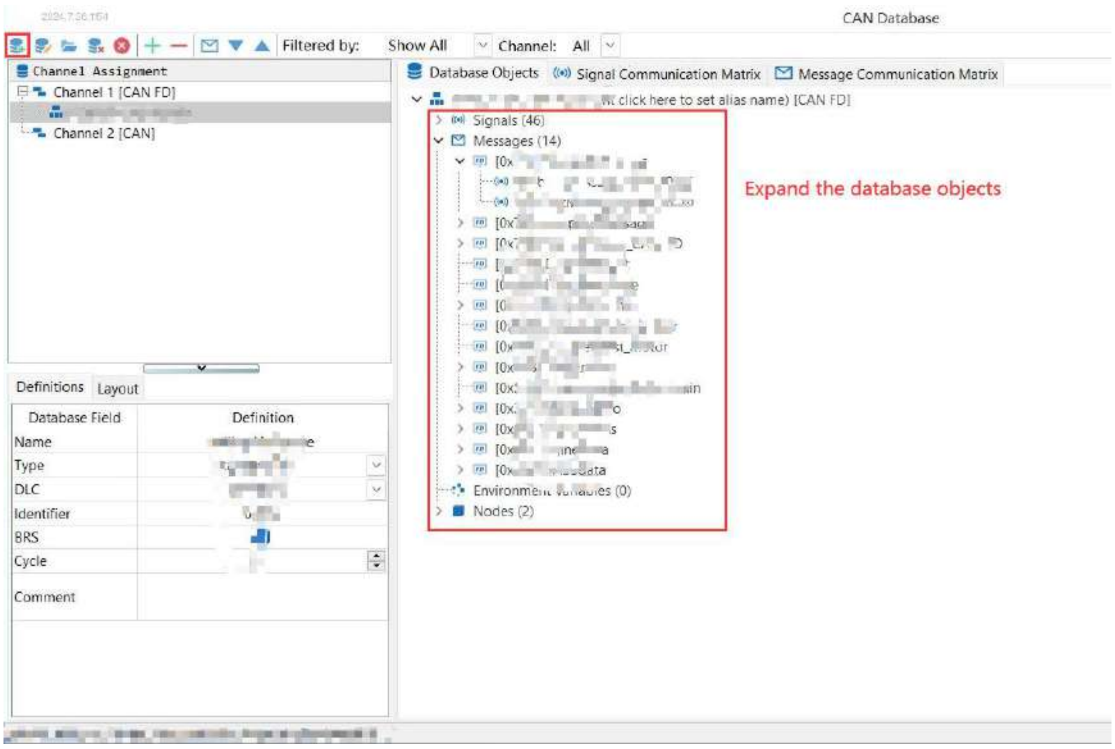
This method essentially provides a mechanism for parsing and injecting DBC information, without being completely bound to the database, making it flexible to use. For example, through the Set_Data function, any message can be injected into a signal, then the signal function calculates the corresponding physical value without having to worry whether the message ID corresponds to the signal's message ID. Therefore, it is used for occasions that require flexible signal parsing, even if the message defined in the database does not match the received message ID, this method can still be used for signal parsing.

Disadvantages:

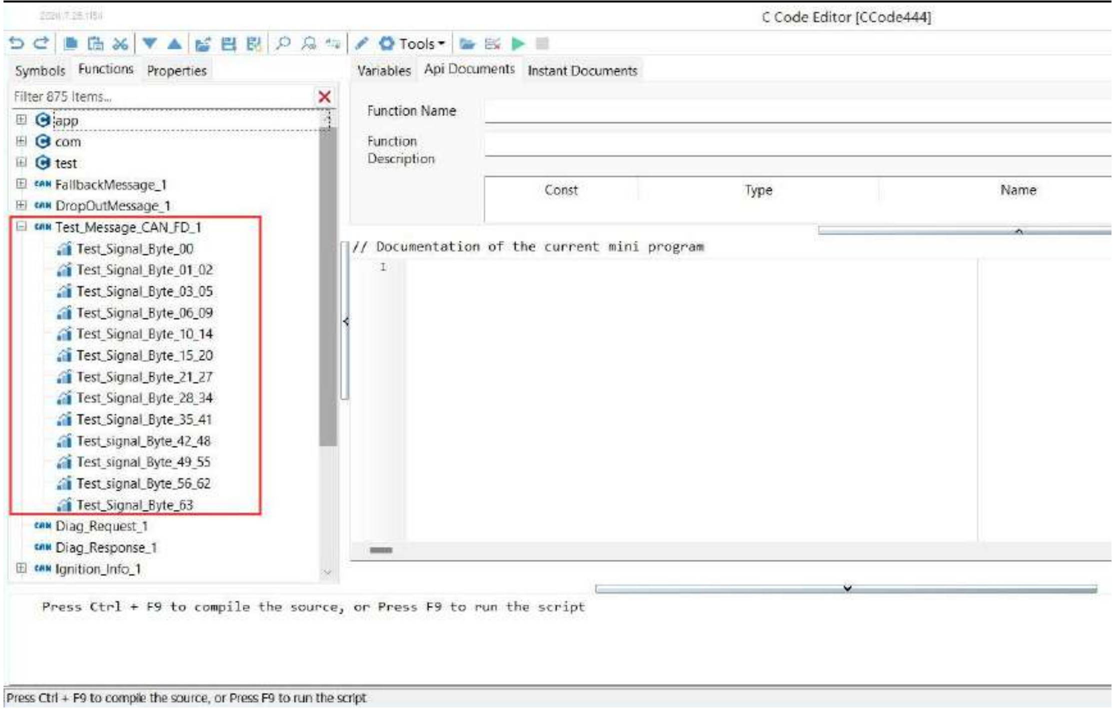
Flexible code may have a relatively more complex process of use, with detailed steps found in step 1 to 5 of this section. Therefore, unless there are special requirements such as testing, this method of using signals is generally not recommended.

The following steps introduce how to operate database variables in the C script of TSMaster.

➢ First step: add the database



After the database is loaded, user can see all the database messages and signals in the quick suggestion window of the script editor, as shown below:



```
// Documentation of the current mini program
```

```
Press Ctrl + F9 to compile the source, or Press F9 to run the script
```

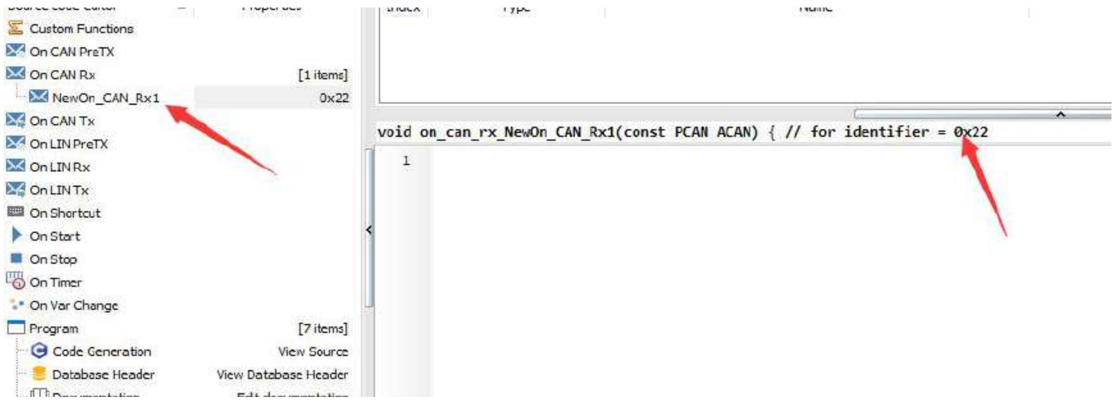
Press Ctrl + F9 to compile the source, or Press F9 to run the script

Explanation: the purpose of adding a database is to dynamically generate C language data type definitions related to the messages and signals in the database, so users do not have to construct the numerous data structures for various CAN messages and signals themselves.

➢ Second step: add message variables

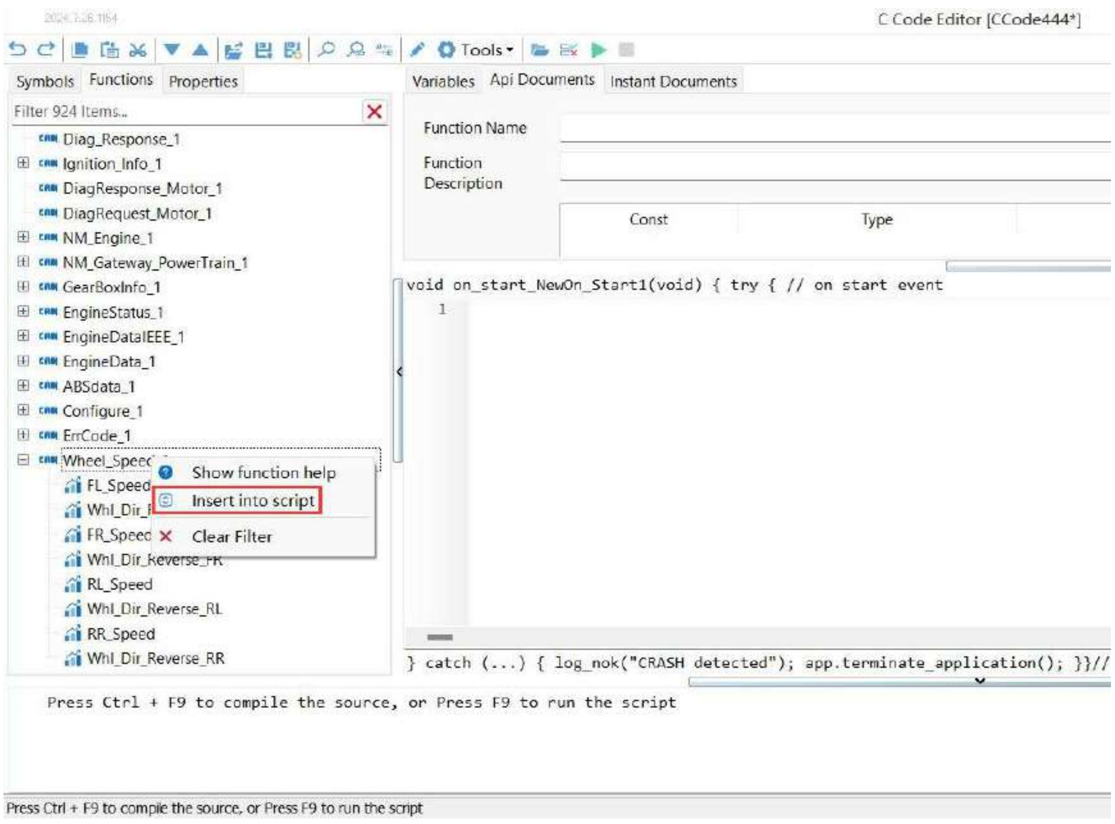
Local Message Variable:

As the name implies, local message variables are used only within the current function. Their scope is limited to the current event function, and the value is refreshed each time the event is triggered. For example, in the CANRX reception event with ID=0x22, a local variable is created, as shown below:



```
void on_can_rx_NewOn_CAN_Rx1(const PCAN ACAN) { // for identifier = 0x22
```

Add message variables through the graphical interface or enter them manually:

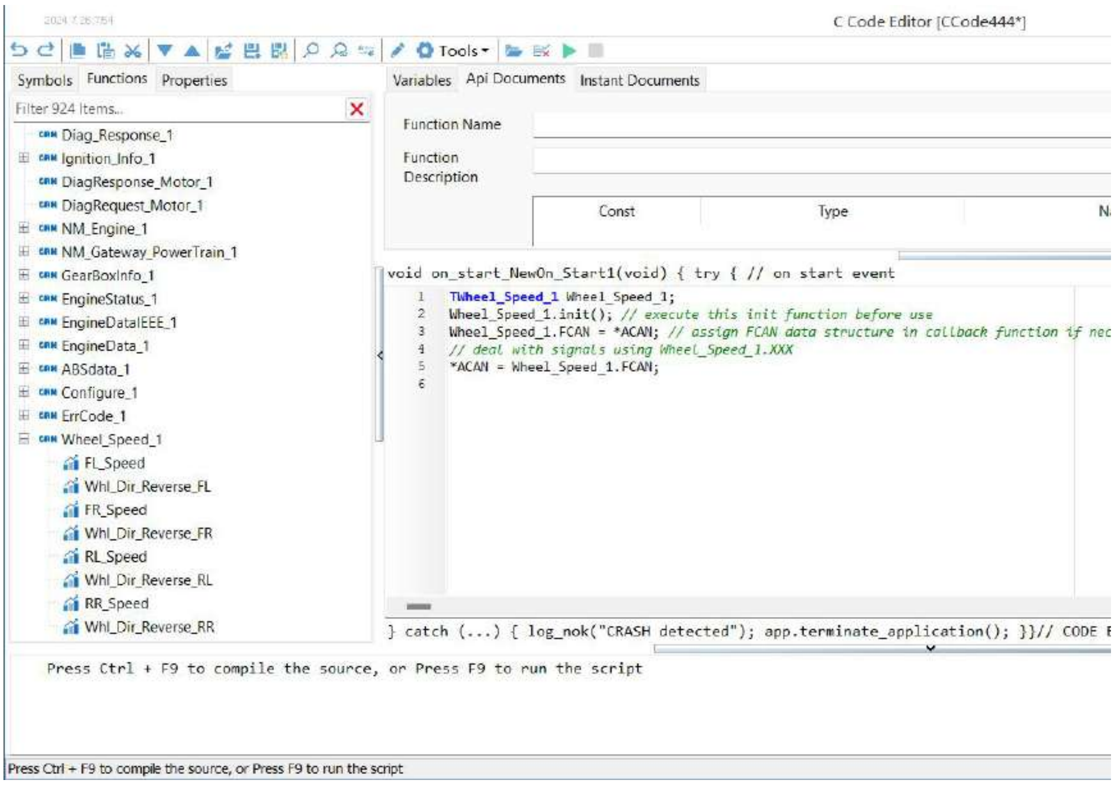


```
void on_start_NewOn_Start1(void) { try { // on start event
} catch (...) { log_nok("CRASH detected"); app.terminate_application(); }}//
```

```
Press Ctrl + F9 to compile the source, or Press F9 to run the script
```

Press Ctrl + F9 to compile the source, or Press F9 to run the script

After adding the variable, it is shown as follows:



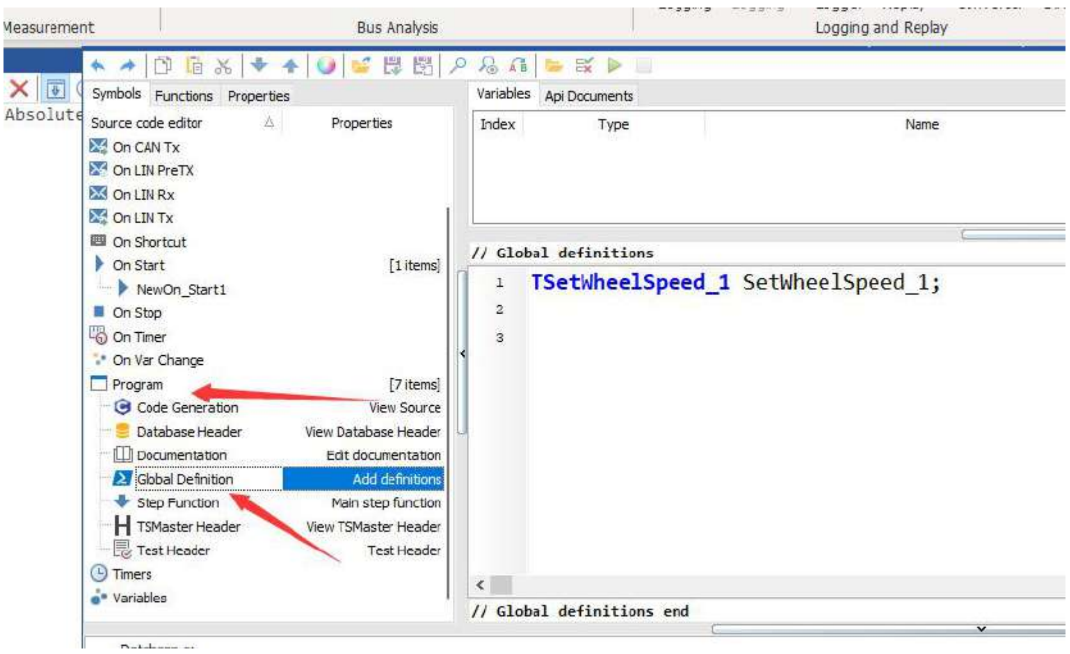
```
void on_start_NewOn_Start1(void) { try { // on start event
TWheel_Speed_1 Wheel_Speed_1;
Wheel_Speed_1.init(); // execute this init function before use
Wheel_Speed_1.FCAN = *ACAN; // assign FCAN data structure in callback function if nec
// deal with signals using Wheel_Speed_1.XXX
*ACAN = Wheel_Speed_1.FCAN;

} catch (...) { log_nok("CRASH detected"); app.terminate_application(); }}// CODE E
```

```
Press Ctrl + F9 to compile the source, or Press F9 to run the script
```

Press Ctrl + F9 to compile the source, or Press F9 to run the script

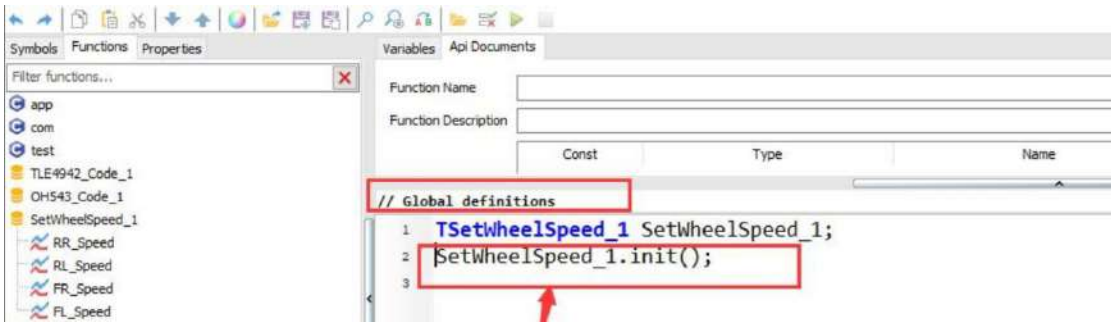
Global Message Variable

As the name implies, global messages are data that can be accessed throughout the entire program, and the program will continuously maintain the memory of this variable during its use. The location to add them is under the: Program -> Global Definition.



```
// Global definitions
TSetWheelSpeed_1 SetWheelSpeed_1;


// Global definitions end
```

The method of adding it is exactly the same as for local variables, but there is something to note: the initialization function of the global variable, such as SetWheelSpeed_1.init(), should not be directly placed in the variable definition file. It needs to be cut and pasted to an event like OnStart to ensure that the variable is initialized before it is used. In the newer version of the script editor, this will be prompted, and if it is not cut and pasted, it will trigger a duplicate definition error message.

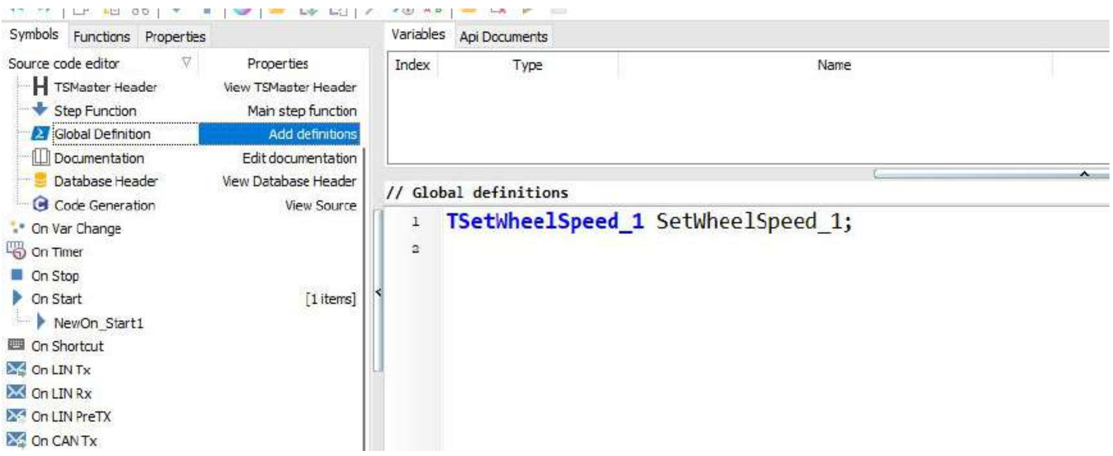


```
// Global definitions
TSetWheelSpeed_1 SetWheelSpeed_1;
SetWheelSpeed_1.init();
```

For example, the second line must be cut out, ensuring that the initialization of the global variable is executed before using it. For example, it can be placed in the OnStart event, where the global variable declared is initialized when the program starts.

After cutting the initialization code, it is as follows:

Declared global variable:

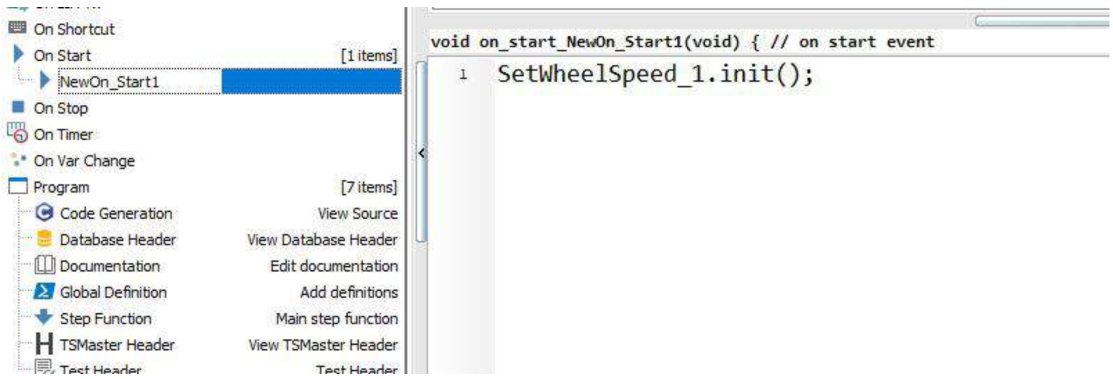


Symbols | Functions | Properties

Variables | Api Documents

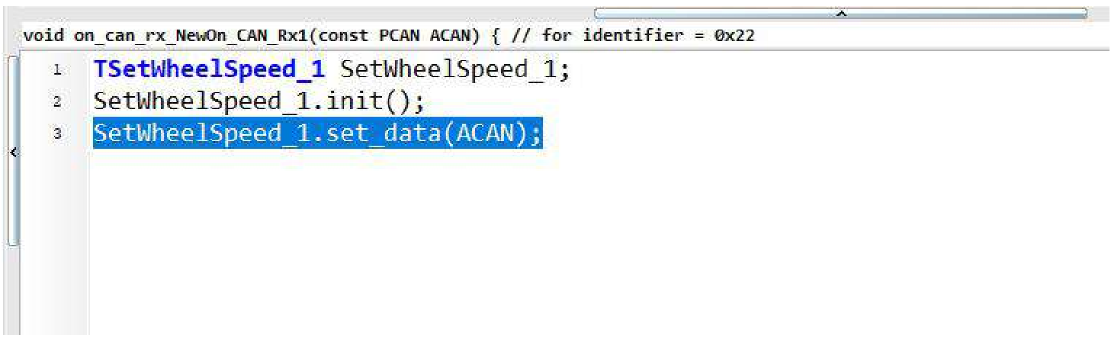
```
// Global definitions
TSetWheelSpeed_1 SetWheelSpeed_1;
```

Initialize global variables:



```
void on_start_NewOn_Start1(void) { // on start event
SetWheelSpeed_1.init();
```

➢ Third step: read CAN raw data into message variables

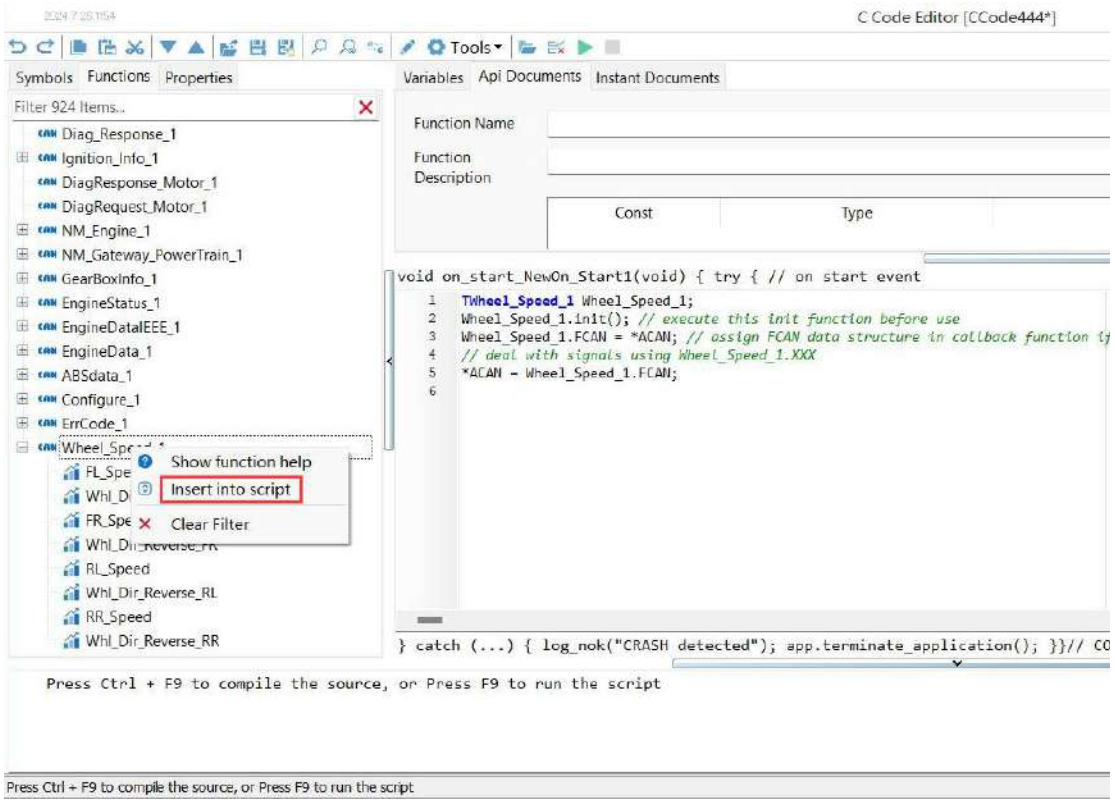


```
void on_can_rx_NewOn_CAN_Rx1(const PCAN ACAN) { // for identifier = 0x22
TSetWheelSpeed_1 SetWheelSpeed_1;
SetWheelSpeed_1.init();
SetWheelSpeed_1.set_data(ACAN);
```

As shown above: this line means that the original message of type TCAN (PCAN is a pointer type to TCAN, defined as follows: typedef TCAN* PCAN;) is filled into the database message (the main purpose is to use the message data and signal data types defined in the database. Of course, if the user does not need the database and wants to operate directly on the TCAN data structure, that is also possible).

➢ Fourth step: Operating Data Based on Signal

Select a signal, right click and select “Insert into script”.

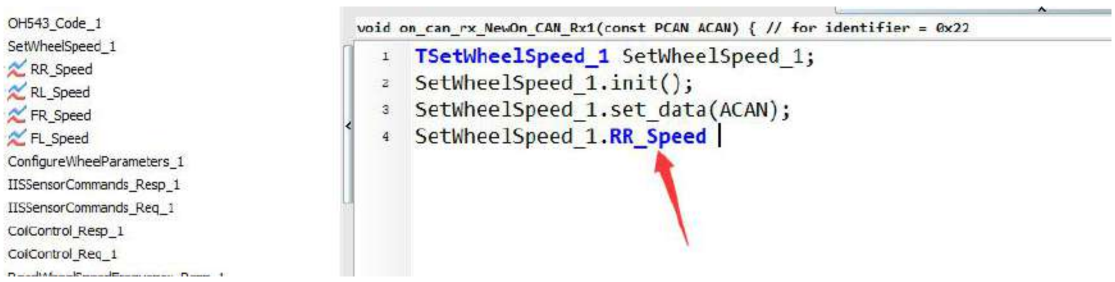


```
void on_start_NewOn_Start1(void) { try { // on start event
TWheel_Speed_1 Wheel_Speed_1;
Wheel_Speed_1.init(); // execute this init function before use
Wheel_Speed_1.FCAN = *ACAN; // assign FCAN data structure in callback function if
// deal with signals using Wheel_Speed_1.XXX
*ACAN = Wheel_Speed_1.FCAN;

} catch (...) { log_nok("CRASH detected"); app.terminate_application(); }}// CO
```

```
Press Ctrl + F9 to compile the source, or Press F9 to run the script
```

Press Ctrl + F9 to compile the source, or Press F9 to run the script

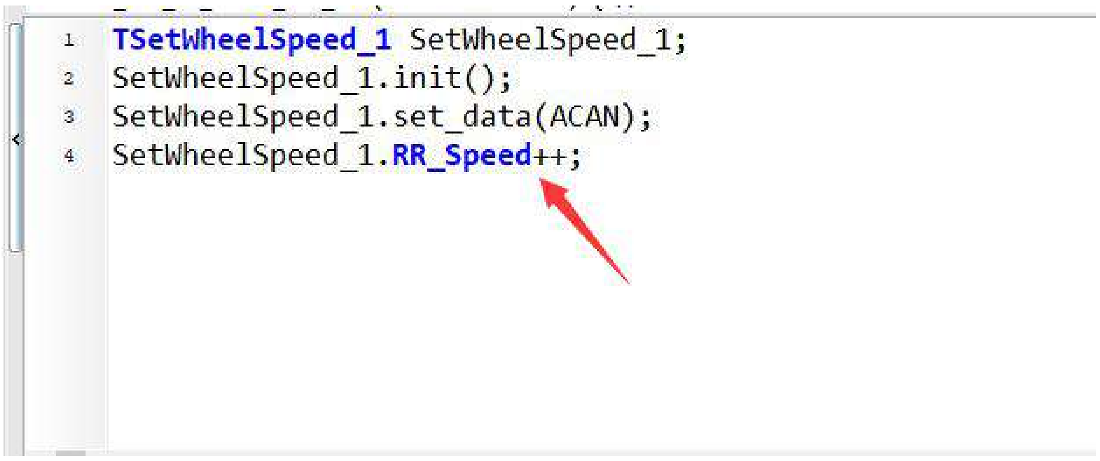
Enter the prompt bar, select the signal, and right-click to add the signal, which will automatically add the signal variable to the code. As shown below:



```
void on_can_rx_NewOn_CAN_Rx1(const PCAN ACAN) { // for identifier = 0x22
TSetWheelSpeed_1 SetWheelSpeed_1;
SetWheelSpeed_1.init();
SetWheelSpeed_1.set_data(ACAN);
SetWheelSpeed_1.RR_Speed
```

The graphical interface is designed to facilitate users in completing code editing. If users are familiar with the definitions of message and signal data structures, they can directly enter them manually.

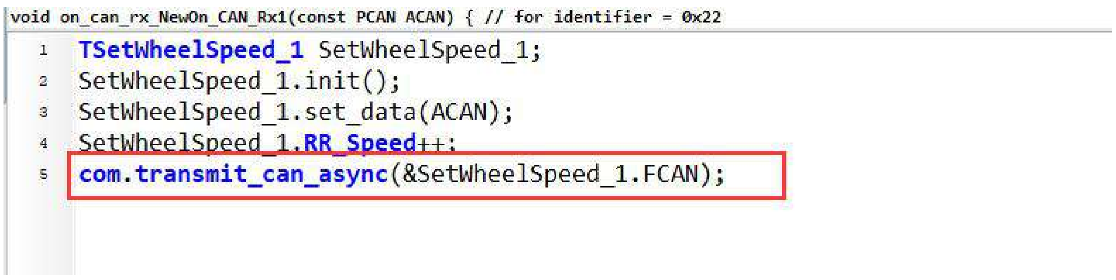
Based on this signal, users can perform the calculations they want, as follows:



```
TSetWheelSpeed_1 SetWheelSpeed_1;
SetWheelSpeed_1.init();
SetWheelSpeed_1.set_data(ACAN);
SetWheelSpeed_1.RR_Speed++;
```

➢ Fifth step: send message

After editing the message, send it back to the bus. Call the transmit function to transmit the FCAN data within the message. As shown below:



```
void on_can_rx_NewOn_CAN_Rx1(const PCAN ACAN) { // for identifier = 0x22
TSetWheelSpeed_1 SetWheelSpeed_1;
SetWheelSpeed_1.init();
SetWheelSpeed_1.set_data(ACAN);
SetWheelSpeed_1.RR_Speed++;
com.transmit_can_async(&SetWheelSpeed_1.FCAN);
```

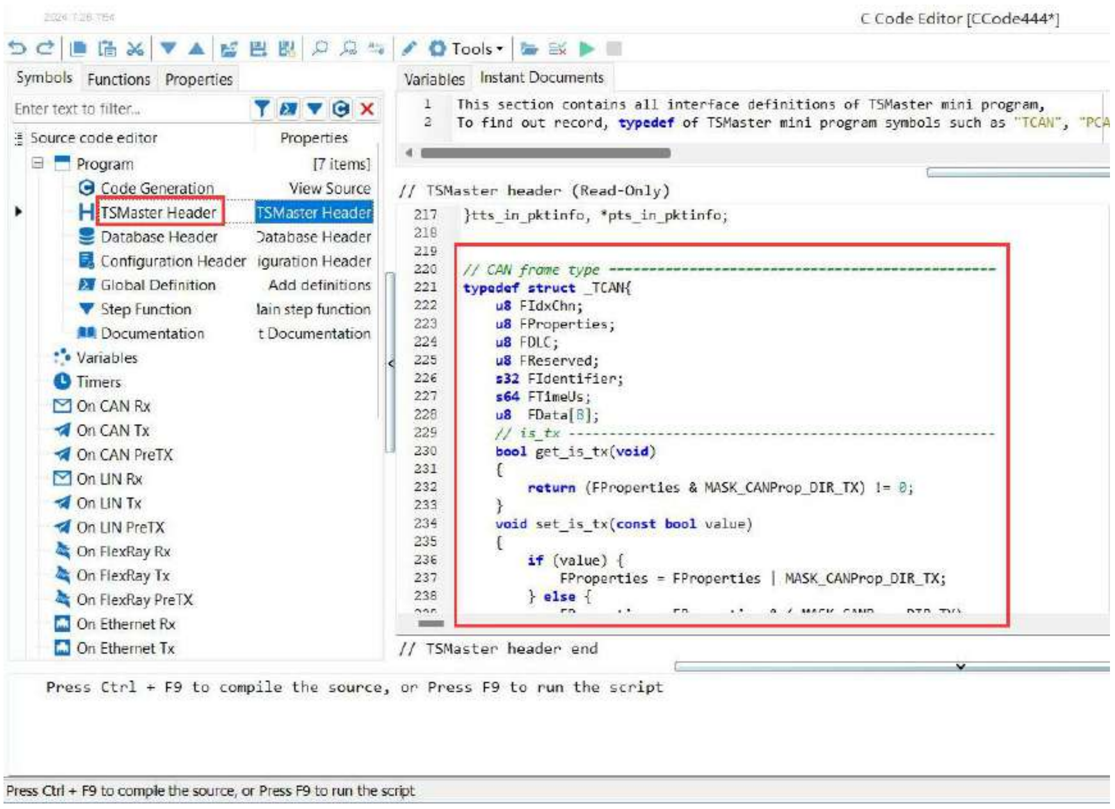
Similarly, the transmit_can_async function can also be added through the help suggestion bar.

If the user is not familiar with writing the send message function, they can refer to the sections 1.14.6.4 Add Message Send Function Through The Send Window and 1.14.6.5 Add Message Send Function Through The Trace Window. TSMaster provides a code generation tool that can automatically generates the corresponding send messages.

1.15.6. C Script Operation Skills

1.15.6.1. View the Definition of System Variable Types (such as CAN, LIN Messages)

The definitions of all built-in system variables in TSMaster C script, such as CAN and LIN messages, are placed in the system file TSMaster Header, as shown below:



```
This section contains all interface definitions of TSMaster mini program,
To find out record, typedef of TSMaster mini program symbols such as "TCAN", "PCA
```

```
// TSMaster header (Read-Only)
}tts_in_pktinfo, *pts_in_pktinfo;


// CAN frame type -----------------------------------------------------
typedef struct _TCAN{
    u8 FIdxChn;
    u8 FProperties;
    u8 FDLC;
    u8 FReserved;
    s32 FIdentifier;
    s64 FTimeUs;
    u8  FData[8];
    // is_tx ----------------------------------------------------------
    bool get_is_tx(void)
    {
        return (FProperties & MASK_CANProp_DIR_TX) != 0;
    }
    void set_is_tx(const bool value)
    {
        if (value) {
            FProperties = FProperties | MASK_CANProp_DIR_TX;
        } else {
// TSMaster header end
```

```
Press Ctrl + F9 to compile the source, or Press F9 to run the script
```

Press Ctrl + F9 to compile the source, or Press F9 to run the script

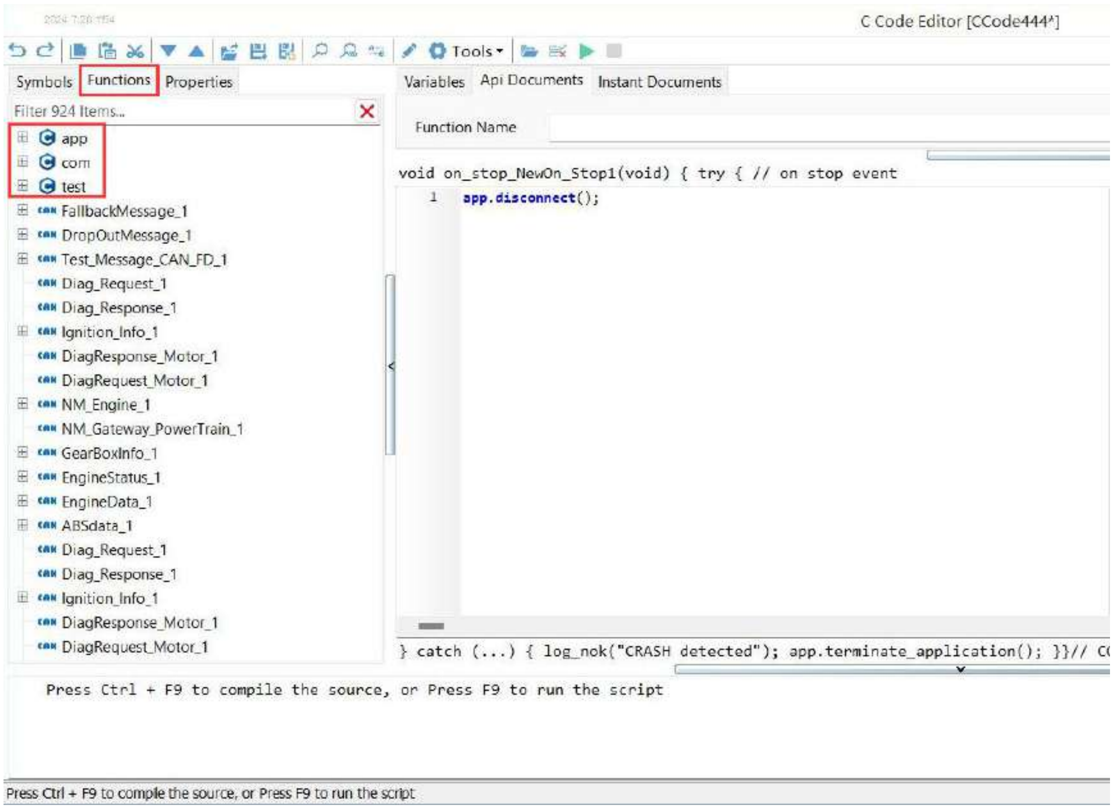
Click “TSMaster Header” to check the CAN frame definition.

For developers who are accustomed to directly processing raw message data, they need to refer to this file to check the data type definitions.

1.15.6.2. Add System Functions

Developers may encounter situations where they do not know how to call system functions. The C script editor of TSMaster provides a help bar to assist users in understanding and adding system functions. The process is as follows:

1. Enter the 'Functions' tab.



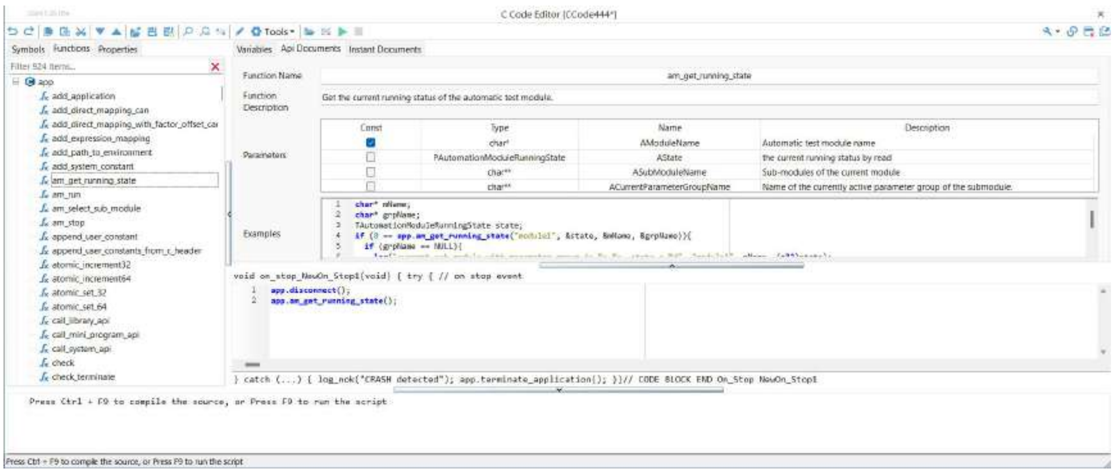
```
void on_stop_NewOn_Stop1(void) { try { // on stop event
app.disconnect();
} catch (...) { log_nok("CRASH detected"); app.terminate_application(); }}// CC
```

```
Press Ctrl + F9 to compile the source, or Press F9 to run the script
```

Press Ctrl + F9 to compile the source, or Press F9 to run the script

app: functions related to script application.
com: functions related to network communication.
test: functions related to testing.

2. Take adding a CANFD configuration parameter as an example to explain how to add system functions, as shown below:



Function Name: am_get_running_state

Function Description: Get the current running status of the automatic test module.

Const	Type	Name	Description
✓	char*	AModuleName	Automatic test module name
☐	PAutomationModuleRunningState	AState	the current running status by read
☐	char**	ASubModuleName	Sub-modules of the current module
☐	char**	ACurrentParameterGroupName	Name of the currently active parameter group of the submodule.

```
char* mName;
char* grpName;
TAutomationModuleRunningState state;
if (0 == app.am_get_running_state("module1", &state, &mName, &grpName)){
  if (grpName == NULL){
```

```
void on_stop_NewOn_Stop1(void) { try { // on stop event
app.disconnect();
app.am_get_running_state();
} catch (...) { log_nok("CRASH detected"); app.terminate_application(); }}// CODE BLOCK END On_Stop NewOn_Stop1
```

```
Press Ctrl + F9 to compile the source, or Press F9 to run the script
```

Press Ctrl + F9 to compile the source, or Press F9 to run the script

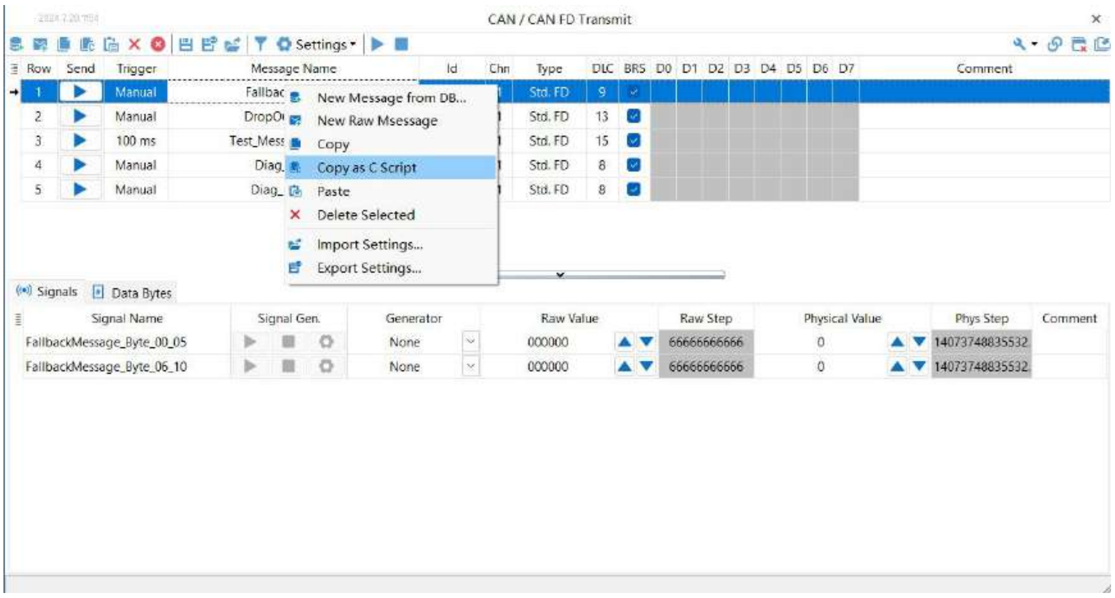
For example, for the am_get_running_state function, select this function and on the top right corner, the parameter description and calling example are given. Double-click the am_get_running_state function to insert it into the script, and user can input the required parameter based on the information on the top right corner.

1.15.6.3. Based on Database Operations

The incorporation of the database allows users to avoid directly dealing with raw CAN messages, making it easier to intuitively modify signal-level data. Detailed operation records can be found in the section: Database Variable Operations.

1.15.6.4. Add Message Send Function through the Send Window

If a user wants to convert the code from the send window into a C script after verification, TSMaster provides the following concise operation method:

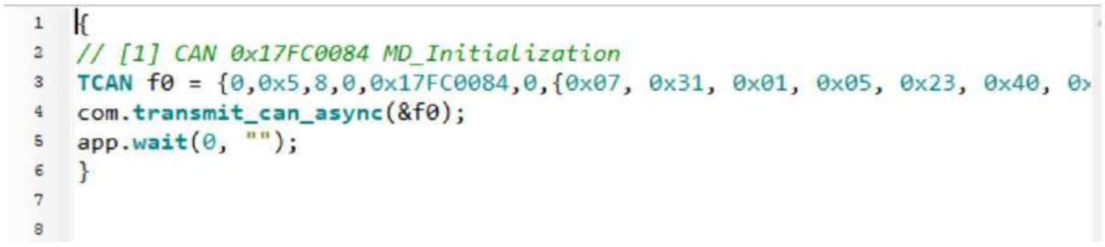


Row	Send	Trigger	Message Name	Type	DLC	BRS
1	▶	Manual	Fallbac	Std. FD	9	✓
2	▶	Manual	DropOi	Std. FD	13	✓
3	▶	100 ms	Test_Mess	Std. FD	15	✓
4	▶	Manual	Diag_	Std. FD	8	✓
5	▶	Manual	Diag_	Std. FD	8	✓

Signal Name	Generator	Raw Value	Raw Step	Physical Value	Phys Step	Comment
FallbackMessage_Byte_00_05	None	000000	66666666666	0	14073748835532	
FallbackMessage_Byte_06_10	None	000000	66666666666	0	14073748835532	

Select the message, right-click and select "Copy as C Script".

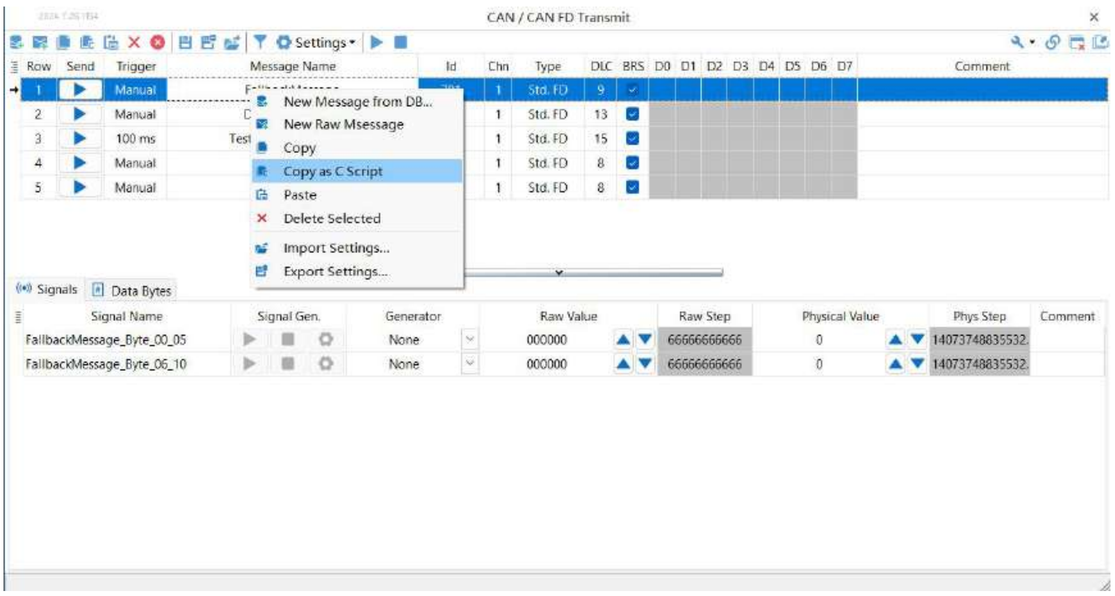
Then, a block for sending the message will be generated, as shown below. Users can simply copy this message block into the C script editor.



```
{
// [1] CAN 0x17FC0084 MD_Initialization
TCAN f0 = {0,0x5,8,0,0x17FC0084,0,{0x07, 0x31, 0x01, 0x05, 0x23, 0x40, 0x
com.transmit_can_async(&f0);
app.wait(0, "");
}
```

1.15.6.5. Add Message Send Function through the Trace Window

When the user is not familiar with how to add a send message function, the user can quickly write a C script through the following mechanism. As shown in the figure below:

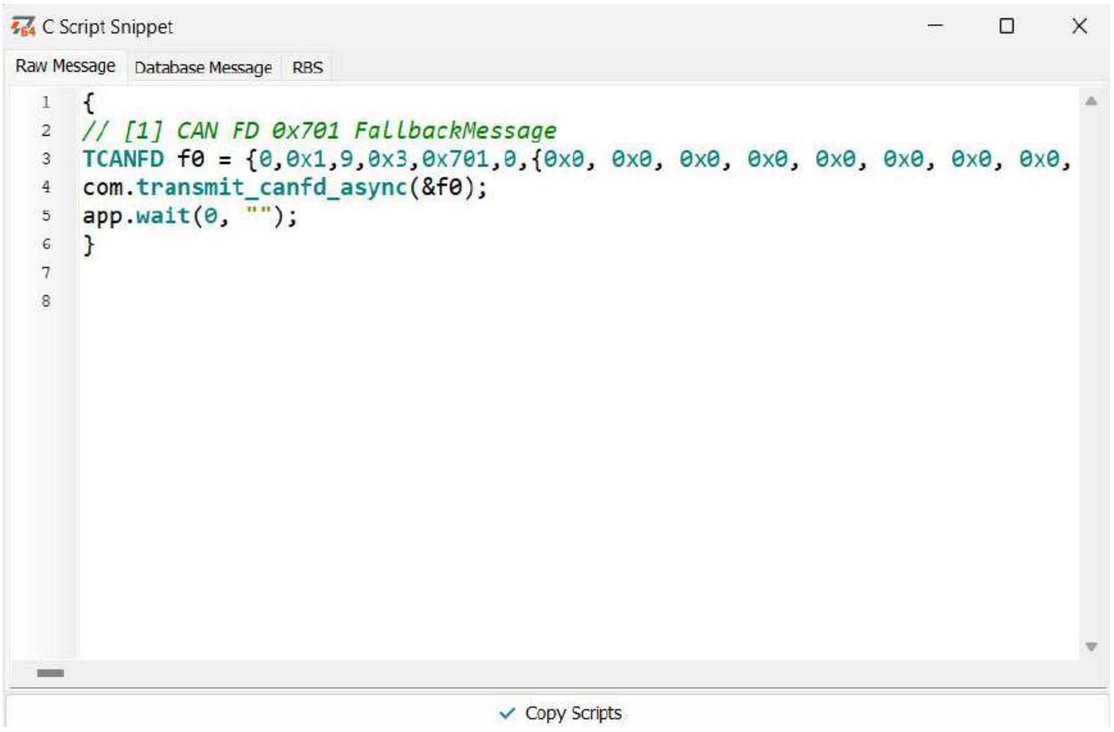


Row	Send	Trigger	Message Name	Id	Chn	Type	DLC	BRS	D0	D1	D2	D3	D4	D5	D6	D7	Comment
1	▶	Manual	F		1	Std. FD	9	✓									
2	▶	Manual	C		1	Std. FD	13	✓									
3	▶	100 ms	Test		1	Std. FD	15	✓									
4	▶	Manual			1	Std. FD	8	✓									
5	▶	Manual			1	Std. FD	8	✓									

Signal Name	Signal Gen.	Generator	Raw Value	Raw Step	Physical Value	Phys Step	Comment
FallbackMessage_Byte_00_05		None	000000	66666666666	0	14073748835532.	
FallbackMessage_Byte_06_10		None	000000	66666666666	0	14073748835532.	

Select the message, right-click and select “Copy as C Script”.

The system automatically generates the C codes for calling the function, as shown below. Users can simply copy the codes into the C script editor.



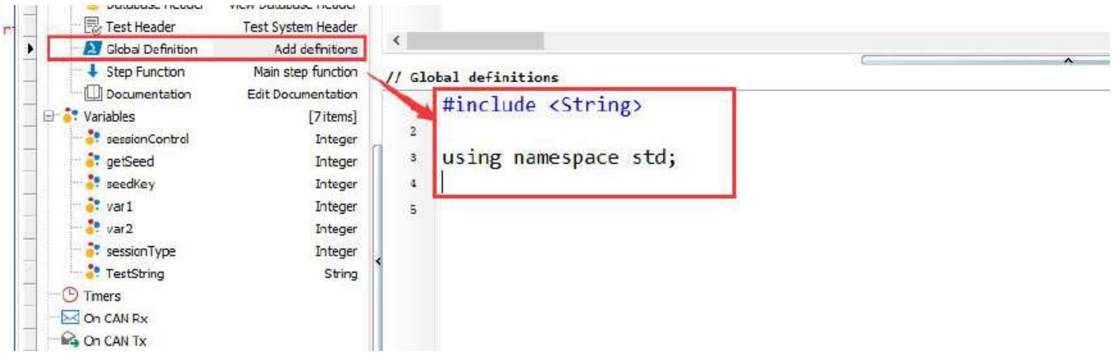
```
{
// [1] CAN FD 0x701 FallbackMessage
TCANFD f0 = {0,0x1,9,0x3,0x701,0,{0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
com.transmit_canfd_async(&f0);
app.wait(0, "");
}
```

1.15.7. Windows Library Files Reference

TSMaster's C script uses standard C/C++ compiler. Therefore, users can reference Windows system resources for any extensions. Here are two examples to show how to use Windows API files:

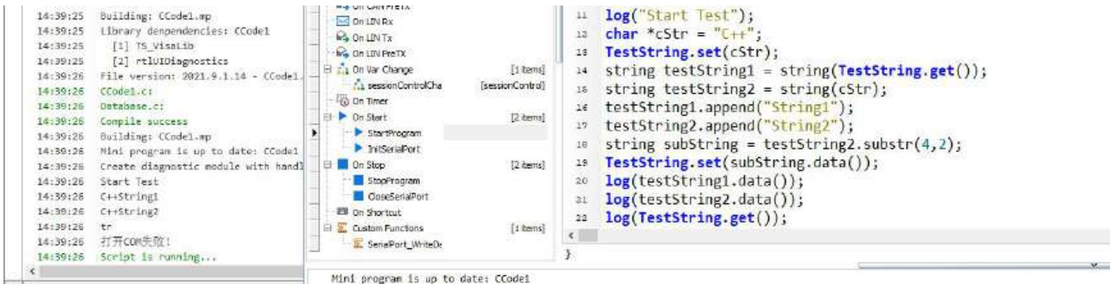
1.15.7.1. Using the String Library File

A. First step: In Global Definitions, add reference to a library file.



```
// Global definitions
#include <String>

using namespace std;
```

B. Second step: At this point, the user can directly use variable types like string in the script, as follows:



```
14:39:25   Building: CCode1.mp
14:39:25   Library denpendencies: CCode1
14:39:25     [1] TS_VisaLib
14:39:25     [2] rtlUIDiagnostics
14:39:26   File version: 2021.9.1.14 - CCode1.
14:39:26   CCode1.c:
14:39:26   Database.c:
14:39:26   Compile success
14:39:26   Building: CCode1.mp
14:39:26   Mini program is up to date: CCode1
14:39:26   Create diagnostic module with handl
14:39:26   Start Test
14:39:26   C++String1
14:39:26   C++String2
14:39:26   tr
14:39:26   打开COM失败!
14:39:26   Script is running...
```

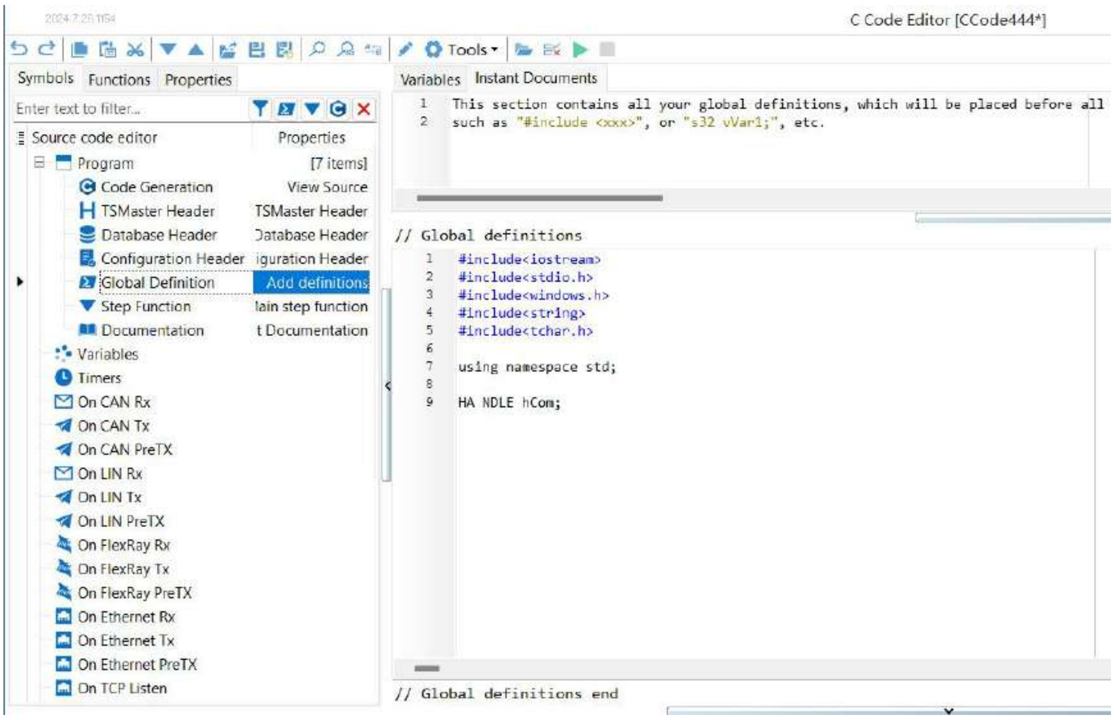
```
log("Start Test");
char *cStr = "C++";
TestString.set(cStr);
string testString1 = string(TestString.get());
string testString2 = string(cStr);
testString1.append("String1");
testString2.append("String2");
string subString = testString2.substr(4,2);
TestString.set(subString.data());
log(testString1.data());
log(testString2.data());
log(TestString.get());
}
```

```
Mini program is up to date: CCode1
```

log("Start Test");
char *cStr = "C++";
TestString.set(cStr);
string testString1 = string(TestString.get());
string testString2 = string(cStr);
testString1.append("String1");
testString2.append("String2");
string subString = testString2.substr(4,2);
TestString.set(subString.data());
log(testString1.data());
log(testString2.data());
log(TestString.get());

1.15.7.2. Using the Computer's Serial Port

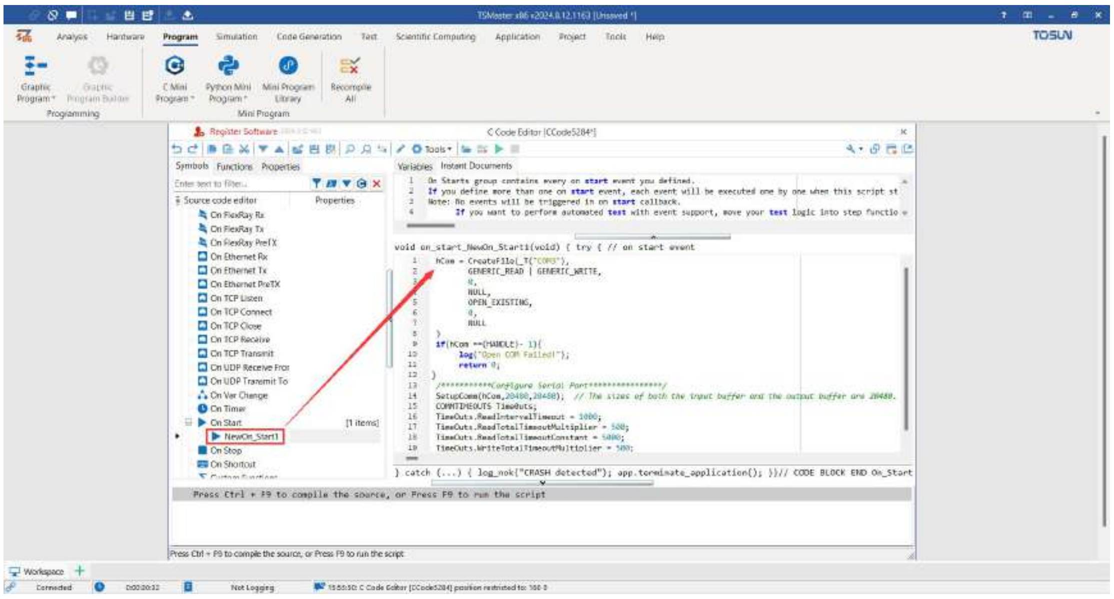
A. In Global Definitions, add reference to a library file.



```
This section contains all your global definitions, which will be placed before all
such as "#include <xxx>", or "s32 vVar1;", etc.
```

```
// Global definitions
#include<iostream>
#include<stdio.h>
#include<windows.h>
#include<string>
#include<tchar.h>

using namespace std;

HA NDLE hCom;
// Global definitions end
```

B. In the startup function, add the initialization code for the serial port.

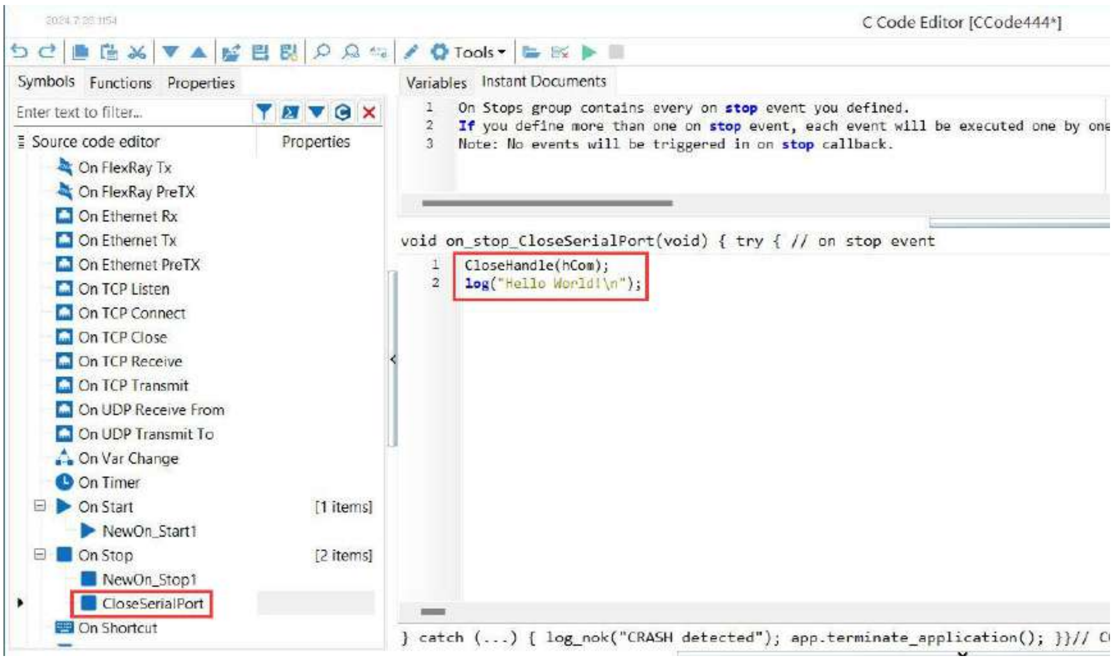


```
On Starts group contains every on start event you defined.
If you define more than one on start event, each event will be executed one by one when this script st
Note: No events will be triggered in on start callback.
      If you want to perform automated test with event support, move your test logic into step functio
```

```
void on_start_NewOn_Start1(void) { try { // on start event
 hCom = CreateFile(_T("COM3"),
        GENERIC_READ | GENERIC_WRITE,
        0,
        NULL,
        OPEN_EXISTING,
        0,
        NULL
 )
 if(hCom ==(HANDLE)- 1){
     log("Open COM Failed!");
     return 0;
}
 /***********Configure Serial Port****************/
 SetupComm(hCom,20480,20480);  // The sizes of both the input buffer and the output buffer are 20480.
 COMMTIMEOUTS TimeOuts;
 TimeOuts.ReadIntervalTimeout = 1000;
 TimeOuts.ReadTotalTimeoutMultiplier = 500;
 TimeOuts.ReadTotalTimeoutConstant = 5000;
 TimeOuts.WriteTotalTimeoutMultiplier = 500;
} catch (...) { log_nok("CRASH detected"); app.terminate_application(); }}// CODE BLOCK END On_Start
```

As shown in the figure above: CreateFile and SetupComm are both native Windows APIs, which can be directly used after including the header files.

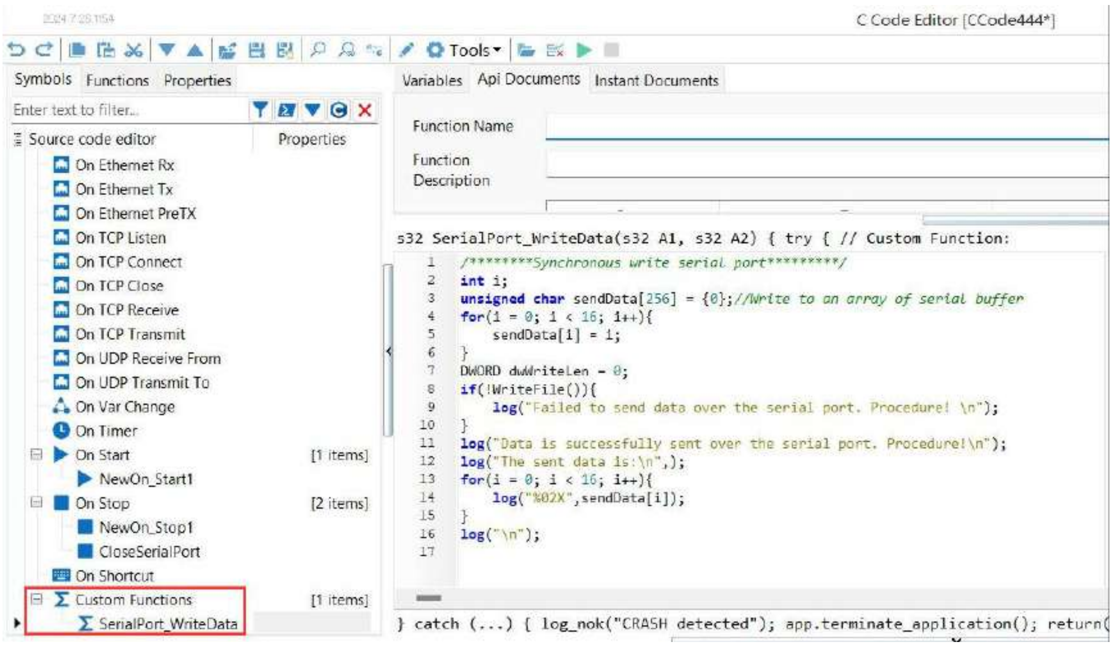
C. In the exit function, add the code to close the serial port.



```
On Stops group contains every on stop event you defined.
If you define more than one on stop event, each event will be executed one by one
Note: No events will be triggered in on stop callback.
```

```
void on_stop_CloseSerialPort(void) { try { // on stop event
 CloseHandle(hCom);
 log("Hello World!\n");
} catch (...) { log_nok("CRASH detected"); app.terminate_application(); }}//
```

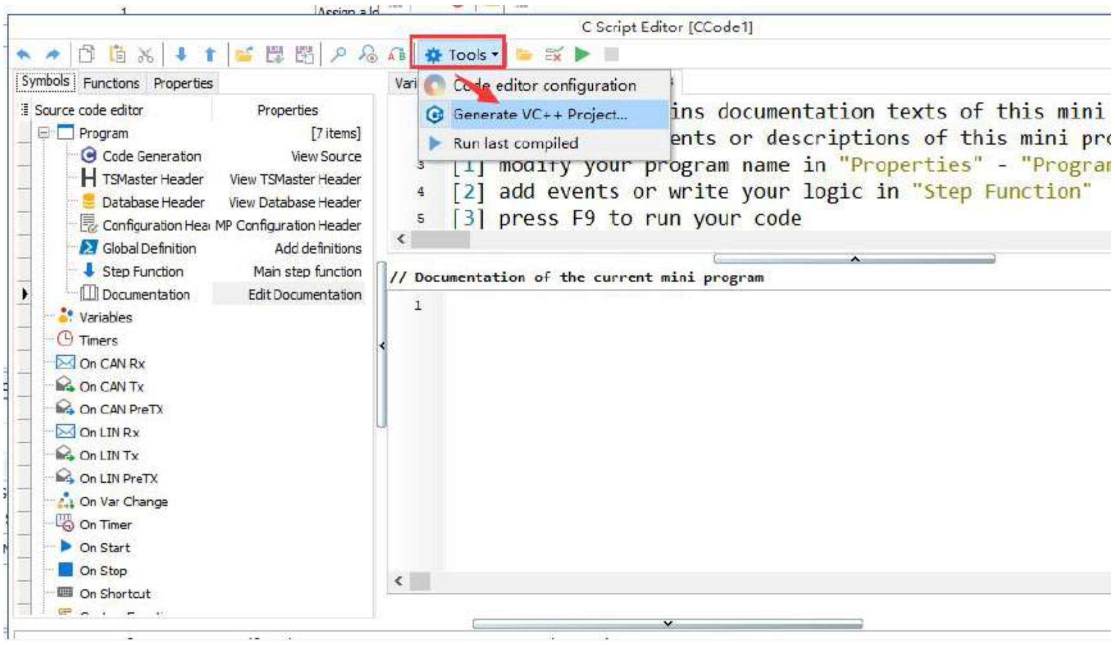
D. Add a function to send data through the serial port.



```
s32 SerialPort_WriteData(s32 A1, s32 A2) { try { // Custom Function:
/********Synchronous write serial port*********/
int i;
unsigned char sendData[256] = {0};//Write to an array of serial buffer
for(i = 0; i < 16; i++){
    sendData[i] = i;
}
DWORD dwWriteLen = 0;
if(!WriteFile()){
    log("Failed to send data over the serial port. Procedure! \n");
}
log("Data is successfully sent over the serial port. Procedure!\n");
log("The sent data is:\n",);
for(i = 0; i < 16; i++){
    log("%02X",sendData[i]);
}
log("\n");

} catch (...) { log_nok("CRASH detected"); app.terminate_application(); return(
```

1.15.8. Export the C Script to a Visual Studio Project for Debugging

1.15.8.1. Generate a Code Debugging Project



```
ins documentation texts of this mini
ents or descriptions of this mini pr
[1] modify your program name in "Properties" - "Progra
[2] add events or write your logic in "Step Function"
[3] press F9 to run your code
```

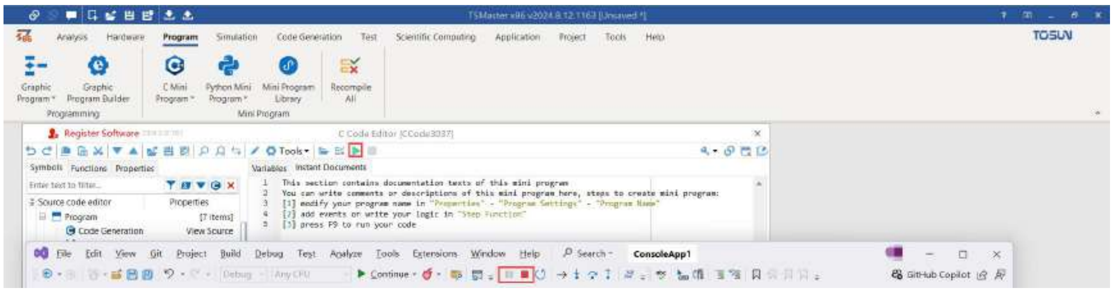
```
// Documentation of the current mini program
```

As shown in the figure above, in the C script panel, select Tools -> Generate VC++ Project, and choose a folder to store the generated project. The generated project has already completed the relevant configuration and association. User can directly run the project to attach it to TSMaster for debugging.

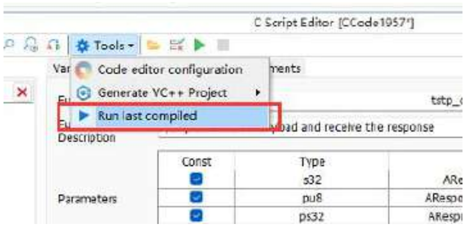
1.15.8.2. Debug the C Script Directly in Visual Studio

After exporting the script to Visual Studio, if a user wants to directly debug the C script codes within Visual Studio to fully utilize Visual Studio's debugging capabilities, they can follow these steps to establish an association between the Visual Studio codes and TSMaster.

- First, compile the code in the Visual Studio project and then click 'Debug' to run. At this point, keep the script in a non-executing state in TSMaster.



- In the C script, select Tools -> Run last compiled. At this point, an association is established between the Visual Studio debugging environment and TSMaster, as shown below:



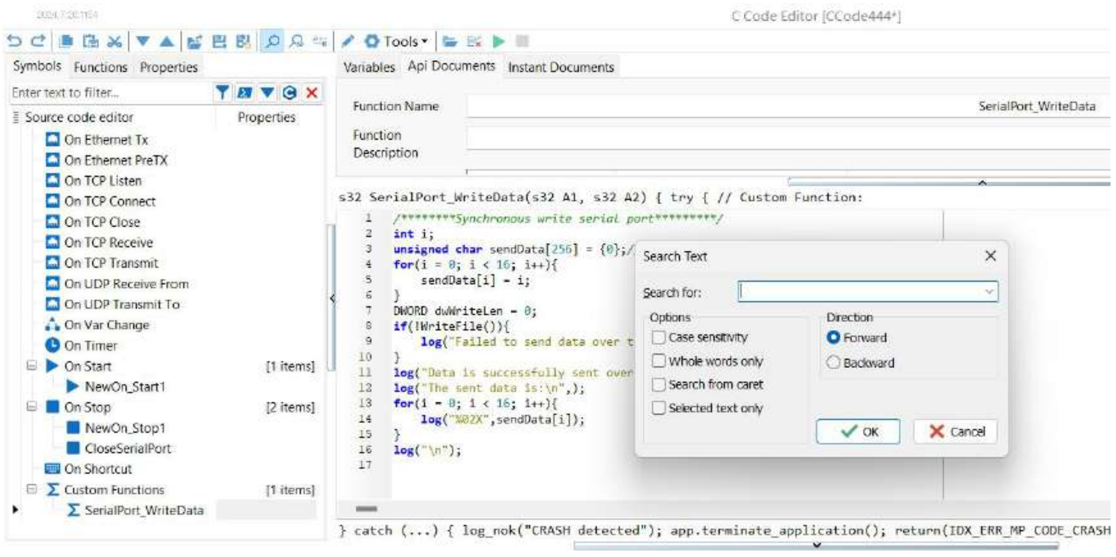
Note: Be sure to follow the steps mentioned above. Otherwise, it will not be possible to establish an association between TSMaster and Visual Studio, which means the debugging features of Visual Studio cannot be utilized.

1.15.9. Code Search/Replace

1.15.9.1. Code Search in the C Script

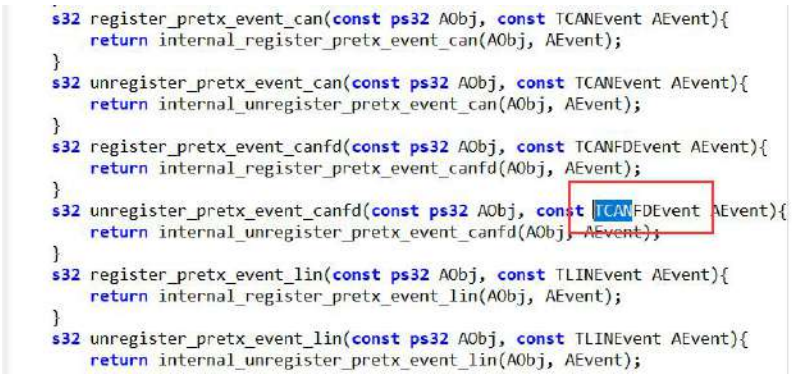
During the process of software development, searching for code is a very common operation. To perform a code search in the mini-program script of TSMaster, the main methods are as follows:

Step one: Summon the search window in the code editor, which can be done in two ways: Ctrl+F, or by clicking on the quick icon on the script toolbar. Both methods have the exact same effect. After execution, the search interface will pop up as shown below:



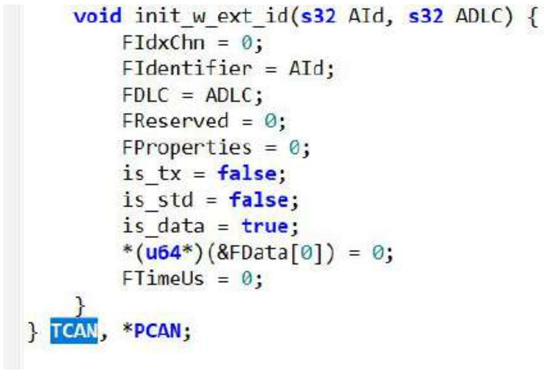
```
s32 SerialPort_WriteData(s32 A1, s32 A2) { try { // Custom Function:
/********Synchronous write serial port*********/
int i;
unsigned char sendData[256] = {0};/
for(i = 0; i < 16; i++){
    sendData[i] = i;
}
DWORD dwWriteLen = 0;
if(!WriteFile()){
    log("Failed to send data over t
}
log("Data is successfully sent over
log("The sent data is:\n",);
for(i = 0; i < 16; i++){
    log("%02X",sendData[i]);
}
log("\n");

} catch (...) { log_nok("CRASH detected"); app.terminate_application(); return(IDX_ERR_MP_CODE_CRASH
```

Step two: Enter the content you want to search for in the search box, for example, if you are searching for the CAN data type definition TCAN, and then click "OK", the first piece of data that meets the criteria will be found, as shown below:



```
s32 register_pretx_event_can(const ps32 AObj, const TCANEvent AEvent){
    return internal_register_pretx_event_can(AObj, AEvent);
}
s32 unregister_pretx_event_can(const ps32 AObj, const TCANEvent AEvent){
    return internal_unregister_pretx_event_can(AObj, AEvent);
}
s32 register_pretx_event_canfd(const ps32 AObj, const TCANFDEvent AEvent){
    return internal_register_pretx_event_canfd(AObj, AEvent);
}
s32 unregister_pretx_event_canfd(const ps32 AObj, const TCANFDEvent AEvent){
    return internal_unregister_pretx_event_canfd(AObj, AEvent);
}
s32 register_pretx_event_lin(const ps32 AObj, const TLINEvent AEvent){
    return internal_register_pretx_event_lin(AObj, AEvent);
}
s32 unregister_pretx_event_lin(const ps32 AObj, const TLINEvent AEvent){
    return internal_unregister_pretx_event_lin(AObj, AEvent);
```

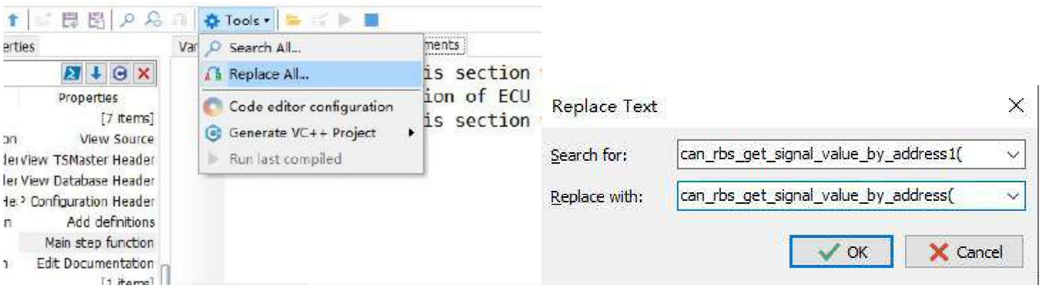
Step three: If user wants to continue searching for this text, there are two methods: the F3 shortcut key or clicking the quick icon on the script tool. Both methods have the same effect. After performing the action, you can continue to search through the subsequent text, as shown below:



```
    void init_w_ext_id(s32 AId, s32 ADLC) {
        FIdxChn = 0;
        FIdentifier = AId;
        FDLC = ADLC;
        FReserved = 0;
        FProperties = 0;
        is_tx = false;
        is_std = false;
        is_data = true;
        *(u64*)(&FData[0]) = 0;
        FTimeUs = 0;
    }
} TCAN, *PCAN;
```

1.15.9.2. Replace Code in the C Script

In the C script, if there are changes to the user function interface or variable names and the variable is used in many places, replacing them one by one can be very troublesome. Therefore, the C script editor in TSMaster provides a way to globally replace. The operation path is: Tools -> Replace All, fill in the parameters to replace, as shown in the following figure:



Search for: can_rbs_get_signal_value_by_address1(

Replace with: can_rbs_get_signal_value_by_address(

1.15.10. Clarification

1.15.10.1. The Mini-Program was Running OK before, but after Upgrading the Version, clicking on it no longer Works

Solution: The upgrade of the project introduced new library files. Recompile the mini-program and then try running it again.

1.15.10.2. Why Can't the Signal Value be Read through the get Function

Situation Description:

In the Transmit window, messages are sent, but the signal changes cannot be observed in the script or on the Panel interface.

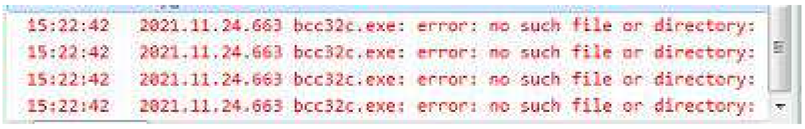
Explanation:

This is because the Transmit function for sending messages is primarily used for debugging purposes, and the messages sent do not synchronize back to the RBS engine. As a result, signals within the CAN RBS will not change in response to messages sent via Transmit. The CAN RBS signals are integrated with the RBS engine, the UI interface, and the C script.

1.15.10.3. Compilation Error

Situation Description:

After editing the script, compiling it results in the following error: "no such file or directory:".



```
15:22:42   2021.11.24.663 bcc32c.exe: error: no such file or directory:
15:22:42   2021.11.24.663 bcc32c.exe: error: no such file or directory:
15:22:42   2021.11.24.663 bcc32c.exe: error: no such file or directory:
15:22:42   2021.11.24.663 bcc32c.exe: error: no such file or directory:
```

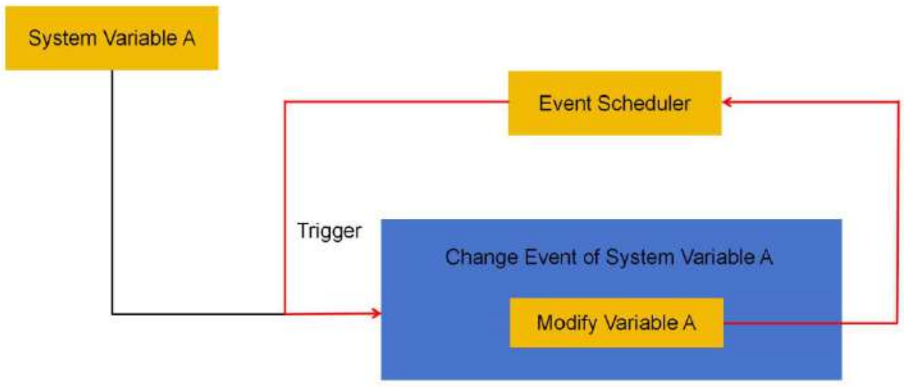
Explanation:

The error occurs because the built-in C script interpreter in TSMaster does not support Chinese characters in file paths. To resolve this issue, simply copy the folder to a path that does not contain Chinese characters, and the problem will be solved.

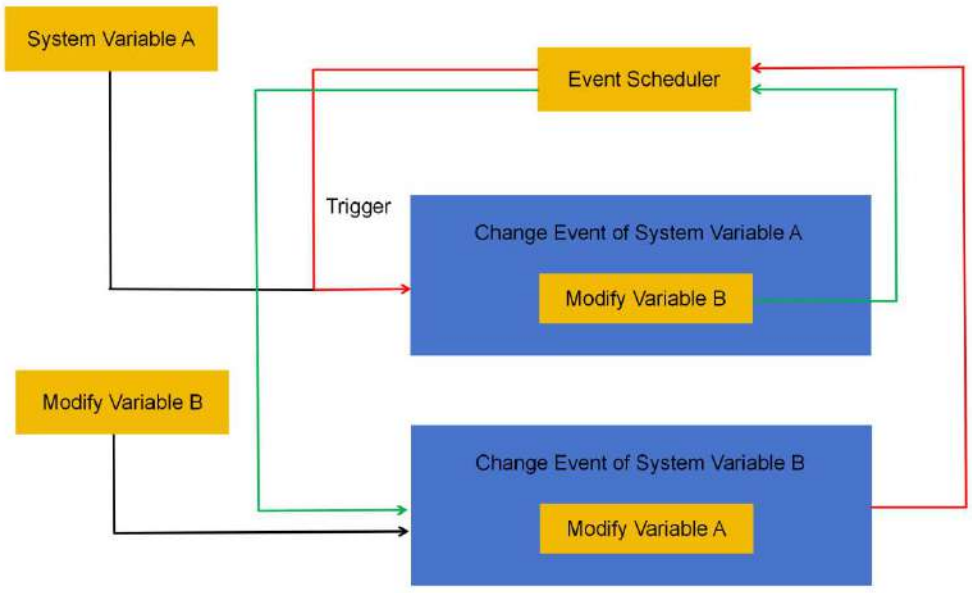
1.15.10.4. System Variable Triggers an Infinite Loop Logic

In TSMaster's C script, variable change events can be triggered by changes in variables. However, if the following situation occurs, it can cause a logical infinite loop.

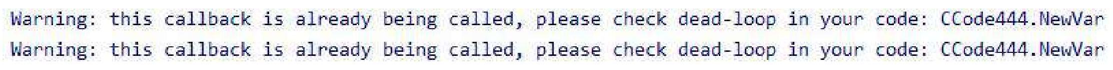
➢ System variable A corresponds to the variable change event A_Event. If a function that directly modifies system variable A itself is executed within A_Event, this will cause a logical infinite loop. As shown in the following figure:



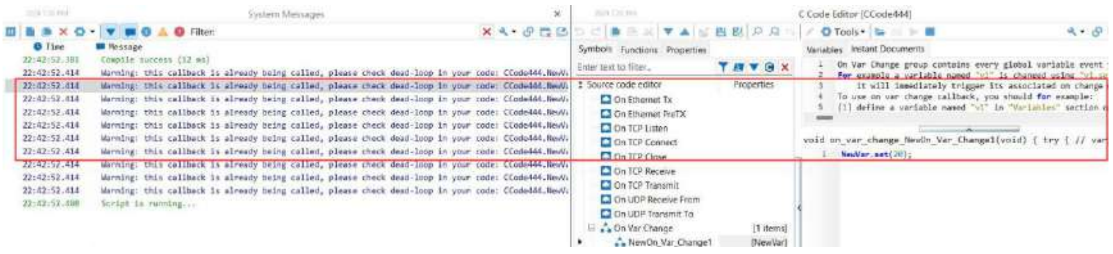
➢ System variable A is associated with the variable change event A_Event; system variable B is associated with the system variable change event B_Event. If a function that directly modifies system variable B itself is executed within A_Event, and then a function that modifies system variable A is executed within B, this will create a logical infinite loop. As shown in the following figure:



After sending the above codes that causes an infinite loop, if no action is taken, it can lead to the software logic becoming unresponsive. TSMaster has a built-in code detection mechanism. When a logical infinite loop occurs, it prints a message in the system messages to provide a prompt. As shown in the following figure:



```
Warning: this callback is already being called, please check dead-loop in your code: CCode444.NewVar
Warning: this callback is already being called, please check dead-loop in your code: CCode444.NewVar
```



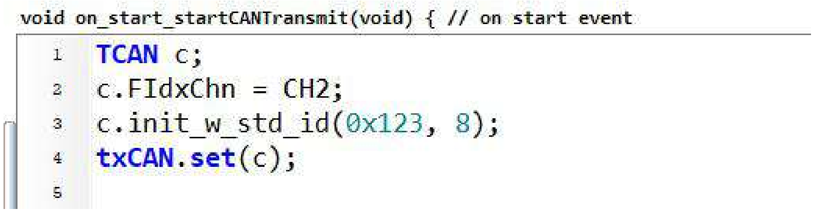
```
Time          Message
22:42:52.381  Compile success (12 ms)
22:42:52.414  Warning: this callback is already being called, please check dead-loop in your code: CCode444.NewV
22:42:52.414  Warning: this callback is already being called, please check dead-loop in your code: CCode444.NewV
22:42:52.414  Warning: this callback is already being called, please check dead-loop in your code: CCode444.NewV
22:42:52.414  Warning: this callback is already being called, please check dead-loop in your code: CCode444.NewV
22:42:52.414  Warning: this callback is already being called, please check dead-loop in your code: CCode444.NewV
22:42:52.414  Warning: this callback is already being called, please check dead-loop in your code: CCode444.NewV
22:42:52.414  Warning: this callback is already being called, please check dead-loop in your code: CCode444.NewV
22:42:52.414  Warning: this callback is already being called, please check dead-loop in your code: CCode444.NewV
22:42:52.414  Warning: this callback is already being called, please check dead-loop in your code: CCode444.NewV
22:42:52.414  Warning: this callback is already being called, please check dead-loop in your code: CCode444.NewV
22:42:53.400  Script is running...
```

```
On Var Change group contains every global variable event
For example a variable named "v1" is changed using "v1.se
    it will immediately trigger its associated on change
To use on var change callback, you should for example:
(1) define a variable named "v1" in "Variables" section
```

```
void on_var_change_NewOn_Var_Change1(void) { try { // var
  NewVar.set(20);
```

When a prompt as described above occurs, users must check the C script code and remove the part of the logic that causes the infinite loop. Otherwise, it will affect the efficiency of the program execution.

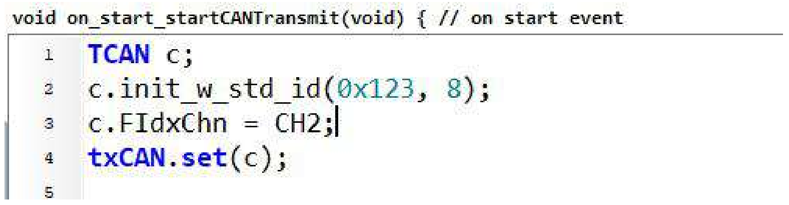
1.15.10.5. The Calling Position of the Initialization Function (init_w_std_id).



```
void on_start_startCANTransmit(void) { // on start event
TCAN c;
c.FIdxChn = CH2;
c.init_w_std_id(0x123, 8);
txCAN.set(c);

```

As shown in the script above, the original intention is: to set this message and send it to channel 2. However, during actual testing, it was found that the message was sent to channel 1. If channel 1 does not have a physical channel set up, the message sending will fail.

The main reason is that the message initialization function, such as init_w_std_id, will reset all internal data of the message, including setting the CAN channel to 0. Therefore, when users initialize the message and set the message attributes, they must first call the init_w_std_id function and then set other attributes. Only in this way are the set attributes valid. Therefore, the corrected code is shown as follows:



```
void on_start_startCANTransmit(void) { // on start event
TCAN c;
c.init_w_std_id(0x123, 8);
c.FIdxChn = CH2;
txCAN.set(c);

```

1.15.10.6. Exception Triggered by Same-named Mini-program

In TSMaster, creating mini-programs with the same name will trigger the following exception:

```
CCode1.c:

Database.c:

Fatal: Could not open C:\Users\XYY\Desktop\FT-Tech\ECHO_R\bin\CCode1.mp (program still running?)

bcc32c.exe: error: unable to remove file: C:/Users/XYY/Desktop/FT-Tech/ECHO_R/bin/CCode1.mp: Can't destroy file: access refuse.

bcc32c.exe: error: linker command failed with exit code 2 (use -Xdriver -v to see invocation)
```

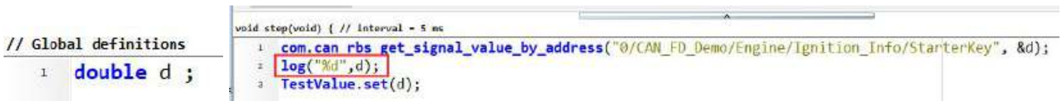
This is because, in TSMaster, each mini-program has its own independent runtime space based on its name. If two mini-programs use the same name, they will simultaneously access the same system resources, which will definitely cause a conflict error.

Solution:

Renaming one of the mini-programs to avoid the situation of having the same name.

1.15.10.7. The Log Function Prints out Zeros regardless of whether there is Data or not.

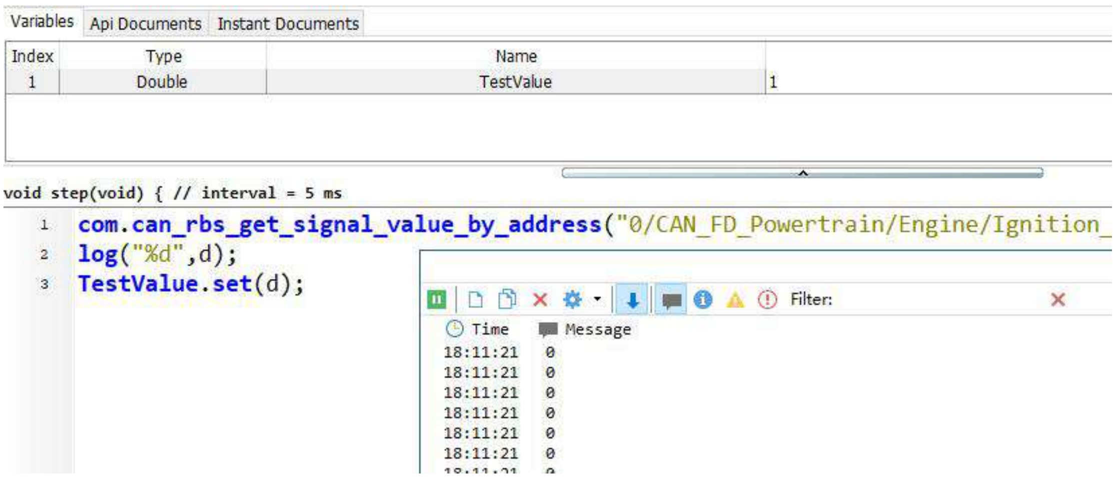
In the C script, the codes are as shown in the figure below:



```
// Global definitions
double d ;
```

```
void step(void) { // interval = 5 ms
com.can_rbs_get_signal_value_by_address("0/CAN_FD_Demo/Engine/Ignition_Info/StarterKey", &d);
log("%d",d);
TestValue.set(d);
```

Upon reviewing the print report, it was observed that regardless of the value of the signal d that is read, the printed value is always 0. The printed value does not match the actual value, as shown in the figure below:

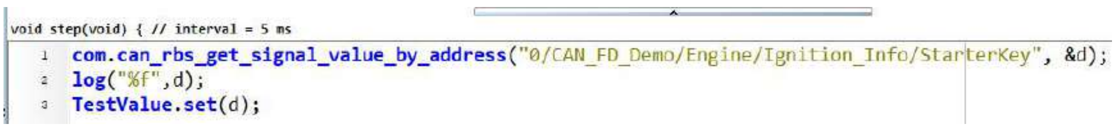


Index	Type	Name	
1	Double	TestValue	1

```
void step(void) { // interval = 5 ms
com.can_rbs_get_signal_value_by_address("0/CAN_FD_Powertrain/Engine/Ignition_
log("%d",d);
TestValue.set(d);
```

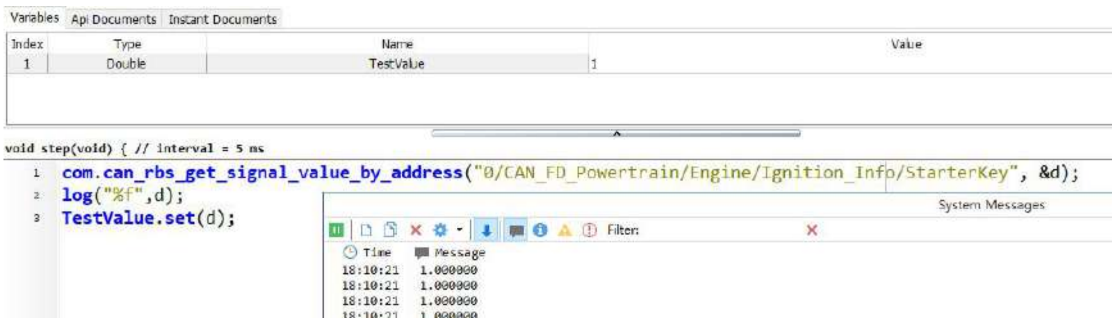
```
Time      Message
18:11:21  0
18:11:21  0
18:11:21  0
18:11:21  0
18:11:21  0
18:11:21  0
```

This is because 'd' is of the double type, but the log function prints using a format string intended for decimal numbers. The solution is to change the print format string from "%d" to "%f" to resolve this issue.



```
void step(void) { // interval = 5 ms
com.can_rbs_get_signal_value_by_address("0/CAN_FD_Demo/Engine/Ignition_Info/StarterKey", &d);
log("%f",d);
TestValue.set(d);
```

After revising the codes, the string printed through the system message matches the actual value, as shown in the figure below:



Index	Type	Name	Value
1	Double	TestValue	1

```
void step(void) { // interval = 5 ms
com.can_rbs_get_signal_value_by_address("0/CAN_FD_Powertrain/Engine/Ignition_Info/StarterKey", &d);
log("%f",d);
TestValue.set(d);
```

```
Time      Message
18:10:21  1.000000
18:10:21  1.000000
18:10:21  1.000000
```

Summary: When using the log function, pay attention to the string formatting.

1.15.11. Appendix

1.15.11.1. TSMaster Header

1.16. Mini-program

To expand the functionality of C script, TSMaster offers a library encapsulation mechanism known as "mini-program." Users can conveniently apply library files encapsulated in C/C++, Pascal, and other development environments (such as Visual Studio) to C script. This mechanism greatly enhances the flexibility of C script. For example, if a user wants to store data in a database or parse data from Excel, these function interfaces can be encapsulated into mini-program libraries and called by C script.

1.17. Invoke External DLL/LIB Library

During the custom development process, it is common to encounter the need to invoke external DLL/LIB library. These files may be written by the user themselves or provided by other suppliers. TSMaster supports the invocation of external binary program libraries, but it must be encapsulated through certain methods. This section demonstrates the method of encapsulating DLL libraries using the TDMS file recording program from NI as an example. The use of LIB libraries is similar to DLLs, and users can perform similar operations in the Visual Studio project to achieve this.

1.17.1. Acquire External Libraries

The TDMS library from NI can be downloaded through the link https://www.ni.com/content/dam/web/product-documentation/c_dll_tdm.zip. When using external libraries, please be aware of the following restrictions:

1. During the process of downloading external program libraries, please pay attention to the usage agreements of the published libraries. TOSUN only provides the environment for using external program libraries and is not responsible for actions that violate the usage agreements of external libraries.

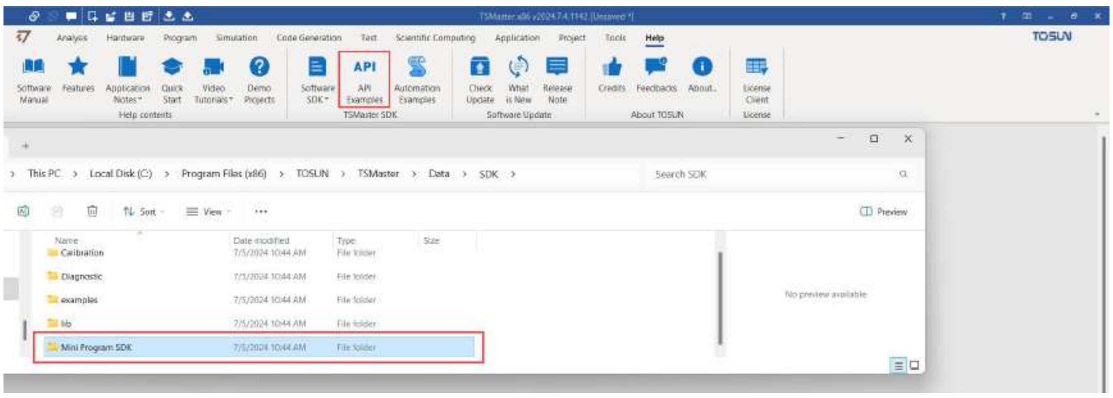
2. Once an external library is loaded by TSMaster, it becomes a part of the main program. If the external library experiences a crash or memory overflow, it can cause the TSMaster program to become unstable or crash. In such cases, it is necessary to reopen the software and uninstall the problematic external library.

3. TSMaster only supports external libraries in the 32-bit Microsoft Visual C++ (msvc) version. Please use the appropriate version of DLL/LIB files; otherwise, it will cause compilation errors.

In the attachment "tdms_example\TDM C DLL" directory, user can find the relevant files of the TDMS external library after extraction.

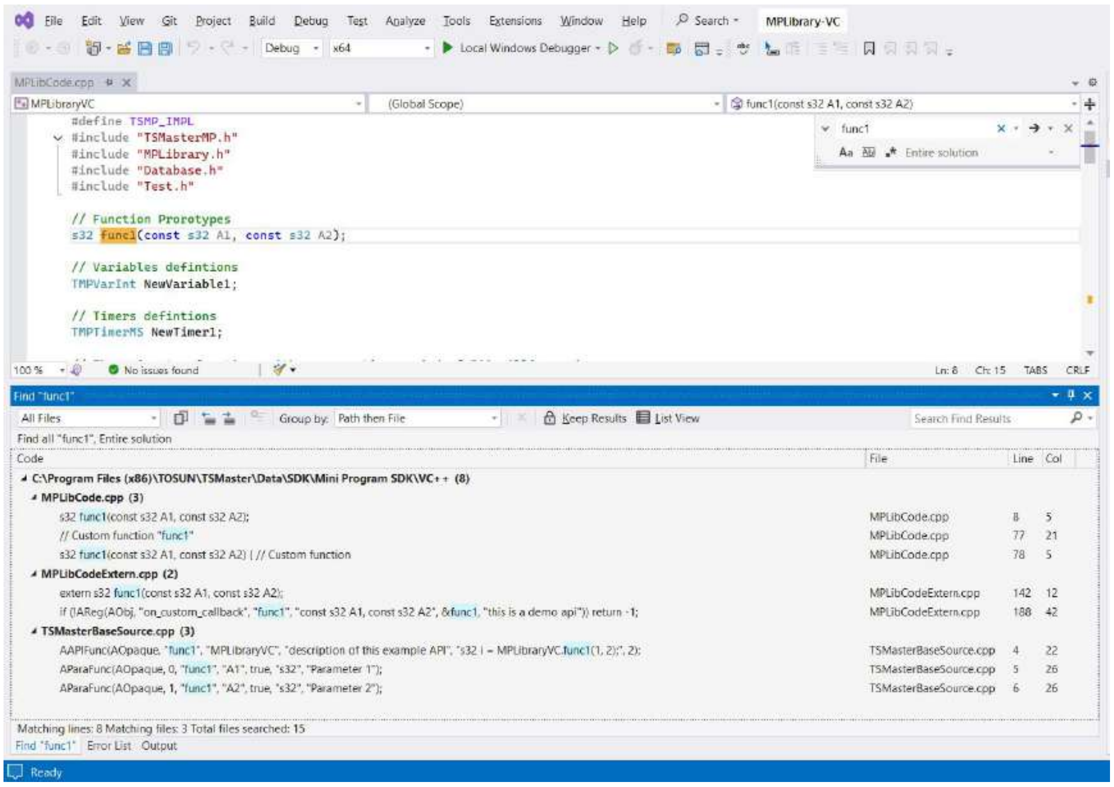
1.17.2. Prepare the Template for Invoking External Libraries

The template can be copied from the TSMaster program by navigating to Help -> API Examples -> Mini Program SDK directory, and under this directory, you will find a project named "VC++". Copy the project to another location that is customized by the user and keep it for future use. In the attachment "tdms_example\tdms_sdk", you can see the project prepared for the TDMS functionality.



1.17.3. Edit the Template and Generate the DLL

Whether the external library is a DLL, a LIB, or both, they can all be called within the template project. It is important to note that in order for TSMaster to recognize them properly, users need to provide not only their own logic but also function comments, parameter descriptions, and other information when preparing the template project. The specific method is to open the default template file, search globally for the keyword 'fun1', which is an example function. Wherever this function appears is where users need to add their custom content.



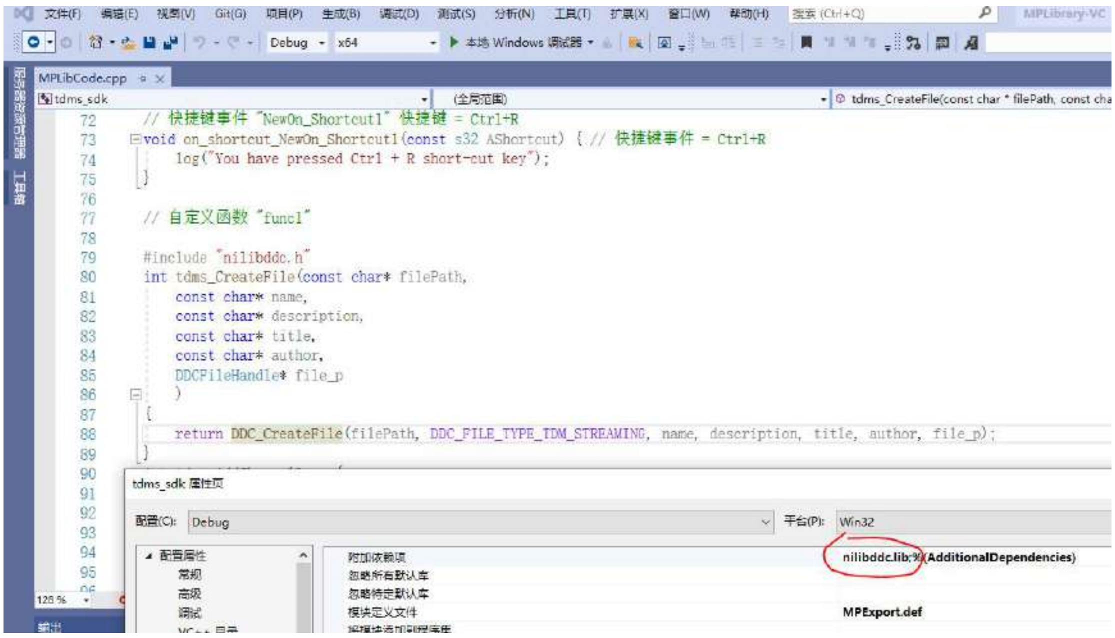
```
#define TSMP_IMPL
#include "TSMasterMP.h"
#include "MPLibrary.h"
#include "Database.h"
#include "Test.h"

// Function Prorotypes
s32 func1(const s32 A1, const s32 A2);

// Variables defintions
TMPVarInt NewVariable1;

// Timers defintions
TMPTimerMS NewTimer1;
```

Find all "func1", Entire solution

```
Code                                                                                    File                      Line  Col
C:\Program Files (x86)\TOSUN\TSMaster\Data\SDK\Mini Program SDK\VC++ (8)
  MPLibCode.cpp (3)
    s32 func1(const s32 A1, const s32 A2);                                              MPLibCode.cpp             8     5
    // Custom function "func1"                                                          MPLibCode.cpp             77    21
    s32 func1(const s32 A1, const s32 A2) { // Custom function                          MPLibCode.cpp             78    5
  MPLibCodeExtern.cpp (2)
    extern s32 func1(const s32 A1, const s32 A2);                                       MPLibCodeExtern.cpp       142   12
    if (!AReg(AObj, "on_custom_callback", "func1", "const s32 A1, const s32 A2", &func1, "this is a demo api")) return -1;   MPLibCodeExtern.cpp   188   42
  TSMasterBaseSource.cpp (3)
    AAPIFunc(AOpaque, "func1", "MPLibraryVC", "description of this example API", "s32 i = MPLibraryVC.func1(1, 2);", 2);   TSMasterBaseSource.cpp   4   22
    AParaFunc(AOpaque, 0, "func1", "A1", true, "s32", "Parameter 1");                   TSMasterBaseSource.cpp    5     26
    AParaFunc(AOpaque, 1, "func1", "A2", true, "s32", "Parameter 2");                   TSMasterBaseSource.cpp    6     26

Matching lines: 8 Matching files: 3 Total files searched: 15
```

As shown in the figure above, the declaration and implementation of the function `fun1` are implemented in the `MPLibCode.cpp` file. In the `MPLibCodeExtern.cpp` file, the existence of the function `fun1` is registered with the DLL management template. In the `TSMasterBaseSource.cpp` file, the relevant parameter information of the function `fun1` is registered with the DLL management template.

During the integration of the TDMS feature, user first need to copy the header files (.h) and library files (.lib) required for the compilation process to the project directory, and then include the .lib file as input in the project linker. For external functions, such as the DDC_CreateFile function that comes with the TDMS library, encapsulate it with a new function in the template, naming it tdms_CreateFile. Although the functions of the .lib file can be directly exported, it is generally recommended to create a new function to encapsulate it. This not only unifies the function names for easier user distinction but also ensures that all API function return values must be of type int. If the native external library does not use the int return type, it is necessary to obtain the return value through methods such as passing pointers. In this case, encapsulation must be used to conform to the required return type.



```
// 快捷键事件 "NewOn_Shortcut1" 快捷键 = Ctrl+R
void on_shortcut_NewOn_Shortcut1(const s32 AShortcut) { // 快捷键事件 = Ctrl+R
    log("You have pressed Ctrl + R short-cut key");
}

// 自定义函数 "func1"

#include "nilibddc.h"
int tdms_CreateFile(const char* filePath,
    const char* name,
    const char* description,
    const char* title,
    const char* author,
    DDCFileHandle* file_p
    )
{
    return DDC_CreateFile(filePath, DDC_FILE_TYPE_TDM_STREAMING, name, description, title, author, file_p);
}
```

tdms_sdk 属性页

配置(C): Debug   平台(P): Win32

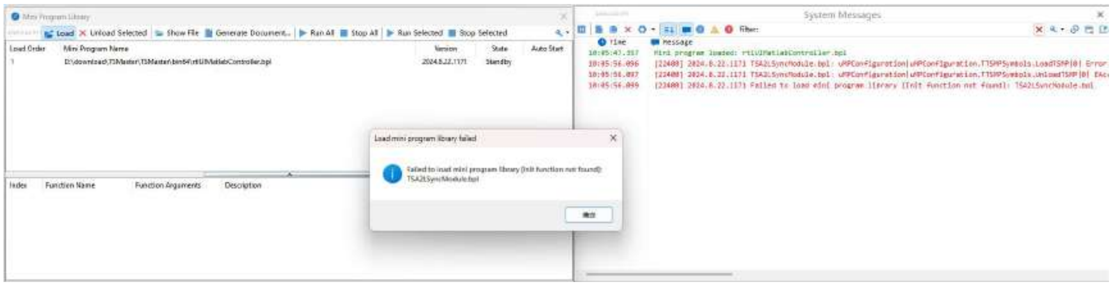
附加依赖项: nilibddc.lib;%(AdditionalDependencies)

模块定义文件: MPExport.def

For the encapsulation process for other TDMS functions, please refer to the example project. For correct operation, it is essential to ensure that the encapsulated function codes and the function registration codes are matched. Based on this template, the required "tdms_sdk.dll" can be generated in the Debug/Release-x86 mode.

1.17.4. Use the Template Dll in TSMaster Project

To use the "tdms_sdk.dll" in TSMaster, user can either drag and drop the DLL directly into the project or load it through the Program -> Mini Program Library -> Load, as shown in the figure below. Direct loading will fail because the template DLL depends on the TDMS runtime DLL files, which are located in the "tdms_example\TDM C DLL\dev\bin\32-bit" directory provided by NI. To correctly load the DLL, copy all files to the TSMaster project's \Plugins\Dependencies directory (this directory needs to be created manually, as TSMaster does not create it by default), and then load the template DLL.



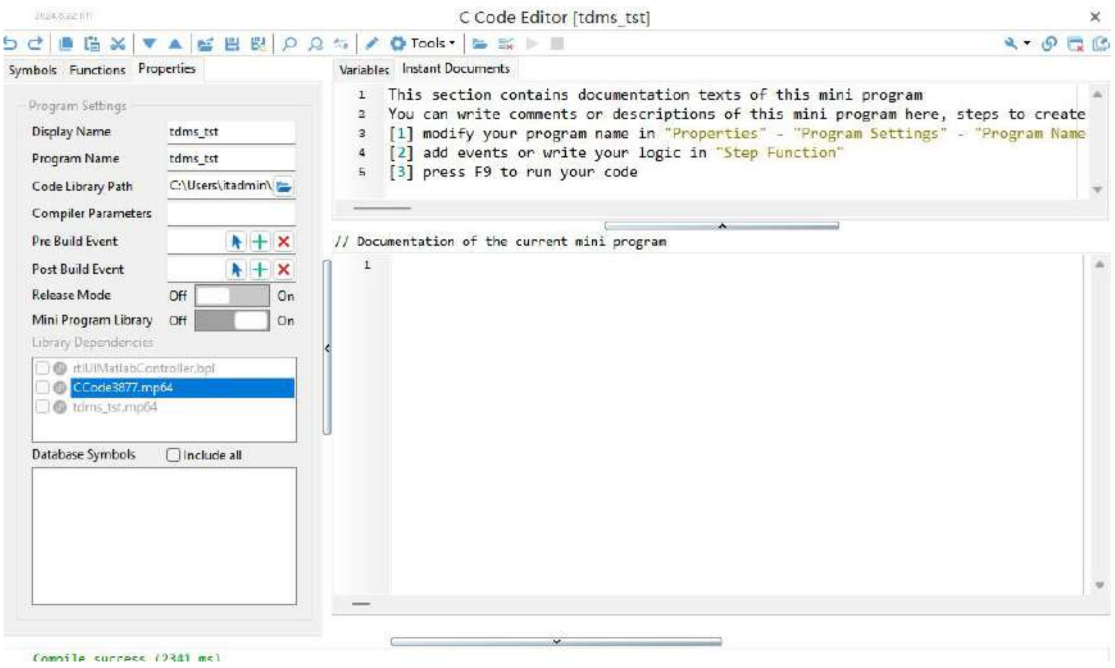
Load Order	Mini Program Name	Version	State	Auto Start
1	D:\download\TSMaster\TSMaster\bin64\rtlUIMatlabController.bpl	2024.8.22.1171	Standby	

Load mini program library failed

Failed to load mini program library (Init function not found): TSA2LSyncModule.bpl

```
10:05:47.357   Mini program loaded: rtlUIMatlabController.bpl
10:05:56.096   [22408] 2024.8.22.1171 TSA2LSyncModule.bpl: uMPConfiguration|uMPConfiguration.TTSMPSymbols.LoadTSMP|0| Error
10:05:56.097   [22408] 2024.8.22.1171 TSA2LSyncModule.bpl: uMPConfiguration|uMPConfiguration.TTSMPSymbols.UnloadTSMP|0| EAc
10:05:56.099   [22408] 2024.8.22.1171 Failed to load mini program library (Init function not found): TSA2LSyncModule.bpl
```

Open the example project "tdms_example\tdms_example", and the first test case in the test system is for generating test code for a TDMS file. Users can also create a new mini-program to call the function, and there is no difference in operation between the two, as both require checking the required external libraries in the properties window, and then calling the necessary functions in the script program. As shown in below figure, after the script is run, an example TDMS file will be created in the root directory of the D drive.



Display Name	tdms_tst
Program Name	tdms_tst
Code Library Path	C:\Users\itadmin\
Compiler Parameters	
Pre Build Event	
Post Build Event	
Release Mode	Off On
Mini Program Library	Off On

Library Dependencies
- rtlUIMatlabController.bpl
- CCode3877.mp64
- tdms_tst.mp64

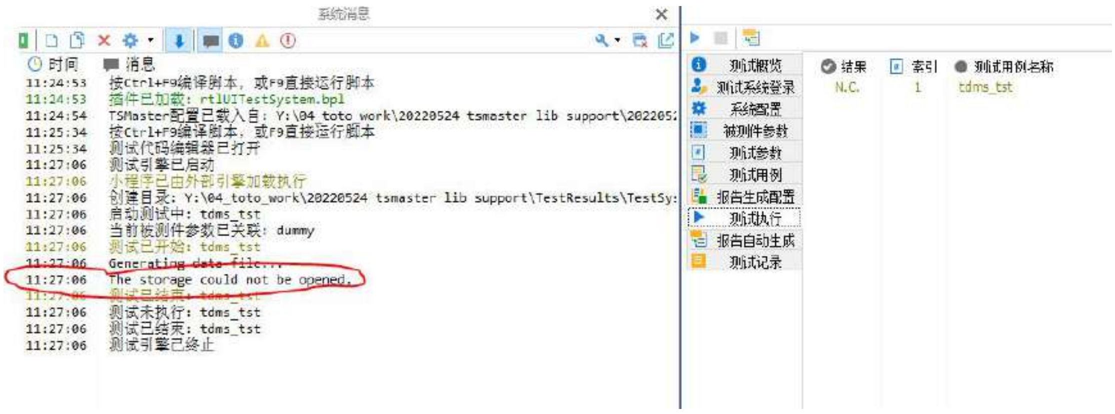
Database Symbols ☐ Include all

```
This section contains documentation texts of this mini program
You can write comments or descriptions of this mini program here, steps to create
[1] modify your program name in "Properties" - "Program Settings" - "Program Name
[2] add events or write your logic in "Step Function"
[3] press F9 to run your code
```

```
// Documentation of the current mini program
```

```
Compile success (2341 ms)
```

Running the script directly, user can find that the program runs but does not create the TDMS file as required. By checking the error messages in the run log file, user can see a message indicating that the storage device could not be opened. This issue is specific to the TDMS library because, in addition to DLLs, its dependencies include a folder named "DataModels". When TSMaster uses the dependent files in the "tdms_example\Plugins\Dependencies" directory, it does not associate with folders, so user needs to manually copy this folder to the TSMaster installation directory, such as "C:\Program Files (x86)\TOSUN\TSMaster\bin". For some external libraries that only have DLL dependencies, manual copying is not required.



```
时间      消息
11:24:53  按Ctrl+F9编译脚本, 或F9直接运行脚本
11:24:53  插件已加载: rtlUITestSystem.bpl
11:24:54  TSMaster配置已载入自: Y:\04 toto work\20220524 tsmaster lib support\202205:
11:25:34  按Ctrl+F9编译脚本, 或F9直接运行脚本
11:25:34  测试代码编辑器已打开
11:27:06  测试引擎已启动
11:27:06  小程序已由外部引擎加载执行
11:27:06  创建目录: Y:\04_toto_work\20220524 tsmaster lib support\TestResults\TestSy:
11:27:06  启动测试中: tdms_tst
11:27:06  当前被测件参数已关联: dummy
11:27:06  测试已开始: tdms_tst
11:27:06  Generating data file...
11:27:06  The storage could not be opened.
11:27:06  测试已结束: tdms_tst
11:27:06  测试未执行: tdms_tst
11:27:06  测试已结束: tdms_tst
11:27:06  测试引擎已终止
```

结果	索引	测试用例名称
N.C.	1	tdms_tst

After resolving the dependencies of the folder, the TDMS file can be correctly generated. Users can refer to the above process to implement their own logic. The entire process requires a certain understanding of the Visual Studio environment. If there are any unclear points, please refer to the example project.

1.17.5. Debugging the Template DLL within the TSMaster Project

During the process of using the template DLL, debugging is also unavoidable. In Visual Studio, user can change the project properties Common -> Output Directory and set the DLL's directory to the "MPLibrary" directory of the project for debugging, such as generating the DLL in the "tdms_example\tdms_example\MPLibrary" of the example "tdms_example". Then first run the TSMaster project, and then start the debugging function in Visual Studio. The debugging process is the same as the mini-program debugging method, and you can refer to the debugging process of the mini-program for guidance.

1.18. Graphical Editing Panel

The TSMaster graphical editing panel allows users to develop their own graphical windows for handling functions such as message transmission and reception, signal parsing, and display.

1.18.1. Toolbar

1.18.1.1. Mode Selection Button, which mainly Includes the Following Modes:

(1) In the depressed state, the current Panel is in edit mode, and users can add or delete controls and edit control properties.

(2) In the released state, the current Panel is in test run mode, displaying the actual state of the panel when it is running, and users cannot perform editing.

(3) In the grayed-out state, the current Panel is in the running state, indicating that TSMaster is currently connected to a device and running. If a user wishes to re-edit the interface, they must disconnect TSMaster from the device before they can enter the edit state.

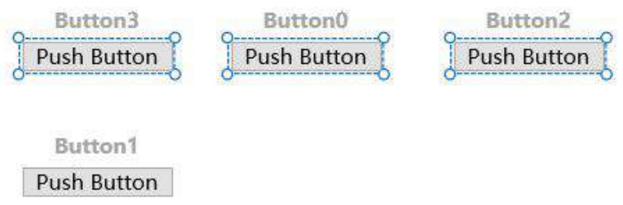
1.18.1.2. Layered Control

When there is an overlap of controls, move some controls to the front and move some controls to the back.

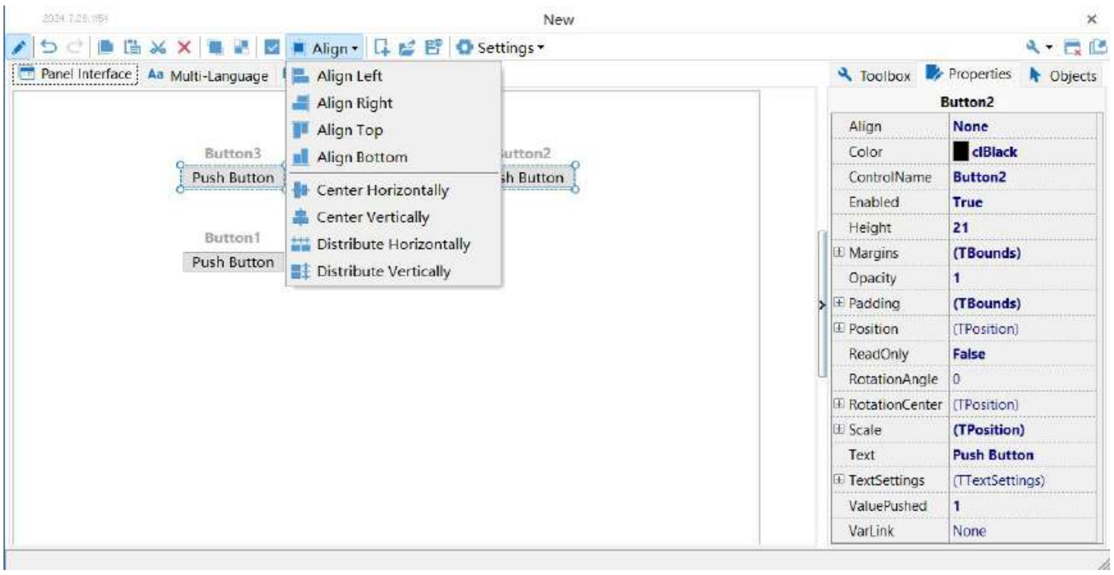
1.18.1.3. Align Align Control

There are two steps to align the controls:

1. First, select the multiple controls you want to align: hold down the Ctrl key and click on each button with the mouse to select multiple controls. As shown below:



2. Select the align button from the top menu bar, as shown below:

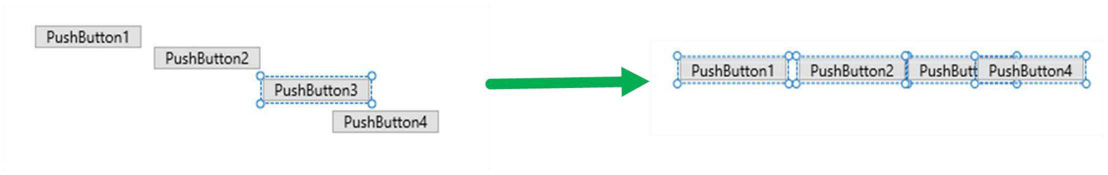


Button2

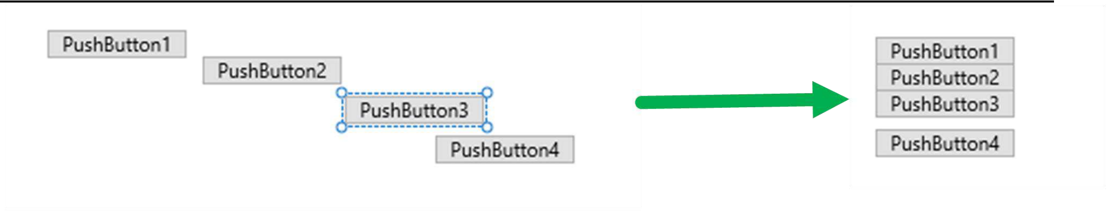
Align	None
Color	clBlack
ControlName	Button2
Enabled	True
Height	21
Margins	(TBounds)
Opacity	1
Padding	(TBounds)
Position	(TPosition)
ReadOnly	False
RotationAngle	0
RotationCenter	(TPosition)
Scale	(TPosition)
Text	Push Button
TextSettings	(TTextSettings)
ValuePushed	1
VarLink	None

Align options:

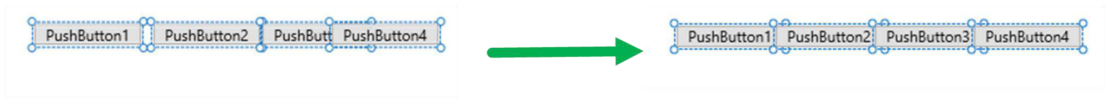
- ➢ Align Left: align the left edges of all selected objects to the same vertical position.
- ➢ Align Right: align the right edges of all selected objects to the same vertical position.
- ➢ Align Top: align the top edges of all selected objects to the same horizontally position.
- ➢ Align Bottom: align the bottom edges of all selected objects to the same horizontally position.
- ➢ Center Horizontally: align horizontally with the middle object as the reference.



- ➢ Center Vertically: align vertically with the middle object as the reference.

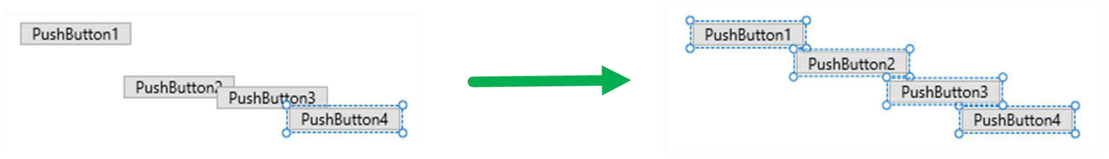


➢ Distribute Horizontally: horizontally and evenly distribute.



Calculate the average spacing between the controls based on the coordinates of the leftmost and rightmost controls, and then arrange them evenly on the horizontal axis.

➢ Distribute Vertically: vertically and evenly distribute.



Calculate the average spacing between the controls based on the coordinates of the topmost and bottommost controls, and then arrange them evenly on the vertical axis.

1.18.1.4. Add Panel

Create a brand new Panel. This action will delete all existing controls in the Panel.

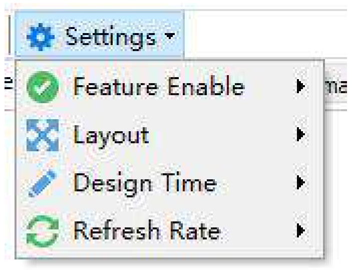
1.18.1.5. Load Configuration

Load an existing Panel configuration file.

1.18.1.6. Save Configuration

Save the current Panel's configuration.

1.18.1.7. Panel Parameter Configurations



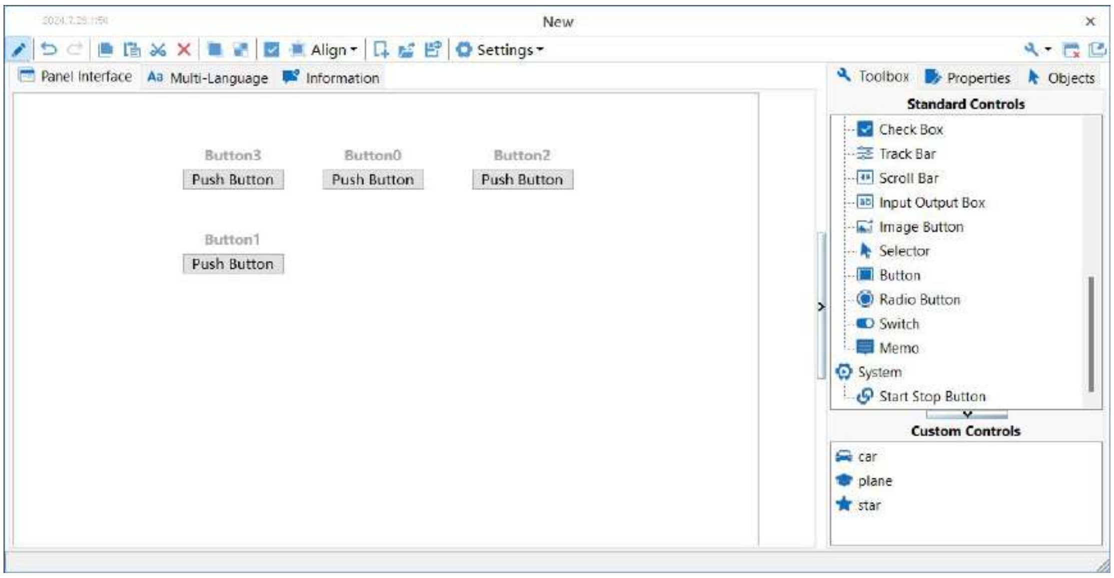
Mainly includes the following parameter configurations:

- Feature Enable: whether to enable Panel.
- Layout: the internal control layout in Panel.
- Design Time: whether to display link labels and control names during design time.
- Refresh Rate: configure the refresh time for the Panel data. A recommended refresh time is 300 milliseconds. If the computer has a higher specification, a higher refresh rate can be configured.

1.18.2. Basic Operations for Control

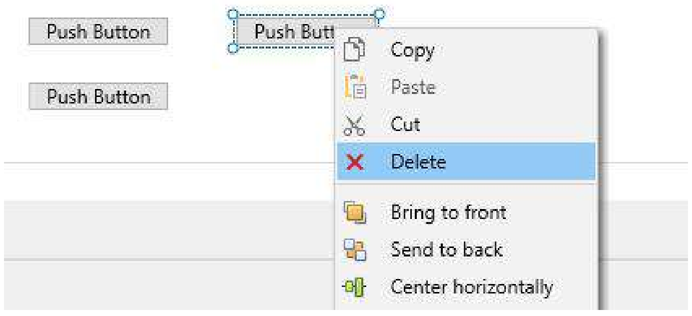
1.18.2.1. Add Control

Select the control and keep holding the left mouse button, then drag it to the target position.



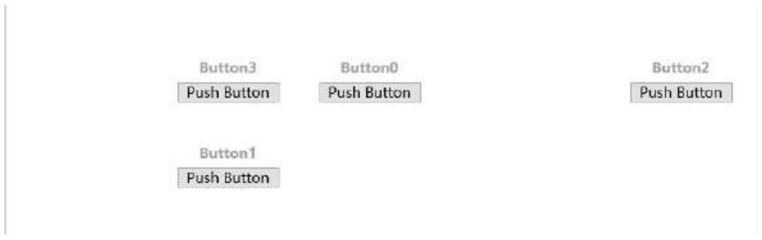
1.18.2.2. Delete Control

To delete a control, user can directly press the Delete key on the keyboard, or right-click and select "Delete".



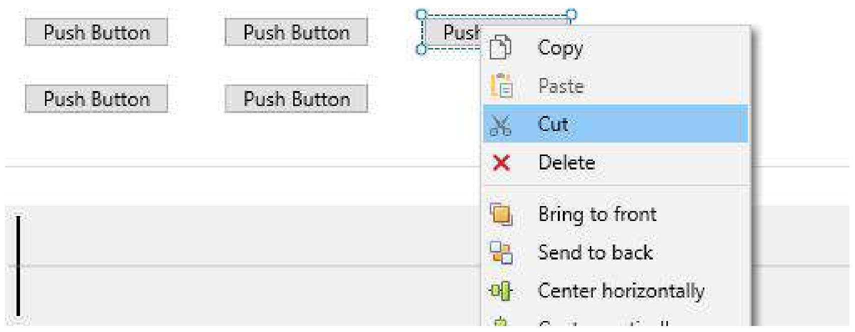
1.18.2.3. Move Control

Each control has its own container, and within the scope of the container, to adjust the position, simply select the control, and keep holding the left mouse button, and move it. As shown below:



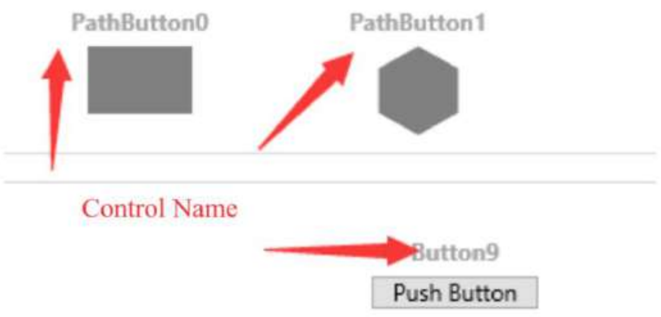
1.18.2.4. Move outside the Container

TSMaster does not support dragging controls directly outside of a container. If user wants to move a control out of a container, use the cut (Ctrl+X) and paste (Ctrl+V) method to move the control outside the container.

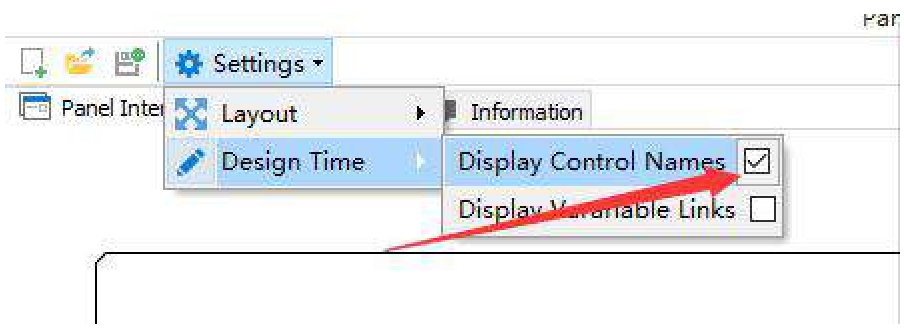


1.18.2.5. Display Control Type Name

During the Design Time phase, by default, the name of the control is displayed (this name serves as the control's unique ID, which is assigned by the system and cannot be modified and also not visible during runtime), as shown in the figure below:

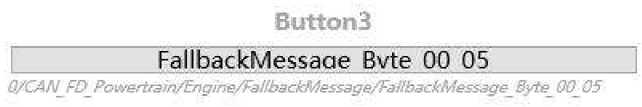


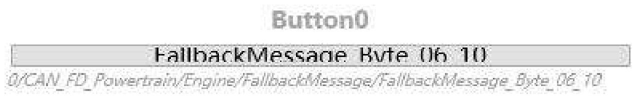
If the user do not wish to see the unique name of the control during the design phase, adjust this configuration in the Settings interface, as shown below:



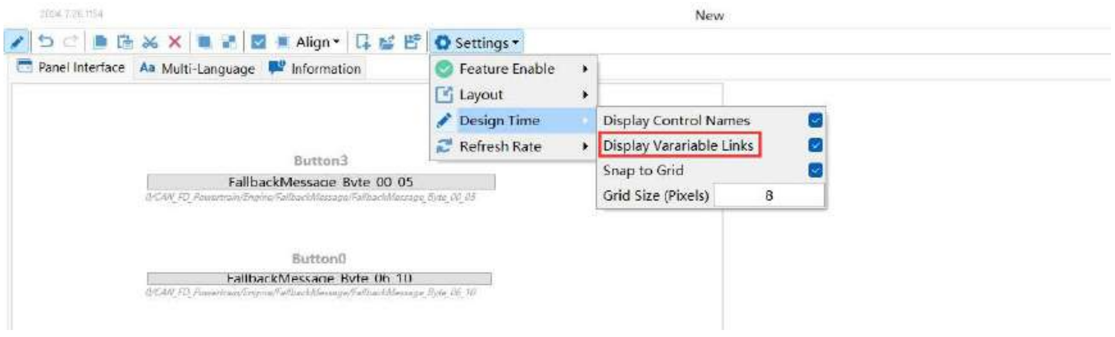
1.18.2.6. Display the Variable Link Associated with the Control

During the Design Time phase, users can set it to display the variables currently associated with the control (such as CAN/LIN signals or system variables), which helps the designer clearly understand the signal value associated with the control, as shown in the figure below:


Button3
FallbackMessage_Byte_00_05
0/CAN_FD_Powertrain/Engine/FallbackMessage/FallbackMessage_Byte_00_05


Button0
FallbackMessage_Byte_06_10
0/CAN_FD_Powertrain/Engine/FallbackMessage/FallbackMessage_Byte_06_10

During the Design Time phase, the operation to enable or disable the display of the associated signal is the same as that for enabling or disabling the display of the control name, as shown in the figure below:



1.18.3. Introduction for Control

1.18.3.1. Graphic Control

1.18.4. Variable Associated Control

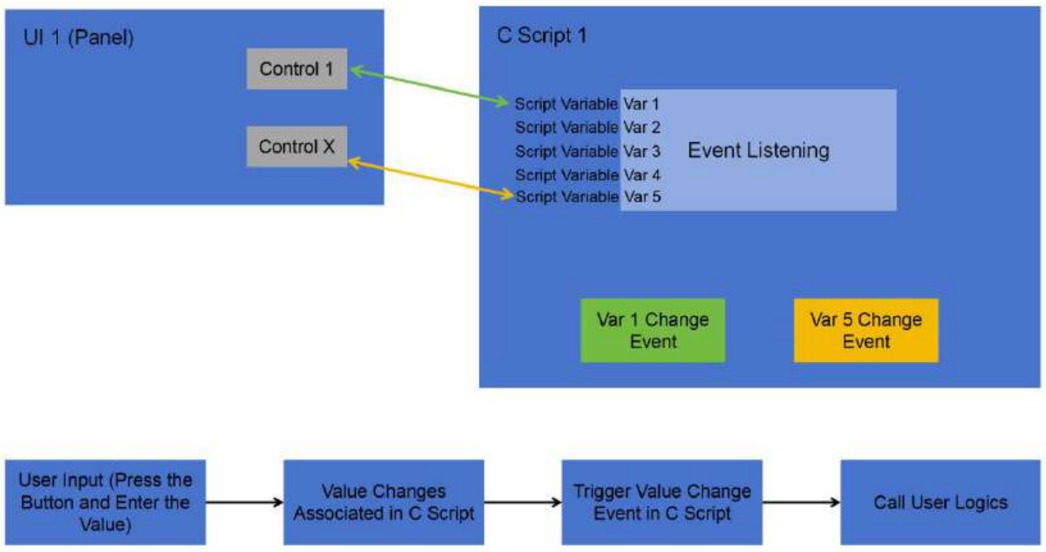
Switches for display only.
TextInputBox for display only.

1.18.5. UI Event

TSMaster offers a rich Panel interface. By associating variables, it is possible to modify the values of variables, or to display the modified values on the UI interface after the modification, which is relatively easy to understand. If users want to implement the function of sending a string of messages after pressing a button in the TSMaster software, it is necessary to work with C scripts to implement UI event mechanisms.

1.18.5.1. UI Event Mechanism

The event mechanism architecture diagram for Panel is shown below:



The implementation of the Panel event mechanism can be summarized as follows: user input in the Panel (pressing a button, entering a value, etc.) -> change the value of the associated mini-program variable -> trigger the value change event in the C script -> execute the codes that the user wants to run within the event. Below is an explanation of the process of adding a UI event, taking the example of sending a message by pressing a button.

1.18.5.2. Example: Message Sending through Pressing the Button

The effect achieved by this example is: pressing the button on the UI interface sends one frame of the message; when the button is released, another frame of the message is sent.

➢ **First step**: prepare the script and Panel.

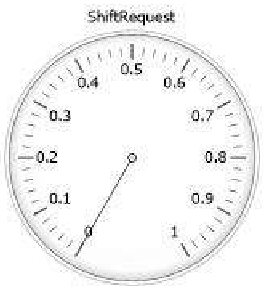
➢ **Second step:**

1.18.6. Clarification

1.18.6.1. Why is the DBC Parsing Correct, but the Control on the Panel Display Incorrect Value

Problem Description:

Why does the signal value parsed from the DBC come out as 12%, but the dashboard always displays 1.0 (100%) ?



Check the signal definition, because the range of the signal value is defined as 0-100, with the unit being "%". The parsed signal value of 12% indicates that the signal value is 12, and the unit is the string "%", not that the signal value is 0.12. Therefore, the range of the dashboard should be 0-100, which is consistent with the range in the signal definition. If the dashboard's range is set to 0-1, then when the signal value is shown as 12%, the dashboard will of course display the maximum value of 1.

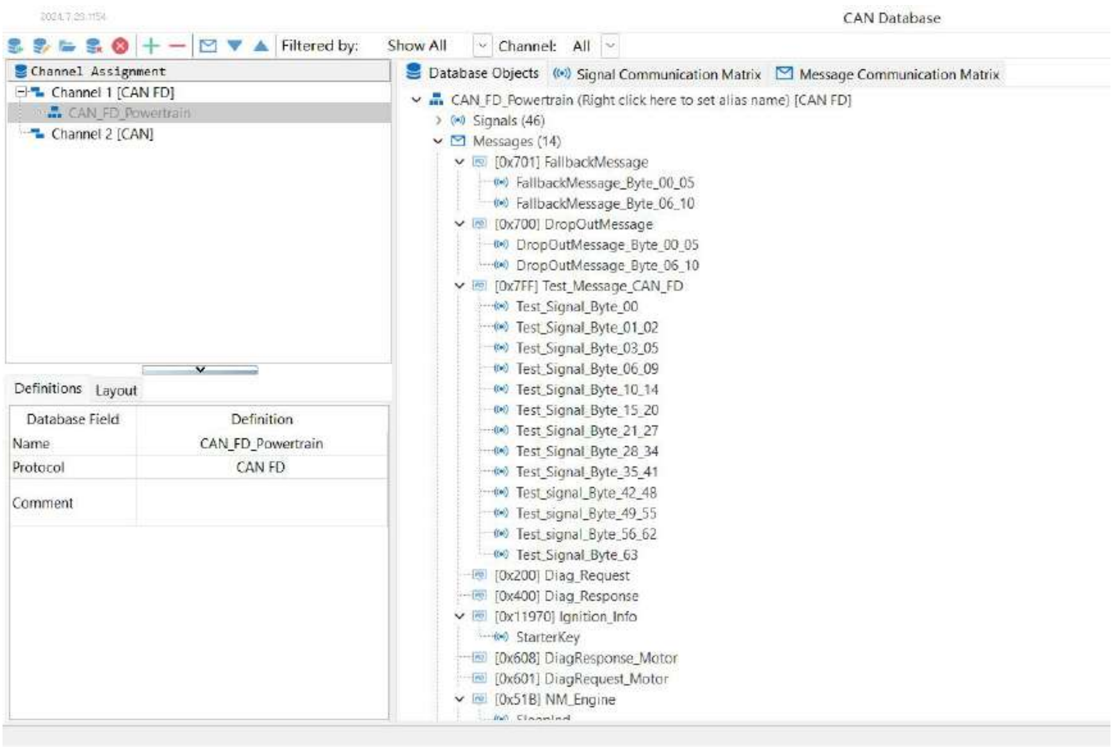
Solution:

Setting the properties of the dashboard control to match the display range with the signal's range. Adjust it from 0-1 to 0-100. After the adjustment, the dashboard display is normal.

1.18.6.2. After Adding the DBC, Can not See any Signal when Trying to Associate Them with the Panel

Situation Description:

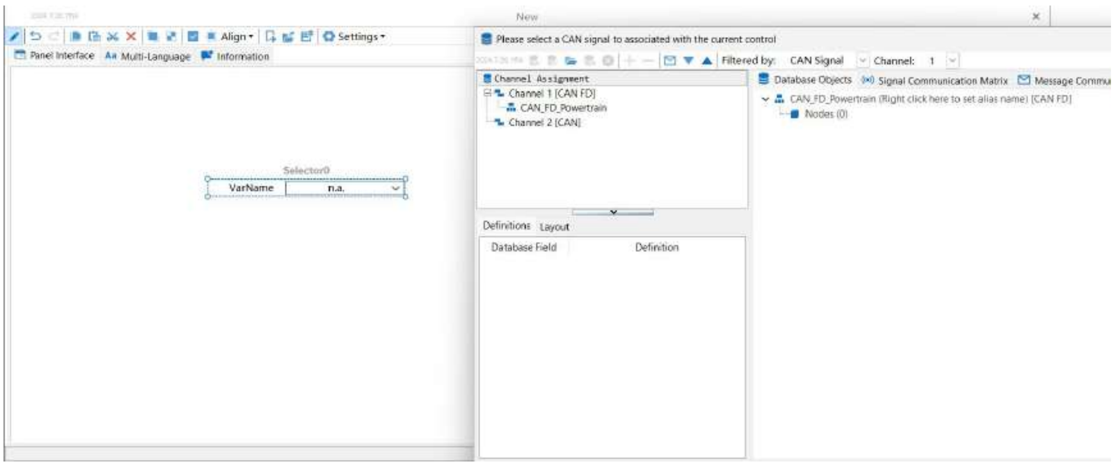
The example database is loaded: CAN_FD_PowerTrain



```
Channel Assignment
  Channel 1 [CAN FD]
    CAN_FD_Powertrain
  Channel 2 [CAN]

CAN_FD_Powertrain (Right click here to set alias name) [CAN FD]
  Signals (46)
  Messages (14)
    [0x701] FallbackMessage
      FallbackMessage_Byte_00_05
      FallbackMessage_Byte_06_10
    [0x700] DropOutMessage
      DropOutMessage_Byte_00_05
      DropOutMessage_Byte_06_10
    [0x7FF] Test_Message_CAN_FD
      Test_Signal_Byte_00
      Test_Signal_Byte_01_02
      Test_Signal_Byte_03_05
      Test_Signal_Byte_06_09
      Test_Signal_Byte_10_14
      Test_Signal_Byte_15_20
      Test_Signal_Byte_21_27
      Test_Signal_Byte_28_34
      Test_Signal_Byte_35_41
      Test_signal_Byte_42_48
      Test_signal_Byte_49_55
      Test_signal_Byte_56_62
      Test_Signal_Byte_63
    [0x200] Diag_Request
    [0x400] Diag_Response
    [0x11970] Ignition_Info
      StarterKey
    [0x608] DiagResponse_Motor
    [0x601] DiagRequest_Motor
    [0x51B] NM_Engine
```

Database Field	Definition
Name	CAN_FD_Powertrain
Protocol	CAN FD
Comment	

It can be seen that there are messages and signals available. In the panel, an InputOutputBox is added to associate with a CAN signal, and the result is showing “n.a.” as follows:

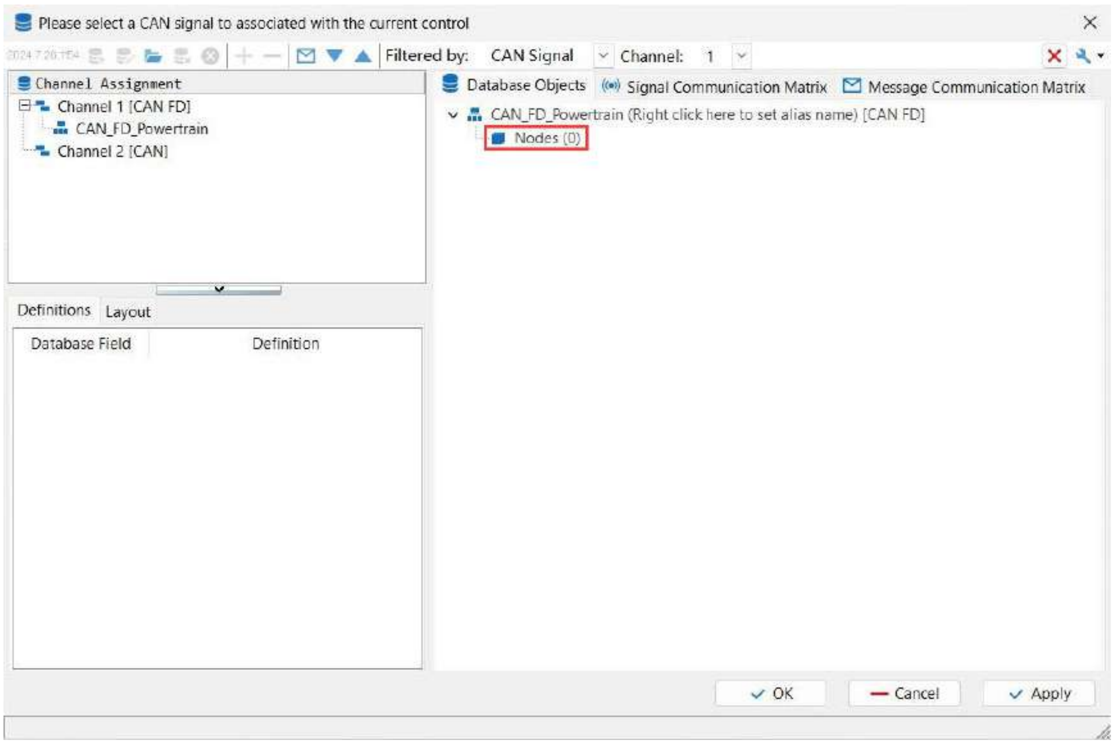


```
Channel Assignment
  Channel 1 [CAN FD]
    CAN_FD_Powertrain
  Channel 2 [CAN]

CAN_FD_Powertrain (Right click here to set alias name) [CAN FD]
  Nodes (0)
```

At this point, if user uses simple display controls, such as a ProgressBar, to associate with the signal, the user can see the signal.

Cause Analysis:

The InputBox is for input signals, logically, this kind of signal needed to be modified and sent onto the bus, so they must be associated with a CAN signal that is linked to a transmitting node. After that, let's continue to examine this database:


Please select a CAN signal to associated with the current control

Filtered by: CAN Signal　Channel: 1

Channel Assignment
- Channel 1 [CAN FD]
  - CAN_FD_Powertrain
- Channel 2 [CAN]

Definitions　Layout

Database Field　Definition

Database Objects　Signal Communication Matrix　Message Communication Matrix

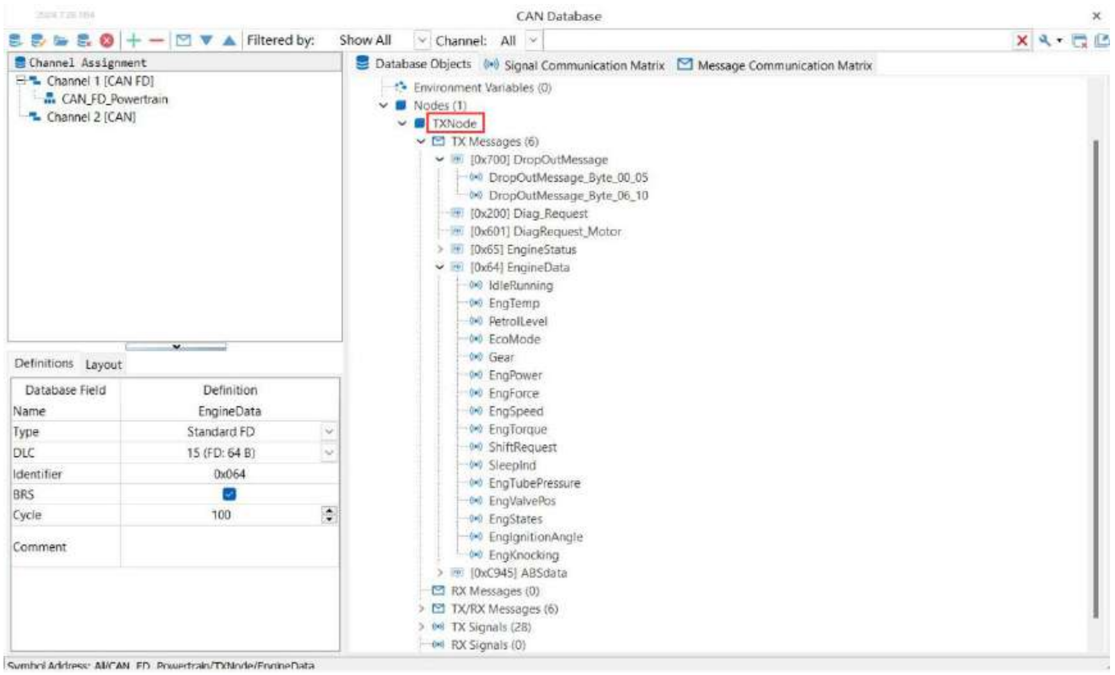
- CAN_FD_Powertrain (Right click here to set alias name) [CAN FD]
  - Nodes (0)

OK　Cancel　Apply

As shown, the number of nodes is 0, which means there are no sending or receiving nodes. In this case, it is impossible to start RBS simulation because there are no active node, and therefore no signals related to transmitting nodes can be seen.

Solution:

By adding a transmitting node and associating the message and signals, the user will be able to see the signals.



Database Field	Definition
Name	EngineData
Type	Standard FD
DLC	15 (FD: 64 B)
Identifier	0x064
BRS	☑
Cycle	100
Comment	

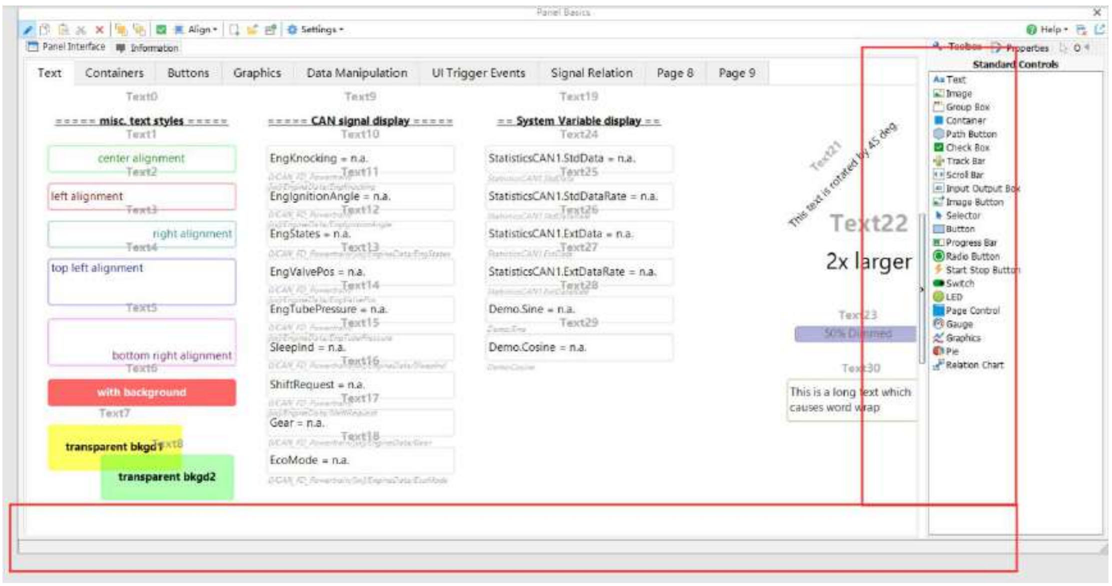
```
Environment Variables (0)
Nodes (1)
  TXNode
    TX Messages (6)
      [0x700] DropOutMessage
        DropOutMessage_Byte_00_05
        DropOutMessage_Byte_06_10
      [0x200] Diag_Request
      [0x601] DiagRequest_Motor
      [0x65] EngineStatus
      [0x64] EngineData
        IdleRunning
        EngTemp
        PetrolLevel
        EcoMode
        Gear
        EngPower
        EngForce
        EngSpeed
        EngTorque
        ShiftRequest
        SleepInd
        EngTubePressure
        EngValvePos
        EngStates
        EngIgnitionAngle
        EngKnocking
      [0xC945] ABSdata
    RX Messages (0)
    TX/RX Messages (6)
    TX Signals (28)
    RX Signals (0)
```

Symbol Address: All/CAN_FD_Powertrain/TXNode/EngineData

1.18.6.3. The Scrollbar is not Visible in the Panel

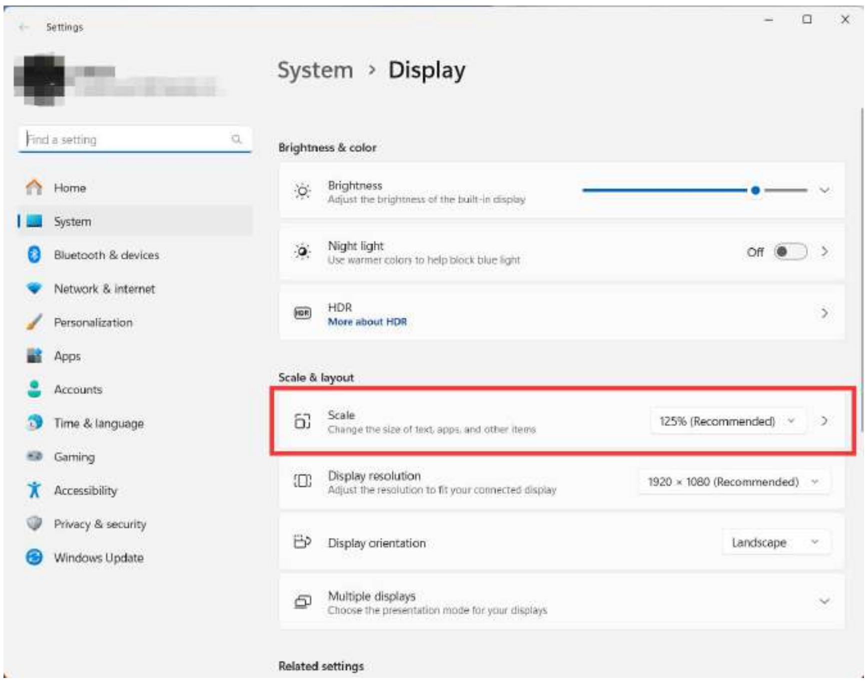
When designing a Panel, there may be situations where the scrollbars (both vertical and horizontal) are not visible. If the Panel's area is large enough to exceed the screen display area, part of the Panel may be obscured.

As shown below, there is no scrollbar on the bottom.



Cause:

The computer screen has been set to scale, which causes the controls inside not to correctly calculate the size relative to the screen, and as a result, the scrollbars cannot be displayed properly. Check the computer settings as follows:



Settings

System › Display

Home
System
Bluetooth & devices
Network & internet
Personalization
Apps
Accounts
Time & language
Gaming
Accessibility
Privacy & security
Windows Update

Brightness & color

Brightness — Adjust the brightness of the built-in display

Night light — Use warmer colors to help block blue light — Off

HDR — More about HDR

Scale & layout

Scale — Change the size of text, apps, and other items — 125% (Recommended)

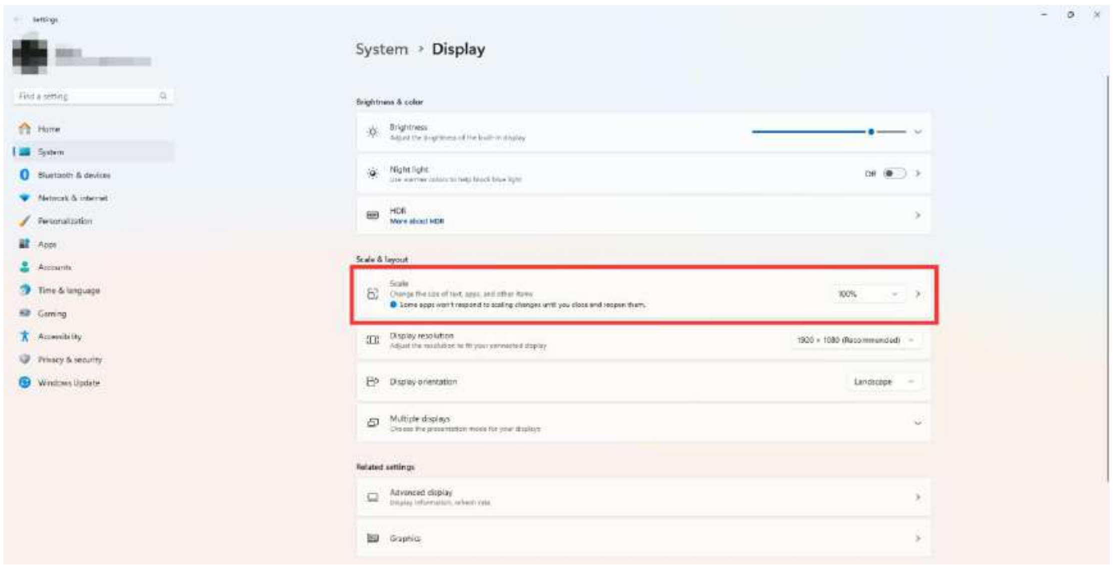
Display resolution — Adjust the resolution to fit your connected display — 1920 × 1080 (Recommended)

Display orientation — Landscape

Multiple displays — Choose the presentation mode for your displays

Related settings

Solution: Turn off screen scaling, set the scaling to 100%, restart the software, and the user will be able to see the scrollbars of the window again.



System › Display

Scale & layout

Scale — Change the size of text, apps, and other items — 100%

Some apps won't respond to scaling changes until you close and reopen them.

Display resolution — Adjust the resolution to fit your connected display — 1920 × 1080 (Recommended)

Display orientation — Landscape

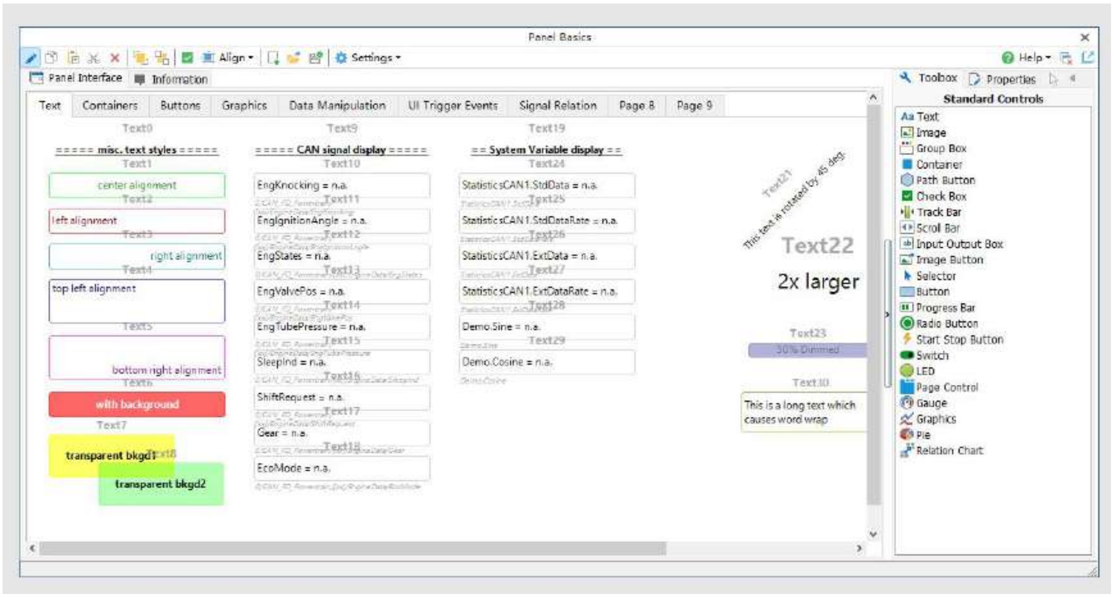
Multiple displays — Choose the presentation mode for your displays

Related settings

Advanced display — Display information, refresh rate

Graphics

As shown below, the scrollbars (both vertical and horizontal) have reappeared on the window:

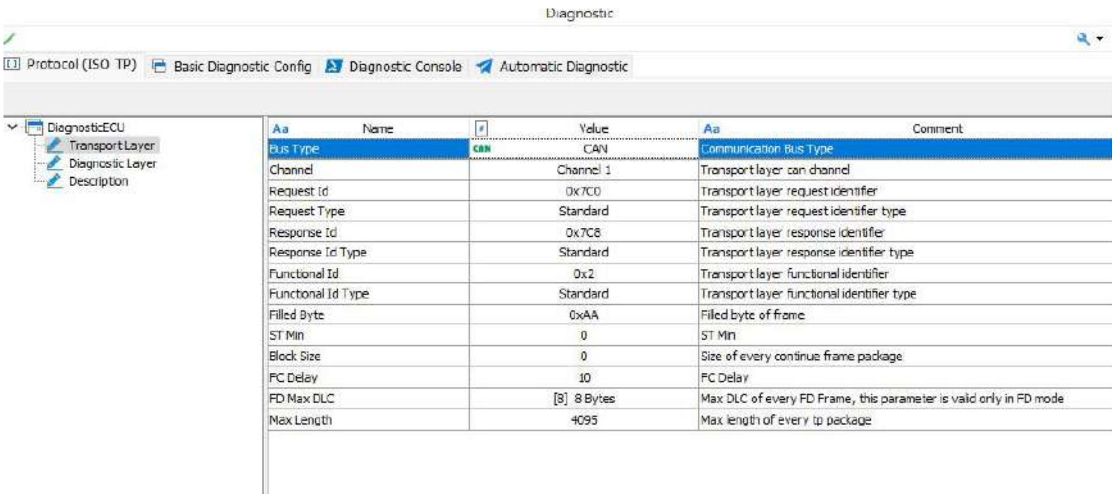

Panel Basics
Align
Settings
Help
Panel Interface
Information
Toolbox
Properties
Standard Controls
Text
Containers
Buttons
Graphics
Data Manipulation
UI Trigger Events
Signal Relation
Page 8
Page 9
===== misc. text styles =====
center alignment
left alignment
right alignment
top left alignment
bottom right alignment
with background
transparent bkgd1
transparent bkgd2
===== CAN signal display =====
EngKnocking = n.a.
EngIgnitionAngle = n.a.
EngStates = n.a.
EngValvePos = n.a.
EngTubePressure = n.a.
SleepInd = n.a.
ShiftRequest = n.a.
Gear = n.a.
EcoMode = n.a.
== System Variable display ==
StatisticsCAN1.StdData = n.a.
StatisticsCAN1.StdDataRate = n.a.
StatisticsCAN1.ExtData = n.a.
StatisticsCAN1.ExtDataRate = n.a.
Demo.Sine = n.a.
Demo.Cosine = n.a.
This text is rotated by 45 deg.
Text22
2x larger
50% Dimmed
This is a long text which causes word wrap
Text
Image
Group Box
Container
Path Button
Check Box
Track Bar
Scroll Bar
Input Output Box
Image Button
Selector
Button
Progress Bar
Radio Button
Start Stop Button
Switch
LED
Page Control
Gauge
Graphics
Pie
Relation Chart

1.19. CAN/CANFD Diagnosis (Diagnostic_CAN)

1.19.1. Diagnostic TP Parameter Configuration

TSMaster provides basic diagnostic console functionalities, allowing users to configure their own send and response requests as needed. Users can perform the following steps to operate:

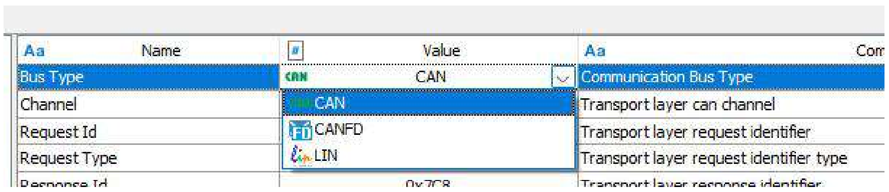
1.19.1.1. Transport Layer Parameters



Name	Value	Comment
Bus Type	CAN	Communication Bus Type
Channel	Channel 1	Transport layer can channel
Request Id	0x7C0	Transport layer request identifier
Request Type	Standard	Transport layer request identifier type
Response Id	0x7C8	Transport layer response identifier
Response Id Type	Standard	Transport layer response identifier type
Functional Id	0x2	Transport layer functional identifier
Functional Id Type	Standard	Transport layer functional identifier type
Filled Byte	0xAA	Filled byte of frame
ST Min	0	ST Min
Block Size	0	Size of every continue frame package
FC Delay	10	FC Delay
FD Max DLC	[8] 8 Bytes	Max DLC of every FD Frame, this parameter is valid only in FD mode
Max Length	4095	Max length of every tp package

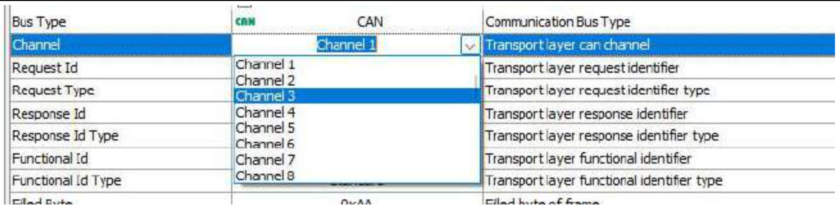
The explanations for each parameter are as follows:

➢ Bus Type: diagnostic transport layer type, currently, CAN/CANFD/LIN is supported, and in the future, it will support Ethernet and FlexRay, etc. User can choose from a drop-down list, as shown in the figure below:



Name	Value	Com
Bus Type	CAN	Communication Bus Type
Channel		Transport layer can channel
Request Id		Transport layer request identifier
Request Type		Transport layer request identifier type
Response Id	0x7C8	Transport layer response identifier

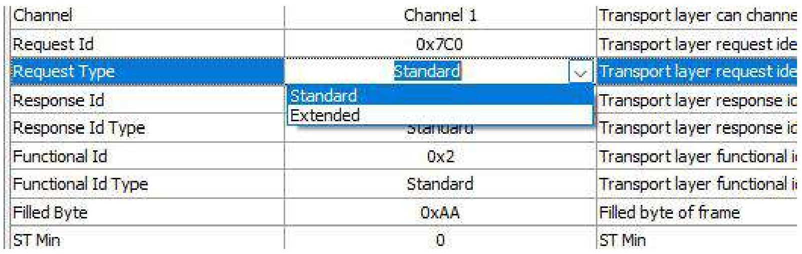
➢ Channel: the logical channels used by the diagnostic module. TSMaster supports multiple diagnostic modules working online simultaneously, and here it is used to select which logical channel of the system the current diagnostic module will use. Selection is made through a drop-down list, as shown in the figure below:



Bus Type	CAN	Communication Bus Type
Channel	Channel 1	Transport layer can channel
Request Id	Channel 1 Channel 2	Transport layer request identifier
Request Type	Channel 3	Transport layer request identifier type
Response Id	Channel 4	Transport layer response identifier
Response Id Type	Channel 5 Channel 6	Transport layer response identifier type
Functional Id	Channel 7	Transport layer functional identifier
Functional Id Type	Channel 8	Transport layer functional identifier type

➢ Setting the ID for diagnostic request, response, and functional frames of the diagnostic module PC tool end.

➢ Request Id Type/Response Id Type /Function Id Type: Setting the ID type for the diagnostic request, response, and functional frames on the PC tool end of the diagnostic module, whether it is a standard frame (11 bits) or an extended frame (29 bits), as shown in the figure below:



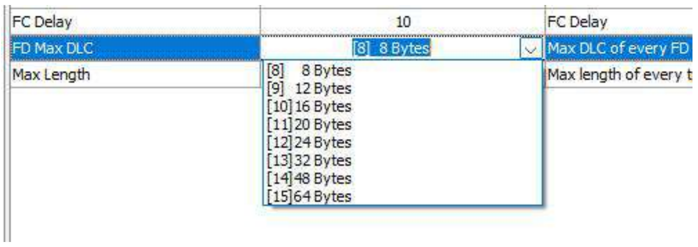
Channel	Channel 1	Transport layer can channe
Request Id	0x7C0	Transport layer request ide
Request Type	Standard	Transport layer request ide
Response Id	Standard	Transport layer response ic
Response Id Type	Extended	Transport layer response ic
Functional Id	0x2	Transport layer functional i
Functional Id Type	Standard	Transport layer functional i
Filled Byte	0xAA	Filled byte of frame
ST Min	0	ST Min

➢ Filled Byte: during the transmission process, when the actual effective bytes are less than a CAN message data frame, the remaining data segment is filled with padding bytes. For example, if a CAN message frame is 8 bytes long and the effective transmission bytes are 【0x02, 0x10, 0x02】, with the padding byte being 0xAA, then the actual message bytes are 【0x02, 0x10, 0x02, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA】.

➢ ST Min: shortest reception time interval. When TSMaster's diagnostic module acts as the receiving end, it supports the shortest time interval between diagnostic frames when receiving continuous frame messages. This parameter is the response to the diagnostic client. Setting it to 0 indicates support for receiving at the shortest possible time interval.

➢ Block Size: The size of the received Block. When TSMaster's diagnostic module, acting as the receiving end, receives continuous frame messages, it indicates the size of the data block that can be received at one time. This parameter is provided as a response to the diagnostic client. Setting it to 0 means that it can receive a data block of any size at once.

➢ FD Max DLC: when the transport layer is set to CAN FD, the maximum number of bytes that can be transmitted in a single frame is 64 bytes (DLC=15), but this parameter is adjustable, with the adjustment range as shown below:



FC Delay	10	FC Delay
FD Max DLC	[8] 8 Bytes	Max DLC of every FD
Max Length		Max length of every t

➢ Max Length: This parameter is meaningless for standard CAN/LIN. During multi-frame transmission, when the DLC length is 8 bytes, the first frame uses the low four bits of the 0th byte + the 8 bits of the first byte, totaling 12 bits to represent the size of the packet for a single transmission, which is up to 4095 bytes, as shown in the figure below:

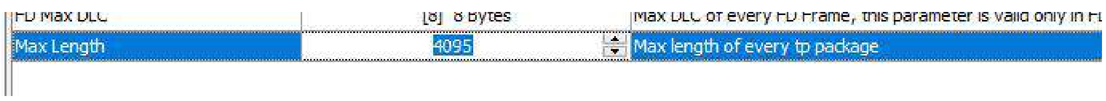
Byte0	Byte1	Byte2--7
0001	FF_DL(12 Bits)	Data segment

However, in FDCAN, when the DLC length is set to more than 8 bytes, more bits can be used to transmit information. Therefore, the transport layer of FDCAN supports the use of the 2nd, 3rd, 4th, and 5th bytes, totaling 32 bits, to transmit the length of a data block. This means that the transport layer of FDCAN can support the transmission of up to 4 gigabytes of data at one time. However, the exact amount supported is configurable by the user.

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6-Byte63
0001	000000000000	FF_DLC(32Bits)				Data segment

Comment: The high four bits of the first byte being set to 1 indicates that the frame is a first frame, which is the case for both FDCAN and Class CAN transport layers.

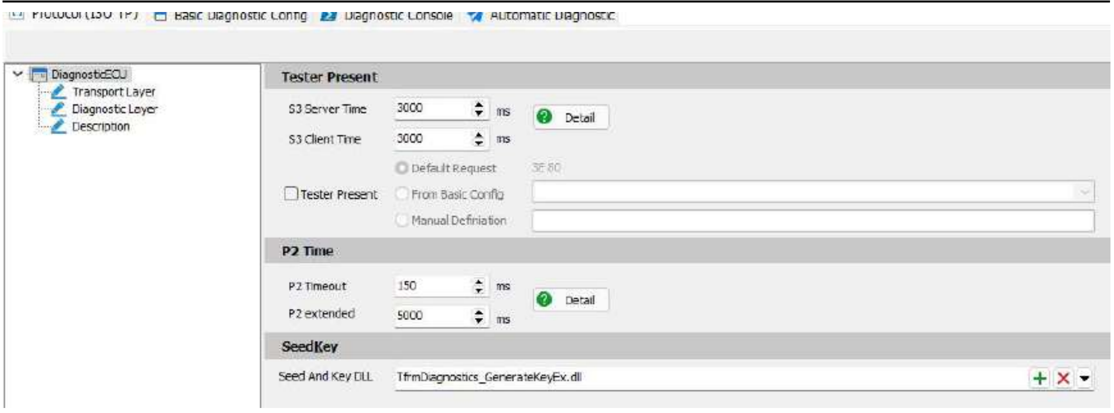
For example, as shown in the figure below, if configured for 4095 bytes, it will be the same as the standard transport layer. However, if configured for more than 4095 bytes, the extended transport layer using FD frames must be used.



FD Max DLC	[8] 8 Bytes	Max DLC of every FD Frame, this parameter is valid only in F
Max Length	4095	Max length of every tp package

1.19.1.2. Service Layer Parameters

Service layer parameters mainly include S3, P2 timing parameters, and the DLL for loading the SeedKey. As shown in the figure below:



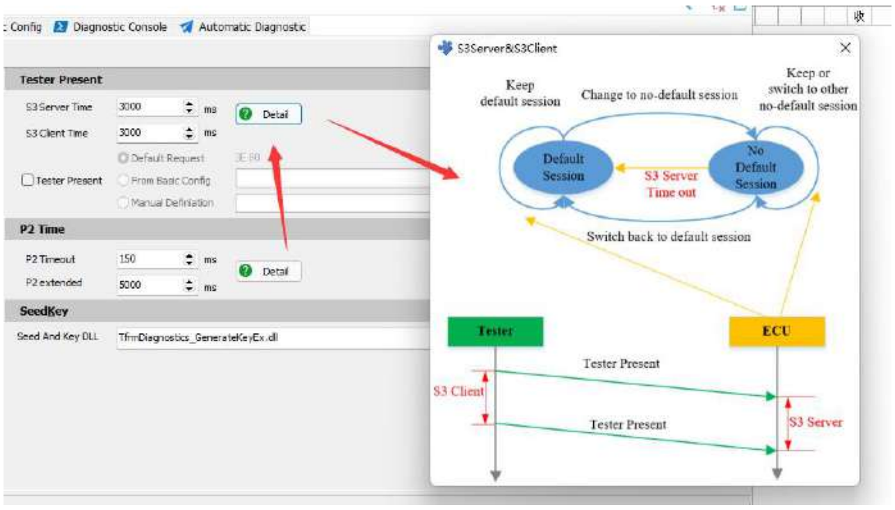
Tester Present
S3 Server Time 3000 ms
S3 Client Time 3000 ms
Tester Present [ ]
P2 Time
P2 Timeout 150 ms
P2 extended 5000 ms
SeedKey
Seed And Key DLL TfrmDiagnostics_GenerateKeyEx.dll

➢ S3 parameter: including S3 Server Time and S3 Client Time.

S3 Server Time: it indicates the timeout period after which the ECU will automatically switch back to the default session from another session.

S3 Client Time: it indicates the time interval at which the TesterPresent frame is sent when acting as the diagnostic Tester end.

The schematic diagram for the two parameters mentioned above can be viewed by clicking the Detail button. As shown in the figure below:



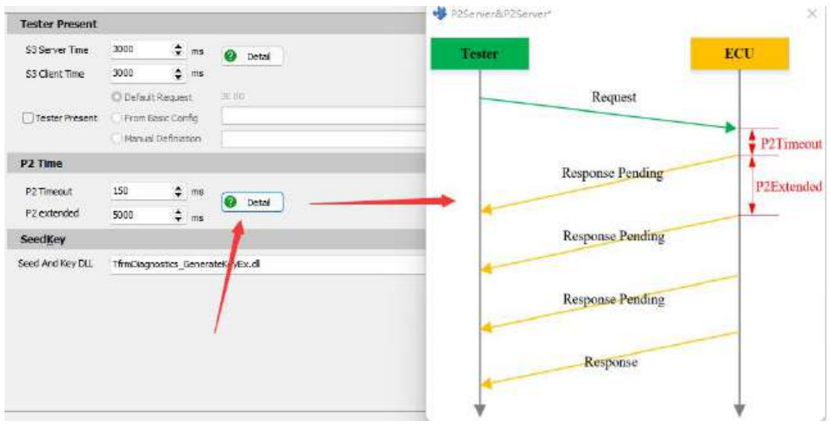
➢ P2 parameter: including P2 Timeout and P2 Extended parameters.

P2 Timeout: it indicates the shortest time interval after which the ECU must reply to a diagnostic request frame. For the diagnostic tool end, this parameter can be used as a timeout judgment parameter after sending a request. For example, if the diagnostic tool sends a diagnostic message and does not receive a reply within the P2Timeout period, it is considered that the request has failed and the process exits due to a timeout.

P2 Extended: when the diagnostic tool sends out a diagnostic message, if the ECU being tested does not have enough time to respond within the P2 Timeout period, it replies with a 7F XX

78 message, informing the diagnostic tool that it is unable to respond in time and that it needs to wait longer before replying. After the ECU sends the delay waiting message, the waiting time parameter is switched to P2Extended. The diagnostic tool's timeout judgment parameter needs to switch to P2Extended after receiving the delay waiting message.

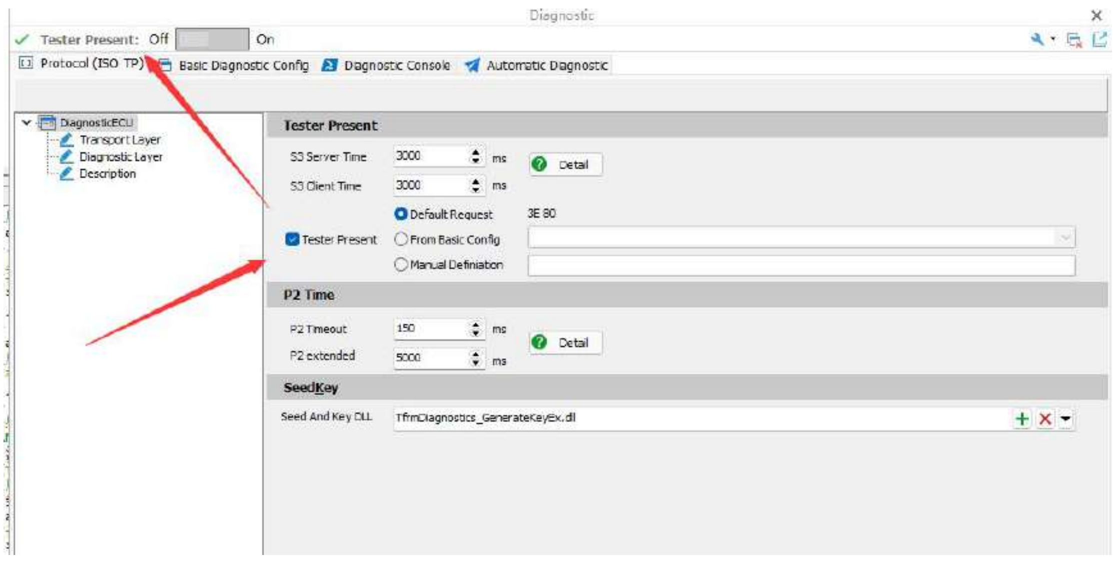
The schematic diagram for the two parameters mentioned above is shown below:



Tester Present
S3 Server Time 3000 ms
S3 Client Time 3000 ms
Default Request 3E 00
Tester Present
From Basic Config
Manual Definition
P2 Time
P2 Timeout 150 ms
P2 extended 5000 ms
SeedKey
Seed And Key DLL TfrmDiagnostics_GenerateKeyEx.dll

➢ Enable the Tester Present command:

In the TSMaster diagnostic module, you can choose to configure and enable the TSMaster Present command, as shown in the figure below:



Tester Present: Off On
Tester Present
S3 Server Time 3000 ms
S3 Client Time 3000 ms
[✓ Default Request] 3E 80
[✓ Tester Present]
From Basic Config
Manual Definiation
P2 Time
P2 Timeout 150 ms
P2 extended 5000 ms
SeedKey
Seed And Key DLL TfrmDiagnostics_GenerateKeyEx.dll

Once the command is enabled, a switch to start the Tester Present command will appear at the top of the module. When Tester Present is activated, the message will be sent at the interval set by the S3ClientTime.

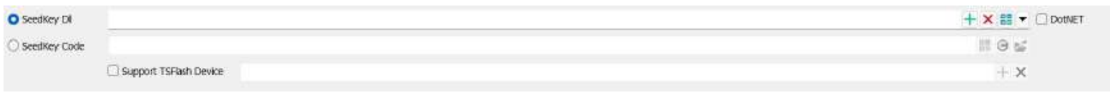
The bytes sent for Tester Present are optional and support three types:

【Default Request】: 0x3E 0x80, the most commonly used bytes.

【From Basic Config】: select the pre-configured 3E command from Basic Config【Manual Definition】: for custom byte.

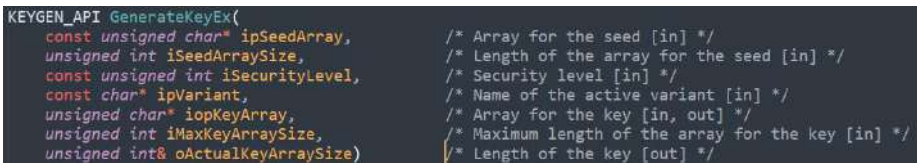
1.19.1.3. Seed&Key

TSMaster provides two methods for handling Seed & Key: the first is the most conventional method of loading the DLL functions of mainstream SeedKey, and the second is to provide a built-in interpreter for Seed & Key, where users can enter encryption and decryption algorithms. In the TP parameter configuration interface, users can choose whether to use the DLL loading method or directly edit the seed & Key source code, as shown below:



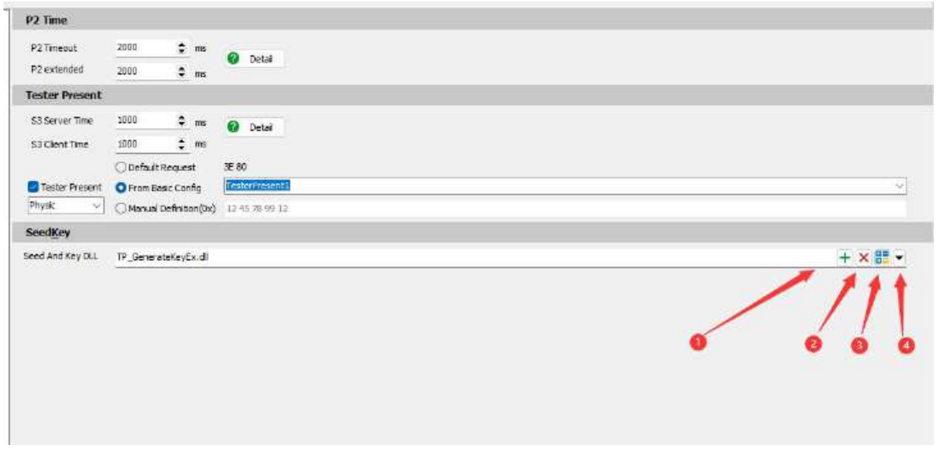
1.19.1.3.1. Load External Seed&Key DLL

During the diagnostic process, issues related to secure access, known as Seed & Key, are encountered. The TSMaster diagnostic module supports the loading of Seed & Key algorithms through a DLL, which is compatible with the calculation interface of mainstream tools. The interface definition is shown in the figure below:



```
KEYGEN_API GenerateKeyEx(
      const unsigned char* ipSeedArray,          /* Array for the seed [in] */
      unsigned int iSeedArraySize,               /* Length of the array for the seed [in] */
      const unsigned int iSecurityLevel,         /* Security level [in] */
      const char* ipVariant,                     /* Name of the active variant [in] */
      unsigned char* iopKeyArray,                /* Array for the key [in, out] */
      unsigned int iMaxKeyArraySize,             /* Maximum length of the array for the key [in] */
      unsigned int& oActualKeyArraySize)         /* Length of the key [out] */
```

The DLL loading interface is shown in the figure below:



P2 Time

P2 Timeout 2000 ms

P2 extended 2000 ms

Detail

Tester Present

S3 Server Time 1000 ms

S3 Client Time 1000 ms

Detail

[✓] Tester Present Physic

Default Request 3E 80

[✓ From Basic Config] TesterPresent1

Manual Definition(0x) 12 45 78 99 12

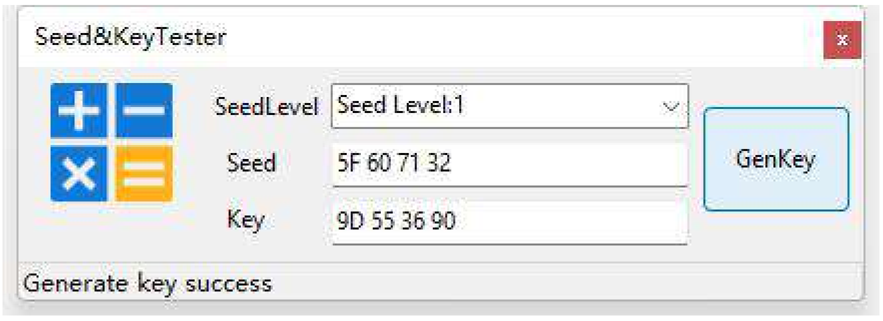
SeedKey

Seed And Key DLL TP_GenerateKeyEx.dll

【1】 Load DLL

【2】 Delete DLL

【3】 The DLL Verifier allows users to determine whether the loaded DLL interface is correct and whether the algorithm meets the design requirements, as shown in the figure below:



Seed&KeyTester

SeedLevel: Seed Level:1

Seed: 5F 60 71 32

Key: 9D 55 36 90

Generate key success

As shown in the interface above, after the user selects the SeedLevel, they enter the Demo Seed value and click on "GenKey" to make a judgment. If the DLL interface matches the template definition interface, it will output a prompt message: "Generate Key Success". Then the user compares the Key value with the target value to further confirm whether the algorithm in the DLL meets the design requirements.

【4】 Navigate to the path where the Seed & Key interface project is located in the TSMaster installation directory. Users can copy this project and add their own Seed & Key algorithms.

Default SeedKey Function Interface

Currently, for the diagnostic module of TSMaster to load the DLL directly, the DLL must implement one of the following three function interfaces:

【1】 Interface 1:

```
unsigned int GenerateKeyEx(
      const unsigned char* ipSeedArray,      /* Array for the seed [in] */
```

```
    unsigned int iSeedArraySize,         /* Length of the array for the seed [in] */
    const unsigned int iSecurityLevel,    /* Security level [in] */
    const char* ipVariant,               /* Name of the active variant [in] */
    unsigned char* iopKeyArray,           /* Array for the key [in, out] */
    unsigned int iMaxKeyArraySize,   /* Maximum length of the array for the key [in] */
    unsigned int& oActualKeyArraySize);   /* Length of the key [out] */
```

【2】 Interface 2:

```
unsigned int GenerateKeyExOpt(
    const unsigned char* ipSeedArray,      /* Array for the seed [in] */
    unsigned int iSeedArraySize,         /* Length of the array for the seed [in] */
    const unsigned int iSecurityLevel,    /* Security level [in] */
    const char* ipVariant,               /* Name of the active variant [in] */
    const char* iPara,                  /* */
    unsigned char* iopKeyArray,           /* Array for the key [in, out] */
    unsigned int iMaxKeyArraySize,   /* Maximum length of the array for the key [in] */
    unsigned int& oActualKeyArraySize)    /* Length of the key [out] */
```

【3】 Interface 3:

```
bool ASAP1A_CCP_ComputeKeyFromSeed(
    const unsigned char* ipSeedArray,      /* Array for the seed [in] */
    unsigned short iSeedArraySize,         /* Length of the array for the seed [in] */
    unsigned char* iopKeyArray,            /* Array for the key [in, out] */
    unsigned short iMaxKeyArraySize, /* Maximum length of the array for the key [in] */
    unsigned short* opSizeKey)             /* Length of the key [out] */
```

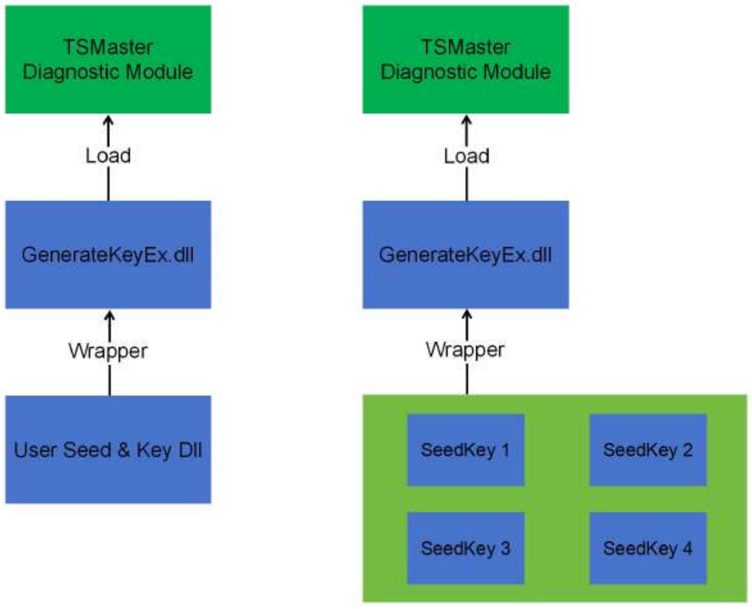
As long as the user's DLL implements any of the above function interfaces, it can be directly loaded into the TP layer module. If there is a failure in loading, the main checking points are:

1. Whether the DLL is published in Release Mode. If the publish mode is Debug, there is often a situation where there is a dependency on debugging DLLs.

2. Whether the target platform of the DLL is x86. Currently, TSMaster supports x86 DLL, therefore, the target platform of the DLL must be x86.

How to be Compatible with Other Function Interfaces

In daily use, it often happens that users have developed a DLL, but the interface of this DLL is not any of the three interfaces mentioned above, which makes it impossible to load directly into the diagnostic module of TSMaster. For this situation, the following solutions are recommended to solve the issue:



Here is an example to shown how to make a user's existing DLL file compatible.

1. The user's existing DLL, named UserSeedKey.dll, contains the following internal API functions:

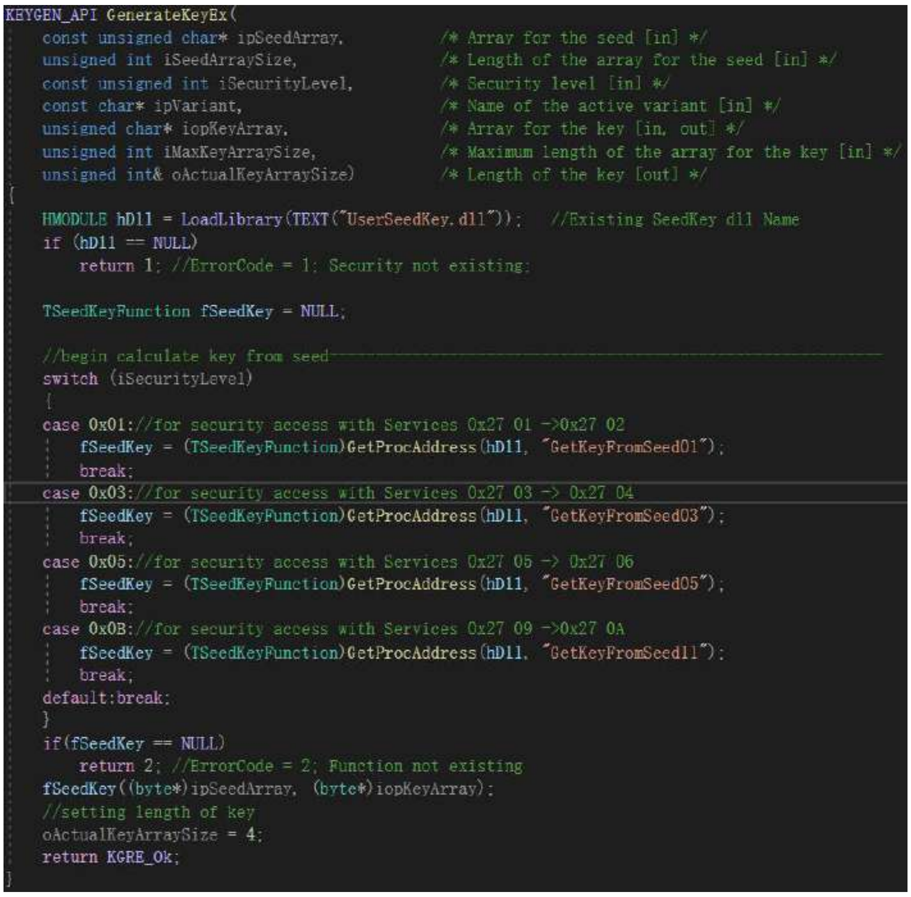
- When the Seed level is 1, call function void GetKeyFromSeed01(byte* ASeed, byte* AKey);
- When the Seed level is 3, call function void GetKeyFromSeed03(byte* ASeed, byte* AKey);
- When the Seed level is 11, call function void GetKeyFromSeed11(byte* ASeed, byte* AKey);

This DLL does not support the aforementioned default loading interfaces and cannot be directly loaded into TSMaster for use. Therefore, it is necessary to encapsulate these DLLs with an additional layer so that they can be loaded into the diagnostic module of TSMaster.

2. Select the GenerateKeyEx template project provided in the TSMaster installation directory and call the function interfaces of the aforementioned DLL within that project. The basic idea is:

- Use Loadlibrary to load the user’s existing dll dynamically.
- Based on the incoming Level parameter, use the GetProcAddress function to dynamically obtain the actual function pointer used for calculating the Key.
- If the function pointer is successfully obtained, then use this function pointer to transmit the Seed value and calculate the corresponding Key value.

A detailed example of the function call is shown in the figure below:



```
KEYGEN_API GenerateKeyEx(
    const unsigned char* ipSeedArray,          /* Array for the seed [in] */
    unsigned int iSeedArraySize,               /* Length of the array for the seed [in] */
    const unsigned int iSecurityLevel,         /* Security level [in] */
    const char* ipVariant,                     /* Name of the active variant [in] */
    unsigned char* iopKeyArray,                /* Array for the key [in, out] */
    unsigned int iMaxKeyArraySize,             /* Maximum length of the array for the key [in] */
    unsigned int& oActualKeyArraySize)         /* Length of the key [out] */
{
    HMODULE hDll = LoadLibrary(TEXT("UserSeedKey.dll"));   //Existing SeedKey dll Name
    if (hDll == NULL)
        return 1; //ErrorCode = 1: Security not existing;

    TSeedKeyFunction fSeedKey = NULL;

    //begin calculate key from seed------------------------------------------------------------
    switch (iSecurityLevel)
    {
    case 0x01://for security access with Services 0x27 01 ->0x27 02
        fSeedKey = (TSeedKeyFunction)GetProcAddress(hDll, "GetKeyFromSeed01");
        break;
    case 0x03://for security access with Services 0x27 03 -> 0x27 04
        fSeedKey = (TSeedKeyFunction)GetProcAddress(hDll, "GetKeyFromSeed03");
        break;
    case 0x05://for security access with Services 0x27 05 -> 0x27 06
        fSeedKey = (TSeedKeyFunction)GetProcAddress(hDll, "GetKeyFromSeed05");
        break;
    case 0x0B://for security access with Services 0x27 09 ->0x27 0A
        fSeedKey = (TSeedKeyFunction)GetProcAddress(hDll, "GetKeyFromSeed11");
        break;
    default:break;
    }
    if(fSeedKey == NULL)
        return 2; //ErrorCode = 2: Function not existing
    fSeedKey((byte*)ipSeedArray, (byte*)iopKeyArray);
    //setting length of key
    oActualKeyArraySize = 4;
    return KGRE_Ok;
}
```

3. After the development of the GenerateKeyEx project is completed, TSMaster directly loads the DLL where GenerateKeyEx belongs to. It is important to note that users need to copy the existing DLLs, such as UserSeedKey.dll, to the TSMaster root directory or the directory where GenerateKeyEx.dll is located. If they are not copied, GenerateKeyEx.dll will fail to find the corresponding dependent DLL when executed, resulting in an unlock failure.

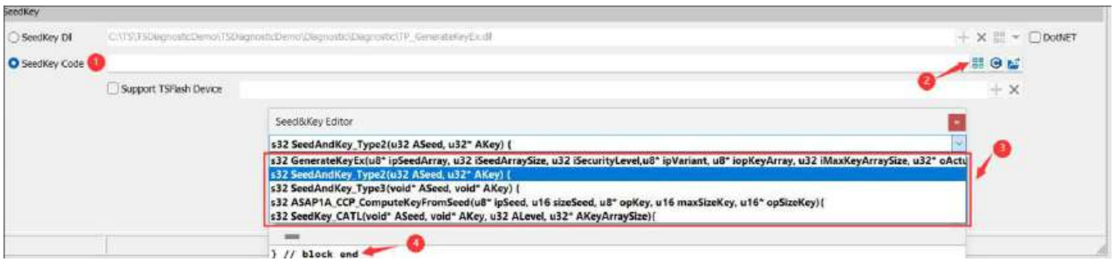
Summary:

In the TSMaster installation directory, there are template projects provided for encapsulating Seed & Key algorithms. Such as GenerateKeyEx, GenerateKeyExOpt, ASAP1A_CCP_ComputeKeyFromSeed, users can develop based on these template projects to obtain DLL functions that can be directly loaded.

Additionally, there are projects provided for secondary encapsulation of DLLs, such as GenerateKeyEx_Wrapper_Demo. This project demonstrates the process of wrapping an existing SeedKey algorithm library to generate a DLL that can be directly loaded into the TSMaster diagnostic module.

1.19.1.3.2. Utilizing the Built-in Algorithm Editor

The basic steps are as follows:


SeedKey
SeedKey Dll
C:\TS\TSDiagnosticDemo\TSDiagnosticDemo\Diagnostic\Diagnostic\TP_GenerateKeyEx.dll
DotNET
SeedKey Code
1
2
Support TSFlash Device
Seed&Key Editor
3
4

```
s32 SeedAndKey_Type2(u32 ASeed, u32* AKey) {
s32 GenerateKeyEx(u8* ipSeedArray, u32 iSeedArraySize, u32 iSecurityLevel,u8* ipVariant, u8* iopKeyArray, u32 iMaxKeyArraySize, u32* oActu
s32 SeedAndKey_Type2(u32 ASeed, u32* AKey) {
s32 SeedAndKey_Type3(void* ASeed, void* AKey) {
s32 ASAP1A_CCP_ComputeKeyFromSeed(u8* ipSeed, u16 sizeSeed, u8* opKey, u16 maxSizeKey, u16* opSizeKey){
s32 SeedKey_CATL(void* ASeed, void* AKey, u32 ALevel, u32* AKeyArraySize){
} // block end
```

1. Select the method for encryption and decryption algorithms.

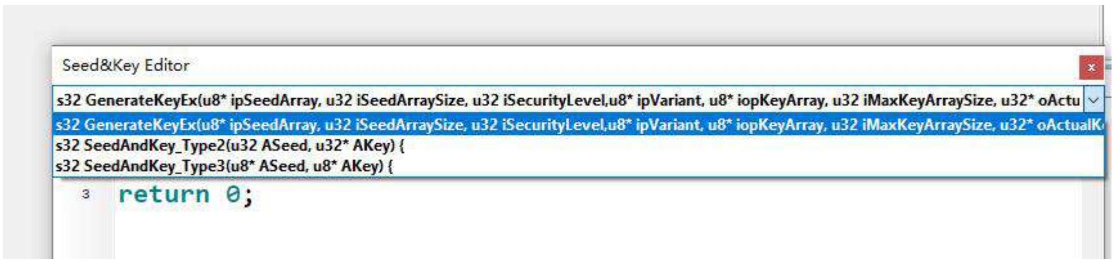
2. Expand the algorithm editor.

3. Select a function interface (the function is expandable).

4. Enter the corresponding encryption and decryption algorithm in the algorithm editor.

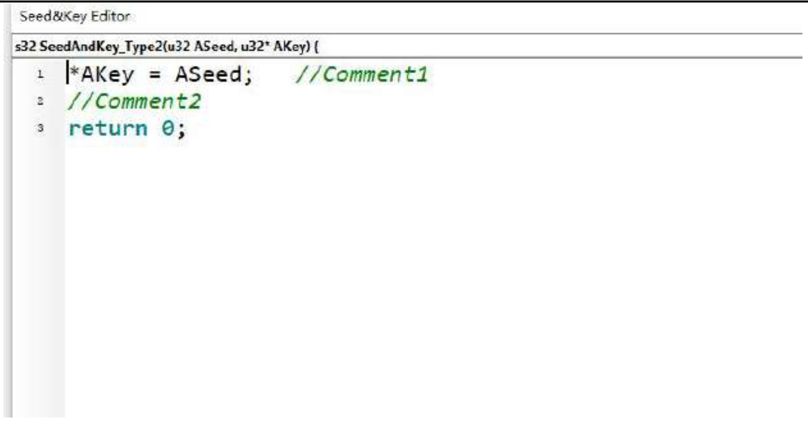
Precautions:

【1】 For the algorithm function interface, TSMaster currently provides the most commonly used interface forms. If users have their own special interface forms that it is not covered, please contact TOSUN Technology Ltd, Shanghai to add their interface to the options.



```
s32 GenerateKeyEx(u8* ipSeedArray, u32 iSeedArraySize, u32 iSecurityLevel,u8* ipVariant, u8* iopKeyArray, u32 iMaxKeyArraySize, u32* oActu
s32 GenerateKeyEx(u8* ipSeedArray, u32 iSeedArraySize, u32 iSecurityLevel,u8* ipVariant, u8* iopKeyArray, u32 iMaxKeyArraySize, u32* oActualK
s32 SeedAndKey_Type2(u32 ASeed, u32* AKey) {
s32 SeedAndKey_Type3(u8* ASeed, u8* AKey) {
return 0;
```

【2】 All interface functions define a return value `s32`. This constraint is added primarily to increase the rigor of the functions. A return value of 0 indicates success, while other values correspond to specific error codes. Users must not forget to enter a return value when editing code in the last line, otherwise, the system will consider the algorithm execution fails and will not proceed with subsequent executions. As shown below:



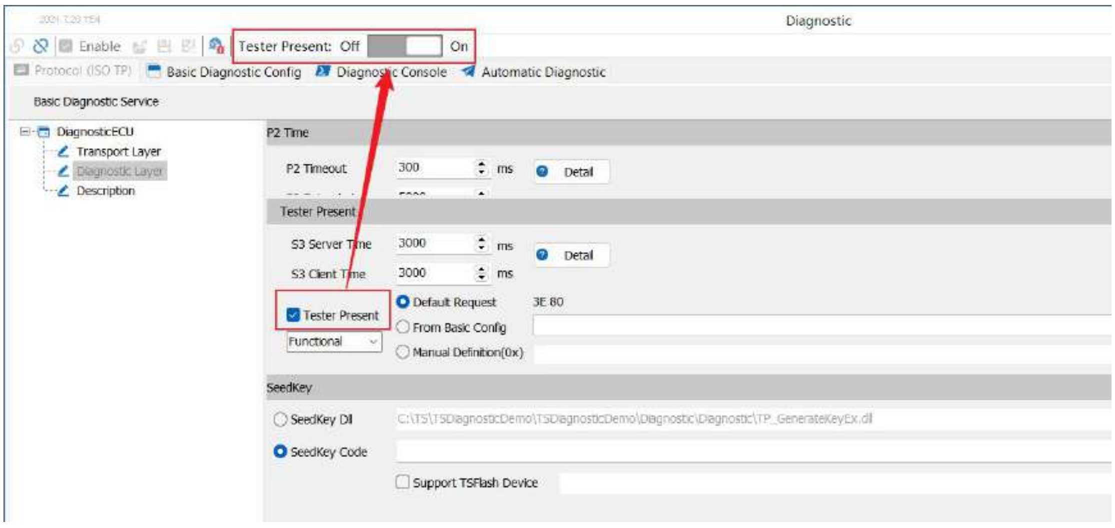
s32 SeedAndKey_Type2(u32 ASeed, u32* AKey) {

```
*AKey = ASeed;    //Comment1
//Comment2
return 0;
```

【3】 After adding the algorithm, click OK to exit. The built-in compiler of TSMaster will automatically interpret the algorithm and prepare it for use in the diagnostic execution process.

1.19.1.4. TesterPresent

In the Diagnostic Tp parameter configuration, enable TesterPresenter, and TSMaster will provide a global switch. Users can turn the Tester Present command on and off directly through this switch, as shown in the figure below:



P2 Time

P2 Timeout 300 ms

Tester Present

S3 Server Time 3000 ms

S3 Client Time 3000 ms

[✓ Tester Present]

Functional

[✓ Default Request] 3E 80

From Basic Config

Manual Definition(0x)

SeedKey

SeedKey Dll C:\TS\TSDiagnosticDemo\TSDiagnosticDemo\Diagnostic\Diagnostic\TP_GenerateKeyEx.dll

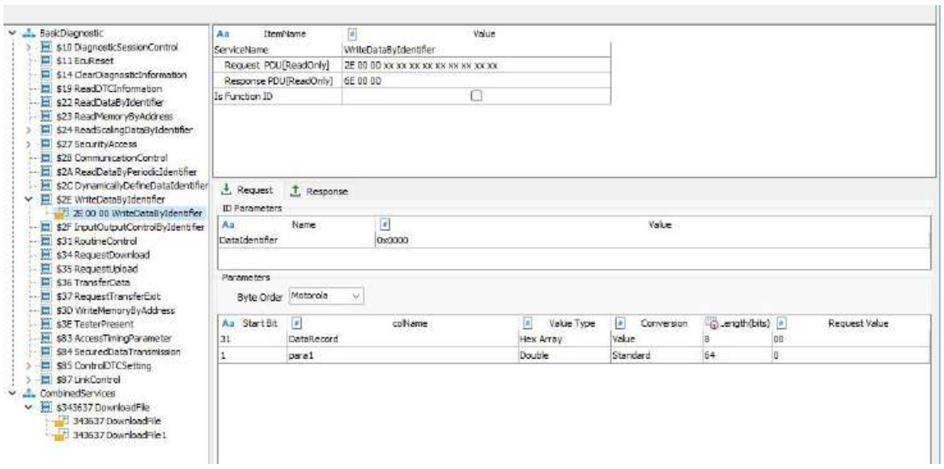
[✓ SeedKey Code]

Support TSFlash Device

In addition to the global switch, if users want more flexible control over the enabling and disabling of the Tester Present command, in subsequent steps of the automated process, TSMaster also provides a step-based way to configure this command, allowing users to choose when to activate and deactivate the Tester Present command at the required steps.

1.19.2. Basic Diagnostic Configuration

This module includes BasicDiagnostic parameters and CombinedService. For commands that are completely independent in their execution process, they are placed in BasicDiagnostic; for commands that require a combination of multiple commands to be completed, they are placed in CombinedService. As shown in the figure below:



- BasicDiagnostic
  - $10 DiagnosticSessionControl
  - $11 EcuReset
  - $14 ClearDiagnosticInformation
  - $19 ReadDTCInformation
  - $22 ReadDataByIdentifier
  - $23 ReadMemoryByAddress
  - $24 ReadScalingDataByIdentifier
  - $27 SecurityAccess
  - $28 CommunicationControl
  - $2A ReadDataByPeriodicIdentifier
  - $2C DynamicallyDefineDataIdentifier
  - $2E WriteDataByIdentifier
    - 2E 00 00 WriteDataByIdentifier
  - $2F InputOutputControlByIdentifier
  - $31 RoutineControl
  - $34 RequestDownload
  - $35 RequestUpload
  - $36 TransferData
  - $37 RequestTransferExit
  - $3D WriteMemoryByAddress
  - $3E TesterPresent
  - $83 AccessTimingParameter
  - $84 SecuredDataTransmission
  - $85 ControlDTCSetting
  - $87 LinkControl
- CombinedServices
  - $343637 DownloadFile
    - 343637 DownloadFile
    - 343637 DownloadFile1

ItemName	Value
ServiceName	WriteDataByIdentifier
Request PDU[ReadOnly]	2E 00 00 xx xx xx xx xx xx xx xx xx
Response PDU[ReadOnly]	6E 00 00
Is Function ID	☐

Request | Response

ID Parameters

Name	Value
DataIdentifier	0x0000

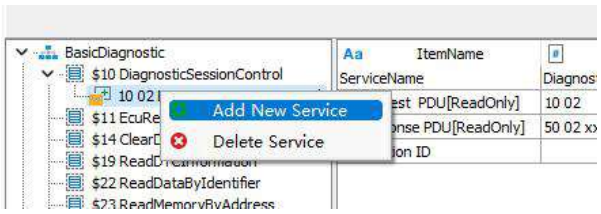
Parameters

Byte Order: Motorola

Start Bit	colName	Value Type	Conversion	Length(bits)	Request Value
31	DataRecord	Hex Array	Value	8	00
1	para1	Double	Standard	64	0

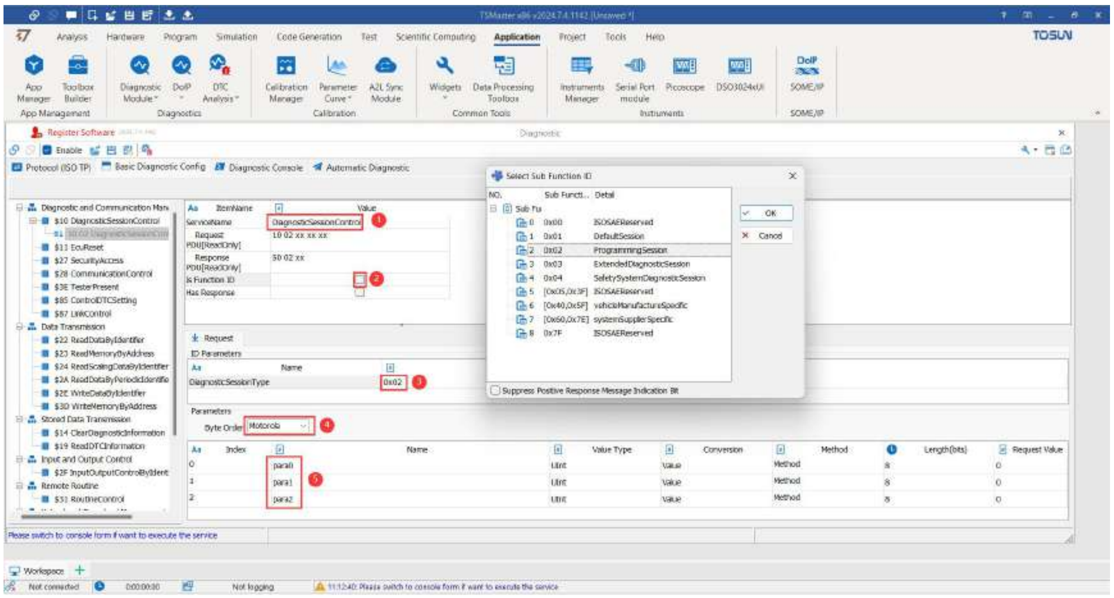
1.19.2.1. Add/Delete Service Commands

Place the mouse over the service command where you want to add or delete, right-click to expand, and select whether to add or remove the service, as shown in the figure below:



1.19.2.2. Configure BasicDiagnostic Service Parameters

Taking Session Control as an example, the main configuration parameters include:



ItemName	Value
ServiceName	DiagnosticSessionControl
Request PDU[ReadOnly]	10 02 xx xx xx
Response PDU[ReadOnly]	50 02 xx
Is Function ID	☐
Has Response	☐

Select Sub Function ID

NO.	Sub Functi...	Detail
0	0x00	ISOSAEReserved
1	0x01	DefaultSession
2	0x02	ProgrammingSession
3	0x03	ExtendedDiagnosticSession
4	0x04	SafetySystemDiagnosticSession
5	[0x05,0x3F]	ISOSAEReserved
6	[0x40,0x5F]	vehicleManufactureSpecific
7	[0x60,0x7E]	systemSupplierSpecific
8	0x7F	ISOSAEReserved

Suppress Positive Response Message Indication Bit

ID Parameters

Name	
DiagnosticSessionType	0x02

Parameters

Byte Order: Motorola

Index	Name	Value Type	Conversion	Method	Length(bits)	Request Value
0	para0	UInt	Value	Method	8	0
1	para1	UInt	Value	Method	8	0
2	para2	UInt	Value	Method	8	0

Please switch to console form if want to execute the service

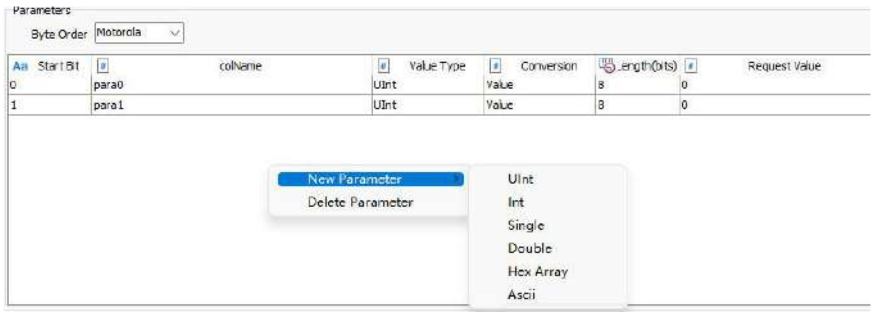
【1】 Service name for configuration: users can configure a service name that is easy to understand and manage.

【2】 Is Function ID: whether this diagnostic service uses Functional ID to send diagnostic requests.

【3】 Select the Sub Function ID: for example, in Session Control, the DiagnosticSessionType includes the types of Sessions as shown in the above figure.

【4】 Byte order in the parameter list: supports Motorola and Intel byte order.

【5】 Parameter List: in addition to the diagnostic ID and Sub Function ID, the diagnostic service can also send parameters to the target ECU. The parameter list includes the parameter list for both request and response frames. The configuration method is shown below, where users can choose to add/delete various types of parameters.



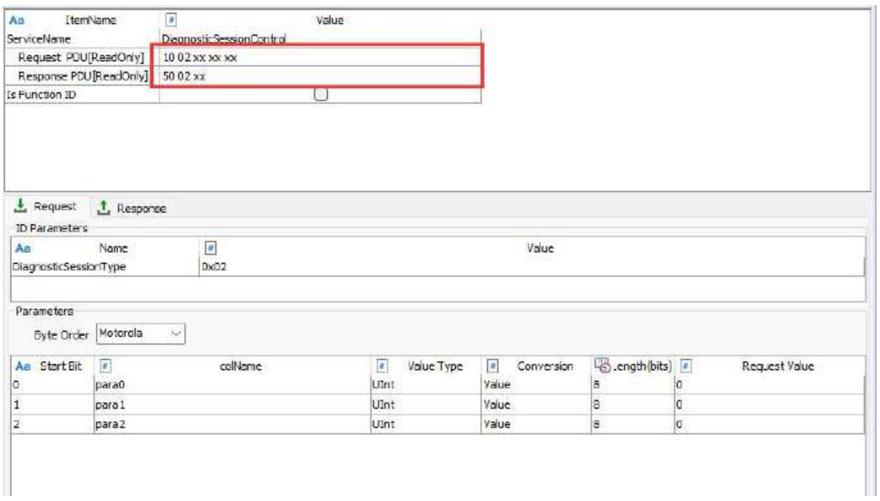
Parameters

Byte Order: Motorola

Start Bit	colName	Value Type	Conversion	Length(bits)	Request Value
0	para0	UInt	Value	8	0
1	para1	UInt	Value	8	0

The Service ID and Sub-Service Type ID, such as the DiagnosticSessionType parameter in Session Control, are mandatory, while the parameter list is optional.

After modifying the configuration, a real-time example of the actual diagnostic message will be displayed at the top of the interface, as shown in the figure below. After completing the configuration as shown, the diagnostic message that the diagnostic device is going to send is: 【10 02 xx xx xx】: xx represents a variable parameter, determined by the data actually entered by the user; the affirmative response message that the diagnostic device is going to receive is 【50 02 xx】.



ItemName	Value
ServiceName	DiagnosticSessionControl
Request PDU[ReadOnly]	10 02 xx xx xx
Response PDU[ReadOnly]	50 02 xx
Is Function ID	☐

Request | Response

ID Parameters

Name	Value
DiagnosticSessionType	0x02

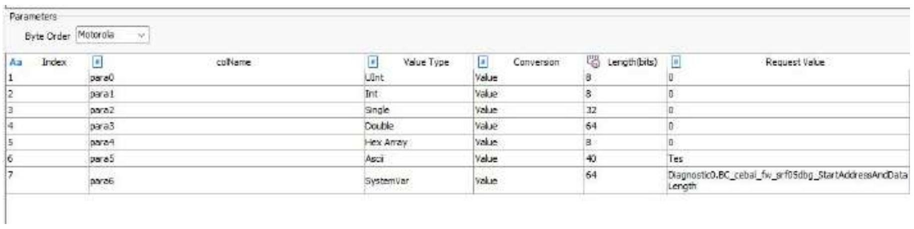
Parameters

Byte Order: Motorola

Start Bit	colName	Value Type	Conversion	Length(bits)	Request Value
0	para0	UInt	Value	8	0
1	para1	UInt	Value	8	0
2	para2	UInt	Value	8	0

1.19.2.2.1. Diagnostic Service Parameters

The diagnostic module parameters support seven types of data types, including: UInt， Int， Single， Double， HexArray， Ascii, and SystemVar.



Parameters

Byte Order: Motorola

Index	colName	Value Type	Conversion	Length(bits)	Request Value
1	para0	UInt	Value	8	0
2	para1	Int	Value	8	0
3	para2	Single	Value	32	0
4	para3	Double	Value	64	0
5	para4	Hex Array	Value	8	0
6	para5	Ascii	Value	40	Tes
7	para6	SystemVar	Value	64	Diagnostic0.BC_cebal_fw_srf05dbg_StartAddressAndDataLength

【1】 UInt: unsigned Integer, its data length must be less than 32 bits and a multiple of 8, and can be 8, 16, 24, 32 bits.

【2】 Int: signed Integer, its data length must be less than 32 bits and a multiple of 8, and can be 8, 16, 24, 32 bits.

【3】 Single: single-precision floating-point number, with a fixed data length of 32 bits. Users directly input and output floating-point data.

【4】 Double: Double-precision floating-point number, with a fixed data length of 64 bits.

Users directly input and output floating-point data.

【5】 Hex Array: A hexadecimal array, where the data length must be a multiple of 8. The input data must conform to hexadecimal data types.

【6】 ASCII: ASCII string, with a data length that is a multiple of 8. The input data is an array of ASCII characters, which is then converted into hexadecimal for transmission.

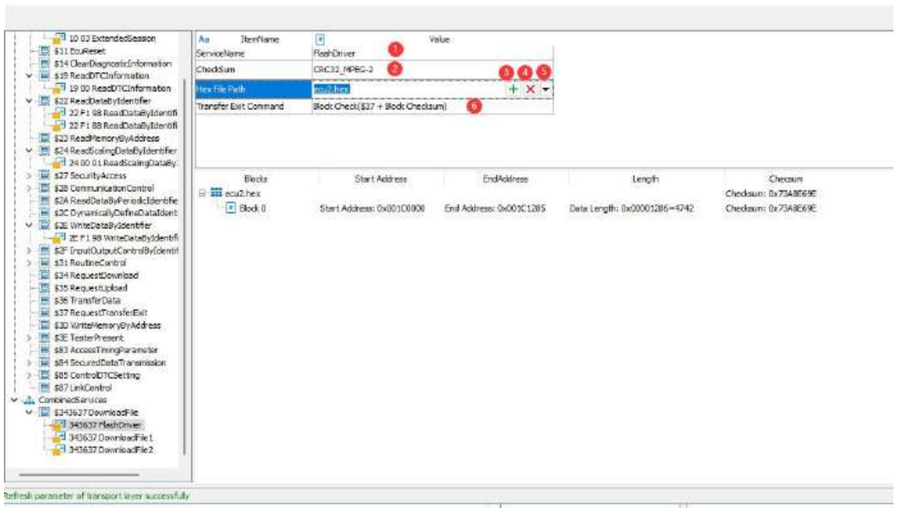
【7】 SystemVar: System variable, with a data length that is a multiple of 8. TSMaster system variables can support various data types such as Uint, Int, Single, Double, UintArray, DoubleArray, HexArray, String, etc. The specific data type is determined by the definition of the system variable itself.

For details on the effects of converting the aforementioned seven types of input data into actual sent byte data, please refer to 1.19.3.3.1 Input Diagnostic Parameters.

1.19.2.3. Configure CombinedServices Message

1.19.2.3.1. Download File

Combined services currently only support the download file service. If users have other combination requirements, they can provide feedback to TOSUN Technology Ltd, Shanghai. Reasonable requirements can be added to the software as standard service modules.



ItemName	Value
ServiceName	FlashDriver
CheckSum	CRC32_MPEG-2
Hex File Path	ecu2.hex
Transfer Exit Command	Block Check($37 + Block Checksum)

Blocks	Start Address	EndAddress	Length	Checsum
ecu2.hex				Checksum: 0x73A8E69E
Block 0	Start Address: 0x001C0000	End Address: 0x001C1285	Data Length: 0x00001286=4742	Checksum: 0x73A8E69E

Refresh parameter of transport layer successfully

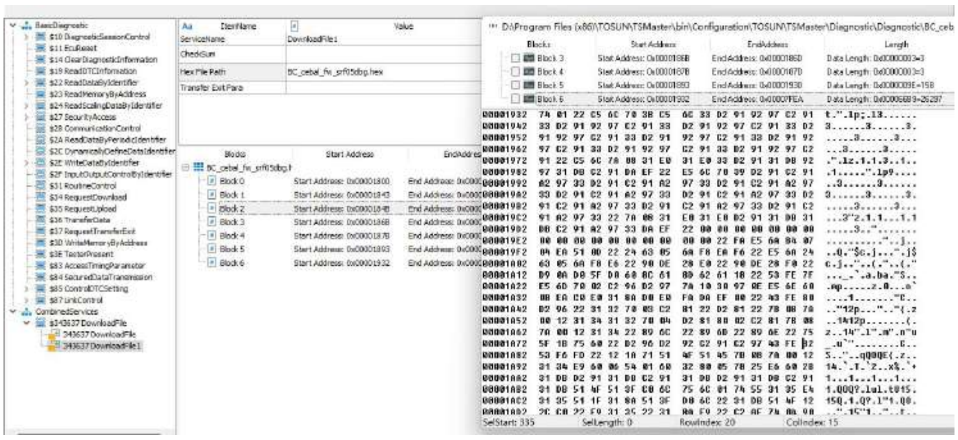
【1】 Configure the name of the service.

【2】 Select the file for CRC verification. Detailed introduction to CRC verification will be provided in the subsequent chapters.

【3】 Load the executable file. TSMaster supports the loading of Hex, S19, Mot, and bin files. After loading, the sections, addresses, lengths, and other information contained in the file can be seen at the bottom of the interface.

【4】 Delete the executable file.

【5】 Open hex viewer. TSMaster has a built-in executable file viewer and editor called TSHexViewer, as shown in the figure below. Users can use this tool to view detailed information of the loaded Hex files.



ItemName	Value
ServiceName	DownloadFile1
CheckSum	
Hex File Path	BC_cebal_fw_srf05dbg.hex
Transfer Exit Para	

Blocks	Start Address	EndAddress
BC_cebal_fw_srf05dbg.h		
Block 0	Start Address: 0x00001800	End Address: 0x000
Block 1	Start Address: 0x00001843	End Address: 0x000
Block 2	Start Address: 0x0000184B	End Address: 0x000
Block 3	Start Address: 0x0000186B	End Address: 0x000
Block 4	Start Address: 0x0000187B	End Address: 0x000
Block 5	Start Address: 0x00001893	End Address: 0x000
Block 6	Start Address: 0x00001932	End Address: 0x000

D:\Program Files (x86)\TOSUN\TSMaster\bin\Configuration\TOSUN\TSMaster\Diagnostic\Diagnostic\BC_ceb

Blocks	Start Address	EndAddress	Length
Block 3	Start Address: 0x0000186B	EndAddress: 0x0000186D	Data Length: 0x00000003=3
Block 4	Start Address: 0x0000187B	EndAddress: 0x0000187D	Data Length: 0x00000003=3
Block 5	Start Address: 0x00001893	EndAddress: 0x00001930	Data Length: 0x0000009E=158
Block 6	Start Address: 0x00001932	EndAddress: 0x00007FEA	Data Length: 0x00006689=26297

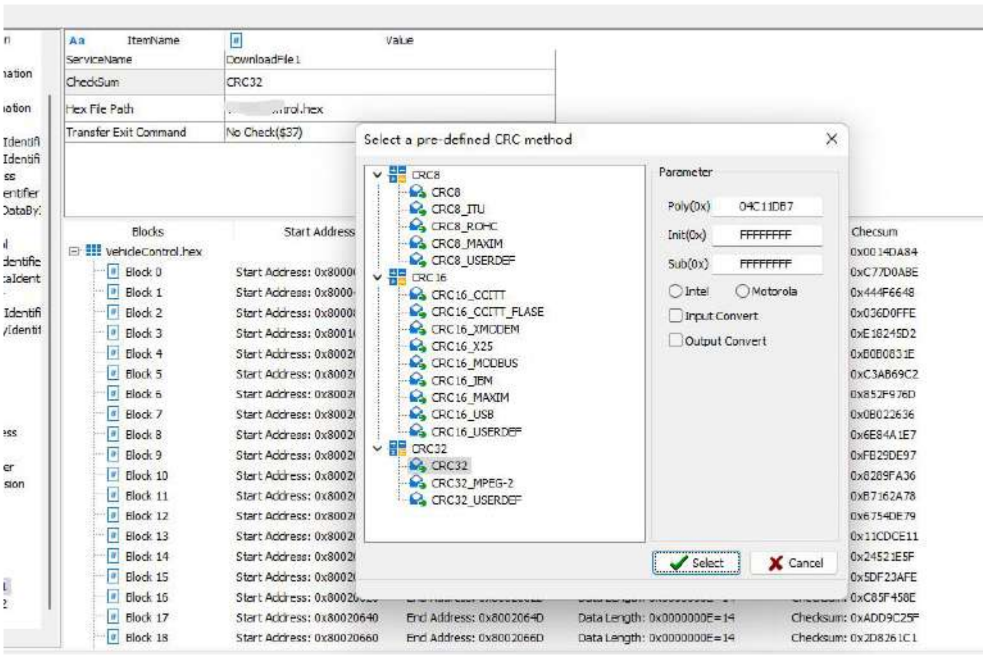
```
00001932  74 01 22 C5 6C 70 3B C5  6C 33 D2 91 92 97 C2 91  t.".lp;.l3......
00001942  33 D2 91 92 97 C2 91 33  D2 91 92 97 C2 91 33 D2  3......3......3.
00001952  91 92 97 C2 91 33 D2 91  92 97 C2 91 33 D2 91 92  .....3......3...
00001962  97 C2 91 33 D2 91 92 97  C2 91 33 D2 91 92 97 C2  ...3......3.....
00001972  91 22 C5 6C 7A 08 31 E0  31 E0 33 D2 91 31 D8 92  .".lz.1.1.3..1..
00001982  97 31 D8 C2 91 DA EF 22  E5 6C 70 39 D2 91 C2 91  .1.....".lp9....
00001992  A2 97 33 D2 91 C2 91 A2  97 33 D2 91 C2 91 A2 97  ..3......3......
000019A2  33 D2 91 C2 91 A2 97 33  D2 91 C2 91 A2 97 33 D2  3......3......3.
000019B2  91 C2 91 A2 97 33 D2 91  C2 91 A2 97 33 D2 91 C2  .....3......3...
000019C2  91 A2 97 33 22 7A 08 31  E0 31 E0 D2 91 31 D8 31  ...3"z.1.1...1.1
000019D2  D8 C2 91 A2 97 33 DA EF  22 00 00 00 00 00 00 00  .....3.."......
000019E2  00 00 00 00 00 00 00 00  00 00 22 FA E5 6A 84 07  ..........."..j..
000019F2  04 EA 51 0D 22 24 63 05  6A F8 EA F6 22 E5 6A 24  ..Q."$c.j...".j$
00001A02  63 05 6A F8 E6 22 90 DE  28 E0 22 90 DE 28 F0 22  c.j.."..(."..(."
00001A12  D9 0A D8 5F DA 60 8C 61  80 62 61 18 22 53 FE 7F  ..._.`.a.ba."S..
00001A22  E5 6D 70 02 C2 96 D2 97  7A 10 30 97 0E E5 6E 60  .mp.....z.0...n`
00001A32  08 EA C0 E0 31 8A D0 E0  FA DA EF 00 22 43 FE 80  ....1......."C..
00001A42  D2 96 22 31 32 70 03 C2  81 22 D2 81 22 7B 08 7A  .."12p..."..."{.z
00001A52  00 12 31 34 31 32 70 04  D2 81 80 02 C2 81 7B 08  ..1412p.......{.
00001A62  7A 00 12 31 34 22 89 6C  22 89 6D 22 89 6E 22 75  z..14".l".m".n"u
00001A72  5F 1B 75 60 22 D2 96 D2  92 C2 91 C2 97 43 FE 82  _.u`"........C..
00001A82  53 F6 FD 22 12 1A 71 51  4F 51 45 7B 08 7A 00 12  S.."..qQOQE{.z..
00001A92  31 34 E9 60 06 54 01 60  32 80 05 78 25 E6 60 28  14.`.T.`2..x%.`+
00001AA2  31 DB D2 91 31 DB C2 91  31 DB D2 91 31 DB C2 91  1...1...1...1...
00001AB2  31 DB 51 4F 51 3F C0 6C  75 6C 01 74 55 31 35 E4  1.QOQ?.lul.tU15.
00001AC2  31 35 51 1F 31 8A 51 3F  D8 6C 22 31 DB 51 4F 12  15Q.1.Q?.l"1.QO.
```

SelStart: 335　SelLength: 0　RowIndex: 20　ColIndex: 15

【6】 Select the parameter type for the TransferExit (0x37) command.

1.19.2.3.2. Checksum

During the program download process, to ensure the integrity of the data, a Checksum algorithm is required to verify the completeness and validity of the data. In the TSMaster diagnostic module's compliance services, the mainstream CRC algorithm is utilized for verification. The selection box for this is shown in the figure below:



ItemName	Value
ServiceName	DownloadFile1
CheckSum	CRC32
Hex File Path	[redacted]ntrol.hex
Transfer Exit Command	No Check($37)

Select a pre-defined CRC method

- CRC8
  - CRC8
  - CRC8_ITU
  - CRC8_ROHC
  - CRC8_MAXIM
  - CRC8_USERDEF
- CRC16
  - CRC16_CCITT
  - CRC16_CCITT_FLASE
  - CRC16_XMODEM
  - CRC16_X25
  - CRC16_MODBUS
  - CRC16_IBM
  - CRC16_MAXIM
  - CRC16_USB
  - CRC16_USERDEF
- CRC32
  - CRC32
  - CRC32_MPEG-2
  - CRC32_USERDEF

Parameter

Poly(0x)	04C11DB7
Init(0x)	FFFFFFFF
Sub(0x)	FFFFFFFF

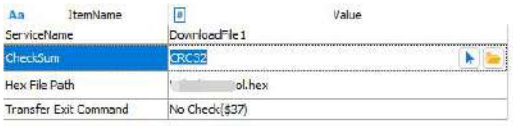
Intel Motorola

Input Convert

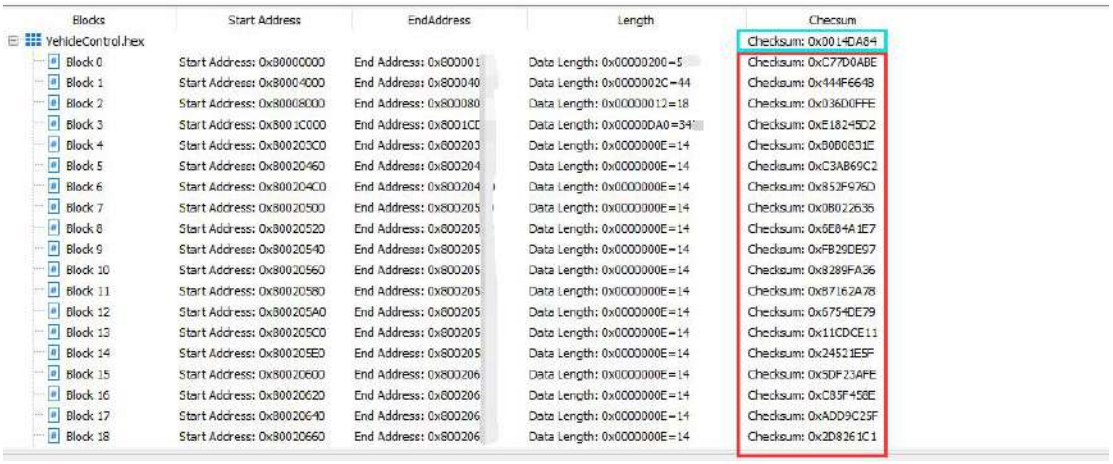
Output Convert

Blocks	Start Address			Checsum
vehicleControl.hex				0x0014DA84
Block 0	Start Address: 0x8000			0xC77D0ABE
Block 1	Start Address: 0x8000			0x444F6648
Block 2	Start Address: 0x8000			0x036D0FFE
Block 3	Start Address: 0x8001			0xE18245D2
Block 4	Start Address: 0x8002			0xB0B0831E
Block 5	Start Address: 0x8002			0xC3AB69C2
Block 6	Start Address: 0x8002			0x852F976D
Block 7	Start Address: 0x8002			0x0B022636
Block 8	Start Address: 0x8002			0x6E84A1E7
Block 9	Start Address: 0x8002			0xFB29DE97
Block 10	Start Address: 0x8002			0x8289FA36
Block 11	Start Address: 0x8002			0xB7162A78
Block 12	Start Address: 0x8002			0x6754DE79
Block 13	Start Address: 0x8002			0x11CDCE11
Block 14	Start Address: 0x8002			0x24521E5F
Block 15	Start Address: 0x8002			0x5DF23AFE
Block 16	Start Address: 0x8002			0xC85F458E
Block 17	Start Address: 0x80020640	End Address: 0x8002064D	Data Length: 0x0000000E=14	Checksum: 0xADD9C25F
Block 18	Start Address: 0x80020660	End Address: 0x8002066D	Data Length: 0x0000000E=14	Checksum: 0x2D8261C1

After the user selects a specific algorithm, the diagnostic module will calculate the Checksum value for the executable file, including the Checksum value for each Block of the file as well as the overall Checksum value for the file. This is shown in the figure below:



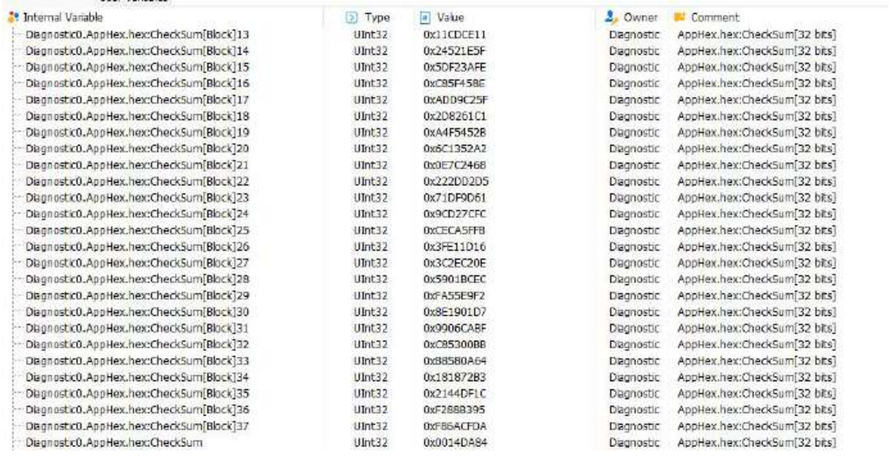
ItemName	Value
ServiceName	DownloadFile1
CheckSum	CRC32
Hex File Path	[redacted]ol.hex
Transfer Exit Command	No Check($37)



Blocks	Start Address	EndAddress	Length	Checsum
VehicleControl.hex				Checksum: 0x0014DA84
Block 0	Start Address: 0x80000000	End Address: 0x800001	Data Length: 0x00000200=5	Checksum: 0xC77D0ABE
Block 1	Start Address: 0x80004000	End Address: 0x800040	Data Length: 0x0000002C=44	Checksum: 0x444F6648
Block 2	Start Address: 0x80008000	End Address: 0x800080	Data Length: 0x00000012=18	Checksum: 0x036D0FFE
Block 3	Start Address: 0x8001C000	End Address: 0x8001CD	Data Length: 0x00000DA0=34	Checksum: 0xE18245D2
Block 4	Start Address: 0x800203C0	End Address: 0x800203	Data Length: 0x0000000E=14	Checksum: 0xB0B0831E
Block 5	Start Address: 0x80020460	End Address: 0x800204	Data Length: 0x0000000E=14	Checksum: 0xC3AB69C2
Block 6	Start Address: 0x800204C0	End Address: 0x800204	Data Length: 0x0000000E=14	Checksum: 0x852F976D
Block 7	Start Address: 0x80020500	End Address: 0x800205	Data Length: 0x0000000E=14	Checksum: 0x0B022636
Block 8	Start Address: 0x80020520	End Address: 0x800205	Data Length: 0x0000000E=14	Checksum: 0x6E84A1E7
Block 9	Start Address: 0x80020540	End Address: 0x800205	Data Length: 0x0000000E=14	Checksum: 0xFB29DE97
Block 10	Start Address: 0x80020560	End Address: 0x800205	Data Length: 0x0000000E=14	Checksum: 0x8289FA36
Block 11	Start Address: 0x80020580	End Address: 0x800205	Data Length: 0x0000000E=14	Checksum: 0xB7162A78
Block 12	Start Address: 0x800205A0	End Address: 0x800205	Data Length: 0x0000000E=14	Checksum: 0x6754DE79
Block 13	Start Address: 0x800205C0	End Address: 0x800205	Data Length: 0x0000000E=14	Checksum: 0x11CDCE11
Block 14	Start Address: 0x800205E0	End Address: 0x800205	Data Length: 0x0000000E=14	Checksum: 0x24521E5F
Block 15	Start Address: 0x80020600	End Address: 0x800206	Data Length: 0x0000000E=14	Checksum: 0x5DF23AFE
Block 16	Start Address: 0x80020620	End Address: 0x800206	Data Length: 0x0000000E=14	Checksum: 0xC85F458E
Block 17	Start Address: 0x80020640	End Address: 0x800206	Data Length: 0x0000000E=14	Checksum: 0xADD9C25F
Block 18	Start Address: 0x80020660	End Address: 0x800206	Data Length: 0x0000000E=14	Checksum: 0x2D8261C1

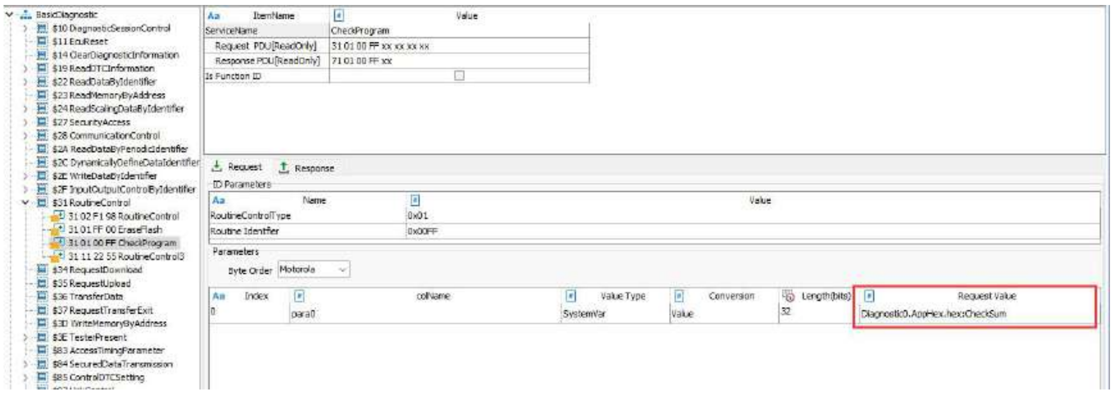
As shown in the figure, 0x0014DA84 is the overall checksum value of the file, and the checksum values below it are the checksum value of each data block, for example, 0xC77D0ABE.

After calculating the Checksum values for each Block and the overall program, these values are further registered into the system variables, as shown in the figure below:



Internal Variable	Type	Value	Owner	Comment
Diagnostic0.AppHex.hex:CheckSum[Block]13	UInt32	0x11CDCE11	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]14	UInt32	0x24521E5F	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]15	UInt32	0x5DF23AFE	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]16	UInt32	0xC85F458E	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]17	UInt32	0xADD9C25F	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]18	UInt32	0x2D8261C1	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]19	UInt32	0xA4F5452B	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]20	UInt32	0x6C1352A2	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]21	UInt32	0x0E7C2468	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]22	UInt32	0x222DD2D5	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]23	UInt32	0x71DF9D61	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]24	UInt32	0x9CD27CFC	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]25	UInt32	0xCECA5FFB	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]26	UInt32	0x3FE11D16	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]27	UInt32	0x3C2EC20E	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]28	UInt32	0x5901BCEC	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]29	UInt32	0xFA55E9F2	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]30	UInt32	0x8E1901D7	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]31	UInt32	0x9906CABF	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]32	UInt32	0xC85300BB	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]33	UInt32	0xB8580A64	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]34	UInt32	0x181872B3	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]35	UInt32	0x2144DF1C	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]36	UInt32	0xF2888395	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum[Block]37	UInt32	0xF86ACFDA	Diagnostic	AppHex.hex:CheckSum[32 bits]
Diagnostic0.AppHex.hex:CheckSum	UInt32	0x0014DA84	Diagnostic	AppHex.hex:CheckSum[32 bits]

TSMaster's diagnostic module can directly use system variables as parameters. Taking the commonly used validity verification of the executable file as an example in diagnostic commands, user can configure the following RoutineControl command to achieve the checking on the validity of the file, as shown below:



- BasicDiagnostic
  - $10 DiagnosticSessionControl
  - $11 EcuReset
  - $14 ClearDiagnosticInformation
  - $19 ReadDTCInformation
  - $22 ReadDataByIdentifier
  - $23 ReadMemoryByAddress
  - $24 ReadScalingDataByIdentifier
  - $27 SecurityAccess
  - $28 CommunicationControl
  - $2A ReadDataByPeriodicIdentifier
  - $2C DynamicallyDefineDataIdentifier
  - $2E WriteDataByIdentifier
  - $2F InputOutputControlByIdentifier
  - $31 RoutineControl
    - 31 02 F1 98 RoutineControl
    - 31 01 FF 00 EraseFlash
    - 31 01 00 FF CheckProgram
    - 31 11 22 55 RoutineControl3
  - $34 RequestDownload
  - $35 RequestUpload
  - $36 TransferData
  - $37 RequestTransferExit
  - $3D WriteMemoryByAddress
  - $3E TesterPresent
  - $83 AccessTimingParameter
  - $84 SecuredDataTransmission
  - $85 ControlDTCSetting

ItemName	Value
ServiceName	CheckProgram
Request PDU[ReadOnly]	31 01 00 FF xx xx xx xx
Response PDU[ReadOnly]	71 01 00 FF xx
Is Function ID	☐

Request Response

ID Parameters

Name	Value
RoutineControlType	0x01
Routine Identifier	0x00FF

Parameters

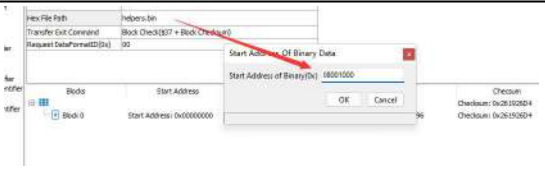
Byte Order: Motorola

Index	colName	Value Type	Conversion	Length(bits)	Request Value
0	para0	SystemVar	Value	32	Diagnostic0.AppHex.hex:CheckSum

The example uses the checksum value of the AppHex as a parameter for the RoutineControl command.

1.19.2.3.3. Load Binary Data File

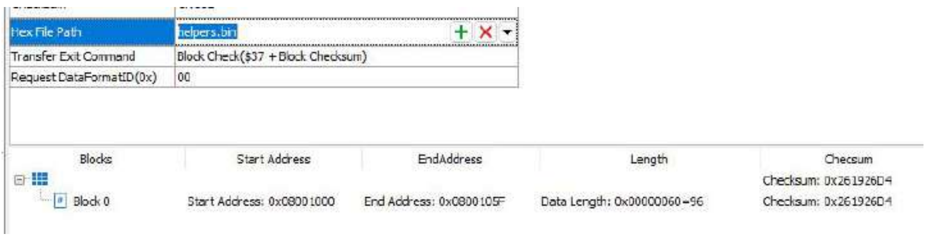
Hex and S19 file formats inherently include data starting addresses and lengths within their structure. However, for Binary-type binary files, they do not contain data starting addresses. Therefore, when loading a Binary data file, the user needs to manually input the starting address and the length of the data segment, as shown below:



Hex File Path	helpers.bin
Transfer Exit Command	Block Check($37 + Block Checksum)
Request DataFormatID(0x)	00
Start Address of Binary(0x)	08001000

Blocks	Start Address	Checsum
		Checksum: 0x261926D4
Block 0	Start Address: 0x00000000	Checksum: 0x261926D4

When loading a Binary file, the software will automatically prompt a window for setting the starting address as shown in the above figure. The address format is hexadecimal, with the address range being [0x00000000, 0xFFFFFFFF]. Users should set the address within this range and click confirm. If user chooses to cancel, the default address 0x00000000 will be used. After setting the address and loading, it is shown as below figure:



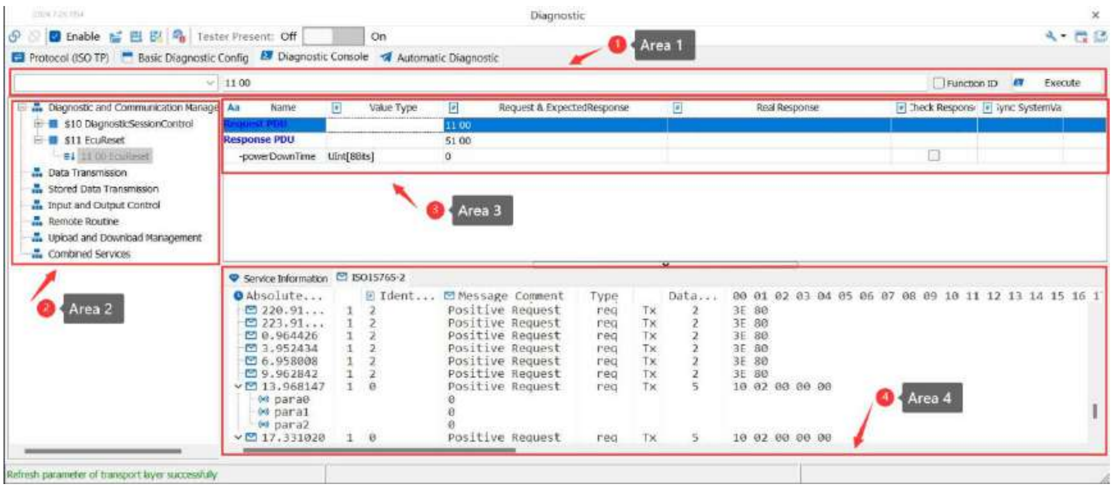
Hex File Path	helpers.bin
Transfer Exit Command	Block Check($37 + Block Checksum)
Request DataFormatID(0x)	00

Blocks	Start Address	EndAddress	Length	Checsum
				Checksum: 0x261926D4
Block 0	Start Address: 0x08001000	End Address: 0x0800105F	Data Length: 0x00000060=96	Checksum: 0x261926D4

As shown, the Binary file now has a starting address for the data segment.

1.19.3. Diagnostic Console

The Diagnostic Console serves as a diagnostic command debugger, allowing users to select each individual service command, edit the transmission of service messages, and receive service messages for testing and verification. It mainly consists of four working areas, as shown in the figure below:



11 00

Diagnostic and Communication Management
$10 DiagnosticSessionControl
$11 EcuReset
11 00 EcuReset
Data Transmission
Stored Data Transmission
Input and Output Control
Remote Routine
Upload and Download Management
Combined Services

Name	Value Type	Request & ExpectedResponse	Real Response	Check Response	Sync SystemVar
Request PDU		11 00			
Response PDU		51 00			
powerDownTime	UInt[8Bits]	0		☐	

```
Absolute...      Ident...  Message Comment   Type      Data...  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 1
 220.91...    1  2         Positive Request  req   Tx  2        3E 80
 223.91...    1  2         Positive Request  req   Tx  2        3E 80
 0.964426     1  2         Positive Request  req   Tx  2        3E 80
 3.952434     1  2         Positive Request  req   Tx  2        3E 80
 6.958008     1  2         Positive Request  req   Tx  2        3E 80
 9.962842     1  2         Positive Request  req   Tx  2        3E 80
v 13.968147   1  0         Positive Request  req   Tx  5        10 02 00 00 00
   para0                   0
   para1                   0
   para2                   0
v 17.331020   1  0         Positive Request  req   Tx  5        10 02 00 00 00
```

Refresh parameter of transport layer successfully

Area 1: manually command input area.

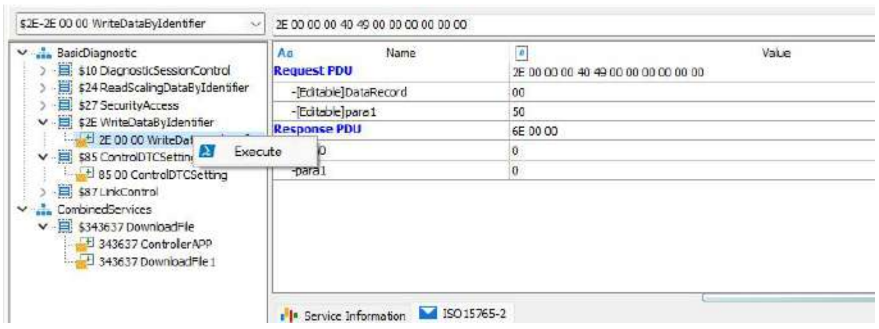
Area 2: service command selection area.

Area 3: diagnostic send/response parameter configuration area.

Area 4: diagnostic information/trace area

1.19.3.1. Service Command Selection Area

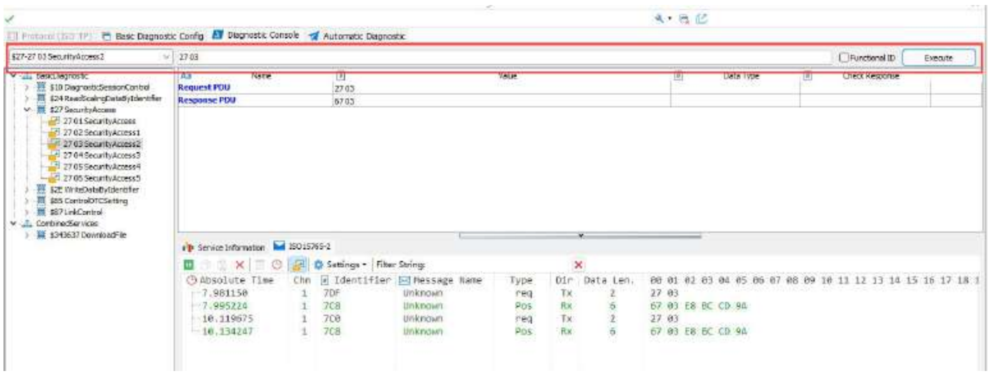
In the Service Command selection area, there is an executable service list generated based on the basic configuration (subsequent ODX/CDD, etc.). Users can double-click to execute the selected service or right-click to choose to execute the service, as shown in the figure below:



Name	Value
Request PDU	2E 00 00 00 40 49 00 00 00 00 00 00
-[Editable]DataRecord	00
-[Editable]para1	50
Response PDU	6E 00 00
0	0
-para1	0

1.19.3.2. Manually Command Input Area

During the testing process, if a user wishes to send arbitrary diagnostic commands, enter any desired message in the manual command input area, as shown in the figure below:



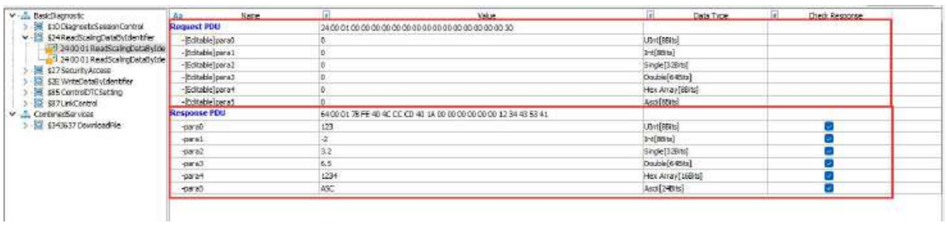
Name	Value	Data Type	Check Response
Request PDU	27 03		
Response PDU	67 03		

```
Absolute Time  Chn  Identifier  Message Name  Type  Dir  Data Len.  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 1
 7.981150       1   7DF         Unknown       req   Tx   2          27 03
 7.995224       1   7C8         Unknown       Pos   Rx   6          67 03 E8 BC CD 9A
 10.119675      1   7C0         Unknown       req   Tx   2          27 03
 10.134247      1   7C8         Unknown       Pos   Rx   6          67 03 E8 BC CD 9A
```

After entering the diagnostic message, click the "Execute" button on the right to send the diagnostic message. To increase the testing flexibility, users can select a checkbox to choose whether to send the diagnostic request message using the physical address or the functional ID.

1.19.3.3. Diagnostic Command Transmission/Response Area

In this area, users can edit the data segments to be sent and the expected data segments to be received, and initiate execution to verify whether the diagnostic response of the ECU under test complies with the actual requirements. Taking the 24 service as an example, six different data types of send parameters have been designed, and six different data types of response parameters have also been designed, as shown in the figure below:

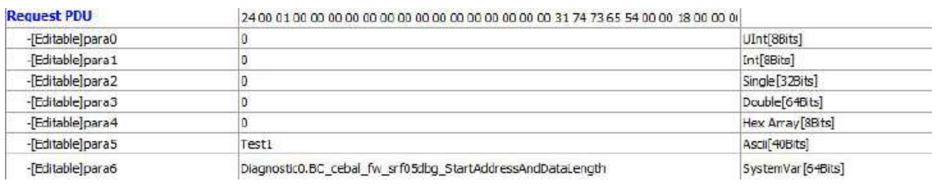


- BasicDiagnostic
  - $10 DiagnosticSessionControl
  - $24 ReadScalingDataByIdentifier
    - 24 00 01 ReadScalingDataByIde
    - 24 00 01 ReadScalingDataByIde
  - $27 SecurityAccess
  - $2E WriteDataByIdentifier
  - $85 ControlDTCSetting
  - $87 LinkControl
- CombinedServices
  - $343637 DownloadFile

Name	Value	Data Type	Check Response
Request PDU	24 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 30		
-[Editable]para0	0	UInt[8Bits]	
-[Editable]para1	0	Int[8Bits]	
-[Editable]para2	0	Single[32Bits]	
-[Editable]para3	0	Double[64Bits]	
-[Editable]para4	0	Hex Array[8Bits]	
-[Editable]para5	0	Ascii[8Bits]	
Response PDU	64 00 01 7B FE 40 4C CC CD 40 1A 00 00 00 00 00 00 12 34 43 53 41		
-para0	123	UInt[8Bits]	✓
-para1	-2	Int[8Bits]	✓
-para2	3.2	Single[32Bits]	✓
-para3	6.5	Double[64Bits]	✓
-para4	1234	Hex Array[16Bits]	✓
-para5	ABC	Ascii[24Bits]	✓

1.19.3.3.1. Input Diagnostic Parameters

Example to input calibration parameter is as follows:



Request PDU	24 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 31 74 73 65 54 00 00 18 00 00 0	
-[Editable]para0	0	UInt[8Bits]
-[Editable]para1	0	Int[8Bits]
-[Editable]para2	0	Single[32Bits]
-[Editable]para3	0	Double[64Bits]
-[Editable]para4	0	Hex Array[8Bits]
-[Editable]para5	Test1	Ascii[40Bits]
-[Editable]para6	Diagnostic0.BC_cebal_fw_srf05dbg_StartAddressAndDataLength	SystemVar[64Bits]

Request PDU: The diagnostic data packet bytes that the diagnostic module needs to send; this part of the values is not editable. After the user enters the parameter values, this part of the data automatically generates the corresponding diagnostic data values.

Diagnostic Parameters: the corresponding relationship is as follows:

【1】 Para0, the data type is UInt and the data length is 8 bits. When the input value is 12, the corresponding byte is 0x0C.

【2】 Para1, the data type is Int and the data length is 8 bits. When the input value is -1， the corresponding byte is 0xFF.

【3】 Para2, the data type is Single, and the data length is 32 bits. When the input value is 3.1, the corresponding byte is 0x40, 0x46, 0x66, 0x66.

【4】 Para3, the data type is Double and the data length is 64 bits. When the input value is 3.2, the corresponding byte is 0x40， 0x09， 0x99， 0x99， 0x99， 0x99， 0x99， 0x9A.

【5】 Para4, the data type is Hex Array, and the data length is 8 bits, when the input value is 0x11, the corresponding byte is 0x11.

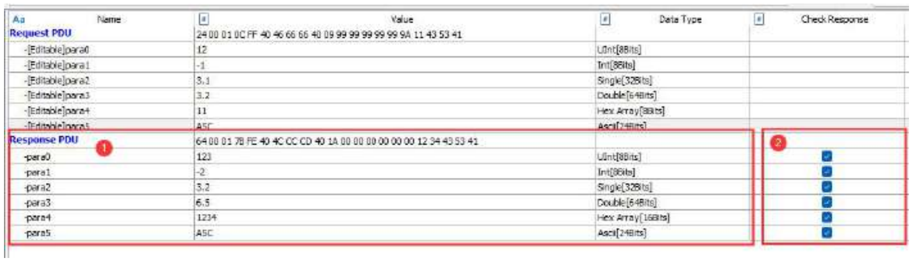
【6】 Para5, the data type is ASCII string, and the data length is 24 bits, when the input string is ”ASC”, the corresponding byte is 0x43, 0x53, 0x41.

【7】 Para6, the data type is system variable. The data length is determined by the value of the extracted system variable, which is 64 bits. The system variable is named "Diag nostic0.BC_cebal_fw_srf05dbg_StartAddressAndDataLength". During the executio n process, the system will automatically extract the actual value of this system variab le based on its name and parse it into the transmission message.

After completing the input of the aforementioned diagnostic parameters, the generated diagnostic request data packet is: 0x24 0x00 0x01 0x0C 0xFF 0x40 0x46 0x66 0x66 0x40 0x09 0x99 0x99 0x99 0x99 0x99 0x9A 0x11 0x43 0x53 0x41, as shown in the figure above.

1.19.3.3.2. Input Response Parameter

Input the response parameter values as shown in the figure below:

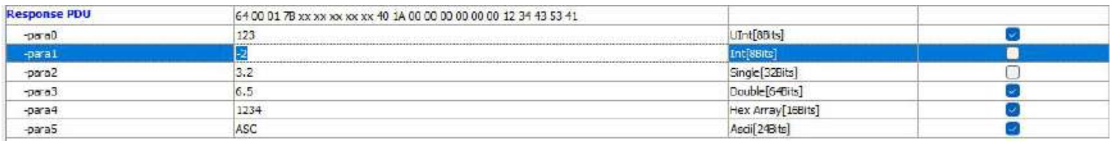


Name	Value	Data Type	Check Response
Request PDU	24 00 01 0C FF 40 46 66 66 40 09 99 99 99 99 99 9A 11 43 53 41		
-[Editable]para0	12	UInt[8Bits]	
-[Editable]para1	-1	Int[8Bits]	
-[Editable]para2	3.1	Single[32Bits]	
-[Editable]para3	3.2	Double[64Bits]	
-[Editable]para4	11	Hex Array[8Bits]	
-[Editable]para5	ASC	Ascii[24Bits]	
Response PDU	64 00 01 7B FE 40 4C CC CD 40 1A 00 00 00 00 00 00 12 34 43 53 41		
-para0	123	UInt[8Bits]	✓
-para1	-2	Int[8Bits]	✓
-para2	3.2	Single[32Bits]	✓
-para3	6.5	Double[64Bits]	✓
-para4	1234	Hex Array[16Bits]	✓
-para5	ASC	Ascii[24Bits]	✓

The first part is identical to the input diagnostic parameters in the previous section and will not be explained again here. However, the response parameters have an additional optional command to select, which is whether to check these parameters. If the Check Response option is selected, the ECU's response must match the configured response parameters for this diagnostic test to pass. If not selected, the diagnostic module will not detect the content of these bytes in the ECU's response.

【1】 When all the above configuration responses are checked, to be recognized by the system as passing the diagnostic test, the ECU's response message must be equal to: 0x64 0x00 0x01 0x7B 0xFE 0x40 0x4C 0xCC 0xCD 0x40 0x1A 0x00 0x00 0x00 0x00 0x00 0x00 0x12 0x34 0x43 0x53 0x41.

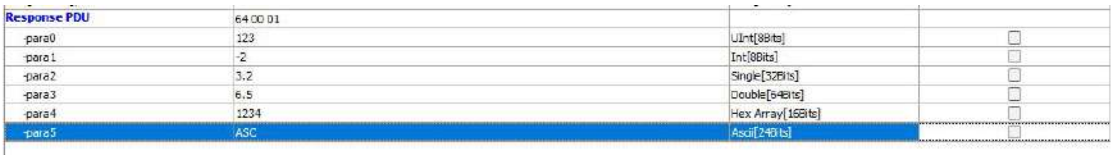
【2】 Uncheck the boxes for checking Para1 and Para2, as shown in the figure below:



Response PDU	64 00 01 7B xx xx xx xx xx 40 1A 00 00 00 00 00 00 12 34 43 53 41		
-para0	123	UInt[8Bits]	☑
-para1	-2	Int[8Bits]	☐
-para2	3.2	Single[32Bits]	☐
-para3	6.5	Double[64Bits]	☑
-para4	1234	Hex Array[16Bits]	☑
-para5	ASC	Ascii[24Bits]	☑

At this point, the ECU response message must be equal to: 0x64 0x00 0x01 0x7B 0xXX 0xXX 0xXX 0xXX 0xXX 0x40 0x1A 0x00 0x00 0x00 0x00 0x00 0x00 0x12 0x34 0x43 0x53 0x41. Where the red parts labeled 0xXX indicate that those bytes are not to be evaluated, and the remaining bytes must equal the configured bytes mentioned above for the system to consider the diagnostic test as passed.

【3】 Uncheck the boxes for checking Para0 to Para5, as shown in the figure below:

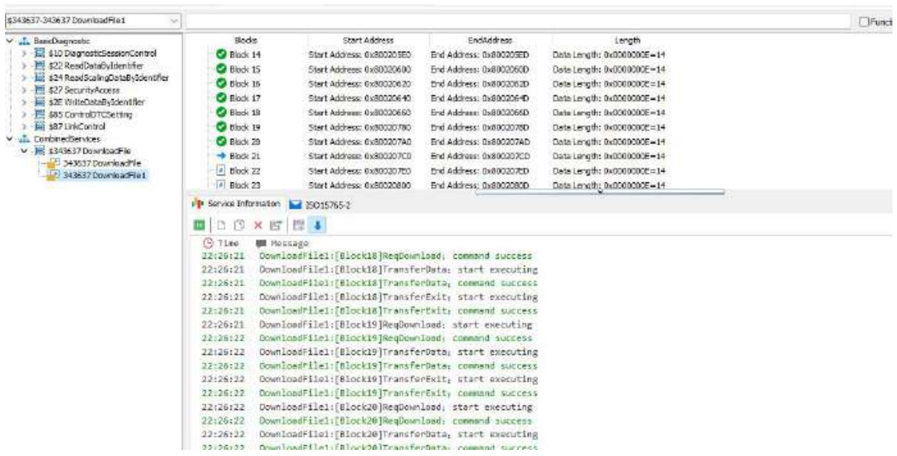


Response PDU	64 00 01		
-para0	123	UInt[8Bits]	☐
-para1	-2	Int[8Bits]	☐
-para2	3.2	Single[32Bits]	☐
-para3	6.5	Double[64Bits]	☐
-para4	1234	Hex Array[16Bits]	☐
-para5	ASC	Ascii[24Bits]	☐

In this case, for the ECU response message to be considered as passing the diagnostic test by the system, it must equal: 0x64 0x00 0x01.

1.19.3.4. Diagnostic Execution

Taking CombinedService as an example, during the diagnostic execution process, it will display the area of the currently completed Block download and show the execution time for each Block written, as shown in the figure below:



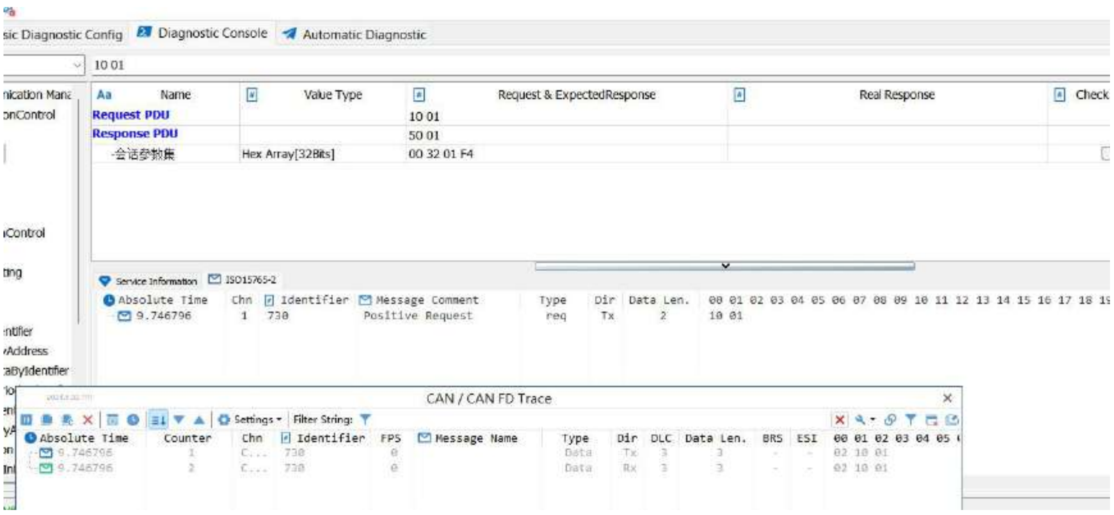
Blocks	Start Address	EndAddress	Length
✓ Block 14	Start Address: 0x800205E0	End Address: 0x800205ED	Data Length: 0x0000000E=14
✓ Block 15	Start Address: 0x80020600	End Address: 0x8002060D	Data Length: 0x0000000E=14
✓ Block 16	Start Address: 0x80020620	End Address: 0x8002062D	Data Length: 0x0000000E=14
✓ Block 17	Start Address: 0x80020640	End Address: 0x8002064D	Data Length: 0x0000000E=14
✓ Block 18	Start Address: 0x80020660	End Address: 0x8002066D	Data Length: 0x0000000E=14
✓ Block 19	Start Address: 0x80020780	End Address: 0x8002078D	Data Length: 0x0000000E=14
✓ Block 20	Start Address: 0x800207A0	End Address: 0x800207AD	Data Length: 0x0000000E=14
→ Block 21	Start Address: 0x800207C0	End Address: 0x800207CD	Data Length: 0x0000000E=14
Block 22	Start Address: 0x800207E0	End Address: 0x800207ED	Data Length: 0x0000000E=14
Block 23	Start Address: 0x80020800	End Address: 0x8002080D	Data Length: 0x0000000E=14

```
Time      Message
22:26:21  DownloadFile1:[Block18]ReqDownload; command success
22:26:21  DownloadFile1:[Block18]TransferData: start executing
22:26:21  DownloadFile1:[Block18]TransferData; command success
22:26:21  DownloadFile1:[Block18]TransferExit; start executing
22:26:21  DownloadFile1:[Block18]TransferExit; command success
22:26:21  DownloadFile1:[Block19]ReqDownload: start executing
22:26:22  DownloadFile1:[Block19]ReqDownload; command success
22:26:22  DownloadFile1:[Block19]TransferData; start executing
22:26:22  DownloadFile1:[Block19]TransferData; command success
22:26:22  DownloadFile1:[Block19]TransferExit; start executing
22:26:22  DownloadFile1:[Block19]TransferExit; command success
22:26:22  DownloadFile1:[Block20]ReqDownload; start executing
22:26:22  DownloadFile1:[Block20]ReqDownload: command success
22:26:22  DownloadFile1:[Block20]TransferData; start executing
22:26:22  DownloadFile1:[Block20]TransferData; command success
```

1.19.3.5. Diagnostic/Trace Area

1.19.3.5.1. Service/Raw Message Trace Comparison

In diagnostics, users will encounter the most primitive CAN/CANFD/LIN messages, as well as service layer messages that have been transmitted through the transport layer. In the TSMaster diagnostic module, the original CAN messages are viewed in the basic Trace module, while the service messages processed by the transport layer are directly viewed in the diagnostic module's Trace area. As shown in the figure below:



Name	Value Type	Request & ExpectedResponse	Real Response	Check
Request PDU		10 01		
Response PDU		50 01		
会话参数集	Hex Array[32Bits]	00 32 01 F4		

```
Absolute Time  Chn  Identifier  Message Comment   Type  Dir  Data Len.  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19
9.746796       1    730         Positive Request  req   Tx   2          10 01
```

```
Absolute Time  Counter  Chn   Identifier  FPS  Message Name  Type  Dir  DLC  Data Len.  BRS  ESI  00 01 02 03 04 05
9.746796       1        C...  730         0                  Data  Tx   3    3          -    -    02 10 01
9.746796       2        C...  730         0                  Data  Rx   3    3          -    -    02 10 01
```

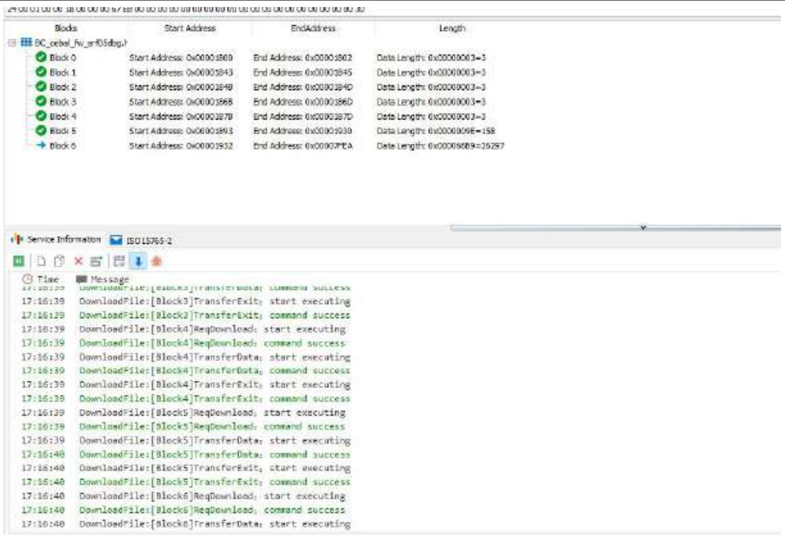
By comparing the above figures, we can see:

➢ The original CAN/CANFD message area also displays information about multi-frame, single-frame, and first-frame transmissions from the transport layer.

➢ The Trace in the diagnostic module presents to the user the service layer messages directly. For users, it is only necessary to be concerned with the service content they are sending, without needing to worry about how this content is specifically divided and sent. Therefore, when performing diagnostic services, it is important to focus on the internal Trace interface within the diagnostic module.

1.19.3.5.2. Operation Prompt Area

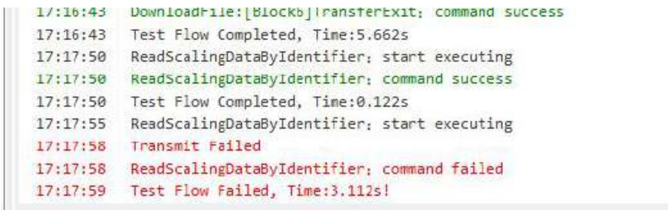
This area displays the current operational steps within the diagnostic module. As shown in the figure below, it shows the process of downloading a hex file, including the internal transmission steps of the program.



Blocks	Start Address	EndAddress	Length
BC_cebal_fw_srf05dbg.h			
Block 0	Start Address: 0x00001800	End Address: 0x00001802	Data Length: 0x00000003=3
Block 1	Start Address: 0x00001843	End Address: 0x00001845	Data Length: 0x00000003=3
Block 2	Start Address: 0x0000184B	End Address: 0x0000184D	Data Length: 0x00000003=3
Block 3	Start Address: 0x0000186B	End Address: 0x0000186D	Data Length: 0x00000003=3
Block 4	Start Address: 0x0000187B	End Address: 0x0000187D	Data Length: 0x00000003=3
Block 5	Start Address: 0x00001893	End Address: 0x00001930	Data Length: 0x0000009E=158
Block 6	Start Address: 0x00001932	End Address: 0x00007FEA	Data Length: 0x000066B9=26297

```
Time      Message
17:16:39  DownloadFile:[Block3]TransferExit: start executing
17:16:39  DownloadFile:[Block3]TransferExit: command success
17:16:39  DownloadFile:[Block4]ReqDownload: start executing
17:16:39  DownloadFile:[Block4]ReqDownload: command success
17:16:39  DownloadFile:[Block4]TransferData: start executing
17:16:39  DownloadFile:[Block4]TransferData: command success
17:16:39  DownloadFile:[Block4]TransferExit: start executing
17:16:39  DownloadFile:[Block4]TransferExit: command success
17:16:39  DownloadFile:[Block5]ReqDownload: start executing
17:16:39  DownloadFile:[Block5]ReqDownload: command success
17:16:39  DownloadFile:[Block5]TransferData: start executing
17:16:40  DownloadFile:[Block5]TransferData: command success
17:16:40  DownloadFile:[Block5]TransferExit: start executing
17:16:40  DownloadFile:[Block5]TransferExit: command success
17:16:40  DownloadFile:[Block6]ReqDownload: start executing
17:16:40  DownloadFile:[Block6]ReqDownload: command success
17:16:40  DownloadFile:[Block6]TransferData: start executing
```

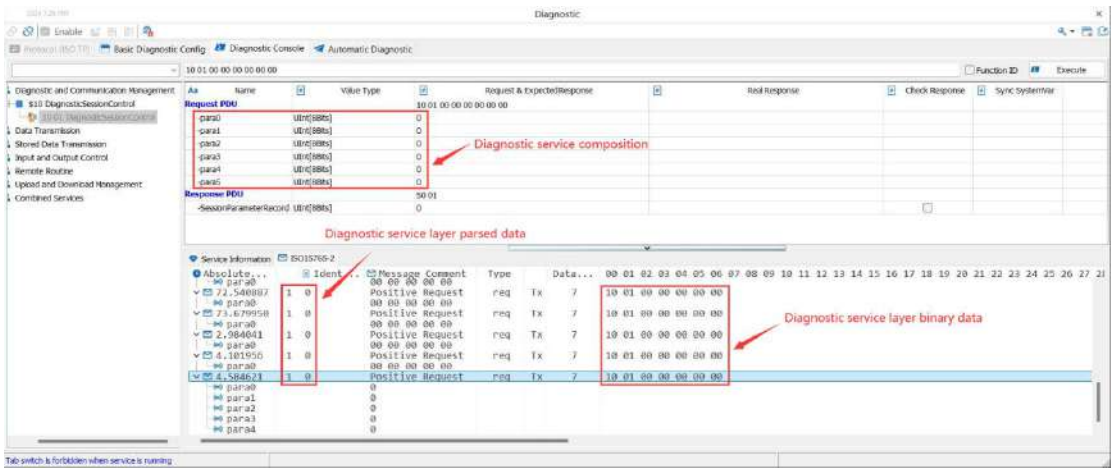
When the diagnostic service does not receive an affirmative response or no response, the error message prompts are shown as follows:



```
17:16:43  DownloadFile:[Block6]TransferExit: command success
17:16:43  Test Flow Completed, Time:5.662s
17:17:50  ReadScalingDataByIdentifier: start executing
17:17:50  ReadScalingDataByIdentifier: command success
17:17:50  Test Flow Completed, Time:0.122s
17:17:55  ReadScalingDataByIdentifier: start executing
17:17:58  Transmit Failed
17:17:58  ReadScalingDataByIdentifier: command failed
17:17:59  Test Flow Failed, Time:3.112s!
```

1.19.3.5.3. ISO15765-2 Service Message Area

This area is used to display detailed service layer message information of the diagnostic module. Combined with the diagnostic database configured earlier, it can also parse the original message data into physical signals for the user, as shown in the figure below:



Diagnostic
Enable
Protocol (ISO TP)
Basic Diagnostic Config
Diagnostic Console
Automatic Diagnostic
Function ID
Execute
Diagnostic and Communication Management
$10 DiagnosticSessionControl
Data Transmission
Stored Data Transmission
Input and Output Control
Remote Routine
Upload and Download Management
Combined Services
Diagnostic service composition
Diagnostic service layer parsed data
Diagnostic service layer binary data
Service Information
ISO15765-2
Tab switch is forbidden when service is running

10 01 00 00 00 00 00 00

Name	Value Type	Request & ExpectedResponse	Real Response	Check Response	Sync SystemVar
Request PDU		10 01 00 00 00 00 00 00			
-para0	UInt[8Bits]	0			
-para1	UInt[8Bits]	0			
-para2	UInt[8Bits]	0			
-para3	UInt[8Bits]	0			
-para4	UInt[8Bits]	0			
-para5	UInt[8Bits]	0			
Response PDU		50 01			
-SessionParameterRecord	UInt[8Bits]	0		☐	

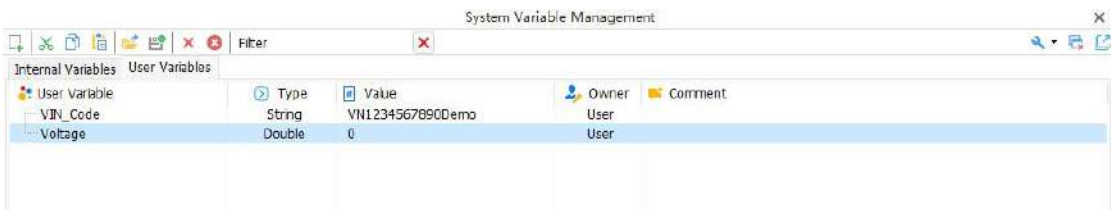
```
Absolute...       Ident...  Message Comment     Type      Data...  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
   para0                    00 00 00 00 00
v 72.540887       1   0     Positive Request    req   Tx   7       10 01 00 00 00 00 00
   para0                    00 00 00 00 00
v 73.679958       1   0     Positive Request    req   Tx   7       10 01 00 00 00 00 00
   para0                    00 00 00 00 00
v 2.984041        1   0     Positive Request    req   Tx   7       10 01 00 00 00 00 00
   para0                    00 00 00 00 00
v 4.101956        1   0     Positive Request    req   Tx   7       10 01 00 00 00 00 00
   para0                    00 00 00 00 00
v 4.584621        1   0     Positive Request    req   Tx   7       10 01 00 00 00 00 00
   para0                    0
   para1                    0
   para2                    0
   para3                    0
   para4                    0
```

1.19.4. Flexible Application of System Variables

1.19.4.1. System Variables as Parameters

System variables have the capability to interact data between internal software and external modules. TSMaster introduces system variables as parameters into the diagnostic module, greatly expanding the data interaction capabilities between the diagnostic module and other modules. Below are several typical application scenarios to describe its functionality:

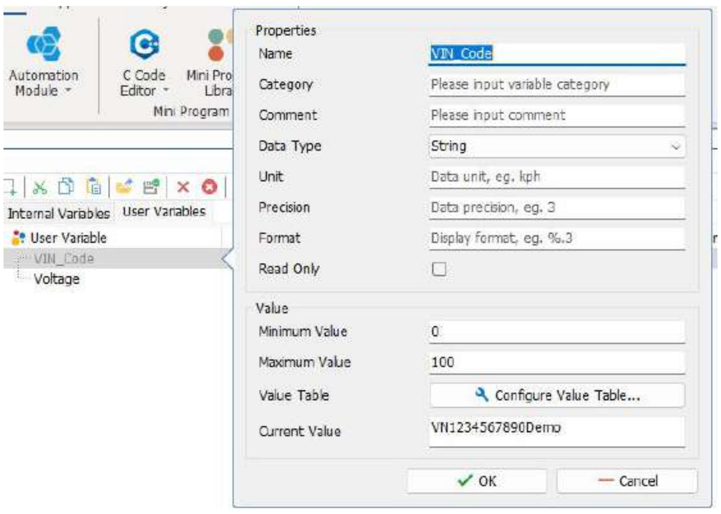
First, create two system variables in the system, VIN_Code and Voltage, as shown below:



System Variable Management

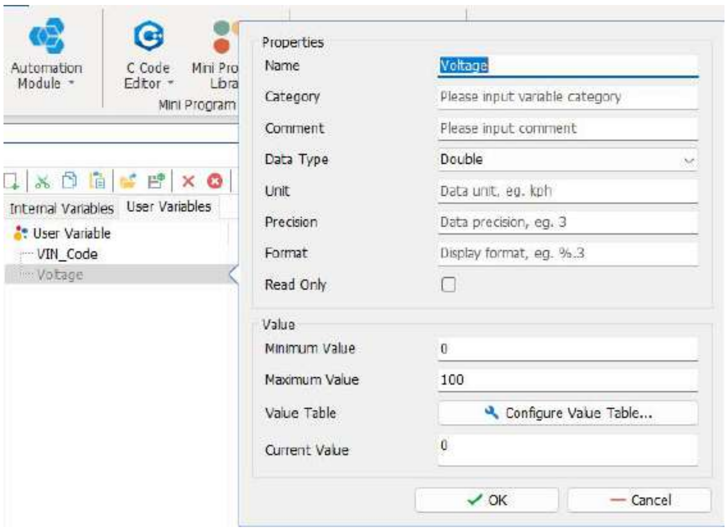
User Variable	Type	Value	Owner	Comment
VIN_Code	String	VN1234567890Demo	User	
Voltage	Double	0	User	

The VIN_Code variable is of the string type:



Properties	
Name	VIN_Code
Category	Please input variable category
Comment	Please input comment
Data Type	String
Unit	Data unit, eg. kph
Precision	Data precision, eg. 3
Format	Display format, eg. %.3
Read Only	☐
Value	
Minimum Value	0
Maximum Value	100
Value Table	Configure Value Table...
Current Value	VN1234567890Demo

The Voltage variable is of the Double type:

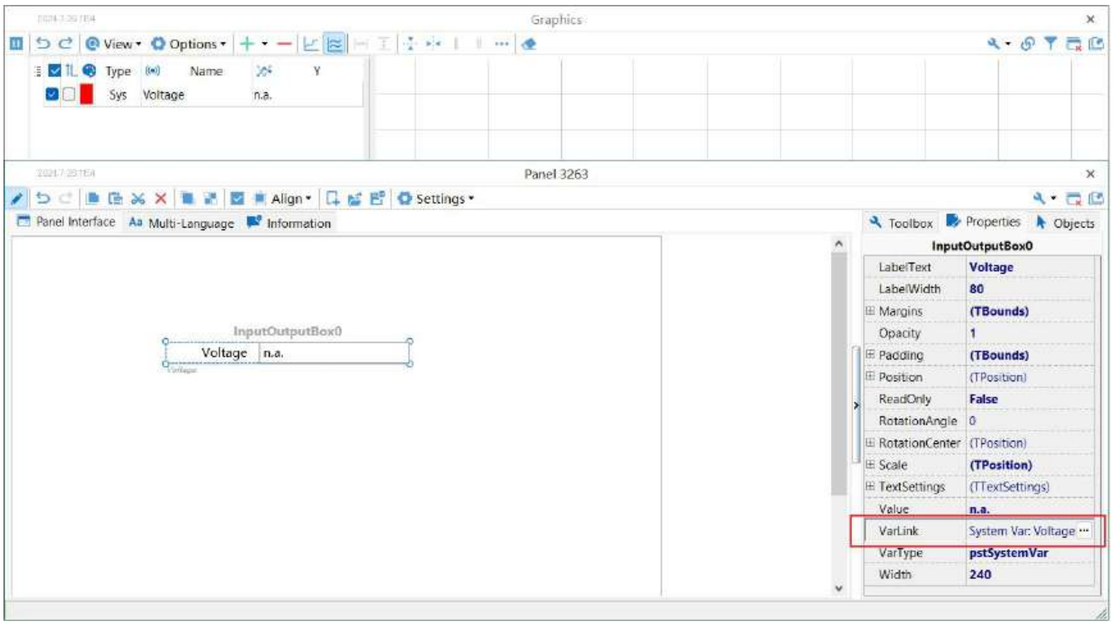

Properties

Name	Voltage
Category	Please input variable category
Comment	Please input comment
Data Type	Double
Unit	Data unit, eg. kph
Precision	Data precision, eg. 3
Format	Display format, eg. %.3
Read Only	☐

Value

Minimum Value	0
Maximum Value	100
Value Table	Configure Value Table...
Current Value	0

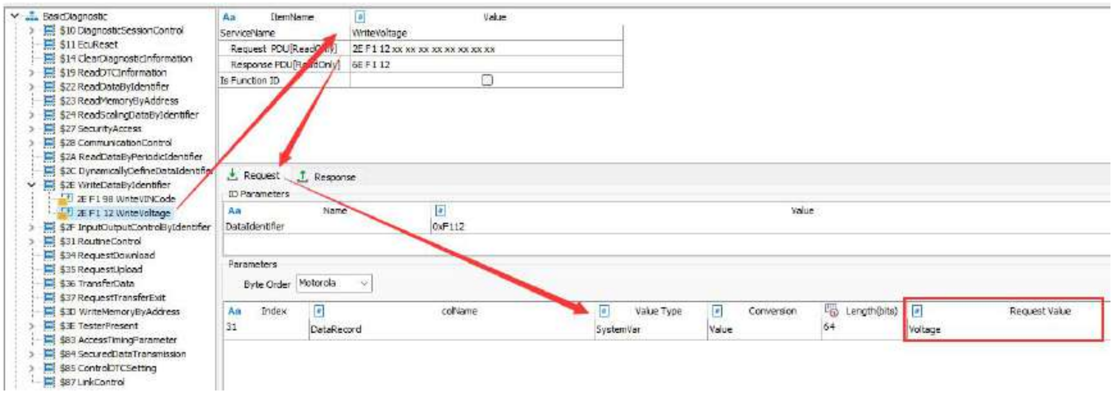
Associate the system variables with the Panel and Graphic, as shown below:


Type	Name	Y
Sys	Voltage	n.a.

InputOutputBox0

LabelText	Voltage
LabelWidth	80
Margins	(TBounds)
Opacity	1
Padding	(TBounds)
Position	(TPosition)
ReadOnly	False
RotationAngle	0
RotationCenter	(TPosition)
Scale	(TPosition)
TextSettings	(TTextSettings)
Value	n.a.
VarLink	System Var: Voltage
VarType	pstSystemVar
Width	240

【1】 In the Panel, set the voltage value for Voltage and write it into the ECU through diagnostics.



ItemName	Value
ServiceName	WriteVoltage
Request PDU[ReadOnly]	2E F1 12 xx xx xx xx xx xx xx xx
Response PDU[ReadOnly]	6E F1 12
Is Function ID	☐

ID Parameters

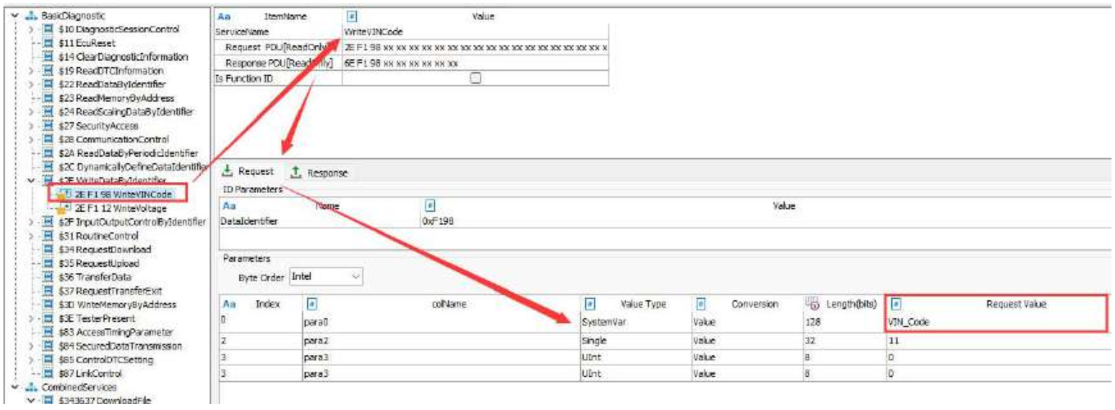
Name	Value
DataIdentifier	0xF112

Parameters

Byte Order: Motorola

Index	colName	Value Type	Conversion	Length(bits)	Request Value
31	DataRecord	SystemVar	Value	64	Voltage

【2】 In the Panel, set the VIN code and write it into the ECU through diagnostics.



ItemName	Value
ServiceName	WriteVINCode
Request PDU[ReadOnly]	2E F1 98 xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx x
Response PDU[ReadOnly]	6E F1 98 xx xx xx xx xx xx
Is Function ID	☐

ID Parameters

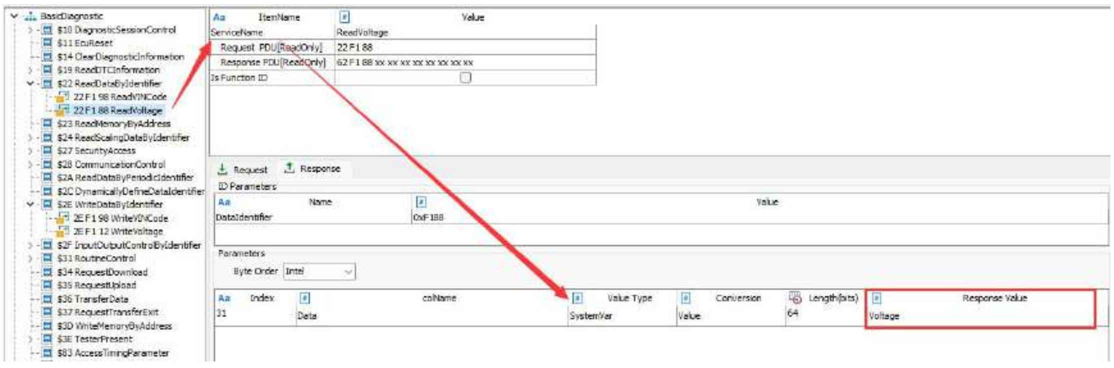
Name	Value
DataIdentifier	0xF198

Parameters

Byte Order: Intel

Index	colName	Value Type	Conversion	Length(bits)	Request Value
0	para0	SystemVar	Value	128	VIN_Code
2	para2	Single	Value	32	11
3	para3	UInt	Value	8	0
3	para3	UInt	Value	8	0

【3】 Read the internal voltage value from the ECU through diagnostics and display it in the Graphic.



ItemName	Value
ServiceName	ReadVoltage
Request PDU[ReadOnly]	22 F1 88
Response PDU[ReadOnly]	62 F1 88 xx xx xx xx xx xx xx xx
Is Function ID	☐

ID Parameters

Name	Value
DataIdentifier	0xF188

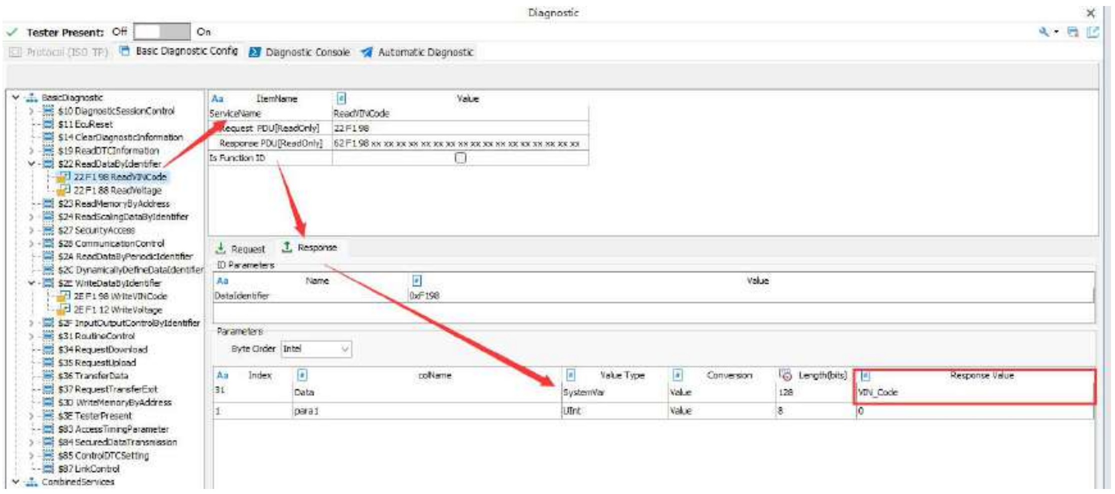
Parameters

Byte Order: Intel

Index	colName	Value Type	Conversion	Length(bits)	Response Value
31	Data	SystemVar	Value	64	Voltage

Note that the read variables need to be manually set by the user to synchronize with the system variables.

【4】 Read the internal VIN code from the ECU through diagnostics and display it in the Panel.



ItemName	Value
ServiceName	ReadVINCode
Request PDU[ReadOnly]	22 F1 98
Response PDU[ReadOnly]	62 F1 98 xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
Is Function ID	☐

ID Parameters

Name	Value
DataIdentifier	0xF198

Parameters

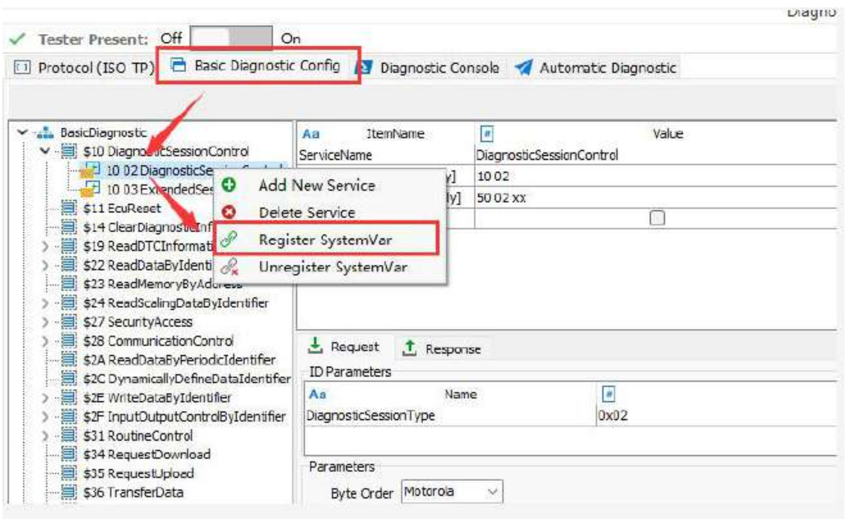
Byte Order: Intel

Index	colName	Value Type	Conversion	Length(bits)	Response Value
31	Data	SystemVar	Value	128	VIN_Code
1	para1	UInt	Value	8	0

1.19.4.2. System Variable Association with Console Service

In the previous chapters, users could flexibly configure diagnostic services in the diagnostic console as needed. After these diagnostic services are configured, users need to double-click to start the diagnostic service in the diagnostic console. However, if users want to start the diagnostic command in the Panel interface, they still need to use system variables. The steps are as follows:

【1】 First, in the Diagnostic BasicConfig window, select the target service, and then register the diagnostic service as a system variable through the right-click menu, as shown below:



ItemName	Value
ServiceName	DiagnosticSessionControl
y]	10 02
y]	50 02 xx
	☐

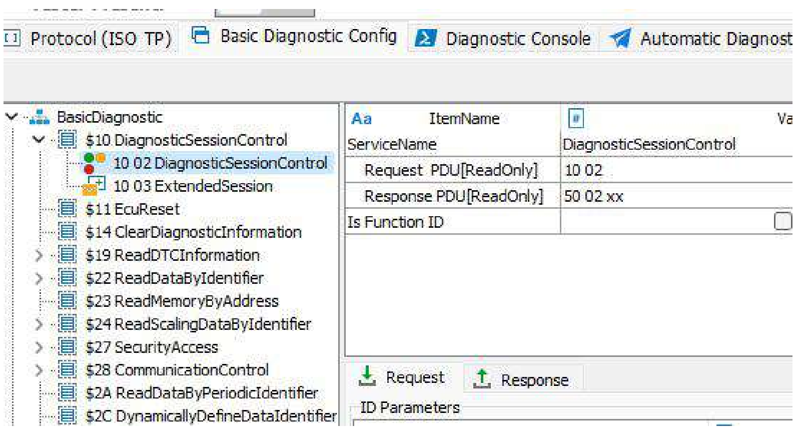
ID Parameters

Name	Value
DiagnosticSessionType	0x02

Parameters

Byte Order: Motorola

After the registration is complete, the service item's icon changes to the following icon, indicating that it has become a service registered with a system variable, as shown below:


Protocol (ISO TP) | Basic Diagnostic Config | Diagnostic Console | Automatic Diagnost

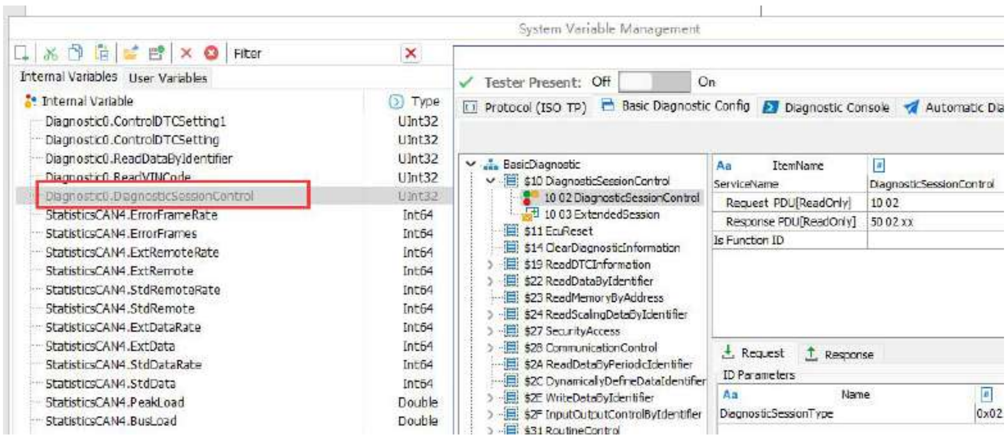
BasicDiagnostic
- $10 DiagnosticSessionControl
  - 10 02 DiagnosticSessionControl
  - 10 03 ExtendedSession
- $11 EcuReset
- $14 ClearDiagnosticInformation
- $19 ReadDTCInformation
- $22 ReadDataByIdentifier
- $23 ReadMemoryByAddress
- $24 ReadScalingDataByIdentifier
- $27 SecurityAccess
- $28 CommunicationControl
- $2A ReadDataByPeriodicIdentifier
- $2C DynamicallyDefineDataIdentifier

ItemName	Va
ServiceName	DiagnosticSessionControl
Request PDU[ReadOnly]	10 02
Response PDU[ReadOnly]	50 02 xx
Is Function ID	☐

Request | Response

ID Parameters

【2】 After the registration is complete, user can see the system variable in System Variable Manager Management, as shown below:


System Variable Management

Filter

Internal Variables | User Variables

Internal Variable	Type
DiagnosticO.ControlDTCSetting1	UInt32
DiagnosticO.ControlDTCSetting	UInt32
DiagnosticO.ReadDataByIdentifier	UInt32
DiagnosticO.ReadVINCode	UInt32
DiagnosticO.DiagnosticSessionControl	UInt32
StatisticsCAN4.ErrorFrameRate	Int64
StatisticsCAN4.ErrorFrames	Int64
StatisticsCAN4.ExtRemoteRate	Int64
StatisticsCAN4.ExtRemote	Int64
StatisticsCAN4.StdRemoteRate	Int64
StatisticsCAN4.StdRemote	Int64
StatisticsCAN4.ExtDataRate	Int64
StatisticsCAN4.ExtData	Int64
StatisticsCAN4.StdDataRate	Int64
StatisticsCAN4.StdData	Int64
StatisticsCAN4.PeakLoad	Double
StatisticsCAN4.BusLoad	Double

Tester Present: Off On

Protocol (ISO TP) | Basic Diagnostic Config | Diagnostic Console | Automatic Dia

BasicDiagnostic
- $10 DiagnosticSessionControl
  - 10 02 DiagnosticSessionControl
  - 10 03 ExtendedSession
- $11 EcuReset
- $14 ClearDiagnosticInformation
- $19 ReadDTCInformation
- $22 ReadDataByIdentifier
- $23 ReadMemoryByAddress
- $24 ReadScalingDataByIdentifier
- $27 SecurityAccess
- $28 CommunicationControl
- $2A ReadDataByPeriodicIdentifier
- $2C DynamicallyDefineDataIdentifier
- $2E WriteDataByIdentifier
- $2F InputOutputControlByIdentifier
- $31 RoutineControl

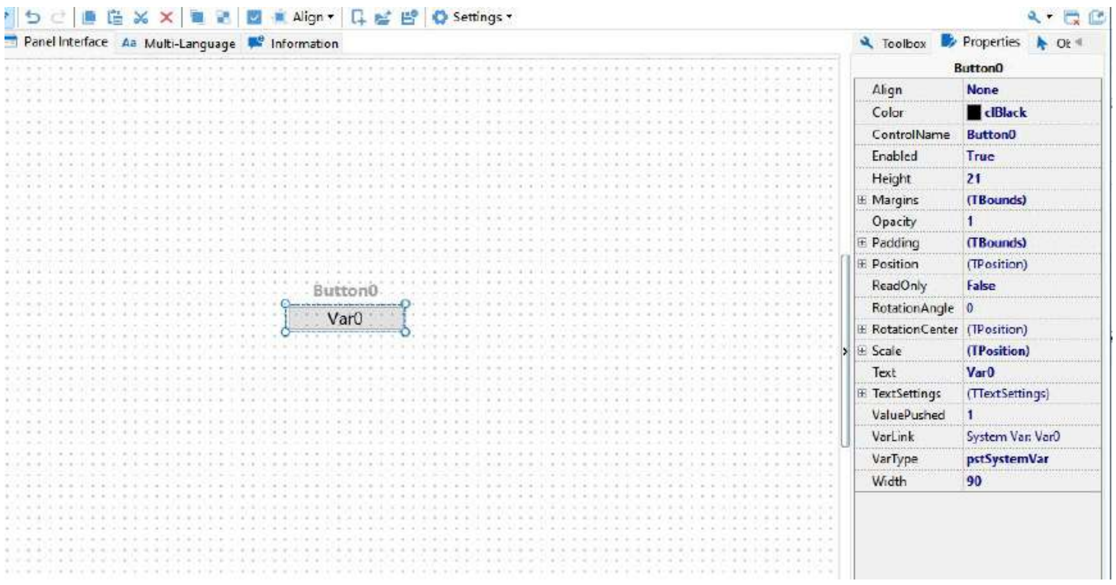
ItemName	
ServiceName	DiagnosticSessionControl
Request PDU[ReadOnly]	10 02
Response PDU[ReadOnly]	50 02 xx
Is Function ID	

Request | Response

ID Parameters

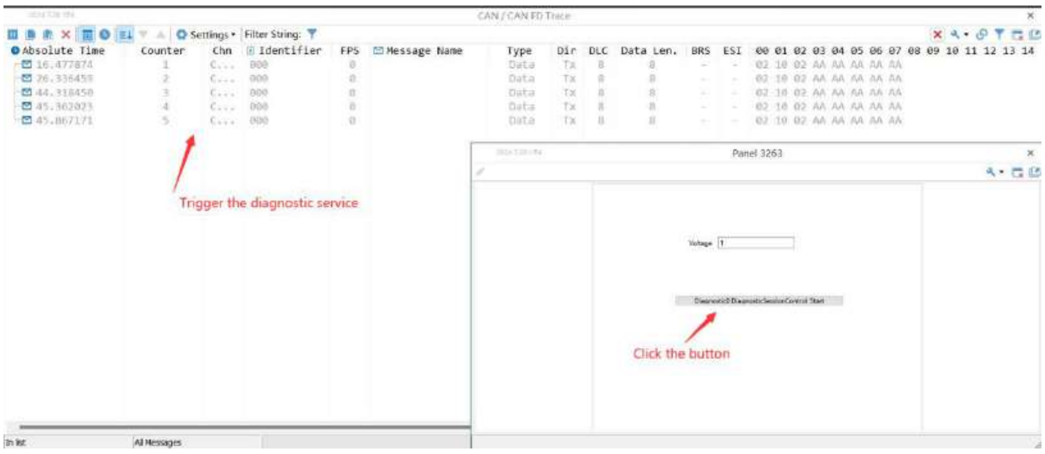
Name	
DiagnosticSessionType	0x02

【3】 In the Panel, add a Button and associate it with the system variable, as shown below:



Button0	
Align	None
Color	■ clBlack
ControlName	Button0
Enabled	True
Height	21
Margins	(TBounds)
Opacity	1
Padding	(TBounds)
Position	(TPosition)
ReadOnly	False
RotationAngle	0
RotationCenter	(TPosition)
Scale	(TPosition)
Text	Var0
TextSettings	(TTextSettings)
ValuePushed	1
VarLink	System Var: Var0
VarType	pstSystemVar
Width	90

【4】 Run the program and click the test button on the Panel. We can see that the diagnostic module has executed the DiagnosticSessionControl service. As shown below:



Absolute Time	Counter	Chn	Identifier	FPS	Message Name	Type	Dir	DLC	Data Len.	BRS	ESI	00 01 02 03 04 05 06 07 08 09 10 11 12 13 14
16.477874	1	C...	000	0		Data	Tx	8	8	-	-	02 10 02 AA AA AA AA AA
26.336459	2	C...	000	0		Data	Tx	8	8	-	-	02 10 02 AA AA AA AA AA
44.318450	3	C...	000	0		Data	Tx	8	8	-	-	02 10 02 AA AA AA AA AA
45.362023	4	C...	000	0		Data	Tx	8	8	-	-	02 10 02 AA AA AA AA AA
45.867171	5	C...	000	0		Data	Tx	8	8	-	-	02 10 02 AA AA AA AA AA

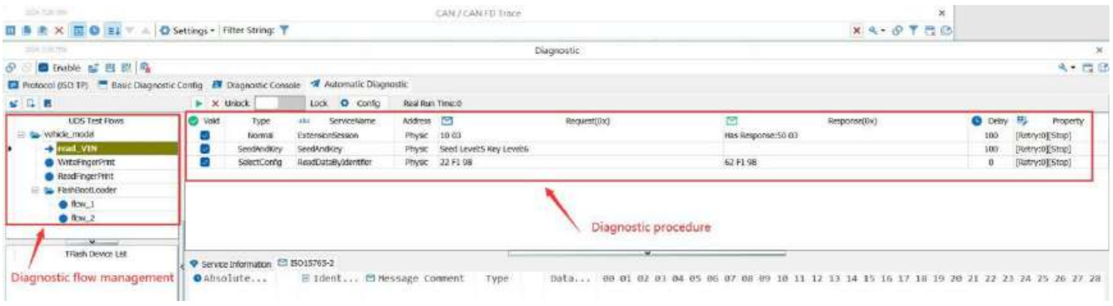
1.19.4.3. Diagnostics Controlled by External Program

The external programs can exchange data with TSMaster through the system variables.

1.19.5. Automatic Diagnostics Process

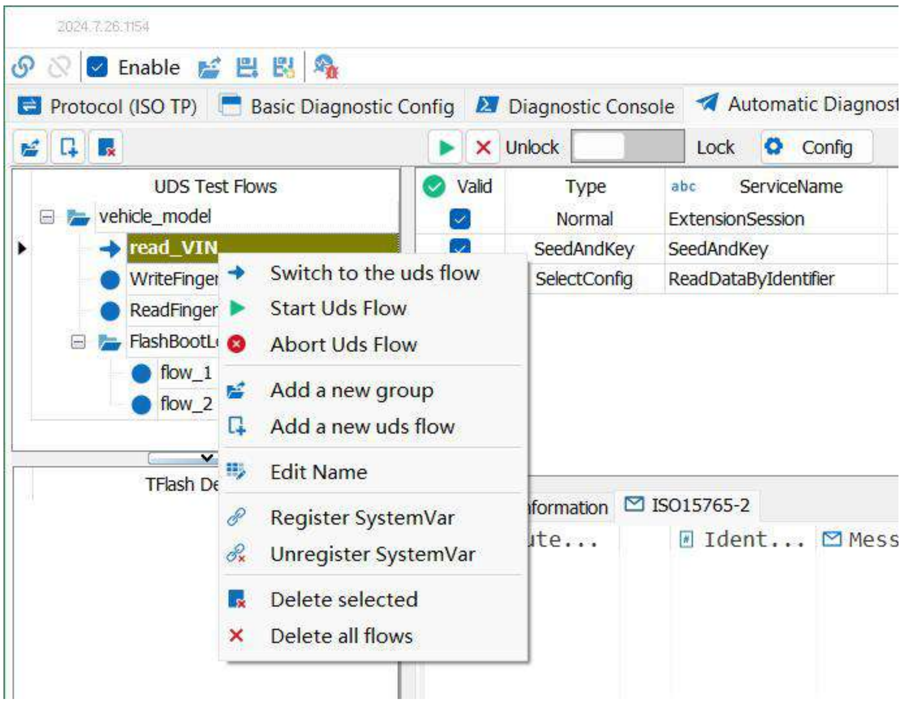
1.19.5.1. Workflow Use Case Management

TSMaster's automated diagnostic process is not only aimed at a specific application but manages the diagnostic process for the entire project. Users can configure test diagnostic process groups according to the needs of the complete project. Each group can contain multiple different diagnostic processes, and it is within a diagnostic process that specific diagnostic steps are included. As shown in the figure below:



Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
✓	Normal	ExtensionSession	Physic	10 03	Has Response:50 03	100	[Retry:0][Stop]
✓	SeedAndKey	SeedAndKey	Physic	Seed Level5 Key Level6		100	[Retry:0][Stop]
✓	SelectConfig	ReadDataByIdentifier	Physic	22 F1 98	62 F1 98	0	[Retry:0][Stop]

Right-click on the UDS process management bar to expand the workflow case management operation menu, as shown in the figure below:

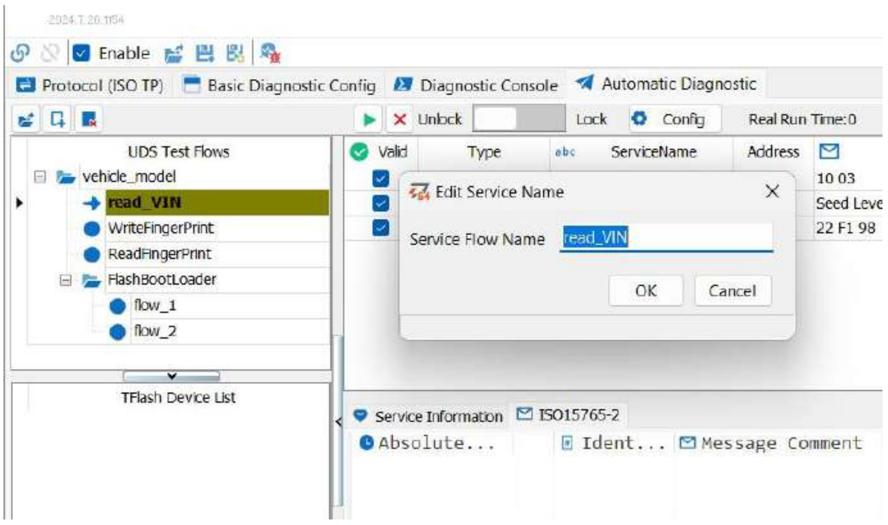


The main operations include:

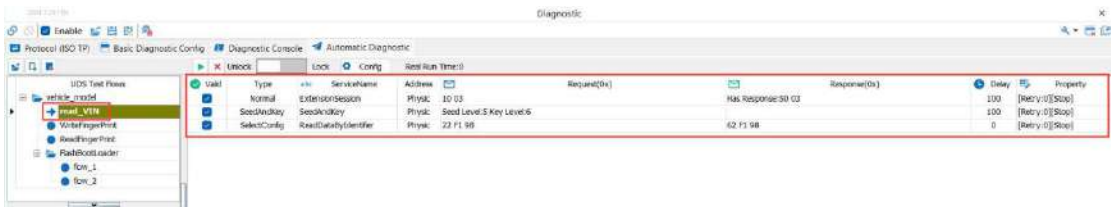
【1】 Add a new group: add a new diagnostic process group. For example, add a new vehicle model 1. Under the diagnostic group, user can further add diagnostic process cases, the case themselves do not contain diagnostic steps.

【2】 Add a new Uds flow: add a new diagnostic process case, and under the diagnostic process case, you can add detailed diagnostic steps.

【3】 Edit name: select a process group or process case, right-click and choose "Edit Name" to edit the name of the node, as shown in the figure below:



【4】 Switch to the current UDS process node. Double-clicking on the node will also switch to this process node. After switching to the node, the node icon and background color will be as shown in the figure below, and at the same time, the detailed diagnostic steps included in this UDS process will be displayed in the node process on the right.



Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
✓	Normal	ExtensionSession	Physic	10 03	Has Response:50 03	100	[Retry:0][Stop]
✓	SeedAndKey	SeedAndKey	Physic	Seed Level:5 Key Level:6		100	[Retry:0][Stop]
✓	SelectConfig	ReadDataByIdentifier	Physic	22 F1 98	62 F1 98	0	[Retry:0][Stop]

【5】 Start Uds flow: start the diagnostic process for this node. After selecting this option, the diagnostic module will automatically execute the diagnostic steps from top to bottom according to the configuration on the right side.

【6】 Abort Uds flow: after clicking on this node, interrupt the diagnostic process step that is currently being executed.

【7】 Delete selected: delete the selected node.

【8】 Delete all flows: clear all nodes.

1.19.5.2. Configure the Diagnostic Process（UDS Flow）

1.19.5.2.1. Basic Configuration Steps

Configure the diagnostic process, the basic steps are shown in the figure below:

【1】 Select a diagnostic process node in the management panel on the left.

【2】 In the editing area on the right, add, delete, and edit diagnostic steps.

【3】 After adding a step, edit the step's name.

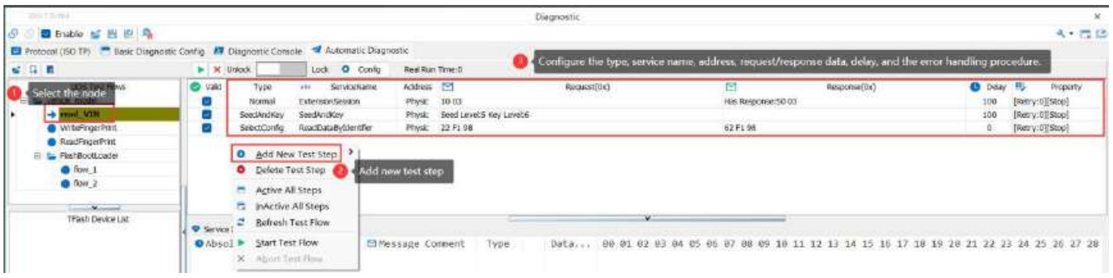
【4】 Select the type of the step.

【5】 Select the type of address for the step, whether it is a physical address or a functional address.

【6】 Configure the detailed diagnostic request and response data packets.

【7】 Configure the waiting time between steps after the completion of this step.

【8】 Configure the error handling method for this step in case of an error.



Configure the type, service name, address, request/response data, delay, and the error handling procedure.

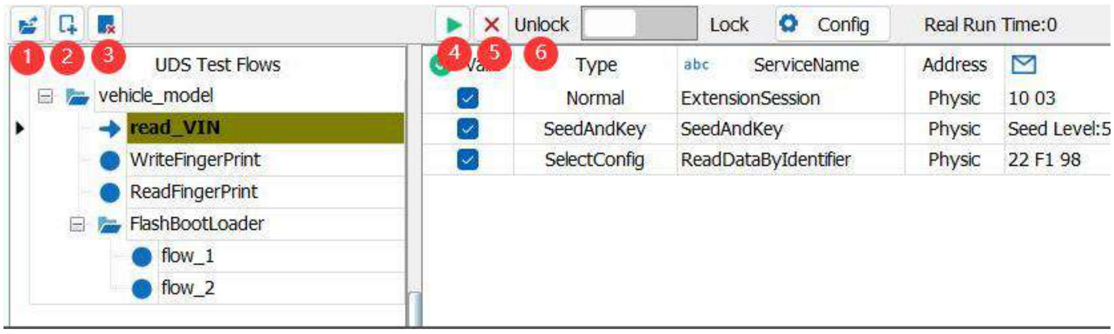
Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
✓	Normal	ExtensionSession	Physic	10 03	Has Response:50 03	100	[Retry:0][Stop]
✓	SeedAndKey	SeedAndKey	Physic	Seed Level:5 Key Level:6		100	[Retry:0][Stop]
✓	SelectConfig	ReadDataByIdentifier	Physic	22 F1 98	62 F1 98	0	[Retry:0][Stop]

```
Message Comment    Type    Data...  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
```

The above steps are the basic steps for configuring a diagnostic process. In actual use, more flexible mechanisms are provided according to the application scenario. Please continue to the following sections for more information.

1.19.5.2.2. Toolbar

The diagnostic process configuration toolbar is shown in the figure below:



Valid	Type	ServiceName	Address	
✓	Normal	ExtensionSession	Physic	10 03
✓	SeedAndKey	SeedAndKey	Physic	Seed Level:5
✓	SelectConfig	ReadDataByIdentifier	Physic	22 F1 98

【1】 Create a new diagnostic process group.

【2】 Create a new diagnostic process case.

【3】 Delete the selected diagnostic process group/case.

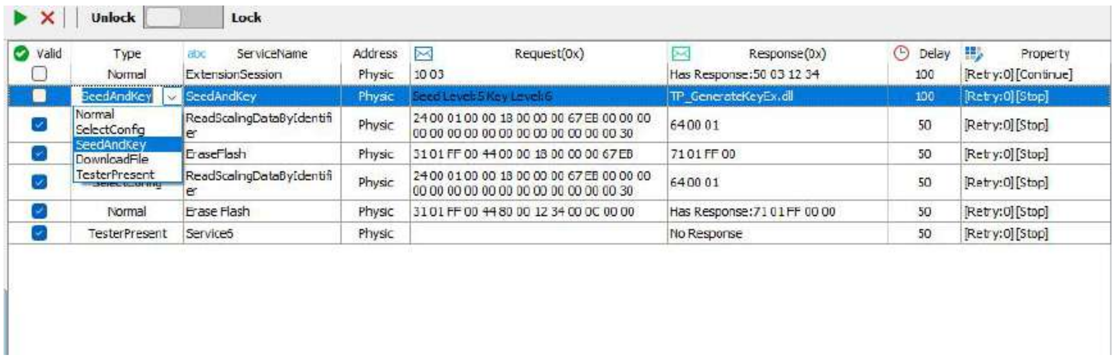
【4】 Start the configured diagnostic process.

【5】 Interrupt the diagnostic process that is currently running.

【6】 Lock/unlock the process configuration area. If the area is locked, the diagnostic process area becomes uneditable.

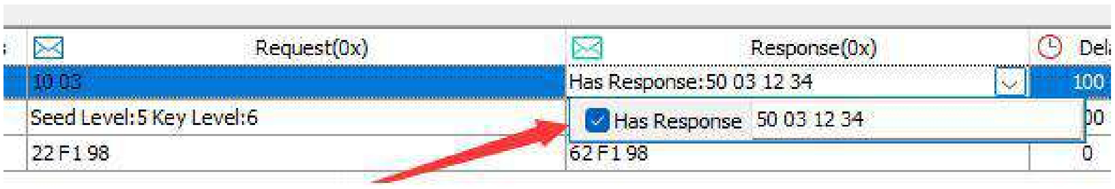
1.19.5.2.3. Diagnostic Step Types

In the test step, to increase the flexibility of diagnostic configuration, 5 types have been designed for selection, as shown in the figure below, mainly including: Normal, SelectConfig, SeedAndKey, DownloadFile, TesterPresent. These 5 types basically cover all the mainstream diagnostic process needs on the market. The following is a detailed introduction to the characteristics of each type.



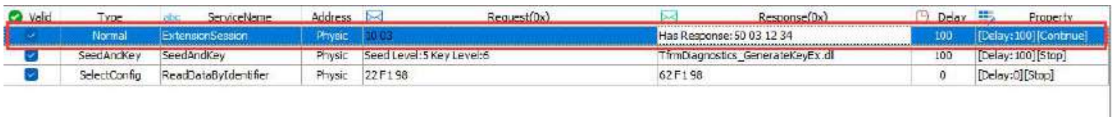
Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
☐	Normal	ExtensionSession	Physic	10 03	Has Response:50 03 12 34	100	[Retry:0] [Continue]
☐	SeedAndKey	SeedAndKey	Physic	Seed Level:5 Key Level:6	TP_GenerateKeyEx.dll	100	[Retry:0] [Stop]
✓		ReadScalingDataByIdentifier	Physic	24 00 01 00 00 18 00 00 00 67 EB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 30	64 00 01	50	[Retry:0] [Stop]
✓		EraseFlash	Physic	31 01 FF 00 44 00 00 18 00 00 00 67 EB	71 01 FF 00	50	[Retry:0] [Stop]
✓		ReadScalingDataByIdentifier	Physic	24 00 01 00 00 18 00 00 00 67 EB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 30	64 00 01	50	[Retry:0] [Stop]
✓	Normal	Erase Flash	Physic	31 01 FF 00 44 80 00 12 34 00 0C 00 00	Has Response:71 01 FF 00 00	50	[Retry:0] [Stop]
✓	TesterPresent	Service6	Physic		No Response	50	[Retry:0] [Stop]

【1】 **Normal**: normal configuration. This configuration is mainly used for simple situations where both the request data and the response data are very clear. For example, if the service request data is 【10 03】 and the service response data is 【50 03 12 34】, user can choose the Normal type. The configuration of Normal is the simplest, just enter the desired request data 【10 03】 in the Request field and the expected response message 【50 03 12 34】 in the Response field. When configuring the response message, expand as shown below:



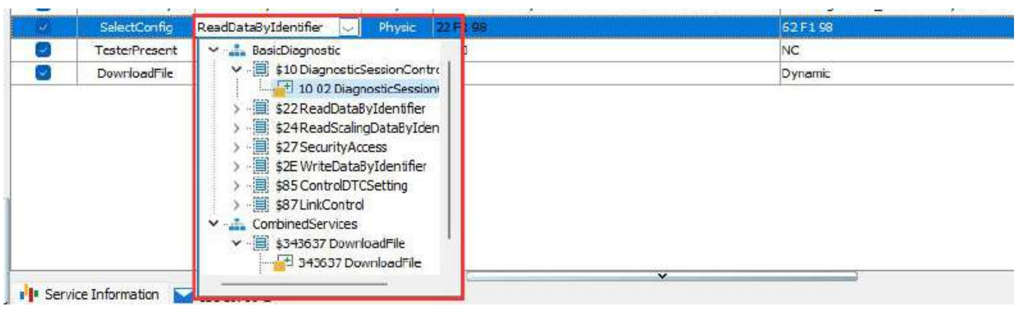
Request(0x)	Response(0x)	Del
10 03	Has Response:50 03 12 34	100
Seed Level:5 Key Level:6		00
22 F1 98	62 F1 98	0

In some test cases, the ECU does not respond. For such situations, users only need to uncheck the "Has Response" option. The effect after completing the configuration is shown below:



Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
✓	Normal	ExtensionSession	Physic	10 03	Has Response: 50 03 12 34	100	[Delay:100] [Continue]
✓	SeedAndKey	SeedAndKey	Physic	Seed Level:5 Key Level:6	TfmDiagnostics_GenerateKeyEx.dll	100	[Delay:100] [Stop]
✓	SelectConfig	ReadDataByIdentifier	Physic	22 F1 98	62 F1 98	0	[Delay:0] [Stop]

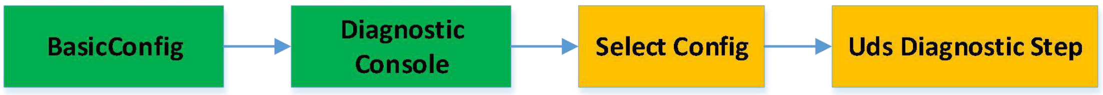
【2】 **SelectConfig**: select an existing configuration. The purpose of this configuration is to allow users to select diagnostic steps that have already been debugged in the Diagnostic Console. The selection process is shown in the figure below:



✓	SelectConfig	ReadDataByIdentifier	Physic	22 F1 98	62 F1 98
✓	TesterPresent				NC
✓	DownloadFile				Dynamic

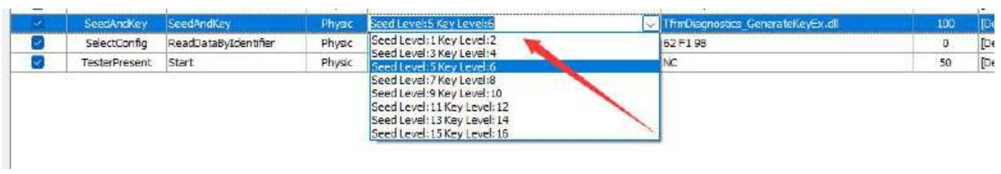
- BasicDiagnostic
  - $10 DiagnosticSessionContr
    - 10 02 DiagnosticSession
  - $22 ReadDataByIdentifier
  - $24 ReadScalingDataByIden
  - $27 SecurityAccess
  - $2E WriteDataByIdentifier
  - $85 ControlDTCSetting
  - $87 LinkControl
- CombinedServices
  - $343637 DownloadFile
    - 343637 DownloadFile

This method is the most recommended configuration approach by TSmaster. Users can first configure and test all the sub-processes in the Diagnostic Console, and then reference this configuration in the automatic diagnostic process. The logic is shown in the figure below:



During the execution of the automatic diagnostic process, its performance will be exactly the same as in the Diagnostic Console.

【3】 **SeedAndKey**: the SeedAndKey is a composite command that cannot be configured directly with a Normal command. Users can either select it from existing configurations via SelectConfig, or choose the SeedAndKey type to directly configure decryption steps within the automated process. The SeedAndKey only requires the selection of the SeedLevel parameter, and the decryption DLL is directly associated with the SeedAndKey DLL loaded in the TP parameter configuration, as shown in the figure below:



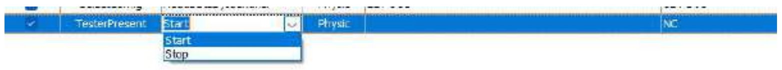
✓	SeedAndKey	SeedAndKey	Physic	Seed Level:5 Key Level:6	TfrmDiagnostics_GenerateKeyEx.dll	100	[De
✓	SelectConfig	ReadDataByIdentifier	Physic		62 F1 98	0	[De
✓	TesterPresent	Start	Physic		NC	50	[De

Seed Level:1 Key Level:2
Seed Level:3 Key Level:4
Seed Level:5 Key Level:6
Seed Level:7 Key Level:8
Seed Level:9 Key Level:10
Seed Level:11 Key Level:12
Seed Level:13 Key Level:14
Seed Level:15 Key Level:16

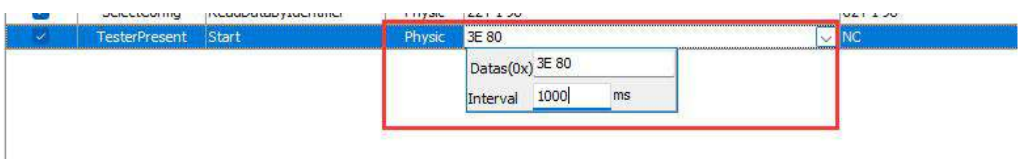
It is evident that the prerequisite for correct operation, whether in the Diagnostic Console module or the Automatic Diagnostic module, is that users must correctly complete the TP layer parameter configuration.

【4】 **TesterPresent:** As previously mentioned, TSMaster provides a global switch for TesterPresent, which allows users to directly activate and deactivate the command. In addition, to support more flexible testing requirements, the automated process steps also offer a way to configure this command based on the step, allowing users to choose when to activate and deactivate the TesterPresent command. After selecting this type, there are mainly two parameters that need to be configured:

➢ Whether to activate/deactivate the command, as follows:



➢ Configure the command data and the periodic interval, as follows:


TesterPresent | Start | Physic | 3E 80 | NC

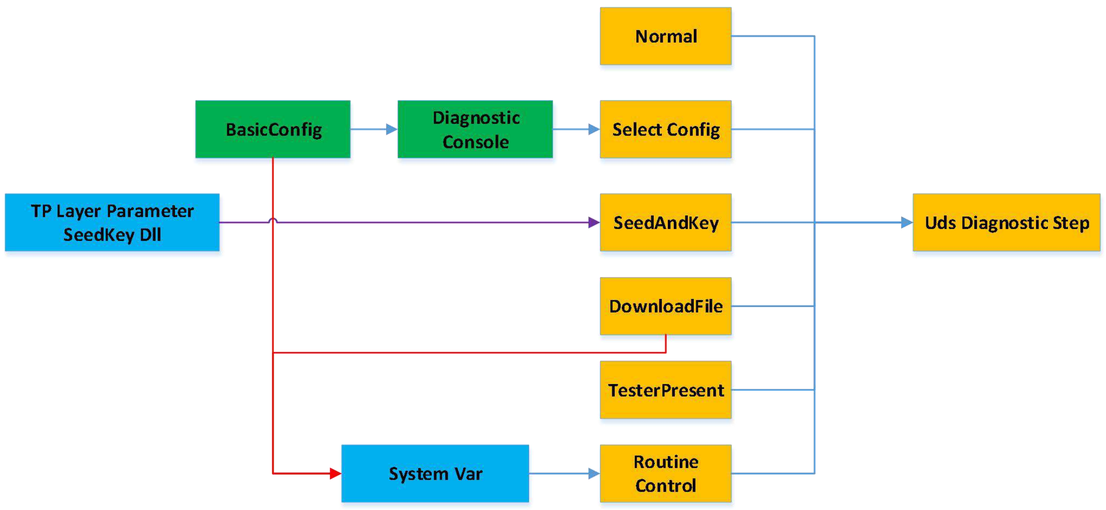
Datas(0x) 3E 80

Interval 1000 ms

【5】 **SystemVarEvent:** to further enhance the extensibility of the diagnostic module, an action based on system variable change events has been added. When the diagnostic module reaches this step, it triggers a variable change event in the script tool by writing parameters into the configured system variables. Within the script, users can extend their own business requirements. For example, in the variable change event, user can implement control over external programmable power supplies or access to data servers.

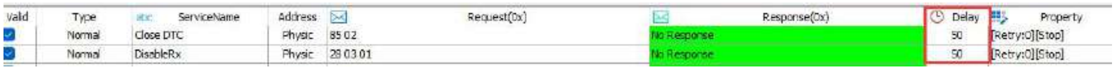
【Diagnostic Step Configuration Summary】

Summarizing the configuration process of the test steps from the previous section, its logical flow is composed as shown in the figure below:



1.19.5.2.4. Step Interval

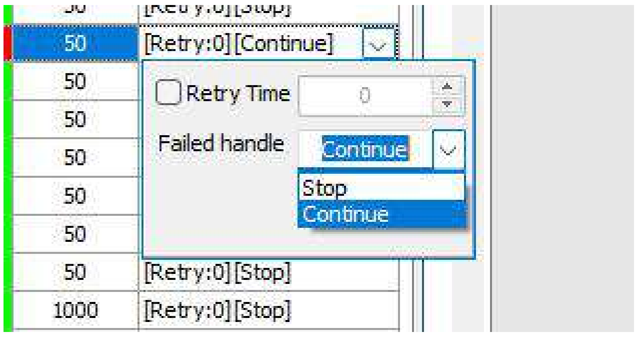
The time interval between steps in the diagnostic process module can be set, as shown in the figure below, with the unit being milliseconds (ms).



Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
✓	Normal	Close DTC	Physic	85 02	No Response	50	[Retry:0][Stop]
✓	Normal	DisableRx	Physic	28 03 01	No Response	50	[Retry:0][Stop]

1.19.5.2.5. Error Handling

At the current stage, error handling mainly includes two parameters: the number of retries after an error and whether to stop or continue running after an error, as shown in the figure below:



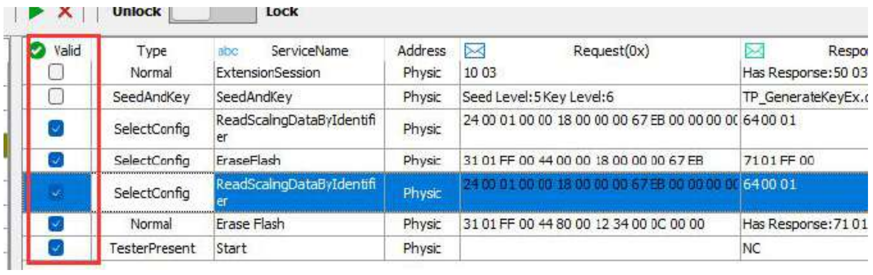
50	[Retry:0][Continue]
50	
50	
50	
50	
50	
50	[Retry:0][Stop]
1000	[Retry:0][Stop]

Retry Time: 0
Failed handle: Continue

In the future product planning, error handling will allow for jumping to a specified process (for example, jumping to an erase process), further increasing the flexibility of the automatic operation process module.

1.19.5.2.6. Enable Step/Position Adjustment

For the diagnostic process steps that have been configured, users can check the boxes on the left to select the diagnostic steps they want to execute, as shown in the figure below:



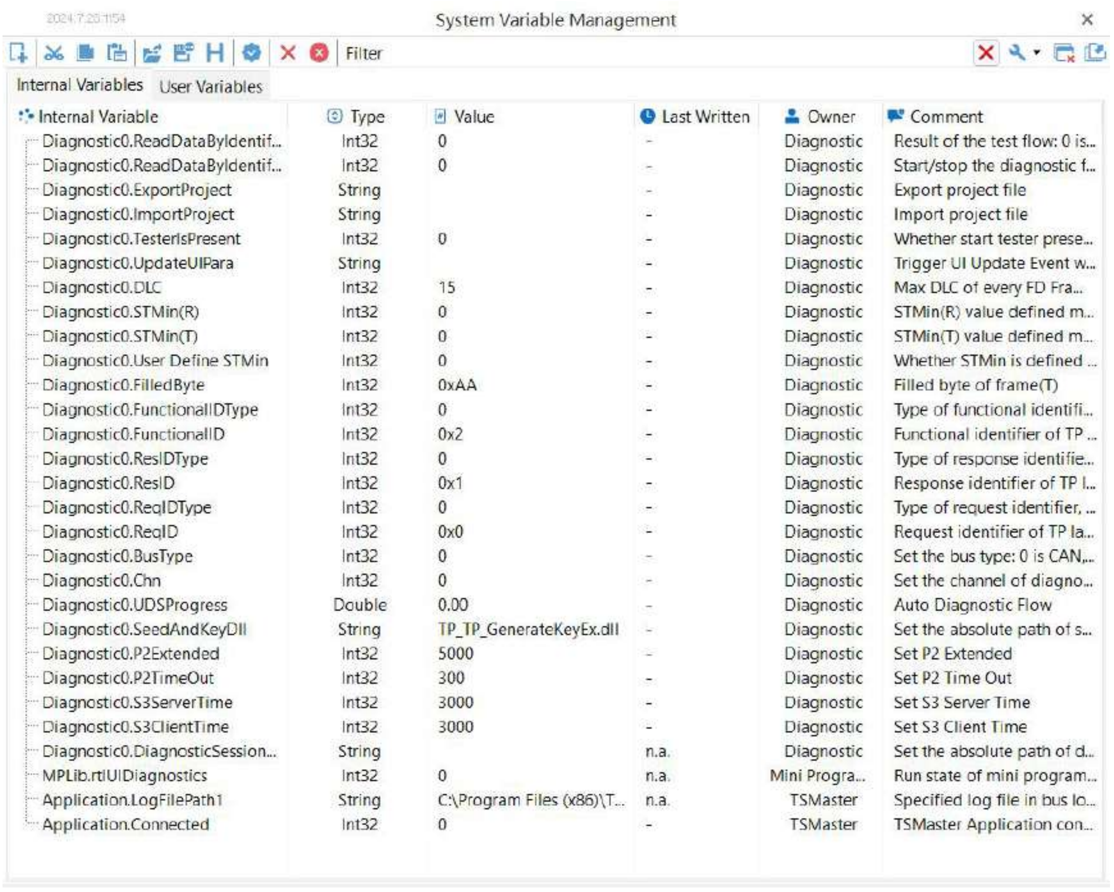
Valid	Type	ServiceName	Address	Request(0x)	Respo
☐	Normal	ExtensionSession	Physic	10 03	Has Response:50 03
☐	SeedAndKey	SeedAndKey	Physic	Seed Level:5 Key Level:6	TP_GenerateKeyEx.c
✓	SelectConfig	ReadScalingDataByIdentifier	Physic	24 00 01 00 00 18 00 00 00 67 EB 00 00 00 0(	64 00 01
✓	SelectConfig	EraseFlash	Physic	31 01 FF 00 44 00 00 18 00 00 00 67 EB	71 01 FF 00
✓	SelectConfig	ReadScalingDataByIdentifier	Physic	24 00 01 00 00 18 00 00 00 67 EB 00 00 00 0(	64 00 01
✓	Normal	Erase Flash	Physic	31 01 FF 00 44 80 00 12 34 00 0C 00 00	Has Response:71 01
✓	TesterPresent	Start	Physic		NC

Regarding the adjustment of execution order: whether it's a test case group, a test case, or specific steps within a test case, when users want to adjust the execution order, they can simply drag and drop the corresponding test case to the desired position.

1.19.5.3. Programmable Automatic Diagnostic Process

1.19.5.3.1. Diagnostic Module Common System Variables

After adding the basic diagnostic module in TSMaster, the System Variable Manager will automatically generate the system variables for this diagnostic module (as shown in below figure). By modifying the system variables, user can configure the corresponding parameters.


System Variable Management

Filter

Internal Variables | User Variables

Internal Variable	Type	Value	Last Written	Owner	Comment
Diagnostic0.ReadDataByIdentif...	Int32	0	-	Diagnostic	Result of the test flow: 0 is...
Diagnostic0.ReadDataByIdentif...	Int32	0	-	Diagnostic	Start/stop the diagnostic f...
Diagnostic0.ExportProject	String		-	Diagnostic	Export project file
Diagnostic0.ImportProject	String		-	Diagnostic	Import project file
Diagnostic0.TesterIsPresent	Int32	0	-	Diagnostic	Whether start tester prese...
Diagnostic0.UpdateUIPara	String		-	Diagnostic	Trigger UI Update Event w...
Diagnostic0.DLC	Int32	15	-	Diagnostic	Max DLC of every FD Fra...
Diagnostic0.STMin(R)	Int32	0	-	Diagnostic	STMin(R) value defined m...
Diagnostic0.STMin(T)	Int32	0	-	Diagnostic	STMin(T) value defined m...
Diagnostic0.User Define STMin	Int32	0	-	Diagnostic	Whether STMin is defined ...
Diagnostic0.FilledByte	Int32	0xAA	-	Diagnostic	Filled byte of frame(T)
Diagnostic0.FunctionalIDType	Int32	0	-	Diagnostic	Type of functional identifi...
Diagnostic0.FunctionalID	Int32	0x2	-	Diagnostic	Functional identifier of TP ...
Diagnostic0.ResIDType	Int32	0	-	Diagnostic	Type of response identifie...
Diagnostic0.ResID	Int32	0x1	-	Diagnostic	Response identifier of TP l...
Diagnostic0.ReqIDType	Int32	0	-	Diagnostic	Type of request identifier, ...
Diagnostic0.ReqID	Int32	0x0	-	Diagnostic	Request identifier of TP la...
Diagnostic0.BusType	Int32	0	-	Diagnostic	Set the bus type: 0 is CAN,...
Diagnostic0.Chn	Int32	0	-	Diagnostic	Set the channel of diagno...
Diagnostic0.UDSProgress	Double	0.00	-	Diagnostic	Auto Diagnostic Flow
Diagnostic0.SeedAndKeyDll	String	TP_TP_GenerateKeyEx.dll	-	Diagnostic	Set the absolute path of s...
Diagnostic0.P2Extended	Int32	5000	-	Diagnostic	Set P2 Extended
Diagnostic0.P2TimeOut	Int32	300	-	Diagnostic	Set P2 Time Out
Diagnostic0.S3ServerTime	Int32	3000	-	Diagnostic	Set S3 Server Time
Diagnostic0.S3ClientTime	Int32	3000	-	Diagnostic	Set S3 Client Time
Diagnostic0.DiagnosticSession...	String		n.a.	Diagnostic	Set the absolute path of d...
MPLib.rtIUIDiagnostics	Int32	0	n.a.	Mini Progra...	Run state of mini program...
Application.LogFilePath1	String	C:\Program Files (x86)\T...	n.a.	TSMaster	Specified log file in bus lo...
Application.Connected	Int32	0	-	TSMaster	TSMaster Application con...

System Variables Corresponding to Diagnostic Module

After creating a diagnostic module, the corresponding system variables will be automatically generated in the System Variable Manager, which mainly include the following types:

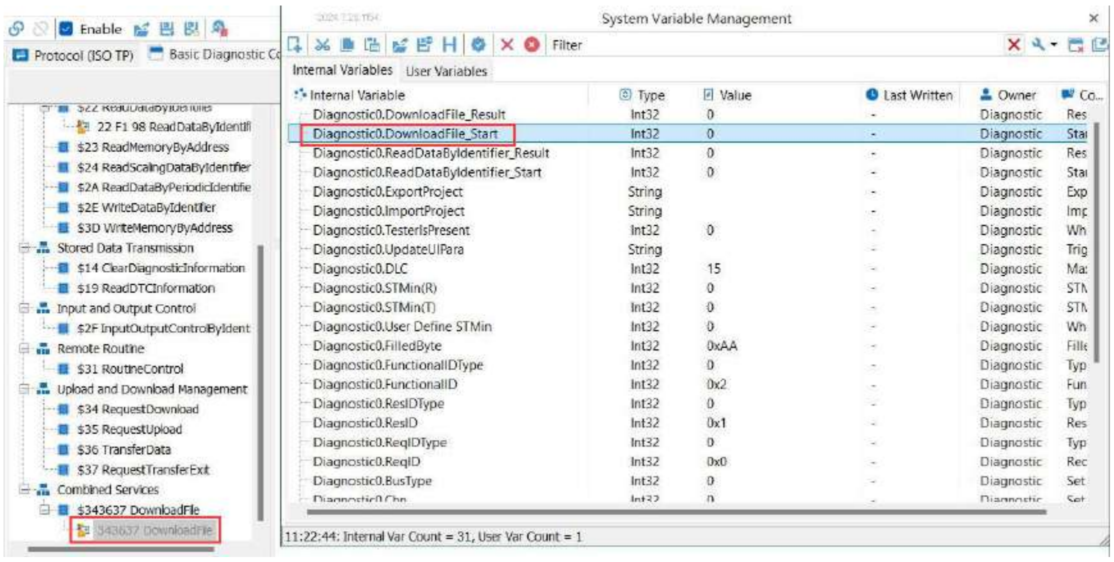
- ➢ Bus Type——BusType（Int32）: CAN = 0, CANFD = 2, LIN = 2, DOIP = 3；
- ➢ Channel——Chn（Int32）: CH1 = 0, CH2 = 1, CH3 = 2, ……；

- ➢ Request ID——ReqID（Int32）: physical addressing ID;
- ➢ Request ID Type——ReqIDType（Int32）: standard frame = 0, extended frame = 1;
- ➢ Response ID——ResID（Int32）: diagnostic response ID;
- ➢ Response ID Type——ResIDType（Int32）: standard frame = 0, extended frame = 1;
- ➢ Function ID——FunctionalID（Int32）: function addressing ID;
- ➢ Function ID Type——FunctionalIDType（Int32）: standard frame = 0, extended frame = 1;
- ➢ Padding Bytes——FilledByte（Int32）: padding byte data in addition to the effective data.
- ➢ Diagnoser Online——TesterPresent（Int32）: enable diagnoser online.
- ➢ Secure Access Seed&Key——SeedAndKeyDll（String）: the physical address of the SeedKey algorithm DLL. Pay attention to escape characters when using it.
- ➢ Automation Process Progress——UDSProgress（Double）: real-time progress of the automatic diagnostic process. This variable is used to obtain the running status of the automatic diagnostic process.

1.19.5.3.2. Diagnostic Service Specific System Variables

The loading and switching of the download files in combined service:

After adding a new service in Basic Diagnostic Config -> Combined Services, the System Variable Manager will also generate the corresponding system variable (ServiceName_DataFile). This variable is the physical address of the download file. Modifying this variable can control the loading and switching of the download file.



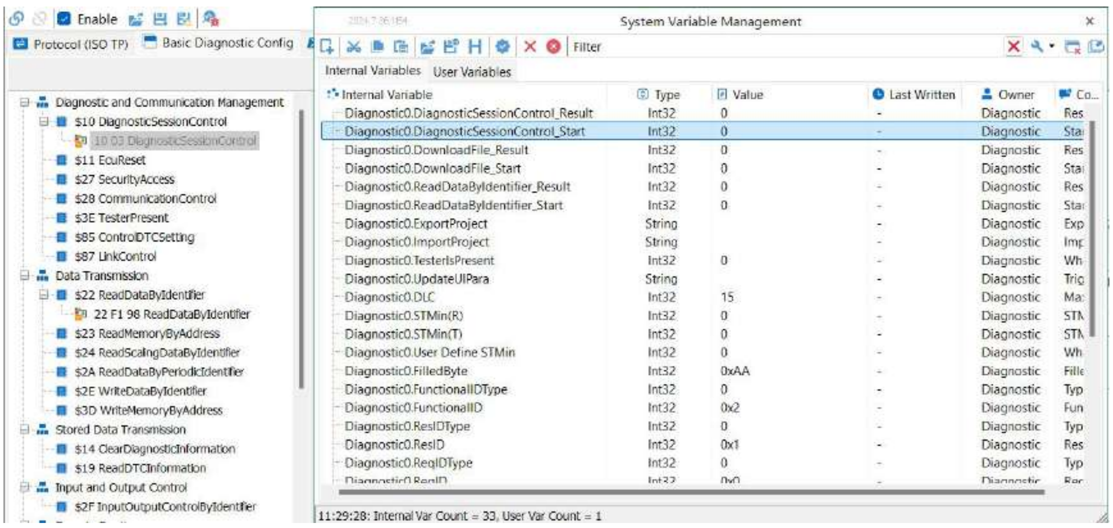
Internal Variable	Type	Value	Last Written	Owner	Co...
Diagnostic0.DownloadFile_Result	Int32	0	-	Diagnostic	Res
Diagnostic0.DownloadFile_Start	Int32	0	-	Diagnostic	Sta
Diagnostic0.ReadDataByIdentifier_Result	Int32	0	-	Diagnostic	Res
Diagnostic0.ReadDataByIdentifier_Start	Int32	0	-	Diagnostic	Sta
Diagnostic0.ExportProject	String		-	Diagnostic	Exp
Diagnostic0.ImportProject	String		-	Diagnostic	Imp
Diagnostic0.TesterIsPresent	Int32	0	-	Diagnostic	Wh
Diagnostic0.UpdateUIPara	String		-	Diagnostic	Trig
Diagnostic0.DLC	Int32	15	-	Diagnostic	Ma:
Diagnostic0.STMin(R)	Int32	0	-	Diagnostic	STN
Diagnostic0.STMin(T)	Int32	0	-	Diagnostic	STN
Diagnostic0.User Define STMin	Int32	0	-	Diagnostic	Wh
Diagnostic0.FilledByte	Int32	0xAA	-	Diagnostic	Fille
Diagnostic0.FunctionalIDType	Int32	0	-	Diagnostic	Typ
Diagnostic0.FunctionalID	Int32	0x2	-	Diagnostic	Fun
Diagnostic0.ResIDType	Int32	0	-	Diagnostic	Typ
Diagnostic0.ResID	Int32	0x1	-	Diagnostic	Res
Diagnostic0.ReqIDType	Int32	0	-	Diagnostic	Typ
Diagnostic0.ReqID	Int32	0x0	-	Diagnostic	Rec
Diagnostic0.BusType	Int32	0	-	Diagnostic	Set
Diagnostic0.Chn	Int32	0	-	Diagnostic	Set

11:22:44: Internal Var Count = 31, User Var Count = 1

Notes: after loading the download file, the system variable controller will generate the checksum for each block and the total checksum, as well as the starting address and length of the download file, based on the selected checksum algorithm. If a combined diagnostic service has been added, a download file has been loaded, and the download file-related variables have been associated in the basic diagnostic service, then when the download file is replaced, these associated variables will also change accordingly.

1.19.5.3.3. Basic Diagnostic Services and Automated Programmable Control of the Diagnostic Process

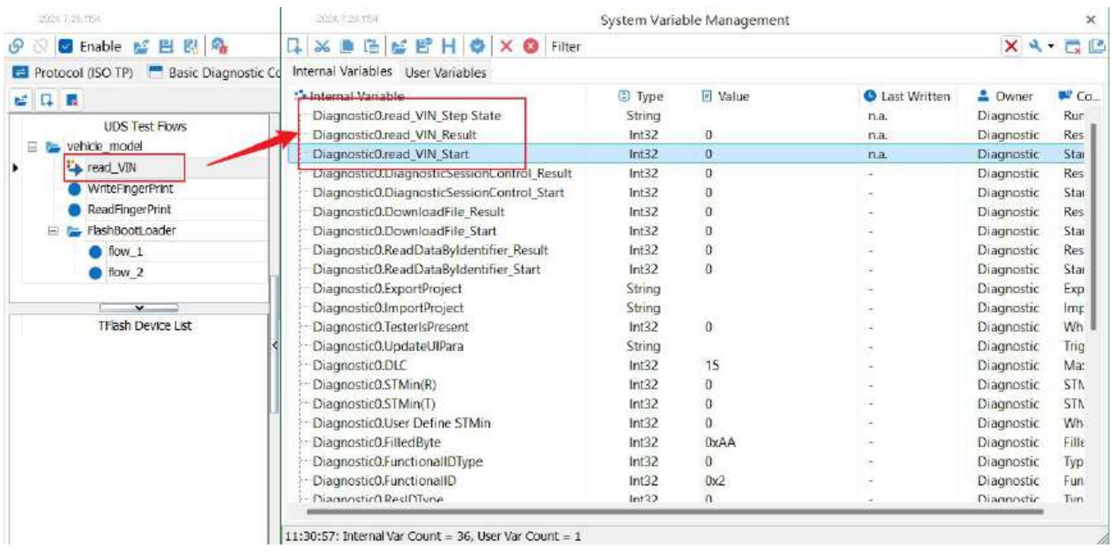
Existing diagnostic services and automatic diagnostic processes can be registered as system variables by right-clicking. After registration, the System Variable Manager will generate programmable control variables for the corresponding service or process (ServiceName_Start, ServiceName_Result). The *Start variable is used to initiate the execution of the service or process (assigning a value of 1 is sufficient), and *Result indicates the execution result of the service or process (0 is default, 1 is running, 2 is success, 3 is failure).



- Diagnostic and Communication Management
  - $10 DiagnosticSessionControl
    - 10 03 DiagnosticSessionControl
  - $11 EcuReset
  - $27 SecurityAccess
  - $28 CommunicationControl
  - $3E TesterPresent
  - $85 ControlDTCSetting
  - $87 LinkControl
- Data Transmission
  - $22 ReadDataByIdentifier
    - 22 F1 98 ReadDataByIdentifier
  - $23 ReadMemoryByAddress
  - $24 ReadScalingDataByIdentifier
  - $2A ReadDataByPeriodicIdentifier
  - $2E WriteDataByIdentifier
  - $3D WriteMemoryByAddress
- Stored Data Transmission
  - $14 ClearDiagnosticInformation
  - $19 ReadDTCInformation
- Input and Output Control
  - $2F InputOutputControlByIdentifier

Internal Variable	Type	Value	Last Written	Owner	Co...
Diagnostic0.DiagnosticSessionControl_Result	Int32	0	-	Diagnostic	Res
Diagnostic0.DiagnosticSessionControl_Start	Int32	0	-	Diagnostic	Sta
Diagnostic0.DownloadFile_Result	Int32	0	-	Diagnostic	Res
Diagnostic0.DownloadFile_Start	Int32	0	-	Diagnostic	Sta
Diagnostic0.ReadDataByIdentifier_Result	Int32	0	-	Diagnostic	Res
Diagnostic0.ReadDataByIdentifier_Start	Int32	0	-	Diagnostic	Sta
Diagnostic0.ExportProject	String		-	Diagnostic	Exp
Diagnostic0.ImportProject	String		-	Diagnostic	Imp
Diagnostic0.TesterIsPresent	Int32	0	-	Diagnostic	Wh
Diagnostic0.UpdateUIPara	String		-	Diagnostic	Trig
Diagnostic0.DLC	Int32	15	-	Diagnostic	Ma
Diagnostic0.STMin(R)	Int32	0	-	Diagnostic	STN
Diagnostic0.STMin(T)	Int32	0	-	Diagnostic	STN
Diagnostic0.User Define STMin	Int32	0	-	Diagnostic	Wh
Diagnostic0.FilledByte	Int32	0xAA	-	Diagnostic	Fille
Diagnostic0.FunctionalIDType	Int32	0	-	Diagnostic	Typ
Diagnostic0.FunctionalID	Int32	0x2	-	Diagnostic	Fun
Diagnostic0.ResIDType	Int32	0	-	Diagnostic	Typ
Diagnostic0.ResID	Int32	0x1	-	Diagnostic	Res
Diagnostic0.ReqIDType	Int32	0	-	Diagnostic	Typ

11:29:28: Internal Var Count = 33, User Var Count = 1



UDS Test Flows
- vehicle_model
  - read_VIN
  - WriteFingerPrint
  - ReadFingerPrint
  - FlashBootLoader
    - flow_1
    - flow_2

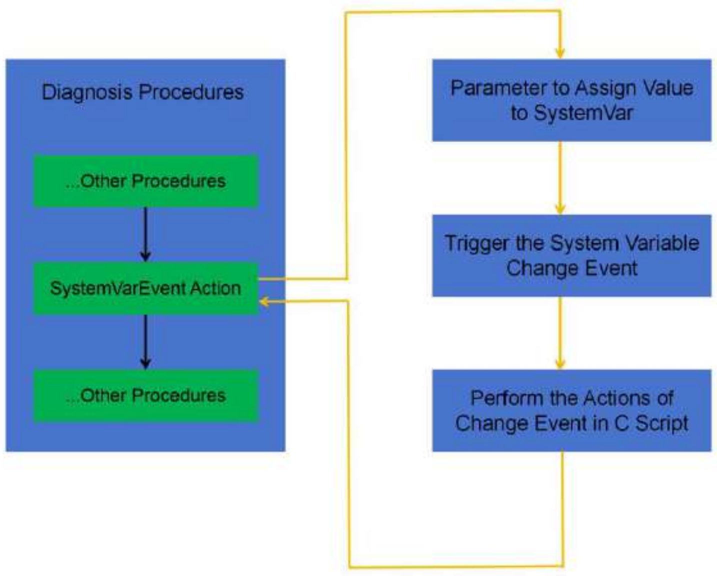
Internal Variable	Type	Value	Last Written	Owner	Co...
Diagnostic0.read_VIN_Step State	String		n.a.	Diagnostic	Rur
Diagnostic0.read_VIN_Result	Int32	0	n.a.	Diagnostic	Res
Diagnostic0.read_VIN_Start	Int32	0	n.a.	Diagnostic	Sta
Diagnostic0.DiagnosticSessionControl_Result	Int32	0	-	Diagnostic	Res
Diagnostic0.DiagnosticSessionControl_Start	Int32	0	-	Diagnostic	Sta
Diagnostic0.DownloadFile_Result	Int32	0	-	Diagnostic	Res
Diagnostic0.DownloadFile_Start	Int32	0	-	Diagnostic	Sta
Diagnostic0.ReadDataByIdentifier_Result	Int32	0	-	Diagnostic	Res
Diagnostic0.ReadDataByIdentifier_Start	Int32	0	-	Diagnostic	Sta
Diagnostic0.ExportProject	String		-	Diagnostic	Exp
Diagnostic0.ImportProject	String		-	Diagnostic	Imp
Diagnostic0.TesterIsPresent	Int32	0	-	Diagnostic	Wh
Diagnostic0.UpdateUIPara	String		-	Diagnostic	Trig
Diagnostic0.DLC	Int32	15	-	Diagnostic	Ma
Diagnostic0.STMin(R)	Int32	0	-	Diagnostic	STN
Diagnostic0.STMin(T)	Int32	0	-	Diagnostic	STN
Diagnostic0.User Define STMin	Int32	0	-	Diagnostic	Wh
Diagnostic0.FilledByte	Int32	0xAA	-	Diagnostic	Fille
Diagnostic0.FunctionalIDType	Int32	0	-	Diagnostic	Typ
Diagnostic0.FunctionalID	Int32	0x2	-	Diagnostic	Fun

11:30:57: Internal Var Count = 36, User Var Count = 1

Note: a Diagnostic License is required for automatic diagnostic process through system variable.

1.19.6. Diagnostic Extension Applications

TSMaster's diagnostic module solves the configuration process of automated diagnostic processes (such as the flashing process) through a graphical configuration method. In the application phase (such as on the production line), in addition to basic diagnostic processes, it is often necessary to interact with external devices, network modules, etc. For example, during the flashing process, it is necessary to control the power-on and power-off of the device under test through the power module; or to pull data from a remote server, etc. The diagnostic module extends this requirement through the behavior of SystemVarEvent, and the basic mechanism of its function is shown below:

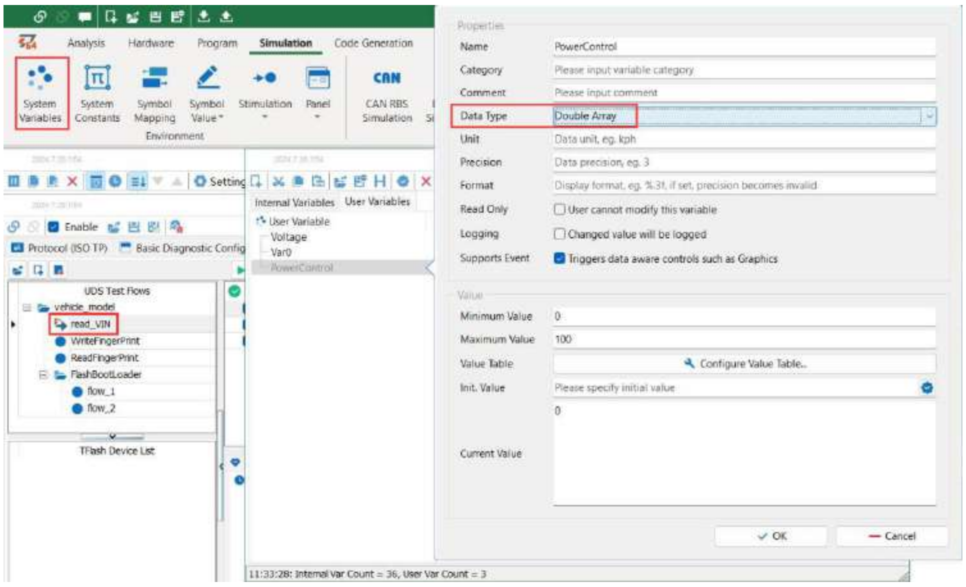

Diagnosis Procedures
...Other Procedures
SystemVarEvent Action
...Other Procedures
Parameter to Assign Value to SystemVar
Trigger the System Variable Change Event
Perform the Actions of Change Event in C Script

In the following sections of this chapter, several typical application examples will be used to introduce this extension application.

1.19.6.1. Controlling the DUT Power during the Diagnostic Flashing Process

This example primarily shows how to control the power on and off through the SystemVarEvent behavior. The steps are as follows:

➢ Firstly, create a system variable named PowerControl. Regarding the data type of the system variable, it should be set according to the actual application scenario. For example, in this requirement, it is necessary to control the power on state and set the output voltage of the power supply, which are two parameters, and among them, the voltage value is a floating-point number. Therefore, the system variable type is set as a double array. The system variable is shown in the figure below:



Properties

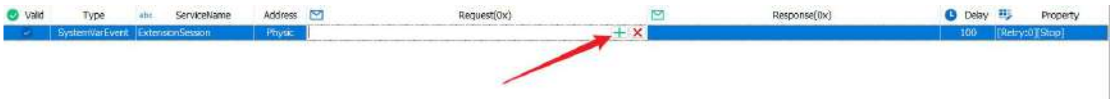
Name	PowerControl
Category	Please input variable category
Comment	Please input comment
Data Type	Double Array
Unit	Data unit, eg. kph
Precision	Data precision, eg. 3
Format	Display format, eg. %.3f, if set, precision becomes invalid
Read Only	☐ User cannot modify this variable
Logging	☐ Changed value will be logged
Supports Event	☑ Triggers data aware controls such as Graphics

Value

Minimum Value	0
Maximum Value	100
Value Table	Configure Value Table..
Init. Value	Please specify initial value
Current Value	0

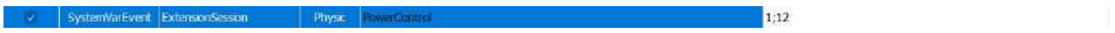
11:33:28: Internal Var Count = 36, User Var Count = 3

➢ In the diagnostic process configuration tool, first select the step type as: SystemVarEvent (System Variable Event), and then associate it with the system variable, as shown in the figure below:



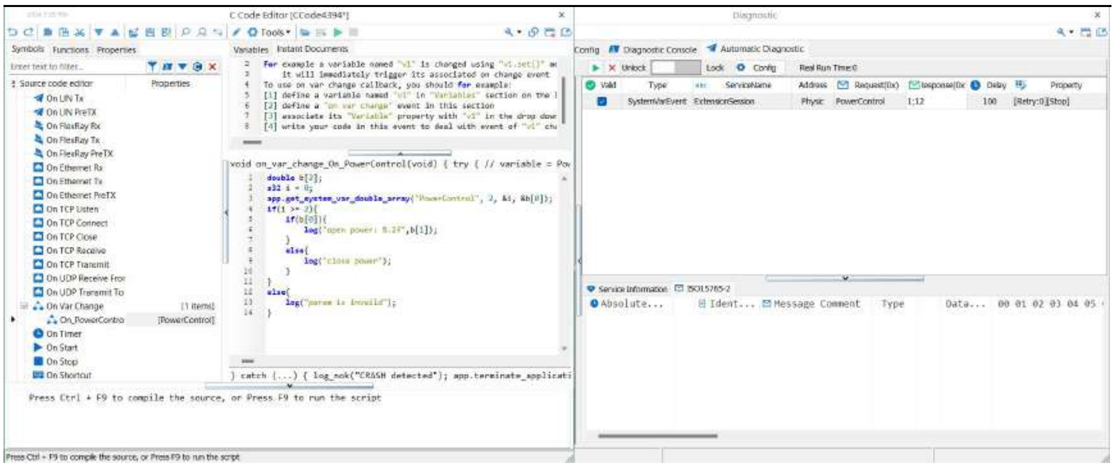
Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
✓	SystemVarEvent	ExtensionSession	Physic			100	[Retry:0][Stop]

➢ Configure the system variable parameter value to: 1, 12.0, which indicates to turn on the power supply and set the output voltage to 12V.



✓	SystemVarEvent	ExtensionSession	Physic	PowerControl	1;12

➢ Open a script tool and create a system variable change event based on this variable, as shown in the figure below:



```
For example a variable named "v1" is changed using "v1.set()" ...
    it will immediately trigger its associated on change event
To use on var change callback, you should for example:
[1] define a variable named "v1" in "Variables" section on the ...
[2] define a "on var change" event in this section
[3] associate its "Variable" property with "v1" in the drop dow...
[4] write your code in this event to deal with event of "v1" ch...
```

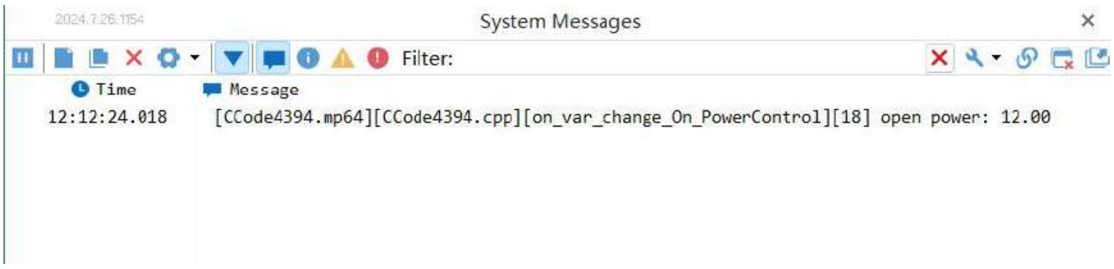
```
void on_var_change_On_PowerControl(void) { try { // variable = Pow...
double b[2];
s32 i = 0;
app.get_system_var_double_array("PowerControl", 2, &i, &b[0]);
if(i >= 2){
    if(b[0]){
        log("open power: %.2f",b[1]);
    }
    else{
        log("close power");
    }
}
else{
    log("param is invalid");
}
} catch (...) { log_nok("CRASH detected"); app.terminate_applicati...
Press Ctrl + F9 to compile the source, or Press F9 to run the script
```

Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
✓	SystemVarEvent	ExtensionSession	Physic	PowerControl	1:12	100	[Retry:0][Stop]

The codes are as follows:

```
double b[2];
s32 i = 2;
app.get_system_var_double_array("Diagnostic0.PowerControl", 2, &i, &b[0]);
if(i >= 2){
  if(b[0]){
    //Real Open Power Codes
    log("Open Power: %.2f", b[1]);
  } else {
     //Real Close Power Codes
    log("Close Power");
  }
}else{
   log("Parameter is invalid");
}
```

➢ Run the diagnostic process, and through the system messages, we can see that the logic for power control has been executed:



```
Time            Message
12:12:24.018    [CCode4394.mp64][CCode4394.cpp][on_var_change_On_PowerControl][18] open power: 12.00
```

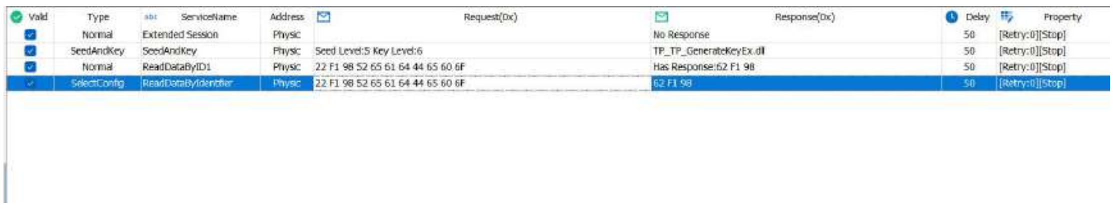
1.19.6.2. Send Several Messages during the Flashing Process

As shown in the previous section, within the script's variable change event, the message sending function is called to send the required CAN messages, as shown below:

1.19.6.3. Interact with the Server after the Flashing Process is Completed (TBD)

1.19.7. Typical Applications

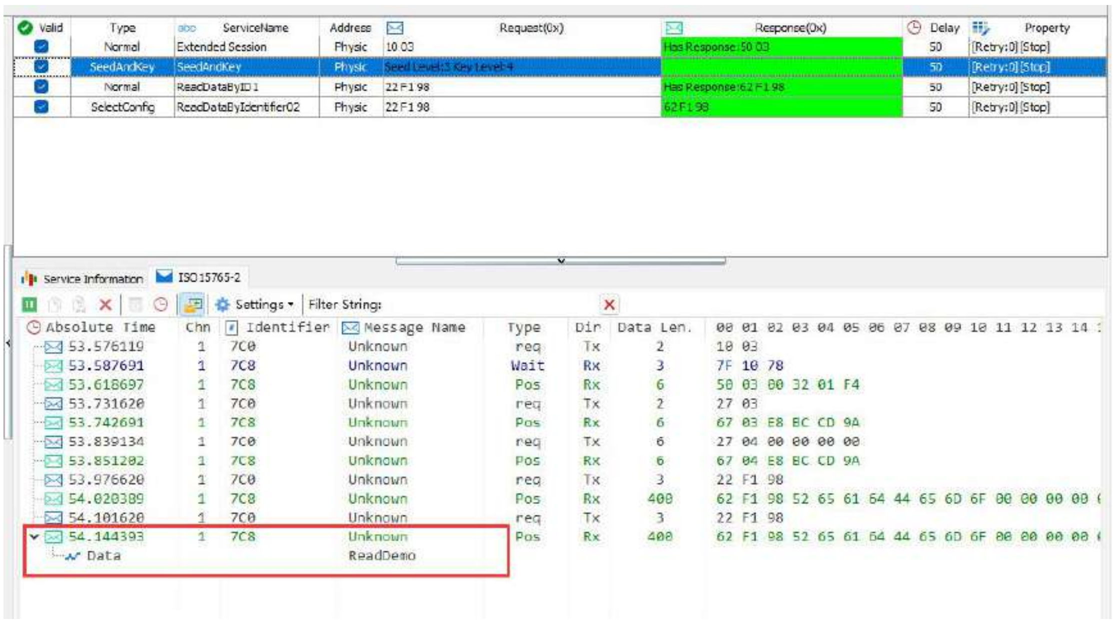
1.19.7.1. Read Vehicle VIN Code



Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
✓	Normal	Extended Session	Physic		No Response	50	[Retry:0][Stop]
✓	SeedAndKey	SeedAndKey	Physic	Seed Level:5 Key Level:6	TP_TP_GenerateKeyEx.dll	50	[Retry:0][Stop]
✓	Normal	ReadDataByID1	Physic	22 F1 98 52 65 61 64 44 65 60 6F	Has Response:62 F1 98	50	[Retry:0][Stop]
✓	SelectConfig	ReadDataByIdentifier	Physic	22 F1 98 52 65 61 64 44 65 60 6F	62 F1 98	50	[Retry:0][Stop]

This application first switch to the extended session, and then get the permission and use the normal reading command to read the VIN code returned by the vehicle (or refer to a previously configured command to read the VIN code, which includes data parsing).

The execution effect is shown as follows:



Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
✓	Normal	Extended Session	Physic	10 03	Has Response:50 03	50	[Retry:0] [Stop]
✓	SeedAndKey	SeedAndKey	Physic	Seed Level:3 Key Level:4		50	[Retry:0] [Stop]
✓	Normal	ReadDataByID1	Physic	22 F1 98	Has Response:62 F1 98	50	[Retry:0] [Stop]
✓	SelectConfig	ReadDataByIdentifier02	Physic	22 F1 98	62 F1 98	50	[Retry:0] [Stop]

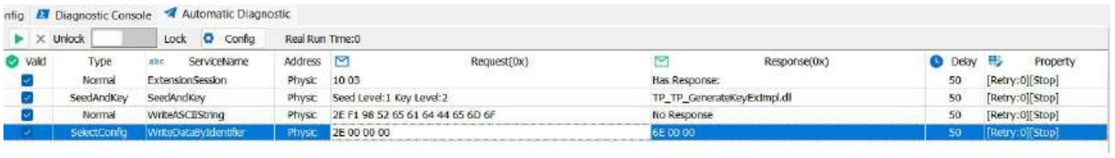
Service Information | ISO15765-2

Settings | Filter String:

Absolute Time	Chn	Identifier	Message Name	Type	Dir	Data Len.	00 01 02 03 04 05 06 07 08 09 10 11 12 13 14
53.576119	1	7C0	Unknown	req	Tx	2	10 03
53.587691	1	7C8	Unknown	Wait	Rx	3	7F 10 78
53.618697	1	7C8	Unknown	Pos	Rx	6	50 03 00 32 01 F4
53.731620	1	7C0	Unknown	req	Tx	2	27 03
53.742691	1	7C8	Unknown	Pos	Rx	6	67 03 E8 BC CD 9A
53.839134	1	7C0	Unknown	req	Tx	6	27 04 00 00 00 00
53.851202	1	7C8	Unknown	Pos	Rx	6	67 04 E8 BC CD 9A
53.976620	1	7C0	Unknown	req	Tx	3	22 F1 98
54.020389	1	7C8	Unknown	Pos	Rx	400	62 F1 98 52 65 61 64 44 65 6D 6F 00 00 00 00
54.101620	1	7C0	Unknown	req	Tx	3	22 F1 98
54.144393	1	7C8	Unknown	Pos	Rx	400	62 F1 98 52 65 61 64 44 65 6D 6F 00 00 00 00
Data			ReadDemo				

It can be seen that, for the same ReadDataByID, in the test step that references BasicConfig, because it includes the corresponding parsing information, the string that can be directly read is "ReadDemo".

1.19.7.2. Write Configuration Information

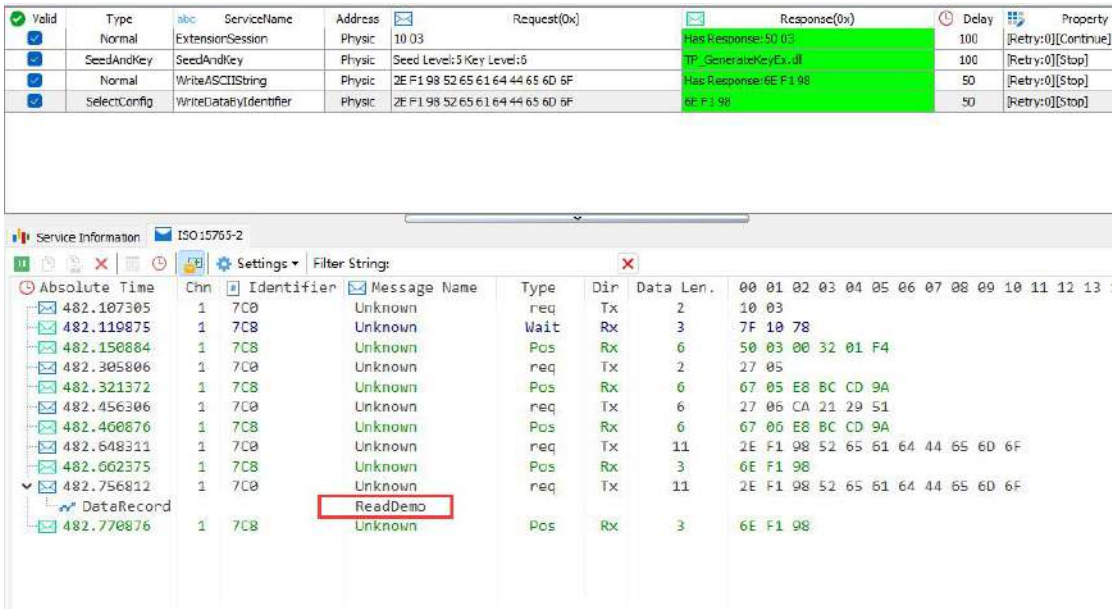


nfig | Diagnostic Console | Automatic Diagnostic

Unlock | Lock | Config | Real Run Time:0

Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
✓	Normal	ExtensionSession	Physic	10 03	Has Response:	50	[Retry:0][Stop]
✓	SeedAndKey	SeedAndKey	Physic	Seed Level:1 Key Level:2	TP_TP_GenerateKeyExImpl.dll	50	[Retry:0][Stop]
✓	Normal	WriteASCIIString	Physic	2E F1 98 52 65 61 64 44 65 6D 6F	No Response	50	[Retry:0][Stop]
✓	SelectConfig	WriteDataByIdentifier	Physic	2E 00 00 00	6E 00 00	50	[Retry:0][Stop]

The execution effect is shown as follows:



Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
✓	Normal	ExtensionSession	Physic	10 03	Has Response: 50 03	100	[Retry:0][Continue]
✓	SeedAndKey	SeedAndKey	Physic	Seed Level:5 Key Level:6	TP_GenerateKeyEx.dll	100	[Retry:0][Stop]
✓	Normal	WriteASCIIString	Physic	2E F1 98 52 65 61 64 44 65 6D 6F	Has Response:6E F1 98	50	[Retry:0][Stop]
✓	SelectConfig	WriteDataByIdentifier	Physic	2E F1 98 52 65 61 64 44 65 6D 6F	6E F1 98	50	[Retry:0][Stop]

Service Information | ISO 15765-2

Settings ▾ | Filter String:

Absolute Time	Chn	Identifier	Message Name	Type	Dir	Data Len.	00 01 02 03 04 05 06 07 08 09 10 11 12 13
482.107305	1	7C0	Unknown	req	Tx	2	10 03
482.119875	1	7C8	Unknown	Wait	Rx	3	7F 10 78
482.150884	1	7C8	Unknown	Pos	Rx	6	50 03 00 32 01 F4
482.305806	1	7C0	Unknown	req	Tx	2	27 05
482.321372	1	7C8	Unknown	Pos	Rx	6	67 05 E8 BC CD 9A
482.456306	1	7C0	Unknown	req	Tx	6	27 06 CA 21 29 51
482.460876	1	7C8	Unknown	Pos	Rx	6	67 06 E8 BC CD 9A
482.648311	1	7C0	Unknown	req	Tx	11	2E F1 98 52 65 61 64 44 65 6D 6F
482.662375	1	7C8	Unknown	Pos	Rx	3	6E F1 98
482.756812	1	7C0	Unknown	req	Tx	11	2E F1 98 52 65 61 64 44 65 6D 6F
DataRecord			ReadDemo				
482.770876	1	7C8	Unknown	Pos	Rx	3	6E F1 98

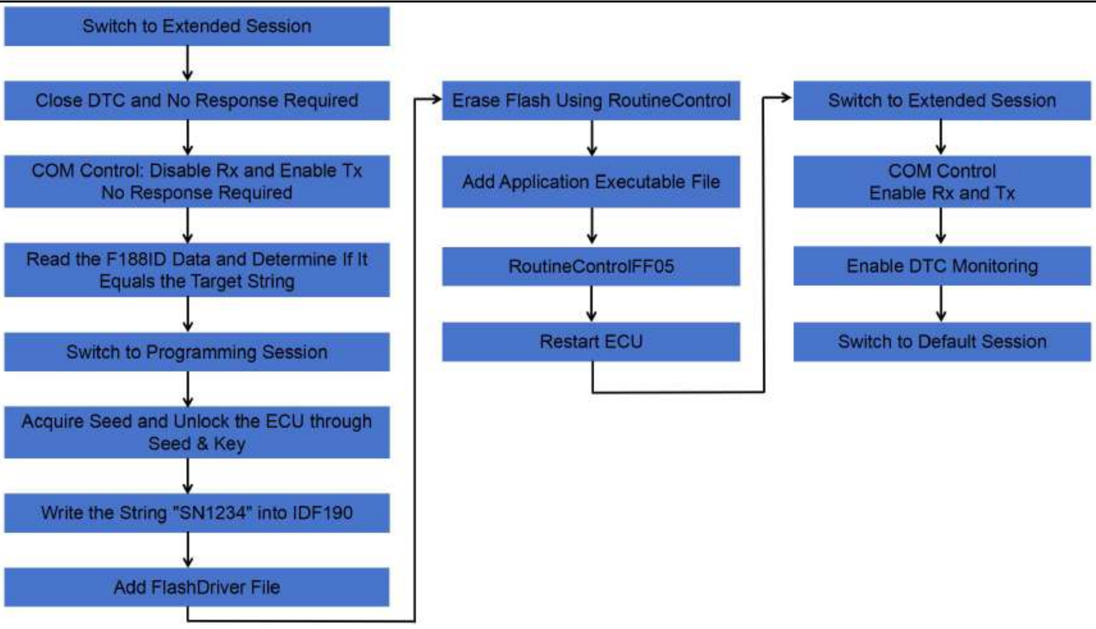
It can be seen that, for the same WriteDataByID, in the test step that references BasicConfig, because it includes the corresponding parsing information, the string that can be directly read is "ReadDemo".

1.19.7.3. Flash Bootloader

This chapter designs a simple Bootloader process to show how to configure a Flash Bootloader process based on the TSMaster diagnostic module.

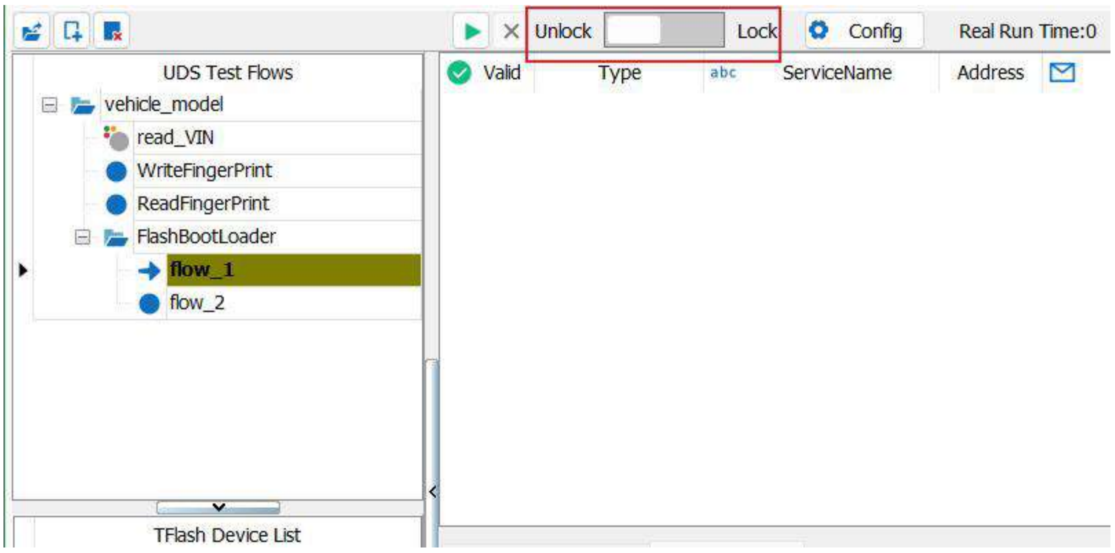
1.19.7.3.1. Flash Bootloader Process

Firstly, the FlashBootloader process is designed as shown below. This is an example process, and users can adjust it according to their actual design specifications.

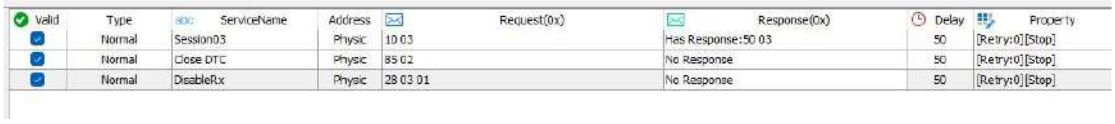

Switch to Extended Session
Close DTC and No Response Required
COM Control: Disable Rx and Enable Tx No Response Required
Read the F188ID Data and Determine If It Equals the Target String
Switch to Programming Session
Acquire Seed and Unlock the ECU through Seed & Key
Write the String "SN1234" into IDF190
Add FlashDriver File
Erase Flash Using RoutineControl
Add Application Executable File
RoutineControlFF05
Restart ECU
Switch to Extended Session
COM Control Enable Rx and Tx
Enable DTC Monitoring
Switch to Default Session

1.19.7.3.2. Configure the Flashing Example

【1】 First, create the Demo1 process: Note to switch the editor to Unlock status, otherwise, it is not allowed to add new process steps.



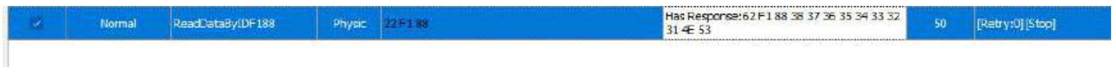
【2】 For the session switching, DTC clearing, and COM control commands shown in the flowchart, user can directly configure them as Normal type commands (note that these commands can, of course, also be configured in BasicConfig and referenced here). As shown below:



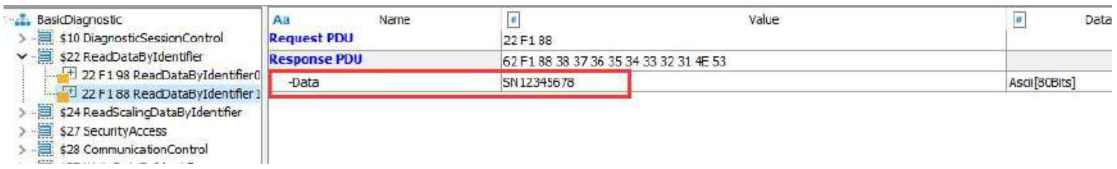
Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
✓	Normal	Session03	Physic	10 03	Has Response:50 03	50	[Retry:0][Stop]
✓	Normal	Close DTC	Physic	85 02	No Response	50	[Retry:0][Stop]
✓	Normal	DisableRx	Physic	28 03 01	No Response	50	[Retry:0][Stop]

【3】 Based on ReadDataByID, read the data at the location with ID=F188, and determine if the data equals, for example, SN12345678. If it matches, it is determined that the part number matches and the process proceeds to the next step. Otherwise, the process is exited.
The configuration is as follows:

Method 1: configure it directly as a Normal type, as shown below:

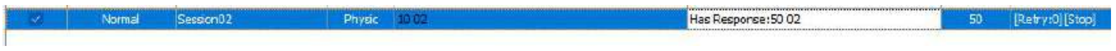

Normal | ReadDataByIDF188 | Physic | 22 F1 88 | Has Response:62 F1 88 38 37 36 35 34 33 32 31 4E 53 | 50 | [Retry:0][Stop]

Method 2: configure it in BasicConfig, and then reference it in the process:

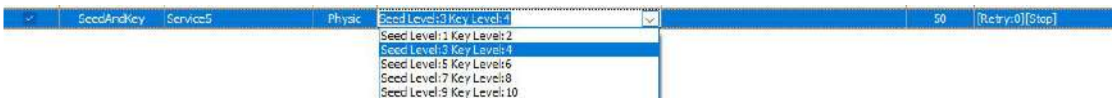

BasicDiagnostic
- $10 DiagnosticSessionControl
- $22 ReadDataByIdentifier
  - 22 F1 98 ReadDataByIdentifier0
  - 22 F1 88 ReadDataByIdentifier1
- $24 ReadScalingDataByIdentifier
- $27 SecurityAccess
- $28 CommunicationControl

Name	Value	Data
Request PDU	22 F1 88	
Response PDU	62 F1 88 38 37 36 35 34 33 32 31 4E 53	
-Data	SN12345678	Ascii[80Bits]

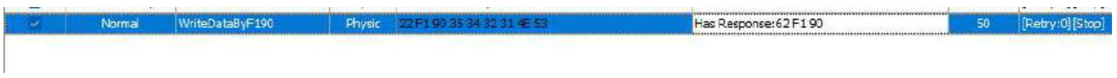
【4】 Switch to the programming session.


Normal | Session02 | Physic | 10 02 | Has Response:50 02 | 50 | [Retry:0][Stop]

【5】 Add a SeedAndKey step to unlock the ECU. The configuration is as follows:

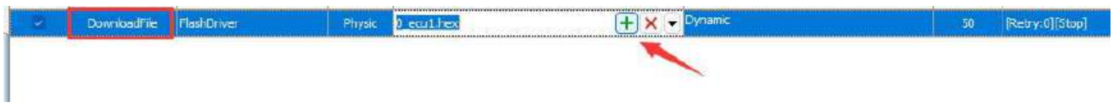

SeedAndKey | Service5 | Physic | Seed Level:3 Key Level:4 | 50 | [Retry:0][Stop]
Seed Level:1 Key Level:2
Seed Level:3 Key Level:4
Seed Level:5 Key Level:6
Seed Level:7 Key Level:8
Seed Level:9 Key Level:10

【6】 After obtaining permission, write the string "SN1234" at the location with ID F190. For such a fixed string that needs to be written, the most straightforward method is to directly configure a Normal step. As shown below:

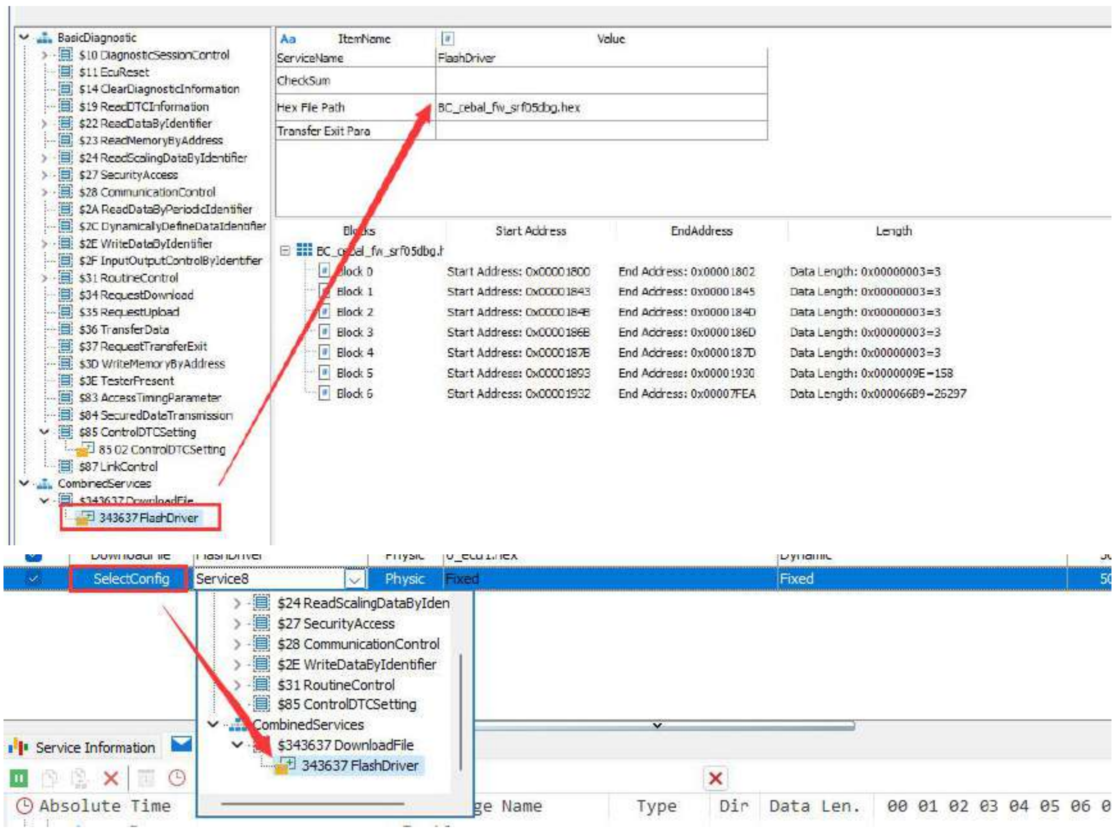

Normal | WriteDataByF190 | Physic | 22 F1 90 35 34 33 31 4E 53 | Has Response:62 F1 90 | 50 | [Retry:0][Stop]

【7】 Add the FlashDriver/ application files. Whether it's a FlashDriver or an application file, the method of adding them is the same. There are two methods:

Method 1: directly add a DownloadFile step, as shown below, add the executable file:


DownloadFile | FlashDriver | Physic | 0_ecu1.hex | Dynamic | 50 | [Retry:0][Stop]

Method 2: Configure BasicConfig, and then reference it in the Flow, as shown below:



- BasicDiagnostic
  - $10 DiagnosticSessionControl
  - $11 EcuReset
  - $14 ClearDiagnosticInformation
  - $19 ReadDTCInformation
  - $22 ReadDataByIdentifier
  - $23 ReadMemoryByAddress
  - $24 ReadScalingDataByIdentifier
  - $27 SecurityAccess
  - $28 CommunicationControl
  - $2A ReadDataByPeriodicIdentifier
  - $2C DynamicallyDefineDataIdentifier
  - $2E WriteDataByIdentifier
  - $2F InputOutputControlByIdentifier
  - $31 RoutineControl
  - $34 RequestDownload
  - $35 RequestUpload
  - $36 TransferData
  - $37 RequestTransferExit
  - $3D WriteMemoryByAddress
  - $3E TesterPresent
  - $83 AccessTimingParameter
  - $84 SecuredDataTransmission
  - $85 ControlDTCSetting
    - 85 02 ControlDTCSetting
  - $87 LinkControl
- CombinedServices
  - $343637 DownloadFile
    - 343637 FlashDriver

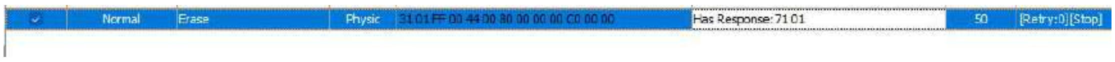
ItemName	Value
ServiceName	FlashDriver
CheckSum	
Hex File Path	BC_rebal_fw_srf05dbg.hex
Transfer Exit Para	

Blocks	Start Address	EndAddress	Length
BC_rebal_fw_srf05dbg.h			
Block 0	Start Address: 0x00001800	End Address: 0x00001802	Data Length: 0x00000003=3
Block 1	Start Address: 0x00001843	End Address: 0x00001845	Data Length: 0x00000003=3
Block 2	Start Address: 0x0000184B	End Address: 0x0000184D	Data Length: 0x00000003=3
Block 3	Start Address: 0x0000186B	End Address: 0x0000186D	Data Length: 0x00000003=3
Block 4	Start Address: 0x0000187B	End Address: 0x0000187D	Data Length: 0x00000003=3
Block 5	Start Address: 0x00001893	End Address: 0x00001930	Data Length: 0x0000009E=158
Block 6	Start Address: 0x00001932	End Address: 0x00007FEA	Data Length: 0x000066B9=26297

✓	SelectConfig	Service8	Physic	Fixed	Fixed	50

- $24 ReadScalingDataByIden
- $27 SecurityAccess
- $28 CommunicationControl
- $2E WriteDataByIdentifier
- $31 RoutineControl
- $85 ControlDTCSetting
- CombinedServices
  - $343637 DownloadFile
    - 343637 FlashDriver

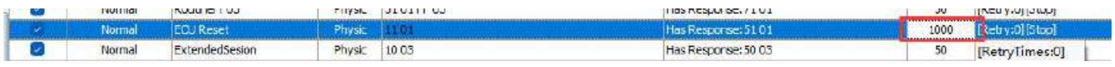
【8】 Use RoutineControl to erase the Flash. Because the address and length here are fixed, user can directly configure the fixed values as follows:



✓	Normal	Erase	Physic	31 01 FF 00 44 00 80 00 00 00 C0 00 00	Has Response: 71 01	50	[Retry:0][Stop]

If the address and length are dynamically changing, please refer to the subsequent sections to solve this issue by introducing system variables.

【9】 Restart the ECU by directly adding a NormalStep as shown below. Noted that the waiting time between the ECUReset and the restart of diagnostics should be adjusted according to the ECU design specifications. Here, the waiting time is set to 1000ms:

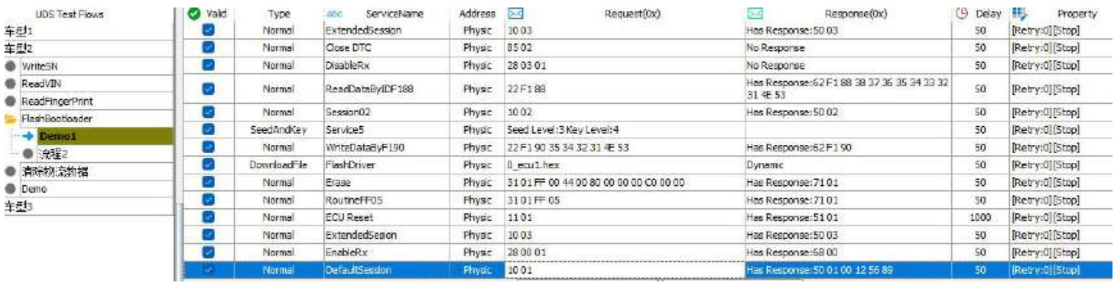


✓	Normal	ECU Reset	Physic	11 01	Has Response: 51 01	1000	[Retry:0][Stop]
✓	Normal	ExtendedSesion	Physic	10 03	Has Response: 50 03	50	[RetryTimes:0]

【10】 The remaining operations such as switching the default session, COM control, DTC control, etc., can be completed following the steps described earlier.

1.19.7.3.3. Summary

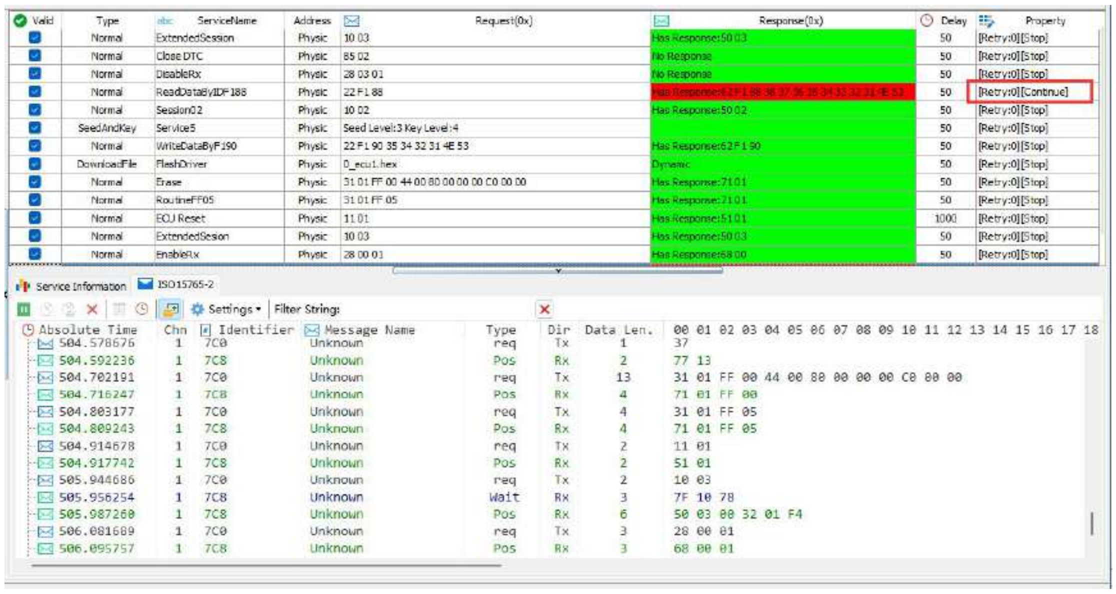
After completing the configuration, the overall configuration process is shown as follows. It can be seen that with the help of TSMaster's diagnostic module, developing diagnostic processes such as Bootloader applications has become a very simple task.



UDS Test Flows
- 车型1
- 车型2
  - WriteSN
  - ReadVIN
  - ReadFingerPrint
  - FlashBootloader
    - Demo1
    - 流程2
  - 清除物流数据
  - Demo
- 车型3

Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
✓	Normal	ExtendedSession	Physic	10 03	Has Response:50 03	50	[Retry:0][Stop]
✓	Normal	Close DTC	Physic	85 02	No Response	50	[Retry:0][Stop]
✓	Normal	DisableRx	Physic	28 03 01	No Response	50	[Retry:0][Stop]
✓	Normal	ReadDataByIDF188	Physic	22 F1 88	Has Response:62 F1 88 38 37 36 35 34 33 32 31 4E 53	50	[Retry:0][Stop]
✓	Normal	Session02	Physic	10 02	Has Response:50 02	50	[Retry:0][Stop]
✓	SeedAndKey	Service5	Physic	Seed Level:3 Key Level:4		50	[Retry:0][Stop]
✓	Normal	WriteDataByF190	Physic	22 F1 90 35 34 32 31 4E 53	Has Response:62 F1 90	50	[Retry:0][Stop]
✓	DownloadFile	FlashDriver	Physic	0_ecu1.hex	Dynamic	50	[Retry:0][Stop]
✓	Normal	Erase	Physic	31 01 FF 00 44 00 80 00 00 00 C0 00 00	Has Response:71 01	50	[Retry:0][Stop]
✓	Normal	RoutineFF05	Physic	31 01 FF 05	Has Response:71 01	50	[Retry:0][Stop]
✓	Normal	ECU Reset	Physic	11 01	Has Response:51 01	1000	[Retry:0][Stop]
✓	Normal	ExtendedSesion	Physic	10 03	Has Response:50 03	50	[Retry:0][Stop]
✓	Normal	EnableRx	Physic	28 00 01	Has Response:68 00	50	[Retry:0][Stop]
✓	Normal	DefaultSession	Physic	10 01	Has Response:50 01 00 12 56 89	50	[Retry:0][Stop]

The actual running effect is shown in the figure below:



Valid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
✓	Normal	ExtendedSession	Physic	10 03	🟩 Has Response:50 03	50	[Retry:0][Stop]
✓	Normal	Close DTC	Physic	85 02	🟩 No Response	50	[Retry:0][Stop]
✓	Normal	DisableRx	Physic	28 03 01	🟩 No Response	50	[Retry:0][Stop]
✓	Normal	ReadDataByIDF188	Physic	22 F1 88	🟥 Has Response:62 F1 88 38 37 36 35 34 33 32 31 4E 53	50	[Retry:0][Continue]
✓	Normal	Session02	Physic	10 02	🟩 Has Response:50 02	50	[Retry:0][Stop]
✓	SeedAndKey	Service5	Physic	Seed Level:3 Key Level:4	🟩	50	[Retry:0][Stop]
✓	Normal	WriteDataByF190	Physic	22 F1 90 35 34 32 31 4E 53	🟩 Has Response:62 F1 90	50	[Retry:0][Stop]
✓	DownloadFile	FlashDriver	Physic	0_ecu1.hex	🟩 Dynamic	50	[Retry:0][Stop]
✓	Normal	Erase	Physic	31 01 FF 00 44 00 80 00 00 00 C0 00 00	🟩 Has Response:71 01	50	[Retry:0][Stop]
✓	Normal	RoutineFF05	Physic	31 01 FF 05	🟩 Has Response:71 01	50	[Retry:0][Stop]
✓	Normal	ECU Reset	Physic	11 01	🟩 Has Response:51 01	1000	[Retry:0][Stop]
✓	Normal	ExtendedSesion	Physic	10 03	🟩 Has Response:50 03	50	[Retry:0][Stop]
✓	Normal	EnableRx	Physic	28 00 01	🟩 Has Response:68 00	50	[Retry:0][Stop]

Service Information | ISO15765-2

Settings ▾ | Filter String:

```
Absolute Time  Chn  Identifier  Message Name   Type  Dir  Data Len.  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18
504.578676     1    7C0         Unknown        req   Tx   1          37
504.592236     1    7C8         Unknown        Pos   Rx   2          77 13
504.702191     1    7C0         Unknown        req   Tx   13         31 01 FF 00 44 00 80 00 00 00 C0 00 00
504.716247     1    7C8         Unknown        Pos   Rx   4          71 01 FF 00
504.803177     1    7C0         Unknown        req   Tx   4          31 01 FF 05
504.809243     1    7C8         Unknown        Pos   Rx   4          71 01 FF 05
504.914678     1    7C0         Unknown        req   Tx   2          11 01
504.917742     1    7C8         Unknown        Pos   Rx   2          51 01
505.944686     1    7C0         Unknown        req   Tx   2          10 03
505.956254     1    7C8         Unknown        Wait  Rx   3          7F 10 78
505.987260     1    7C8         Unknown        Pos   Rx   6          50 03 00 32 01 F4
506.081689     1    7C0         Unknown        req   Tx   3          28 00 01
506.095757     1    7C8         Unknown        Pos   Rx   3          68 00 01
```

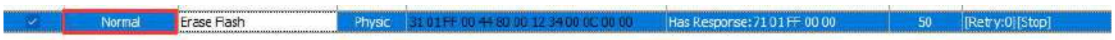
Because the part number of the tested device does not match, to allow the testing process to continue, an error handling setting is configured here to continue execution after an error occurs.

1.19.8. Common Issues Summary

1.19.8.1. Erase Address Configuration

【1】 Fixed Address and Length

If the erase address is a fixed address, the method of handling is the simplest. Directly configure a content as “Normal” service, and fill in the original data directly. If the erase address is 0x00801234 and the erase length is 0x0000C000, then the fixed values to be filled in are as follows:

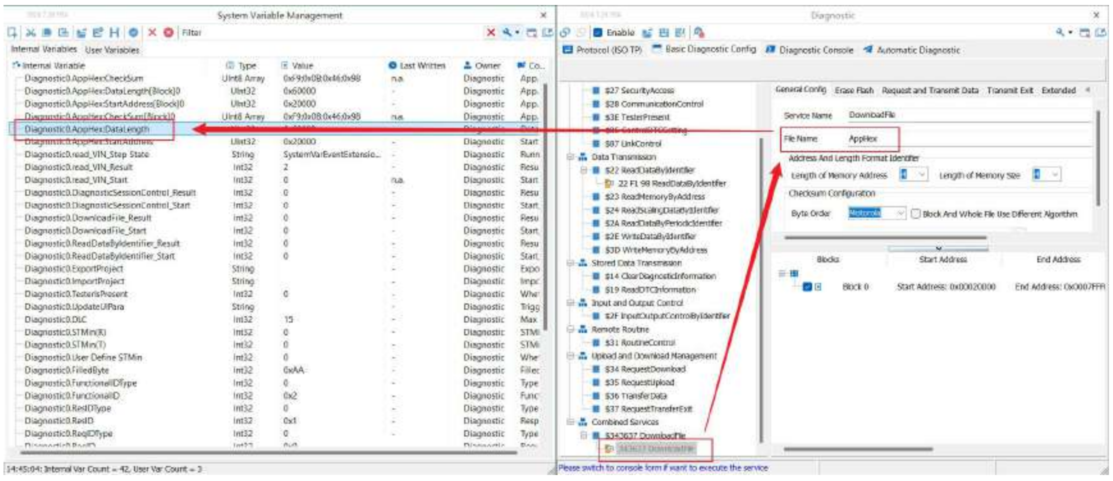


Normal	Erase Flash	Physic	31 01 FF 00 44 80 00 12 34 00 0C 00 00	Has Response:71 01 FF 00 00	50	[Retry:0][Stop]

That is to say, fill in the values that need to be sent and the expected response values directly into the service queue.

【2】 Variable Address and Length

The address changes based on the different Hex files loaded, with the corresponding data address and length being variable. For this case, system variables are required. Taking a sample Hex file as an example, each time the diagnostic module loads a Hex file, it automatically extracts some characteristic information of the Hex as system variables registered into the system (currently, only the address + length are registered; for other needs, please provide feedback directly to TOSUN for assessment). As shown below:



System Variable Management

Internal Variables | User Variables

Internal Variable	Type	Value	Last Written	Owner	Co...
Diagnostic0.AppHex:CheckSum	UInt8 Array	0xF9,0x0B,0x46,0x98	n.a.	Diagnostic	App.
Diagnostic0.AppHex:DataLength[Block]0	UInt32	0x60000	-	Diagnostic	App.
Diagnostic0.AppHex:StartAddress[Block]0	UInt32	0x20000	-	Diagnostic	App.
Diagnostic0.AppHex:CheckSum[Block]0	UInt8 Array	0xF9,0x0B,0x46,0x98	n.a.	Diagnostic	App.
Diagnostic0.AppHex:DataLength	[illegible]	[illegible]	-	[illegible]	[illegible]
Diagnostic0.AppHex:StartAddress	UInt32	0x20000	-	Diagnostic	Start
Diagnostic0.read_VIN_Step State	String	SystemVarEventExtensio...	-	Diagnostic	Runn
Diagnostic0.read_VIN_Result	Int32	2	-	Diagnostic	Resu
Diagnostic0.read_VIN_Start	Int32	0	n.a.	Diagnostic	Start
Diagnostic0.DiagnosticSessionControl_Result	Int32	0	-	Diagnostic	Resu
Diagnostic0.DiagnosticSessionControl_Start	Int32	0	-	Diagnostic	Start
Diagnostic0.DownloadFile_Result	Int32	0	-	Diagnostic	Resu
Diagnostic0.DownloadFile_Start	Int32	0	-	Diagnostic	Start
Diagnostic0.ReadDataByIdentifier_Result	Int32	0	-	Diagnostic	Resu
Diagnostic0.ReadDataByIdentifier_Start	Int32	0	-	Diagnostic	Start
Diagnostic0.ExportProject	String		-	Diagnostic	Expo
Diagnostic0.ImportProject	String		-	Diagnostic	Impc
Diagnostic0.TesterIsPresent	Int32	0	-	Diagnostic	Whe
Diagnostic0.UpdateUIPara	String		-	Diagnostic	Trigg
Diagnostic0.DLC	Int32	15	-	Diagnostic	Max
Diagnostic0.STMin(R)	Int32	0	-	Diagnostic	STMi
Diagnostic0.STMin(T)	Int32	0	-	Diagnostic	STMi
Diagnostic0.User Define STMin	Int32	0	-	Diagnostic	Whe
Diagnostic0.FilledByte	Int32	0xAA	-	Diagnostic	Fillec
Diagnostic0.FunctionalIDType	Int32	0	-	Diagnostic	Type
Diagnostic0.FunctionalID	Int32	0x2	-	Diagnostic	Func
Diagnostic0.ResIDType	Int32	0	-	Diagnostic	Type
Diagnostic0.ResID	Int32	0x1	-	Diagnostic	Resp
Diagnostic0.ReqIDType	Int32	0	-	Diagnostic	Type

14:45:04: Internal Var Count = 42, User Var Count = 3

Diagnostic

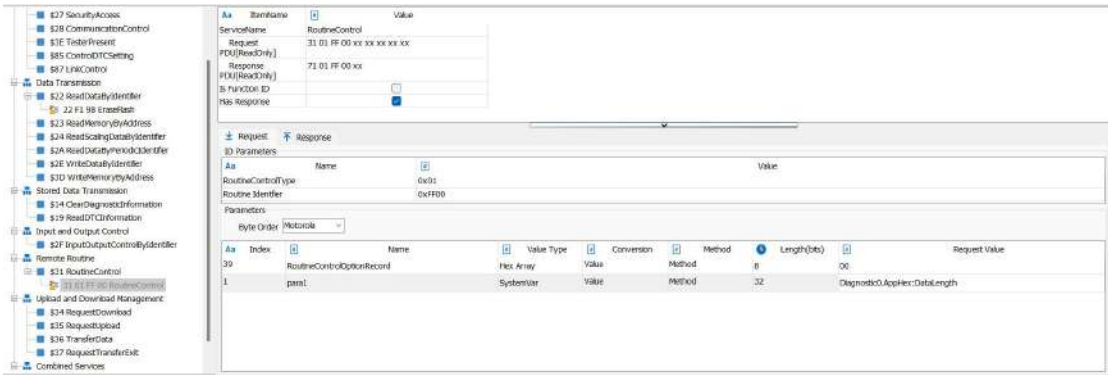
General Config | Erase Flash | Request and Transmit Data | Transmit Exit | Extended

Service Name: DownloadFile
File Name: AppHex
Address And Length Format Identifier
Length of Memory Address: 4 Length of Memory Size: 4
Checksum Configuration
Byte Order: Motorola ☐ Block And Whole File Use Different Algorithm

Blocks	Start Address	End Address
Block 0	Start Address: 0x00020000	End Address: 0x0007FFFF

Please switch to console form if want to execute the service

Then, configure the following in BasicConfig:



ItemName	Value
ServiceName	RoutineControl
Request PDU[ReadOnly]	31 01 FF 00 xx xx xx xx xx
Response PDU[ReadOnly]	71 01 FF 00 xx
Is Function ID	☐
Has Response	☑

Request | Response

ID Parameters

Name	Value
RoutineControlType	0x01
Routine Identifier	0xFF00

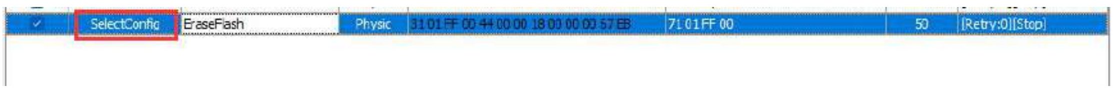
Parameters

Byte Order: Motorola

Index	Name	Value Type	Conversion	Method	Length(bits)	Request Value
39	RoutineControlOptionRecord	Hex Array	Value	Method	8	00
1	para1	SystemVar	Value	Method	32	Diagnostic0.AppHex:DataLength

1. Configure the corresponding RoutineControlType.
2. Configure the corresponding Routine Identifier.
3. Configure para1.
4. Configure another parameter, indicating a Hex variable at this address and length.

Finally, reference this configuration in the automated workflow as shown below:



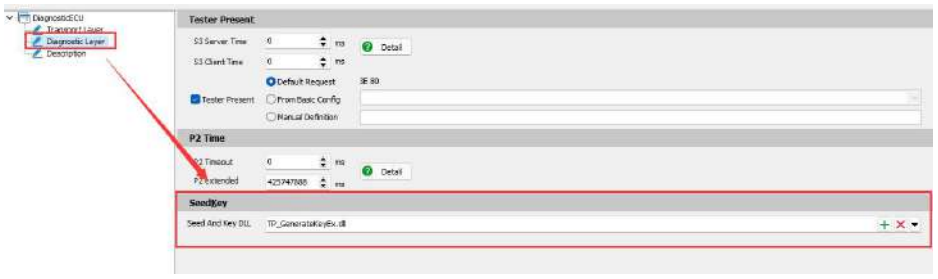
✓	SelectConfig	EraseFlash	Physic	31 01 FF 00 44 00 00 18 00 00 00 57 EB	71 01 FF 00	50	[Retry:0][Stop]

During the automatic operation process, the system will automatically read the values of the current system variables and fill them into the sending service, and therefore achieving the loading of dynamic parameters.

1.19.8.2. Value of Seed&Key

If the Seed & Key values are fixed, simply use the Normal mode and enter the fixed values. This section mainly explains how to dynamically calculate the Key value based on Seed & Key. It mainly includes the following steps:

【1】 Firstly, when configuring the transport layer parameters, load the corresponding algorithm DLL. This DLL is a common library used for all Seed & Key algorithms within this diagnostic module. Therefore, users need to incorporate various level Key calculation methods into this function library.



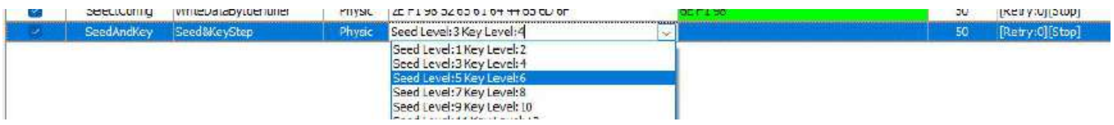
DiagnosticECU
Transport Layer
Diagnostic Layer
Description

Tester Present
S3 Server Time: 0 ms
S3 Client Time: 0 ms
[✓ Tester Present]
[✓ Default Request] 3E 80
From Basic Config
Manual Definition

P2 Time
P2 Timeout: 0 ms
P2 Extended: 425747888 ms

SeedKey
Seed And Key DLL: TP_GenerateKeyEx.dll

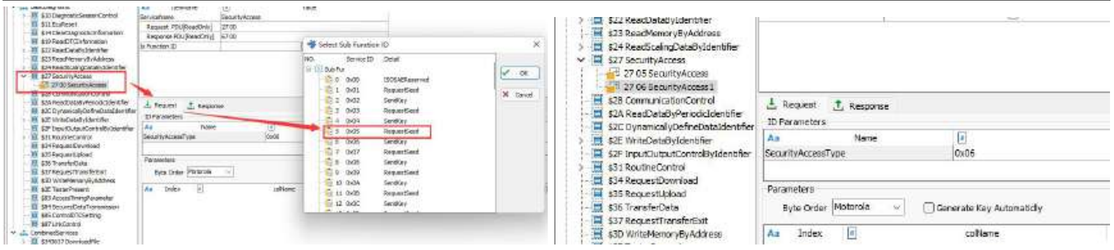
【2】 Method 1: in the automatic process steps, add a SeedAndKey type operation step, and then select the level for GetSeed, as shown below:



✓	SelectConfig	WriteDataByIdentifier	Physic	2E F1 98 52 65 61 64 44 65 6D 6F	6E F1 98	50	[Retry:0][Stop]
✓	SeedAndKey	Seed&KeyStep	Physic	Seed Level:3 Key Level:4		50	[Retry:0][Stop]

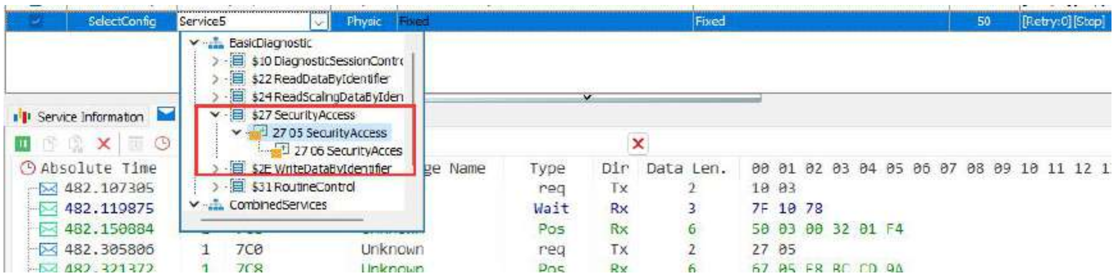
Seed Level:1 Key Level:2
Seed Level:3 Key Level:4
Seed Level:5 Key Level:6
Seed Level:7 Key Level:8
Seed Level:9 Key Level:10

【3】 Method 2: in BasicConfig, add the 0x27 GetSeed and SendKey services (note, these two services must be configured in pairs), as shown below:



In the automated test process, add a SelectConfig step and select the previously configured 0x27 service.

As shown below:



```
Absolute Time    ...  ge Name   Type  Dir  Data Len.  00 01 02 03 04 05 06 07 08 09 10 11 12 1
482.107305                      req   Tx   2          10 03
482.119875                      Wait  Rx   3          7F 10 78
482.150884                      Pos   Rx   6          50 03 00 32 01 F4
482.305806       1  7C0  Unknown  req   Tx   2          27 05
482.321372       1  7C8  Unknown  Pos   Rx   6          67 05 E8 BC CD 9A
```

1.19.8.3. Why does the DLL become Unusable after being Copied to Another Computer

This case, in short: it lacks the dependent library files required for the DLL to run. The main causes for this situation include the following scenarios:

- ➢ The DLL relies on other library functions for its operation. When it was copied, only the DLL itself was copied, and the dependent library functions were not, hence it cannot run on the new computer. This situation is quite apparent, and in most cases, it is relatively easy to resolve.

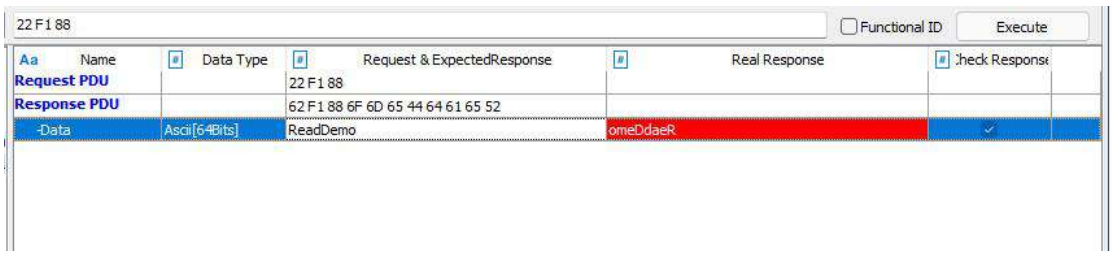
- ➢ When the DLL is compiled, it is not published in Release mode but rather in Debug mode. This issue is quite subtle. For example, if the user develops a Test.dll with C++ and all the algorithms are contained within this DLL with no other external dependencies, it still may not work when copied to another computer. This is because a C++ developed DLL also relies on the C++ runtime library during execution, such as vcruntime140.dll, which typically exists on most computers. However, if the DLL is published in Debug mode, it depends on the runtime library vcruntime140d.dll. Note the 'd' highlighted in red, this runtime library is only present on computers that have a development environment like Visual Studio installed. As a result, a situation can arise where the DLL runs perfectly on your development computer but fails to run when copied to another computer.

Summary: 1. Ensure that all dependent DLLs have been copied. 2. Make sure to use the Release mode to publish DLL.

1.19.8.4. DownloadFile Debugging and File Handling

1.19.8.5. Why is the Retrieved String Reversed

The user expected to read the string "ReadDemo," but the string that was retrieved is "omeDdeaR," which is completely reversed, as shown below:



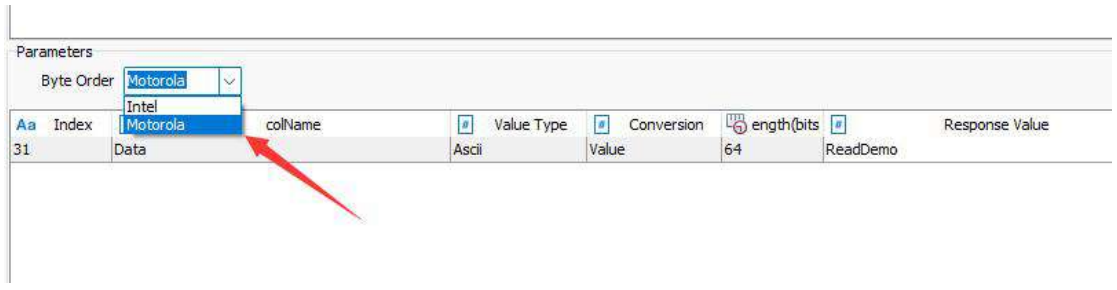
22 F1 88

Name	Data Type	Request & ExpectedResponse	Real Response	Check Response
Request PDU		22 F1 88		
Response PDU		62 F1 88 6F 6D 65 44 64 61 65 52		
-Data	Ascii[64Bits]	ReadDemo	omeDdaeR	✓

This happens because the string parsing order configured does not match the actual storage order of the characters, so the parsed string also comes out reversed.

Solution:

Adjust the string parsing order, for example, if it was previously Motorola, now change it to Intel. After the modification, the retrieved string will match the expected string.



Parameters

Byte Order: Motorola

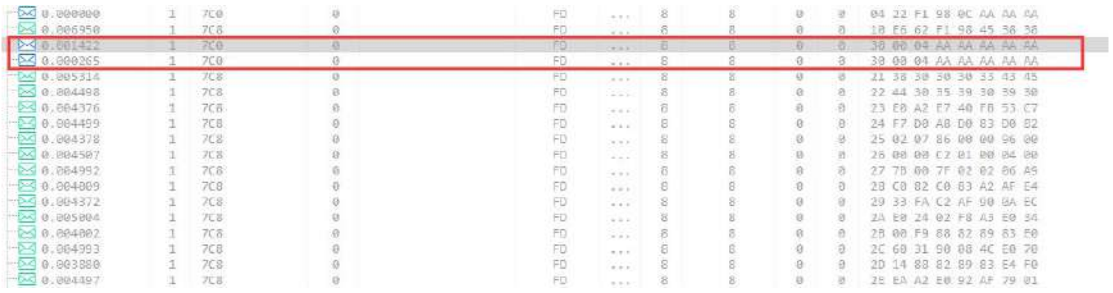
Index	colName	Value Type	Conversion	Length(bits)	Response Value
31	Data	Ascii	Value	64	ReadDemo

1.19.8.6. Why Messages are Seen on the Bus and the Message Length is Sufficient, but the Displayed String is Empty

The case is shown in the figure below:

1.19.8.7. Repeated Flow Control Frame Responses

When using the diagnostic module, user may notice in the Trace process that there are repetition, such as flow control frames. As shown below, there are duplicate sets of two or more flow control frames occurring.

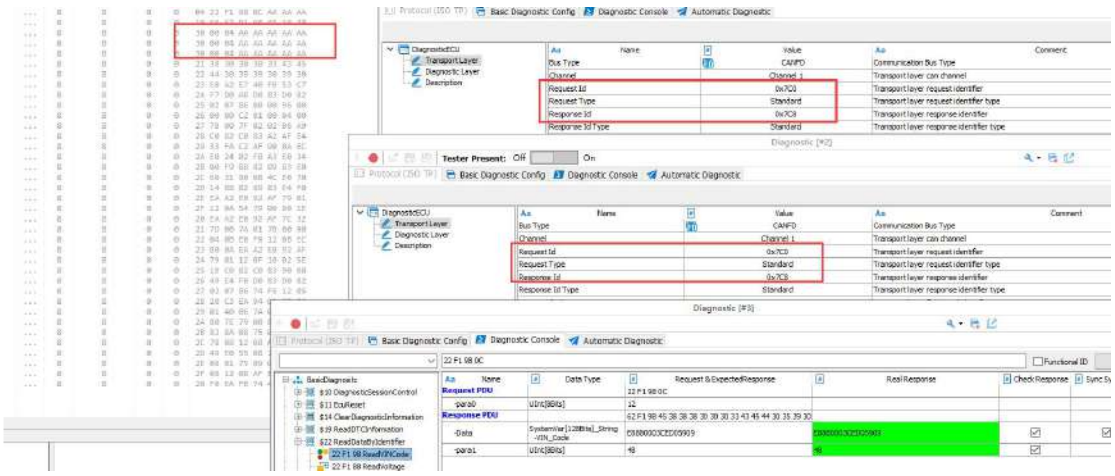


```
0.000000  1  7C0  0  FD  ...  8  8  0  0  04 22 F1 98 0C AA AA AA
0.006950  1  7C8  0  FD  ...  8  8  0  0  10 E6 62 F1 98 45 38 38
0.001422  1  7C0  0  FD  ...  8  8  0  0  30 00 04 AA AA AA AA AA
0.000265  1  7C0  0  FD  ...  8  8  0  0  30 00 04 AA AA AA AA AA
0.005314  1  7C8  0  FD  ...  8  8  0  0  21 38 30 30 30 33 43 45
0.004498  1  7C8  0  FD  ...  8  8  0  0  22 44 30 35 39 30 39 30
0.004376  1  7C8  0  FD  ...  8  8  0  0  23 E8 A2 E7 40 FB 53 C7
0.004499  1  7C8  0  FD  ...  8  8  0  0  24 F7 D0 A8 D0 83 D0 82
0.004378  1  7C8  0  FD  ...  8  8  0  0  25 02 07 86 00 00 96 00
0.004507  1  7C8  0  FD  ...  8  8  0  0  26 00 00 C2 01 00 04 00
0.004992  1  7C8  0  FD  ...  8  8  0  0  27 7B 00 7F 02 02 86 A9
0.004009  1  7C8  0  FD  ...  8  8  0  0  28 C0 82 C0 83 A2 AF E4
0.004372  1  7C8  0  FD  ...  8  8  0  0  29 33 FA C2 AF 90 BA EC
0.005004  1  7C8  0  FD  ...  8  8  0  0  2A E0 24 02 F8 A3 E0 34
0.004002  1  7C8  0  FD  ...  8  8  0  0  2B 00 F9 88 82 89 83 E0
0.004993  1  7C8  0  FD  ...  8  8  0  0  2C 60 31 90 08 4C E0 70
0.003880  1  7C8  0  FD  ...  8  8  0  0  2D 14 88 82 89 83 E4 F0
0.004497  1  7C8  0  FD  ...  8  8  0  0  2E EA A2 E0 92 AF 79 01
```

The symptoms described above could be caused by multiple diagnostic modules responding to the ECU. TSMaster supports the simultaneous operation of multiple diagnostic modules, but if not managed properly, it can lead to situations where diagnostic modules respond at the same time. The common causes of such situations are mainly as follows:

1.19.8.7.1. Created Multiple Diagnostic Modules Using the Same ID and Communication Channel

If multiple diagnostic modules are created using the same ID (request, response, and functional ID) and the same communication channel, then these modules will process the diagnostic messages from the ECU simultaneously. For example, during a diagnostic transmission, when the ECU requests a diagnostic response, multiple modules may respond at the same time. This can cause multiple functional frames to be seen as being sent out simultaneously in the Trace. As shown below:



Protocol (ISO TP) | Basic Diagnostic Config | Diagnostic Console | Automatic Diagnostic

Name	Value	Comment
Bus Type	CANFD	Communication Bus Type
Channel	Channel 1	Transport layer can channel
Request Id	0x7C0	Transport layer request identifier
Request Type	Standard	Transport layer request identifier type
Response Id	0x7C8	Transport layer response identifier
Response Id Type	Standard	Transport layer response identifier type

Diagnostic [#2]

Tester Present: Off On

Protocol (ISO TP) | Basic Diagnostic Config | Diagnostic Console | Automatic Diagnostic

Name	Value	Comment
Bus Type	CANFD	Communication Bus Type
Channel	Channel 1	Transport layer can channel
Request Id	0x7C0	Transport layer request identifier
Request Type	Standard	Transport layer request identifier type
Response Id	0x7C8	Transport layer response identifier
Response Id Type	Standard	Transport layer response identifier type

Diagnostic [#3]

Protocol (ISO TP) | Basic Diagnostic Config | Diagnostic Console | Automatic Diagnostic

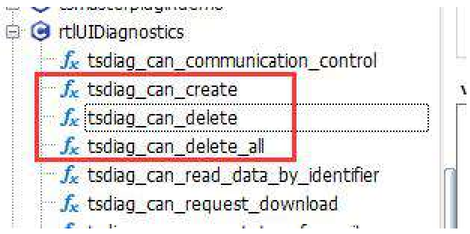
22 F1 98 0C

Name	Data Type	Request & ExpectedResponse	RealResponse	Check Response
Request PDU		22 F1 98 0C		
-para0	UInt[8Bits]	12		
Response PDU		62 F1 98 45 38 38 38 30 30 30 33 43 45 44 30 35 39 30		
-Data	SystemVar[128Bits]_String -VIN_Code	E8880003CED05909	E8880003CED05909	✓
-para1	UInt[8Bits]	48	48	✓

Although the diagnostic request originated from diagnostic module 3, because these three diagnostic modules are configured with identical parameters, so that the users can observe that three identical flow control frames were sent from the PC side.

1.19.8.7.2. In the Script, Diagnostic Modules were Created but were not Released

To support the simultaneous operation of multiple diagnostic modules, TSMaster's scripting system provides the following API functions:



rtlUIDiagnostics
tsdiag_can_communication_control
tsdiag_can_create
tsdiag_can_delete
tsdiag_can_delete_all
tsdiag_can_read_data_by_identifier
tsdiag_can_request_download

tsdiag_can_create is used to create a diagnostic module. After this function is executed successfully, the user will obtain a unique handle for the diagnostic module, as shown below:

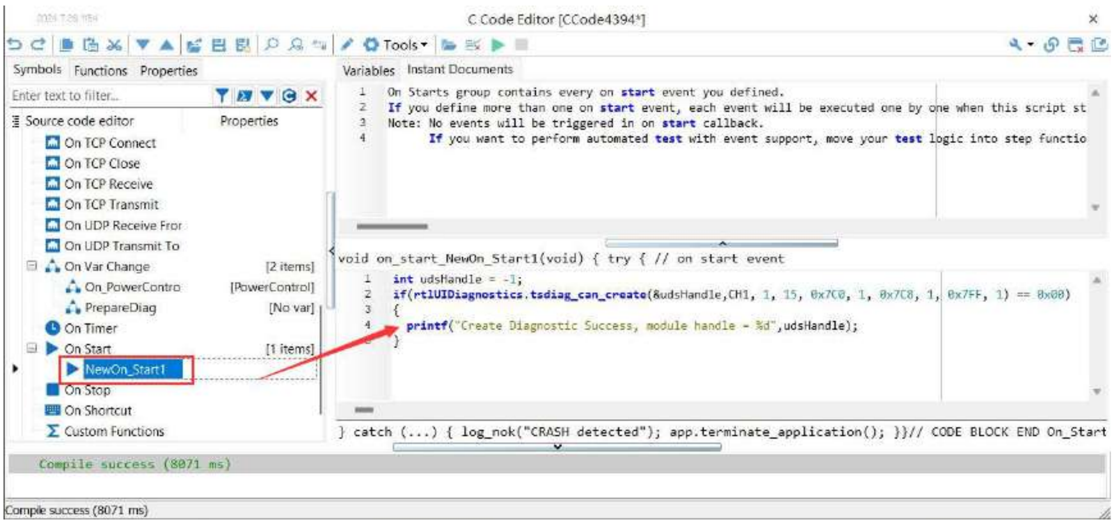
```
if(rtlUIDiagnostics.tsdiag_can_create(&udsHandle,CH1,0,8,0x7C0,1,0x7C8,1,1,1) == 0x00)
{
  printf("Create Diagnostic Success, module handle = %d",udsHandle);
}
```

All subsequent diagnostic actions using this diagnostic module must be based on this unique handle. As shown below, for example, to perform a read operation to retrieve data from the memory address 0xF198:

```
u8 reqDataArray[] = {0x22,0xF1,0x98,0x0C,0x00};
u8 responseArray[600];
int responseArraySize = 600;
if(rtlUIDiagnostics.tstp_can_request_and_get_response(udsHandle, reqDataArray,5,
responseArray, &responseArraySize) == 0x00)
{
  log("send diagnostic payload and get response success! %d datas received", responseArraySize);
}
else
{
  log("send diagnostic payload and get response failed! %d datas received", responseArraySize);
}
```

However, if a user forgets to call the tsdiag_can_delete function to delete the diagnostic module after use, that module will continue to reside in the system and respond to diagnostic actions. Therefore, when using a diagnostic module, tsdiag_can_create and tsdiag_can_delete must always be used in pairs. For example, in the script system, if a diagnostic module is created upon script startup, then the function to delete that diagnostic module must be executed during the script's exit event, as shown below:

During the program startup, add a diagnostic module and obtain a unique handle, udsHandle, as follows:



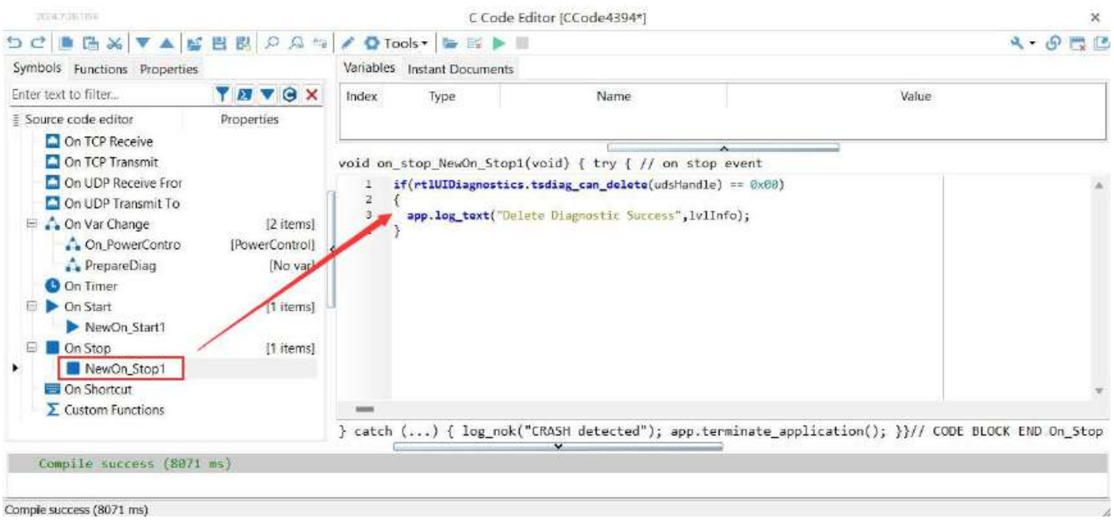
```
On Starts group contains every on start event you defined.
If you define more than one on start event, each event will be executed one by one when this script st
Note: No events will be triggered in on start callback.
      If you want to perform automated test with event support, move your test logic into step functio
```

```
void on_start_NewOn_Start1(void) { try { // on start event
int udsHandle = -1;
if(rtlUIDiagnostics.tsdiag_can_create(&udsHandle,CH1, 1, 15, 0x7C0, 1, 0x7C8, 1, 0x7FF, 1) == 0x00)
{
  printf("Create Diagnostic Success, module handle = %d",udsHandle);
}
} catch (...) { log_nok("CRASH detected"); app.terminate_application(); }}// CODE BLOCK END On_Start
```

```
Compile success (8071 ms)
```

Compile success (8071 ms)

In the program's exit event, use the unique handle udsHandle to delete the diagnostic module, as follows:



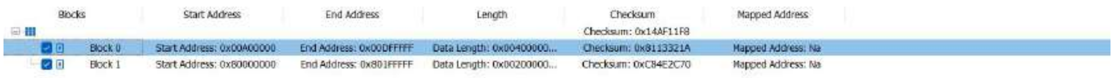
```
void on_stop_NewOn_Stop1(void) { try { // on stop event
if(rtlUIDiagnostics.tsdiag_can_delete(udsHandle) == 0x00)
{
  app.log_text("Delete Diagnostic Success",lvlInfo);
}
} catch (...) { log_nok("CRASH detected"); app.terminate_application(); }}// CODE BLOCK END On_Stop
```

```
Compile success (8071 ms)
```

Compile success (8071 ms)

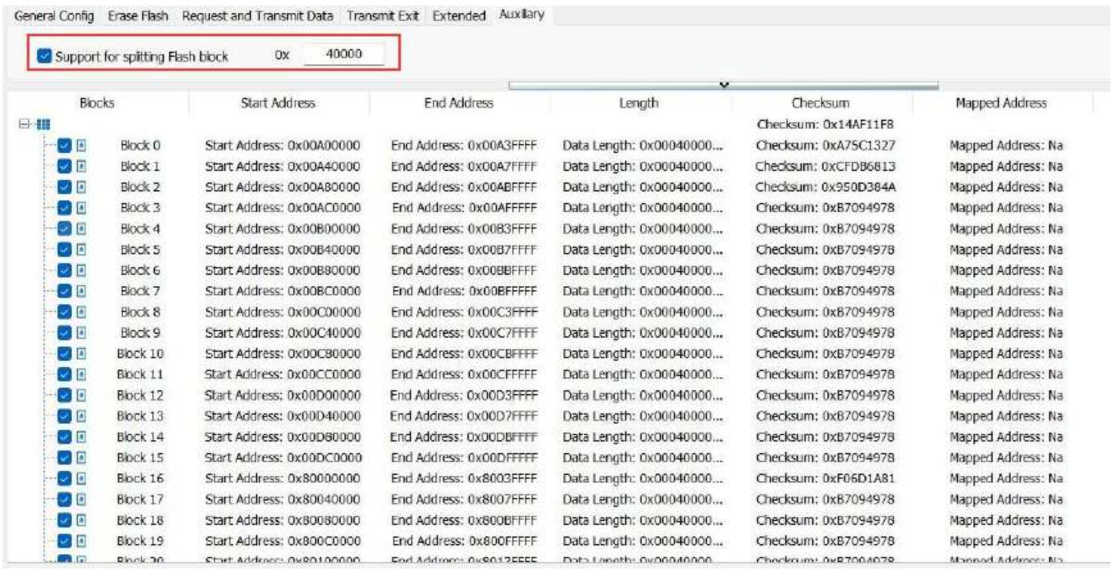
1.19.8.8. The Downloaded File Segments are too Large for the ECU to Process in Time

When configuring the download file, there may be situations where the length of the file segment data for download is too long for the ECU to process. For example, if a hex data file contains two address segments: one with a length of 0x00400000 and another with a length of 0x00200000, as shown in the figure below:



Blocks	Start Address	End Address	Length	Checksum	Mapped Address
				Checksum: 0x14AF11F8	
Block 0	Start Address: 0x00A00000	End Address: 0x00DFFFFF	Data Length: 0x00400000...	Checksum: 0x8113321A	Mapped Address: Na
Block 1	Start Address: 0x80000000	End Address: 0x801FFFFF	Data Length: 0x00200000...	Checksum: 0xC84E2C70	Mapped Address: Na

For some ECUs, the length of the first segment is too long to be processed all at once (this situation is not common and is related to the internal design of some customers' ECUs. Normally, this situation can be avoided by limiting the transfer block size [BS parameter] at the service layer). In this case, it is necessary to subdivide the data segment. Therefore, TSMaster's diagnostic module provides a configuration parameter: whether to support the division of the Flash area. After the user selects to support this feature and sets the maximum allowed size of the Flash area, the software will automatically further subdivide the complete Flash block into multiple logical Flash sub-blocks. The operation is shown in the figure below:



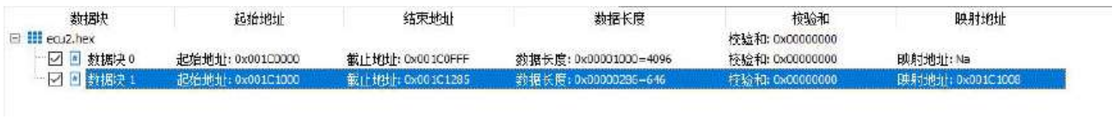
✓ Support for splitting Flash block 0x 40000

Blocks	Start Address	End Address	Length	Checksum	Mapped Address
				Checksum: 0x14AF11F8	
Block 0	Start Address: 0x00A00000	End Address: 0x00A3FFFF	Data Length: 0x00040000...	Checksum: 0xA75C1327	Mapped Address: Na
Block 1	Start Address: 0x00A40000	End Address: 0x00A7FFFF	Data Length: 0x00040000...	Checksum: 0xCFDB6813	Mapped Address: Na
Block 2	Start Address: 0x00A80000	End Address: 0x00ABFFFF	Data Length: 0x00040000...	Checksum: 0x950D384A	Mapped Address: Na
Block 3	Start Address: 0x00AC0000	End Address: 0x00AFFFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 4	Start Address: 0x00B00000	End Address: 0x00B3FFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 5	Start Address: 0x00B40000	End Address: 0x00B7FFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 6	Start Address: 0x00B80000	End Address: 0x00BBFFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 7	Start Address: 0x00BC0000	End Address: 0x00BFFFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 8	Start Address: 0x00C00000	End Address: 0x00C3FFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 9	Start Address: 0x00C40000	End Address: 0x00C7FFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 10	Start Address: 0x00C80000	End Address: 0x00CBFFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 11	Start Address: 0x00CC0000	End Address: 0x00CFFFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 12	Start Address: 0x00D00000	End Address: 0x00D3FFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 13	Start Address: 0x00D40000	End Address: 0x00D7FFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 14	Start Address: 0x00D80000	End Address: 0x00DBFFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 15	Start Address: 0x00DC0000	End Address: 0x00DFFFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 16	Start Address: 0x80000000	End Address: 0x8003FFFF	Data Length: 0x00040000...	Checksum: 0xF06D1A81	Mapped Address: Na
Block 17	Start Address: 0x80040000	End Address: 0x8007FFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 18	Start Address: 0x80080000	End Address: 0x800BFFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 19	Start Address: 0x800C0000	End Address: 0x800FFFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 20	Start Address: 0x80100000	End Address: 0x8013FFFF	Data Length: 0x00040000	Checksum: 0xB7094978	Mapped Address: Na

As shown above, select to support the division of the Flash area, and set each Flash block size to 0x40000. The software automatically divides the previous 0x00400000 into 16 Flash blocks of 0x40000 size each. After this processing, when performing the flashing operation, the software will handle it according to 48 separate Flash blocks.

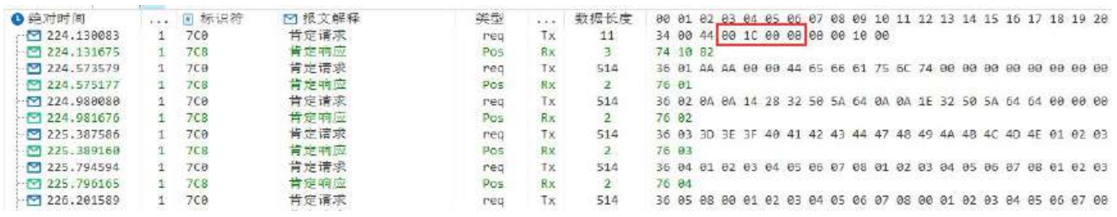
1.19.8.9. The Downloaded File Segment Addresses have Offset

During the program's flashing process, there may be situations where the data address for flashing needs to be offset. For example, the data segment in the Flash data file from address 0x800000 to 0x860000 actually needs to be stored in the address segment from 0x900000 to 0x960000. Common methods to handle this involve processing the source data file or specially handling this part of the address segment data in the flashing software, both of which can be quite troublesome in practice. TSMaster provides a method for address offsetting, allowing users to select data block address segments for flexible address mapping. As shown in the figure below:



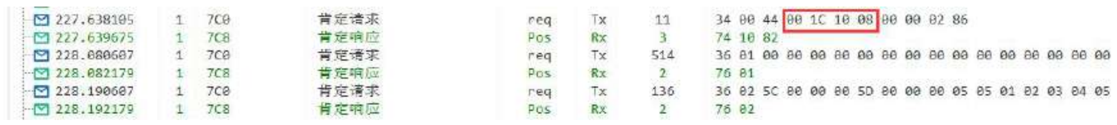
数据块	起始地址	结束地址	数据长度	校验和	映射地址
ecu2.hex				校验和: 0x00000000	
数据块 0	起始地址: 0x001C0000	截止地址: 0x001C0FFF	数据长度: 0x00001000=4096	校验和: 0x00000000	映射地址: Na
数据块 1	起始地址: 0x001C1000	截止地址: 0x001C1285	数据长度: 0x00000286=646	校验和: 0x00000000	映射地址: 0x001C1008

Segment 0 has the original starting address of 0x001C0000, and the mapped address is Na, which indicates no offset, and segment 1 has the original starting address of 0x001C1000, and the mapped address is programmed to 0x001C1008. When executing the flashing operation, the data transfer is as follows:



```
绝对时间      ...  标识符  报文解释   类型  ...  数据长度  00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20
224.130083    1    7C0    肯定请求   req   Tx   11       34 00 44 00 1C 00 08 08 00 10 00
224.131675    1    7C8    肯定响应   Pos   Rx   3        74 10 82
224.573579    1    7C0    肯定请求   req   Tx   514      36 01 AA AA 00 00 44 65 66 61 75 6C 74 00 00 00 00 00 00 00 00
224.575177    1    7C8    肯定响应   Pos   Rx   2        76 01
224.980080    1    7C0    肯定请求   req   Tx   514      36 02 0A 0A 14 28 32 50 5A 64 0A 0A 1E 32 50 5A 64 64 00 00 00
224.981676    1    7C8    肯定响应   Pos   Rx   2        76 02
225.387586    1    7C0    肯定请求   req   Tx   514      36 03 3D 3E 3F 40 41 42 43 44 47 48 49 4A 4B 4C 4D 4E 01 02 03
225.389160    1    7C8    肯定响应   Pos   Rx   2        76 03
225.794594    1    7C0    肯定请求   req   Tx   514      36 04 01 02 03 04 05 06 07 08 01 02 03 04 05 06 07 08 01 02 03
225.796165    1    7C8    肯定响应   Pos   Rx   2        76 04
226.201589    1    7C0    肯定请求   req   Tx   514      36 05 08 00 01 02 03 04 05 06 07 08 00 01 02 03 04 05 06 07 08
```

It can be seen that for segment 0, its transport layer address is 0x001C0000, and the address has not been offset.

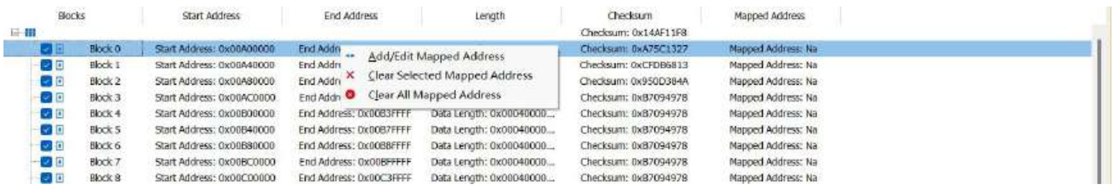


```
227.638105    1    7C0    肯定请求   req   Tx   11       34 00 44 00 1C 10 08 00 00 02 86
227.639675    1    7C8    肯定响应   Pos   Rx   3        74 10 82
228.080607    1    7C0    肯定请求   req   Tx   514      36 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
228.082179    1    7C8    肯定响应   Pos   Rx   2        76 01
228.190607    1    7C0    肯定请求   req   Tx   136      36 02 5C 00 00 00 5D 00 00 00 05 05 01 02 03 04 05
228.192179    1    7C8    肯定响应   Pos   Rx   2        76 02
```

It can be seen that for segment 1, the transport layer address is 0x001C1008, not 0x001C1000, indicating that an offset has occurred in the address.

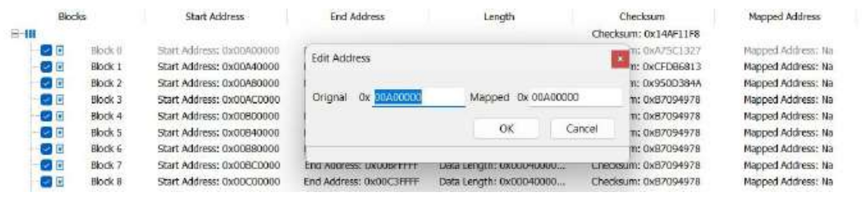
1.19.8.9.1. Mapping Operation

In the download configuration interface's Flash data segment interface, right-click to pop up the mapping menu. It mainly includes three types of operations: add/edit mapped address, clear selected mapped address, and clear all mapped address, as shown in the figure below:



Blocks	Start Address	End Address	Length	Checksum	Mapped Address
				Checksum: 0x14AF11F8	
Block 0	Start Address: 0x00A00000	End Addr		Checksum: 0xA75C1327	Mapped Address: Na
Block 1	Start Address: 0x00A40000	End Addr		Checksum: 0xCFDB6813	Mapped Address: Na
Block 2	Start Address: 0x00A80000	End Addr		Checksum: 0x950D384A	Mapped Address: Na
Block 3	Start Address: 0x00AC0000	End Addr		Checksum: 0xB7094978	Mapped Address: Na
Block 4	Start Address: 0x00B00000	End Address: 0x00B3FFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 5	Start Address: 0x00B40000	End Address: 0x00B7FFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 6	Start Address: 0x00B80000	End Address: 0x00BBFFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 7	Start Address: 0x00BC0000	End Address: 0x00BFFFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 8	Start Address: 0x00C00000	End Address: 0x00C3FFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na

- Add/edit mapped address: select the specified Flash, right-click to pop up the address editing window, as shown in the figure below:



Edit Address

Orignal 0x 00A00000 Mapped 0x 00A00000

Blocks	Start Address	End Address	Length	Checksum	Mapped Address
				Checksum: 0x14AF11F8	
Block 0	Start Address: 0x00A00000			m: 0xA75C1327	Mapped Address: Na
Block 1	Start Address: 0x00A40000			m: 0xCFDB6813	Mapped Address: Na
Block 2	Start Address: 0x00A80000			m: 0x950D384A	Mapped Address: Na
Block 3	Start Address: 0x00AC0000			m: 0xB7094978	Mapped Address: Na
Block 4	Start Address: 0x00B00000			m: 0xB7094978	Mapped Address: Na
Block 5	Start Address: 0x00B40000			m: 0xB7094978	Mapped Address: Na
Block 6	Start Address: 0x00B80000			m: 0xB7094978	Mapped Address: Na
Block 7	Start Address: 0x00BC0000	End Address: 0x00BFFFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na
Block 8	Start Address: 0x00C00000	End Address: 0x00C3FFFF	Data Length: 0x00040000...	Checksum: 0xB7094978	Mapped Address: Na

After adding new mapping information, the mapping address field will display the corresponding mapped address. If there is no mapping information, it will display as "Na".

- Clear selected mapped address: clear the mapping information of the specified Flash block.

- Clear all mapped address: clear all the mapping information of data blocks in the Flash file.

1.19.9. In the Application of EOL and Non-standard Projects

1.19.9.1. Export Configuration File

After completing the diagnostic process development in TSMaster, TSMaster supports the export of configuration files (*.tflash).

1.19.9.2. Load Configuration File in TFlash

1.19.9.3. The External Program Calls the TFlash Automated Server

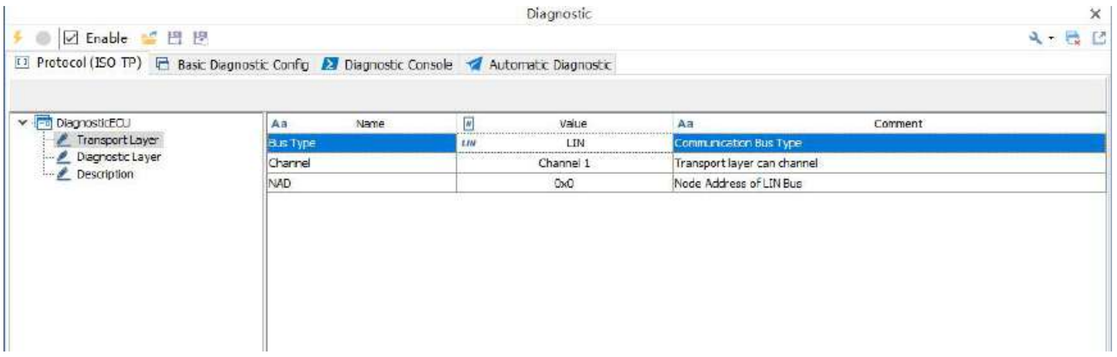
1.20. LIN Diagnosis（Diagnostic_LIN）

1.20.1. Diagnostic TP Parameter Configuration

TSMaster provides the basic functionality of a diagnostic console, allowing users to configure their own send and response requests according to their needs. Follow these steps to operate. Many repetitive steps will not be explained in this chapter, and users can refer to 1.18 CAN/CANFD Diagnosis (Diagnostic_CAN).

1.20.1.1. Transport Layer Parameter

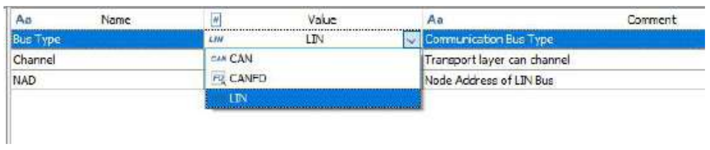
Select the bus type as LIN, and after clicking "OK", the interface will be as shown below:



Name	Value	Comment
Bus Type	LIN	Communication Bus Type
Channel	Channel 1	Transport layer can channel
NAD	0x0	Node Address of LIN Bus

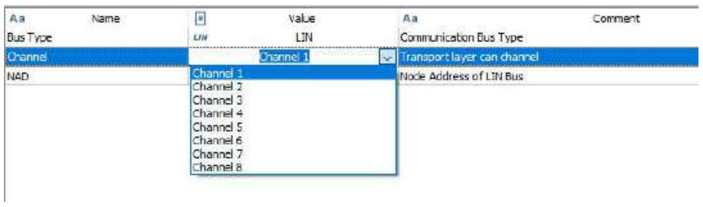
The explanations for each parameter are as follows:

➢ Bus Type: diagnostic transport layer types, currently support CAN/CANFD/LIN, and in the future, Ethernet and FlexRay, etc will be supported. Choose from the drop-down list, as shown in the figure below:



Name	Value	Comment
Bus Type	LIN	Communication Bus Type
Channel		Transport layer can channel
NAD		Node Address of LIN Bus

➢ Channel: the logical channel used by the diagnostic module. TSMaster supports multiple diagnostic modules working online simultaneously, and here it is used to select which logical channel of the system the current diagnostic module uses. Choose from the drop-down list, as shown in the figure below:

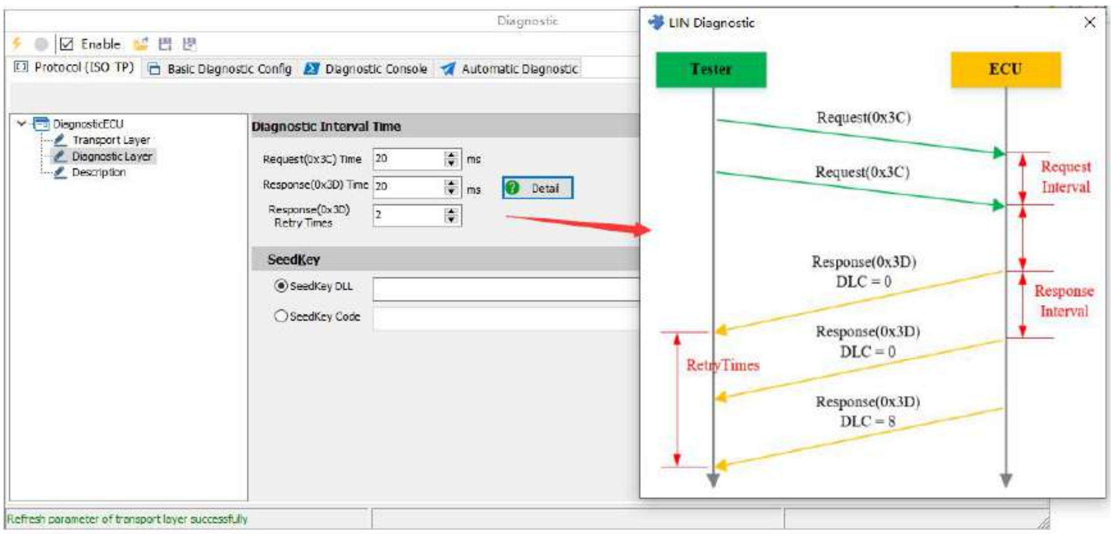


Name	Value	Comment
Bus Type	LIN	Communication Bus Type
Channel	Channel 1	Transport layer can channel
NAD		Node Address of LIN Bus

➢ NAD: node address. In the LIN bus, all nodes send diagnostic request messages through 0x3C and diagnostic response messages through 0x3D. To distinguish the target node for this diagnosis, the parameter NAD is introduced. Only through NAD can it be determined which specific LIN node the diagnostic message is intended for.

1.20.1.2. Service Layer Parameters

Service layer parameters primarily include the interval time parameters for requests (0x3C) and responses (0x3D), the number of response repetitions, and the loading of the SeedKey DLL. As shown in the figure below:



Diagnostic Interval Time
Request(0x3C) Time: 20 ms
Response(0x3D) Time: 20 ms
Response(0x3D) Retry Times: 2

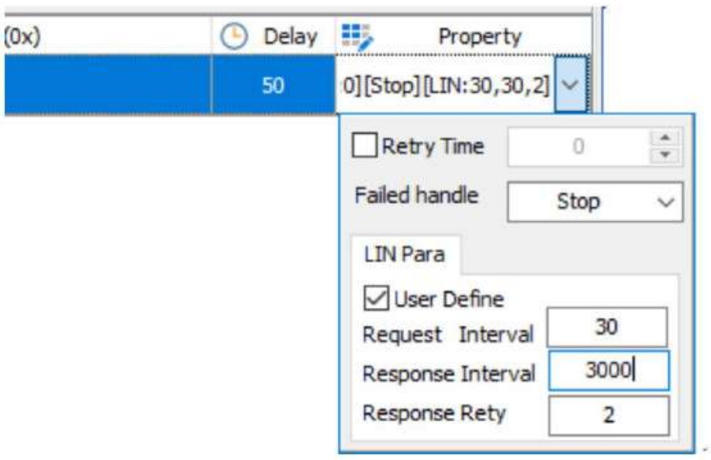
SeedKey
(●) SeedKey DLL
( ) SeedKey Code

Refresh parameter of transport layer successfully

➢ Interval Time（IntervalTime）: as shown in the details of the figure above, the interval time refers to the time interval between the diagnostic and response messages. For ease of use by the user, the request interval and response interval can be configured separately.

➢ Response Retry Time: when the master node sends a 0x3D, if the ECU is not ready to respond with data, it will not reply. In this case, by setting this parameter, the ECU will attempt to obtain a response multiple times, and the specific number of attempts is determined by this parameter.

The time configured above is for diagnostic requests, and the intervals for responses and the number of retry attempts are default parameters of the diagnostic module. In actual use, there may be cases where a longer diagnostic response interval is needed, such as when PC software sends an erase Flash command to the ECU via a diagnostic command, but the ECU requires a longer time to respond. In this case, a longer interval time for 0x3D is needed. Therefore, the TestFlow module also provides a configuration window, allowing users to set the desired diagnostic request response interval and the number of retries for that step, as shown in the figure below. If not configured, the default parameters will be used.



(0x) | Delay | Property
50 | 0][Stop][LIN:30,30,2]

☐ Retry Time: 0
Failed handle: Stop
LIN Para
☑ User Define
Request Interval: 30
Response Interval: 3000
Response Rety: 2

As shown in the figure, the diagnostic response interval is configured to three seconds.

1.20.1.3. Seed&Key

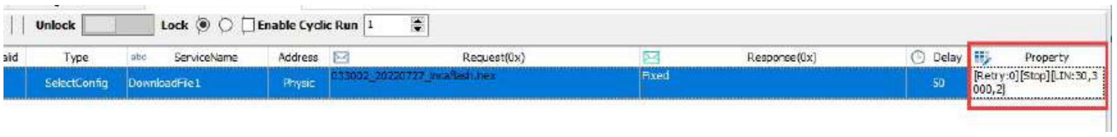
Refer to 1.18.1.3 Seed&Key in the CAN chapter.

1.20.2. Basic Diagnostic Configuration

In the diagnostic configuration section, the concept and method of configuration are equivalent to those for CAN. The aspects that differ from the CAN bus configuration are as follows:

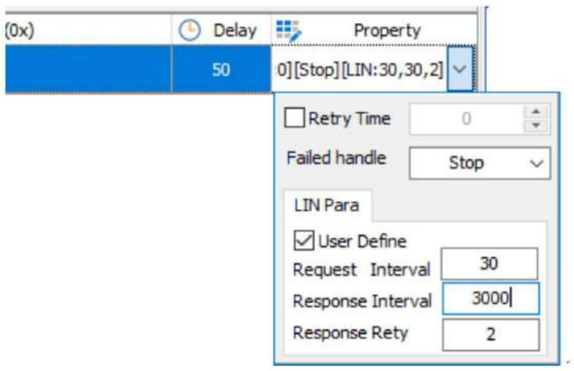
1.20.2.1. In TestFlow, Control Parameter Configuration has been Added

In the TestFlow configuration window, select a configuration process, choose a configuration item, and open the Property attribute window of that item, as shown below:



aid	Type	ServiceName	Address	Request(0x)	Response(0x)	Delay	Property
	SelectConfig	DownloadFile1	Physic	033002_20220727_incaflash.hex	Fixed	50	[Retry:0][Stop][LIN:30,3000,2]

As shown in the following window, the diagnostic response interval is configured to three seconds. The properties have added a LIN Para option. If the user wants to use the default parameters, they should not check the "Use Define" box. If they want to use custom parameters in this step, they should check the "User Define" box and enter the parameters.



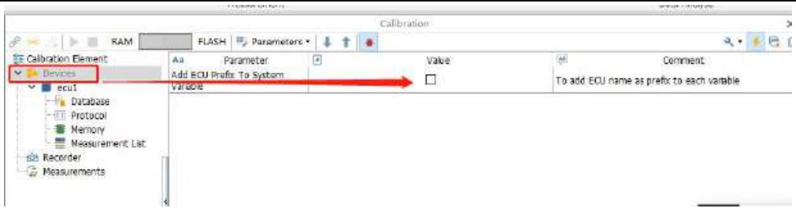
Delay: 50
Property: 0][Stop][LIN:30,30,2]
Retry Time: 0 (unchecked)
Failed handle: Stop
LIN Para
User Define: ✓
Request Interval: 30
Response Interval: 3000
Response Rety: 2

The parameters configured above will only take effect for this step. Once the users enter a new step, the default parameters will continue to be used. The biggest advantage of this approach is that it allows users to flexibly choose the interval time for diagnostic messages. For example, for the Session Control command, if the ECU processing is relatively simple, user can directly use the default parameters. Then, for the Erase Flash command, if ECU processing takes quite a lot of times, user can use custom parameters. Following that, for the Transmit Data command, the ECU can requires a longer processing time, user can also use custom parameters, entering a longer response interval time and a greater number of response attempts.

1.21. Calibration

1.21.1. How to Distinguish when Calibrating Multiple ECUs Simultaneously

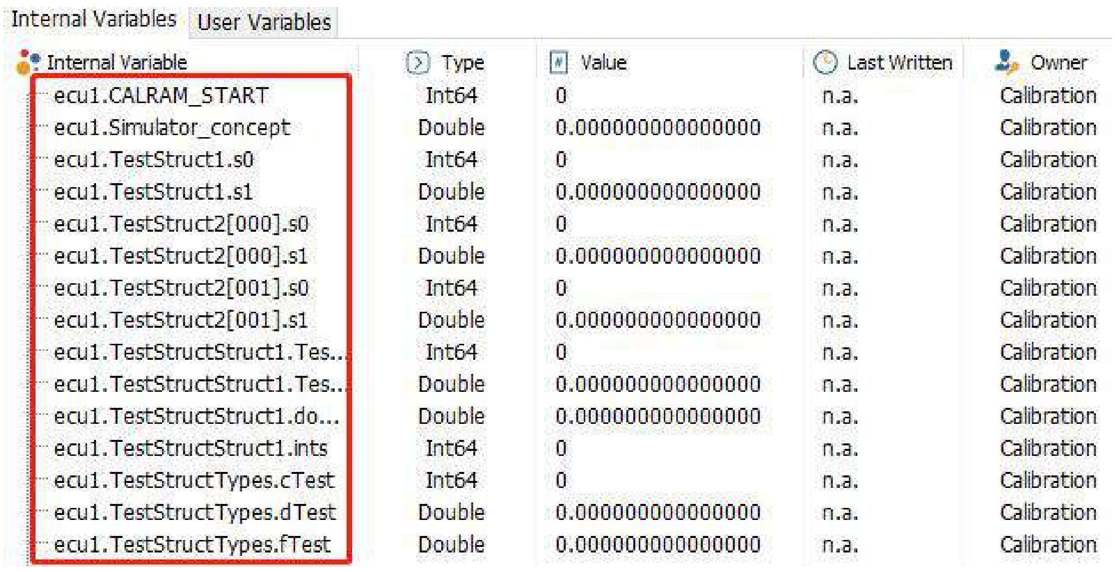
When multiple ECUs are calibrated simultaneously, and the same A2L file is used or there are duplicate variables in the file, it is necessary to distinguish between calibration and measurement variables from different ECUs. The solution provided by TSMaster is to check the option to add an ECU name prefix in the calibration configuration tool:



Parameter	Value	Comment
Add ECU Prefix To System Variable	☐	To add ECU name as prefix to each variable

Select the "Add ECU Prefix To System Variable" Option

After selecting this option, restart the software. The system variables related to calibration will automatically add the corresponding prefixes, as shown below:

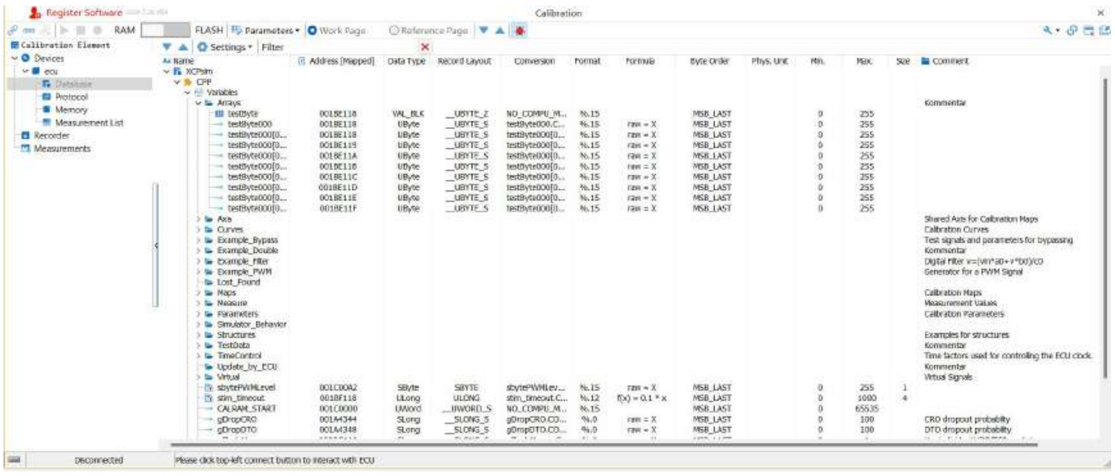


Internal Variable	Type	Value	Last Written	Owner
ecu1.CALRAM_START	Int64	0	n.a.	Calibration
ecu1.Simulator_concept	Double	0.000000000000000	n.a.	Calibration
ecu1.TestStruct1.s0	Int64	0	n.a.	Calibration
ecu1.TestStruct1.s1	Double	0.000000000000000	n.a.	Calibration
ecu1.TestStruct2[000].s0	Int64	0	n.a.	Calibration
ecu1.TestStruct2[000].s1	Double	0.000000000000000	n.a.	Calibration
ecu1.TestStruct2[001].s0	Int64	0	n.a.	Calibration
ecu1.TestStruct2[001].s1	Double	0.000000000000000	n.a.	Calibration
ecu1.TestStructStruct1.Tes...	Int64	0	n.a.	Calibration
ecu1.TestStructStruct1.Tes...	Double	0.000000000000000	n.a.	Calibration
ecu1.TestStructStruct1.do...	Double	0.000000000000000	n.a.	Calibration
ecu1.TestStructStruct1.ints	Int64	0	n.a.	Calibration
ecu1.TestStructTypes.cTest	Int64	0	n.a.	Calibration
ecu1.TestStructTypes.dTest	Double	0.000000000000000	n.a.	Calibration
ecu1.TestStructTypes.fTest	Double	0.000000000000000	n.a.	Calibration

As shown, system variables have been updated with the added ECU prefix.

1.21.2. Database View

In TSMaster, the A2L database view has two modes: tree mode and list mode. As the name suggests, the tree mode expands the variables inside the A2L database in a tree structure, allowing the user to see the hierarchical relationships of the variables, as shown in the figure below:

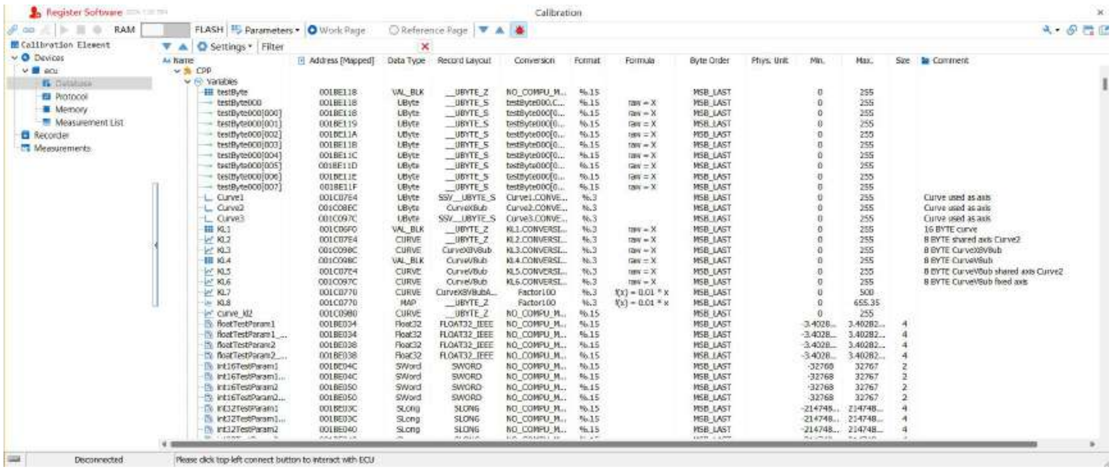


Calibration Element
- Devices
  - ecu
    - Database
    - Protocol
    - Memory
    - Measurement List
- Recorder
- Measurements

Name	Address [Mapped]	Data Type	Record Layout	Conversion	Format	Formula	Byte Order	Phys. Unit	Min.	Max.	Size	Comment
XCPsim												
CPP												
Variables												
Arrays												Kommentar
testByte	001BE118	VAL_BLK	__UBYTE_Z	NO_COMPU_M...	%.15		MSB_LAST		0	255		
testByte000	001BE118	UByte	__UBYTE_S	testByte000.C...	%.15	raw = X	MSB_LAST		0	255		
testByte000[0...	001BE118	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
testByte000[0...	001BE119	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
testByte000[0...	001BE11A	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
testByte000[0...	001BE11B	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
testByte000[0...	001BE11C	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
testByte000[0...	001BE11D	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
testByte000[0...	001BE11E	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
testByte000[0...	001BE11F	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
Axis												Shared Axis for Calibration Maps
Curves												Calibration Curves
Example_Bypass												Test signals and parameters for bypassing
Example_Double												Kommentar
Example_Filter												Digital Filter v=(vin*a0+v*b0)/c0
Example_PWM												Generator for a PWM Signal
Lost_Found												
Maps												Calibration Maps
Measure												Measurement Values
Parameters												Calibration Parameters
Simulator_Behavior												
Structures												Examples for structures
TestData												Kommentar
TimeControl												Time factors used for controlling the ECU clock
Update_by_ECU												Kommentar
Virtual												Virtual Signals
sbytePWMLevel	001C00A2	SByte	SBYTE	sbytePWMLev...	%.15	raw = X	MSB_LAST		0	255	1	
stim_timeout	001BF118	ULong	ULONG	stim_timeout.C...	%.12	f(x) = 0.1 * x	MSB_LAST		0	1000	4	
CALRAM_START	001C0000	UWord	__UWORD_S	NO_COMPU_M...	%.15		MSB_LAST		0	65535		
gDropCRO	001A4344	SLong	__SLONG_S	gDropCRO.CO...	%.0	raw = X	MSB_LAST		0	100		CRO dropout probability
gDropDTO	001A4348	SLong	__SLONG_S	gDropDTO.CO...	%.0	raw = X	MSB_LAST		0	100		DTO dropout probability

Disconnected | Please click top-left connect button to interact with ECU

List mode simply displays all the measured and calibrated quantities in the A2L database in a flat layout, as shown in the figure below:



Calibration Element
- Devices
  - ecu
    - Database
    - Protocol
    - Memory
    - Measurement List
- Recorder
- Measurements

Name	Address [Mapped]	Data Type	Record Layout	Conversion	Format	Formula	Byte Order	Phys. Unit	Min.	Max.	Size	Comment
CPP												
Variables												
testByte	001BE118	VAL_BLK	__UBYTE_Z	NO_COMPU_M...	%.15		MSB_LAST		0	255		
testByte000	001BE118	UByte	__UBYTE_S	testByte000.C...	%.15	raw = X	MSB_LAST		0	255		
testByte000[000]	001BE118	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
testByte000[001]	001BE119	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
testByte000[002]	001BE11A	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
testByte000[003]	001BE11B	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
testByte000[004]	001BE11C	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
testByte000[005]	001BE11D	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
testByte000[006]	001BE11E	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
testByte000[007]	001BE11F	UByte	__UBYTE_S	testByte000[0...	%.15	raw = X	MSB_LAST		0	255		
Curve1	001C07E4	UByte	SSV__UBYTE_S	Curve1.CONVE...	%.3		MSB_LAST		0	255		Curve used as axis
Curve2	001C08EC	UByte	CurveXBub	Curve2.CONVE...	%.3		MSB_LAST		0	255		Curve used as axis
Curve3	001C097C	UByte	SSV__UBYTE_S	Curve3.CONVE...	%.3		MSB_LAST		0	255		Curve used as axis
KL1	001C06F0	VAL_BLK	__UBYTE_Z	KL1.CONVERSI...	%.3	raw = X	MSB_LAST		0	255		16 BYTE curve
KL2	001C07E4	CURVE	__UBYTE_Z	KL2.CONVERSI...	%.3	raw = X	MSB_LAST		0	255		8 BYTE shared axis Curve2
KL3	001C098C	CURVE	CurveX8VBub	KL3.CONVERSI...	%.3	raw = X	MSB_LAST		0	255		8 BYTE CurveX8VBub
KL4	001C098C	VAL_BLK	CurveVBub	KL4.CONVERSI...	%.3	raw = X	MSB_LAST		0	255		8 BYTE CurveVBub
KL5	001C07E4	CURVE	CurveVBub	KL5.CONVERSI...	%.3	raw = X	MSB_LAST		0	255		8 BYTE CurveVBub shared axis Curve2
KL6	001C097C	CURVE	CurveVBub	KL6.CONVERSI...	%.3	raw = X	MSB_LAST		0	255		8 BYTE CurveVBub fixed axis
KL7	001C0770	CURVE	CurveX8VBubA...	Factor100	%.3	f(x) = 0.01 * x	MSB_LAST		0	500		
KL8	001C0770	MAP	__UBYTE_Z	Factor100	%.3	f(x) = 0.01 * x	MSB_LAST		0	655.35		
curve_kl2	001C0980	CURVE	__UBYTE_Z	NO_COMPU_M...	%.15		MSB_LAST		0	255		
floatTestParam1	001BE034	Float32	FLOAT32_IEEE	NO_COMPU_M...	%.15		MSB_LAST		-3.4028...	3.40282...	4	
floatTestParam1_...	001BE034	Float32	FLOAT32_IEEE	NO_COMPU_M...	%.15		MSB_LAST		-3.4028...	3.40282...	4	
floatTestParam2	001BE038	Float32	FLOAT32_IEEE	NO_COMPU_M...	%.15		MSB_LAST		-3.4028...	3.40282...	4	
floatTestParam2_...	001BE038	Float32	FLOAT32_IEEE	NO_COMPU_M...	%.15		MSB_LAST		-3.4028...	3.40282...	4	
int16TestParam1	001BE04C	SWord	SWORD	NO_COMPU_M...	%.15		MSB_LAST		-32768	32767	2	
int16TestParam1...	001BE04C	SWord	SWORD	NO_COMPU_M...	%.15		MSB_LAST		-32768	32767	2	
int16TestParam2	001BE050	SWord	SWORD	NO_COMPU_M...	%.15		MSB_LAST		-32768	32767	2	
int16TestParam2...	001BE050	SWord	SWORD	NO_COMPU_M...	%.15		MSB_LAST		-32768	32767	2	
int32TestParam1	001BE03C	SLong	SLONG	NO_COMPU_M...	%.15		MSB_LAST		-214748...	214748...	4	
int32TestParam1...	001BE03C	SLong	SLONG	NO_COMPU_M...	%.15		MSB_LAST		-214748...	214748...	4	
int32TestParam2	001BE040	SLong	SLONG	NO_COMPU_M...	%.15		MSB_LAST		-214748...	214748...	4	

Disconnected | Please click top-left connect button to interact with ECU

The advantage of using the tree mode is that it is convenient for viewing the hierarchical relationships of variables. However, the disadvantage is that it is not convenient to select multiple variables at the same time. For example, if a user wants to add hundreds or all variables to the measurement list at once, in the tree view, user would need to expand each one and add them one by one. In this case, it is necessary to switch to the list mode, where user can simply select all and add them directly.

1.21.3. CCP DAQ

When users cannot directly obtain DAQ-related configurations from the A2L, they can refer to existing projects and manually add the description configuration for DAQ, which mainly consists of two parts: Event Channel + DAQ. The process is as follows:

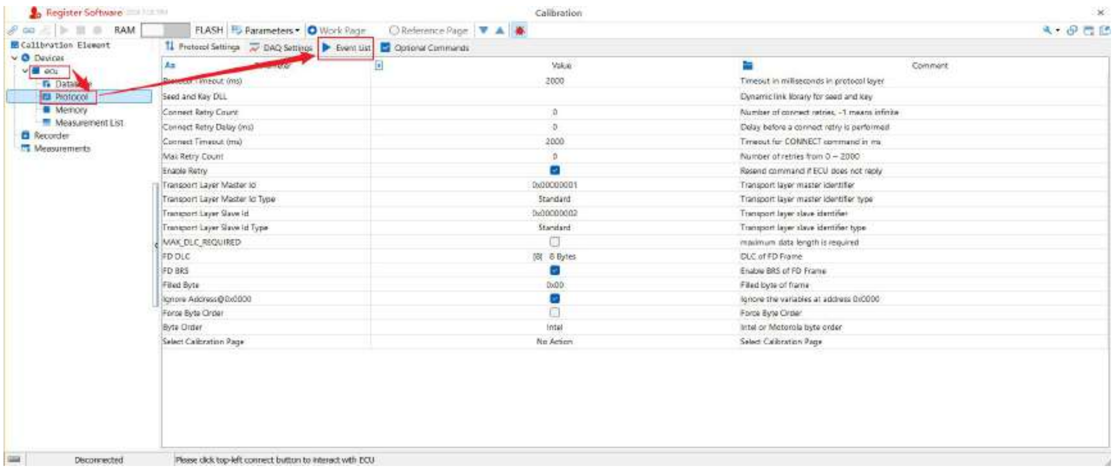
1.21.3.1. Event Channel

An event channel is a mechanism for event triggering that is defined during the development of the ECU CCP module. Therefore, user can manually add an event channel based on the description file. The main attributes of an event channel include:

- Channel Name: this name is arbitrarily named by the user, mainly to avoid repetition and to be clearly indicative.
- Event channel: the channel number is determined during the ECU design and matches the trigger speed. For example, the event corresponding to channel 0 defined in the ECU is matched with a 10ms trigger rate, and the correspondence between the two must be correct.
- Trigger rate: select the speed that corresponds to the channel number.
- Unit: select a 1ms rate.

Adding steps:

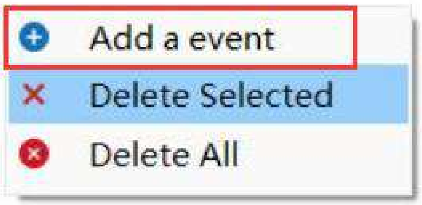
1. Through ecu->Protocol->Event List.



Parameter	Value	Comment
Protocol Timeout (ms)	2000	Timeout in milliseconds in protocol layer
Seed and Key DLL		Dynamic link library for seed and key
Connect Retry Count	0	Number of connect retries, -1 means infinite
Connect Retry Delay (ms)	0	Delay before a connect retry is performed
Connect Timeout (ms)	2000	Timeout for CONNECT command in ms
Max Retry Count	0	Number of retries from 0 ~ 2000
Enable Retry	✓	Resend command if ECU does not reply
Transport Layer Master Id	0x00000001	Transport layer master identifier
Transport Layer Master Id Type	Standard	Transport layer master identifier type
Transport Layer Slave Id	0x00000002	Transport layer slave identifier
Transport Layer Slave Id Type	Standard	Transport layer slave identifier type
MAX_DLC_REQUIRED	☐	maximum data length is required
FD DLC	[8] 8 Bytes	DLC of FD Frame
FD BRS	✓	Enable BRS of FD Frame
Filled Byte	0x00	Filled byte of frame
Ignore Address@0x0000	✓	Ignore the variables at address 0x0000
Force Byte Order	☐	Force Byte Order
Byte Order	Intel	Intel or Motorola byte order
Select Calibration Page	No Action	Select Calibration Page

Disconnected | Please click top-left connect button to interact with ECU

2. Right-click to add an event.



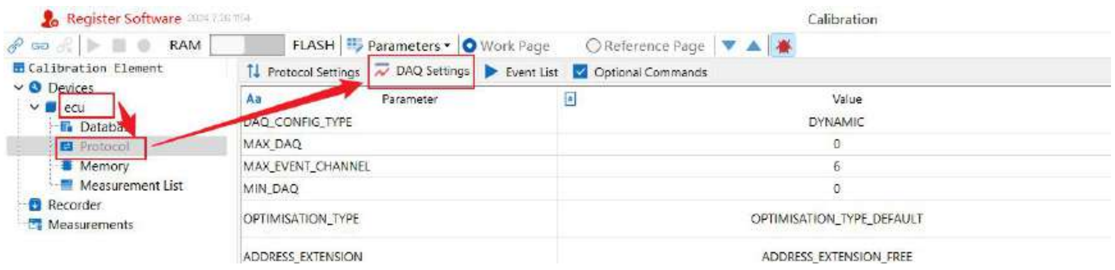
3. Enter the name, select the event channel, trigger rate, and unit.

1s	2	1000	1 ms
New Event	3	100	ms

1.21.3.2. DAQ Configuration

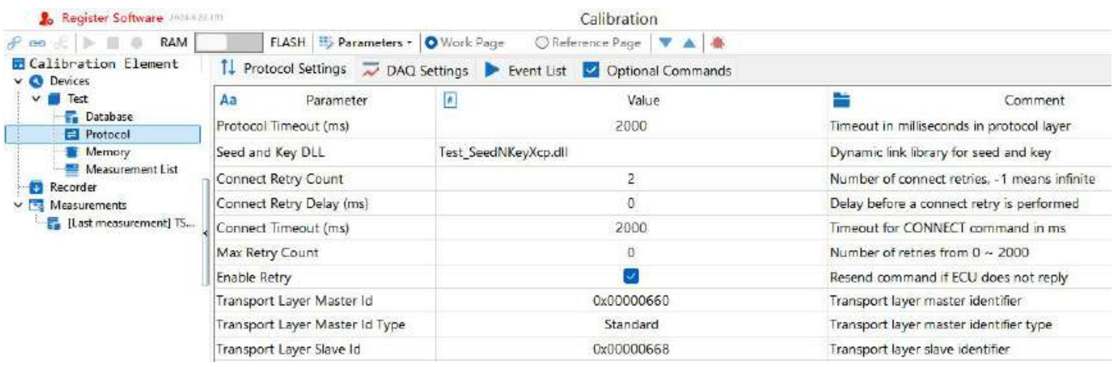
When the DAQ list cannot be automatically read from the ECU configuration, users need to manually add a DAQ list description as follows:

1. Through ecu->Protocol->DAQ Settings, and enter the settings interface:



Parameter	Value
DAQ_CONFIG_TYPE	DYNAMIC
MAX_DAQ	0
MAX_EVENT_CHANNEL	6
MIN_DAQ	0
OPTIMISATION_TYPE	OPTIMISATION_TYPE_DEFAULT
ADDRESS_EXTENSION	ADDRESS_EXTENSION_FREE

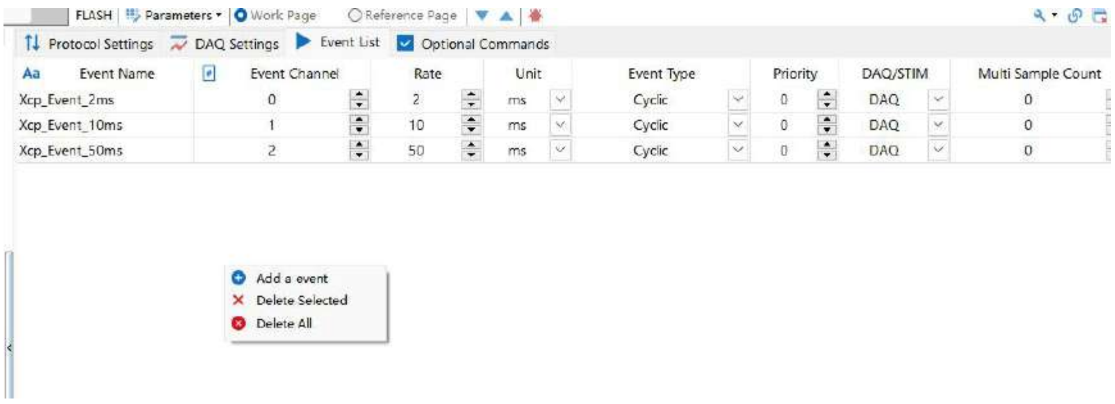
2. Uncheck the option for automatic detection as follows:



Parameter	Value	Comment
Protocol Timeout (ms)	2000	Timeout in milliseconds in protocol layer
Seed and Key DLL	Test_SeedNKeyXcp.dll	Dynamic link library for seed and key
Connect Retry Count	2	Number of connect retries, -1 means infinite
Connect Retry Delay (ms)	0	Delay before a connect retry is performed
Connect Timeout (ms)	2000	Timeout for CONNECT command in ms
Max Retry Count	0	Number of retries from 0 ~ 2000
Enable Retry	✓	Resend command if ECU does not reply
Transport Layer Master Id	0x00000660	Transport layer master identifier
Transport Layer Master Id Type	Standard	Transport layer master identifier type
Transport Layer Slave Id	0x00000668	Transport layer slave identifier

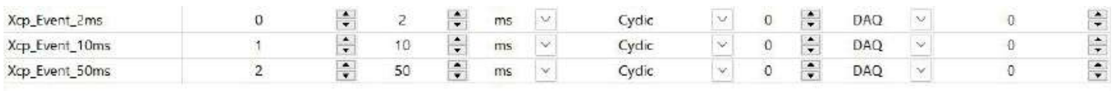
If the checkbox is not unchecked, the ODT list is dynamically obtained from the ECU during the download process through commands, and the ODT list is dynamically assigned. In this case, the properties do not need to be manually configured, and therefore, it is not possible to configure the properties.

3. Right-click on the blank area to add or delete ODT list descriptions.



Event Name	Event Channel	Rate	Unit	Event Type	Priority	DAQ/STIM	Multi Sample Count
Xcp_Event_2ms	0	2	ms	Cyclic	0	DAQ	0
Xcp_Event_10ms	1	10	ms	Cyclic	0	DAQ	0
Xcp_Event_50ms	2	50	ms	Cyclic	0	DAQ	0

4. Set ODT list properties.



Xcp_Event_2ms	0	2	ms	Cydic	0	DAQ	0
Xcp_Event_10ms	1	10	ms	Cydic	0	DAQ	0
Xcp_Event_50ms	2	50	ms	Cydic	0	DAQ	0

If user is not clear about the detailed meaning of the ODT list, refer to the existing project and configure the above description table to be the same.

5. Precautions:

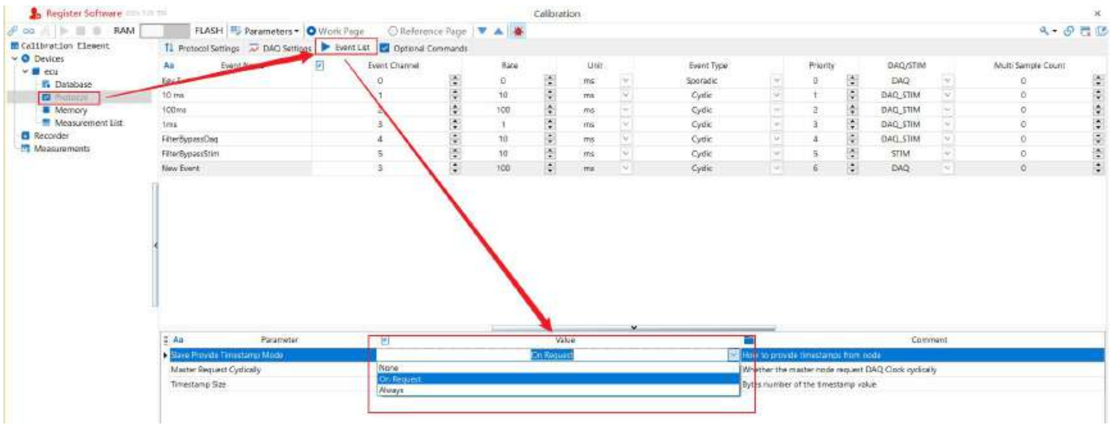
After setting the channel in the event configuration, user needs to save the project, restart the software, and then the corresponding event channel properties can be seen in the DAQ settings dropdown box.

1.21.4. XCP DAQ

1.21.4.1. DAQ TimeStamp

In the XCP calibration protocol, there is a DAQ timestamp feature. When this feature is activated, the calibration device reads the current timestamp from the ECU node being tested to correct the timestamp on the calibration device side. In TSMaster, the related configuration is as follows:

1. Firstly, the path for related configurations is: Protocol -> Event List -> Save Provide Timestamp Mode. As shown, it mainly has three configuration parameters: the mode for the device to provide timestamps, whether the master device continuously sends requests, and the size of the timestamp.



Event Name	Event Channel	Rate	Unit	Event Type	Priority	DAQ/STIM	Multi Sample Count
Key T	0	0	ms	Sporadic	0	DAQ	0
10 ms	1	10	ms	Cyclic	1	DAQ_STIM	0
100ms	2	100	ms	Cyclic	2	DAQ_STIM	0
1ms	3	1	ms	Cyclic	3	DAQ_STIM	0
FilterBypassDaq	4	10	ms	Cyclic	4	DAQ_STIM	0
FilterBypassStim	5	10	ms	Cyclic	5	STIM	0
New Event	3	100	ms	Cyclic	6	DAQ	0

Parameter	Value	Comment
Slave Provide Timestamp Mode	On Request	How to provide timestamps from node
Master Request Cyclically		Whether the master node request DAQ Clock cyclically
Timestamp Size		Bytes number of the timestamp value

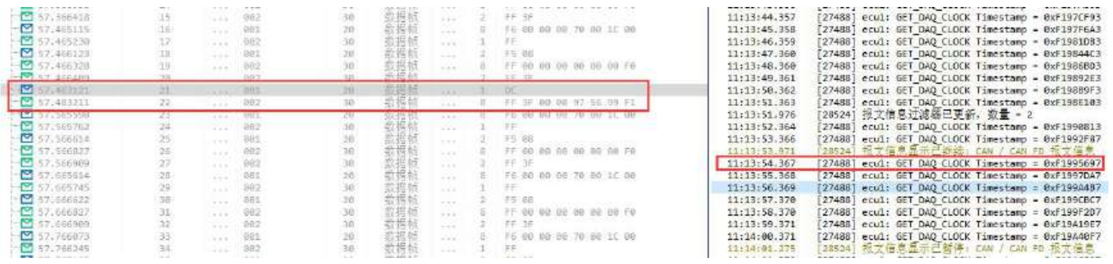
2. The mode for the device to provide timestamps:

- None: regardless of whether the device has the capability to provide a timestamp, turn off its DAQ TimeStamp feature.

- OnRequest/Always: If the slave device has the capability to provide a timestamp, then activate its DAQ Timestamp feature.

3. The master device continuously sends requests:

If this configuration is set to True, then the calibration master device sends a GetDAQClock command to the slave device every 1 second, and the slave device returns the current corresponding DAQ timestamp value. Reflected in the message, it is shown in the figure below:



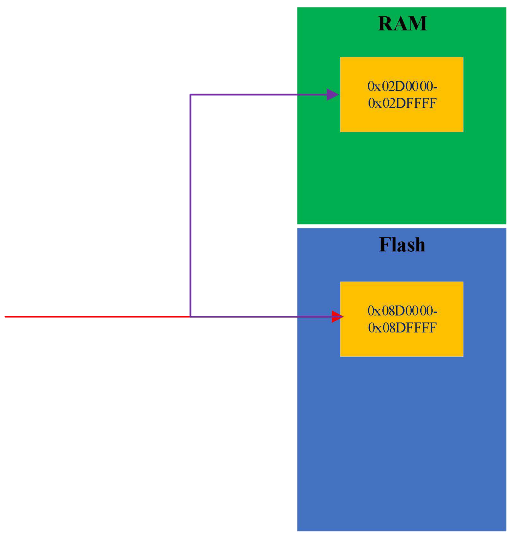
```
11:13:44.357   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF197CF93
11:13:45.358   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF197F6A3
11:13:46.359   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF1981DB3
11:13:47.360   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF19844C3
11:13:48.360   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF1986BD3
11:13:49.361   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF19892E3
11:13:50.362   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF198B9F3
11:13:51.363   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF198E103
11:13:51.976   [28524] 报文信息过滤器已更新，数量 = 2
11:13:52.364   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF1990813
11:13:53.366   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF1992F87
11:13:53.871   [28524] 报文信息显示已继续：CAN / CAN FD 报文信息
11:13:54.367   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF1995697
11:13:55.368   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF1997DA7
11:13:56.369   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF199A4B7
11:13:57.370   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF199CBC7
11:13:58.370   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF199F2D7
11:13:59.371   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF19A19E7
11:14:00.371   [27488] ecu1: GET_DAQ_CLOCK Timestamp = 0xF19A40F7
11:14:01.275   [28524] 报文信息显示已暂停：CAN / CAN FD 报文信息
```

As shown in the figure above, the calibration device requests the DAQ timestamp value from the slave device every 1 second.

1.21.5. Switching between RAM and ROM

1.21.5.1. Cause Analysis

In some ECUs, it is necessary to switch the memory address mapping during calibration, such as switching to RAM for calibration (which is related to the internal CCP architecture of the ECU, and not all calibration require this mechanism). As shown below:

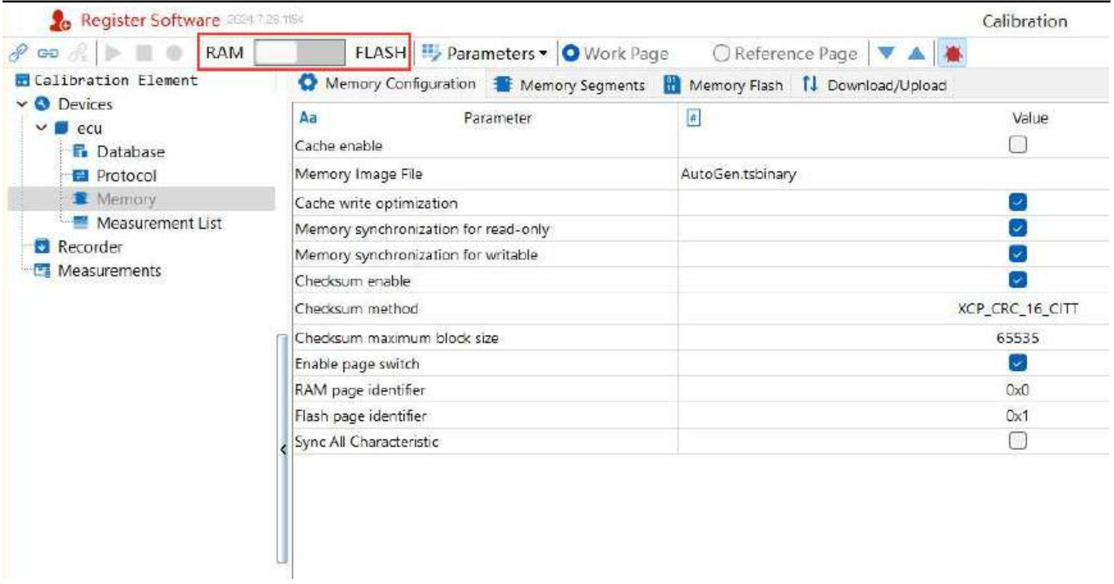


As shown in the figure above, the ECU has set a memory mapping, so when calibrating the Flash address of 0x08D0000-0x08DFFFF, the actual direct calibration is of the RAM address of 0x02D0000-0x02DFFFF. After the calibration is completed, a Hex file is formed on the PC side, which is then written to the Flash address through the Bootloader.

If the switch is not made, the CCP module of the ECU will directly calibrate at the Flash address received. Because Flash is different from RAM in that it cannot be written randomly, it will trigger a hardware error in Flash writing, directly causing the ECU calibration module to exit abnormally.

1.21.5.2. Switch Calibration Page

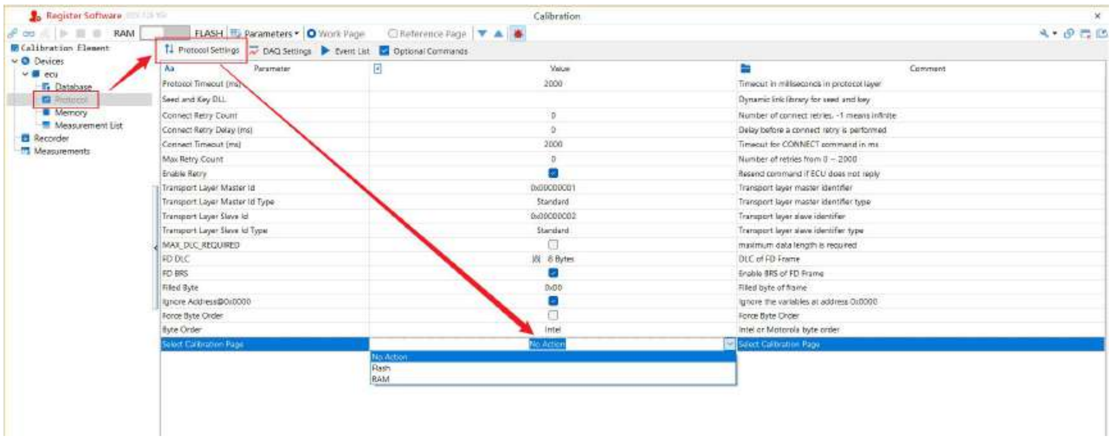
When the ECU is powered on for the first time, after completing the connection with TSMaster, if the ECU is still in the Flash calibration state, user only needs to click the switch button on the calibration manager to use the CCP command to switch the ECU to RAM calibration. As shown in the figure below:



Parameter		Value
Cache enable		☐
Memory Image File	AutoGen.tsbinary	
Cache write optimization		✓
Memory synchronization for read-only		✓
Memory synchronization for writable		✓
Checksum enable		✓
Checksum method		XCP_CRC_16_CITT
Checksum maximum block size		65535
Enable page switch		✓
RAM page identifier		0x0
Flash page identifier		0x1
Sync All Characteristic		☐

1.21.5.3. Automatic Switching During Connection Process

To perform the calibration of data normally, the calibrated ECU needs to switch to the corresponding page. This switching process can be manually done by the user after the ECU connection is completed, or the user can choose to complete the switching action during the ECU connection process. The specific method to be used needs to be selected in the Protocol -> Protocol Settings -> Select Calibration Page configuration, as shown in the figure below:



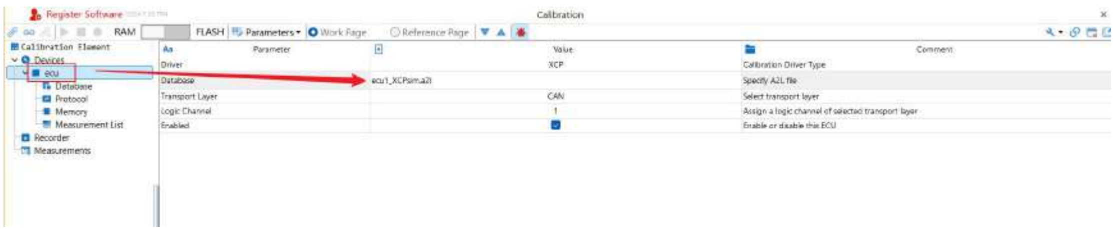
Parameter	Value	Comment
Protocol Timeout (ms)	2000	Timeout in milliseconds in protocol layer
Seed and Key DLL		Dynamic link library for seed and key
Connect Retry Count	0	Number of connect retries. -1 means infinite
Connect Retry Delay (ms)	0	Delay before a connect retry is performed
Connect Timeout (ms)	2000	Timeout for CONNECT command in ms
Max Retry Count	0	Number of retries from 0 ~ 2000
Enable Retry	✓	Resend command if ECU does not reply
Transport Layer Master Id	0x00000001	Transport layer master identifier
Transport Layer Master Id Type	Standard	Transport layer master identifier type
Transport Layer Slave Id	0x00000002	Transport layer slave identifier
Transport Layer Slave Id Type	Standard	Transport layer slave identifier type
MAX_DLC_REQUIRED	☐	maximum data length is required
FD DLC	8 Bytes	DLC of FD Frame
FD BRS	✓	Enable BRS of FD Frame
Filled Byte	0x00	Filled byte of frame
Ignore Address@0x0000	✓	Ignore the variables at address 0x0000
Force Byte Order	☐	Force Byte Order
Byte Order	Intel	Intel or Motorola byte order
Select Calibration Page	No Action	Select Calibration Page

No Action
Flash
RAM

1.21.6. A2L File

1.21.6.1. Load New A2L File

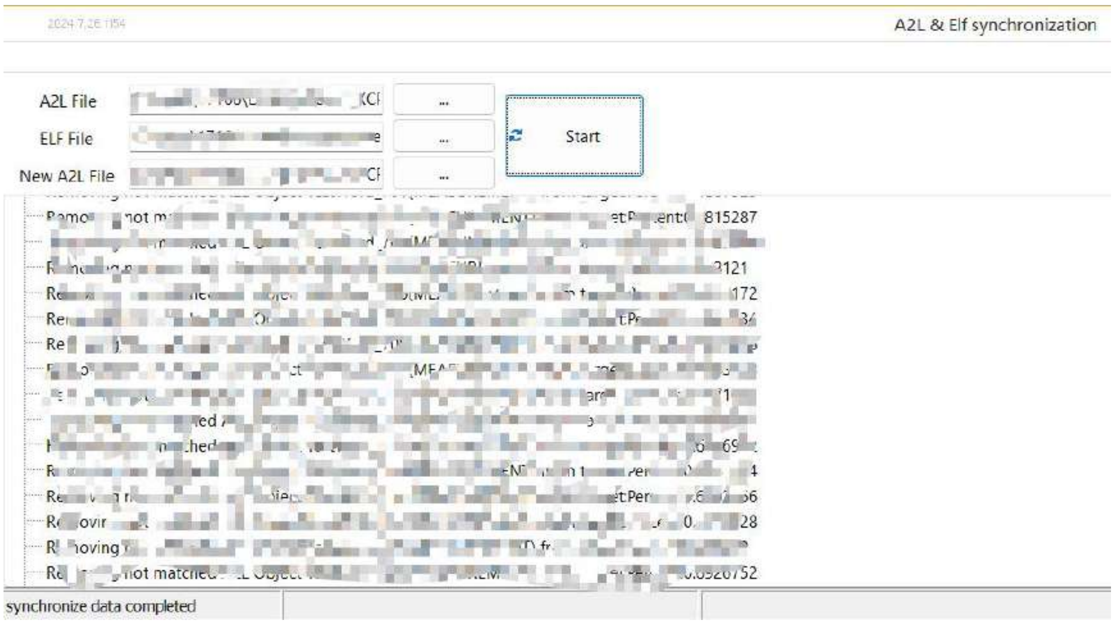
After completing the above configuration, save a basic template project. When there are subsequent changes to the A2L file, simply update the A2L file on this basic project. As shown in the figure below:



Parameter		Value	Comment
Driver		XCP	Calibration Driver Type
Database	ecu1_XCPsim.a2l		Specify A2L file
Transport Layer		CAN	Select transport layer
Logic Channel		1	Assign a logic channel of selected transport layer
Enabled		✓	Enable or disable this ECU

1.21.6.2. Update A2L File

Update the internal addresses in the A2L file. TSMaster provides an A2L synchronization module that allows users to synchronize the new data addresses to a new A2L file after updating the Elf data, as shown below:



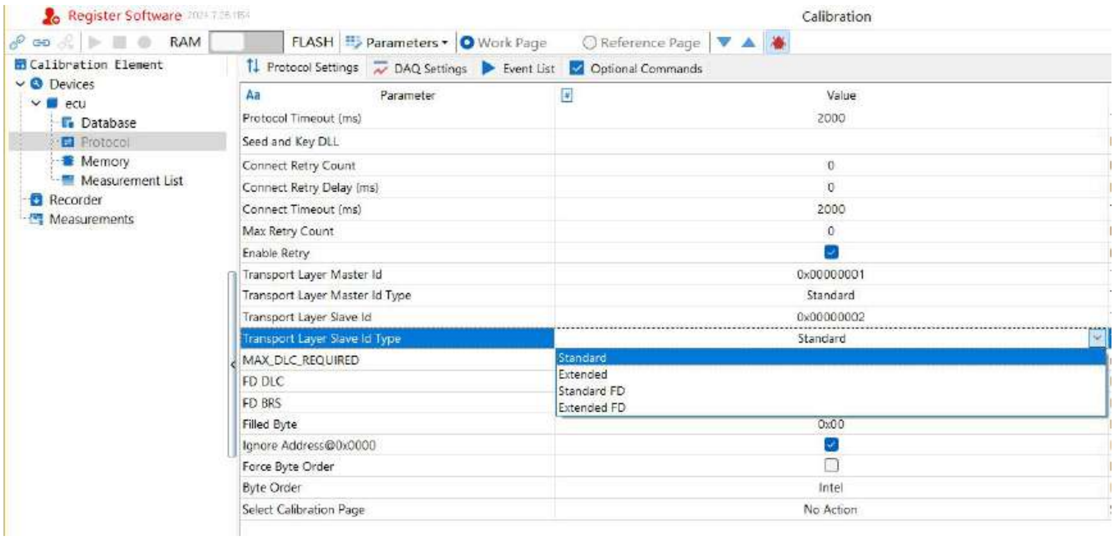
A2L File [redacted]
ELF File [redacted]
New A2L File [redacted]

[redacted]

synchronize data completed

1.21.7. Use CANFD Channel

TSMaster supports XCP calibration over CAN FD. To set up the calibration features for CAN FD, user needs to go to the calibration module -> ecu -> Protocol, and select the transport layer master/slave Id type. As shown in the figure below, choose standard FD/Extended FD:

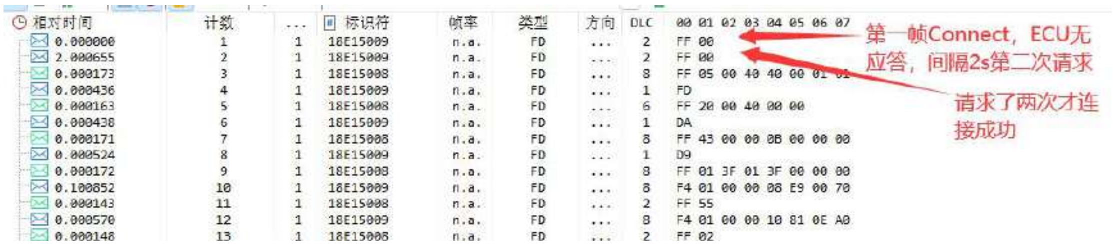


Parameter	Value
Protocol Timeout (ms)	2000
Seed and Key DLL	
Connect Retry Count	0
Connect Retry Delay (ms)	0
Connect Timeout (ms)	2000
Max Retry Count	0
Enable Retry	✓
Transport Layer Master Id	0x00000001
Transport Layer Master Id Type	Standard
Transport Layer Slave Id	0x00000002
Transport Layer Slave Id Type	Standard
MAX_DLC_REQUIRED	Standard / Extended / Standard FD / Extended FD
FD DLC	
FD BRS	
Filled Byte	0x00
Ignore Address@0x0000	✓
Force Byte Order	☐
Byte Order	Intel
Select Calibration Page	No Action

1.21.8. ECU Connection Fail (Connect Fail)

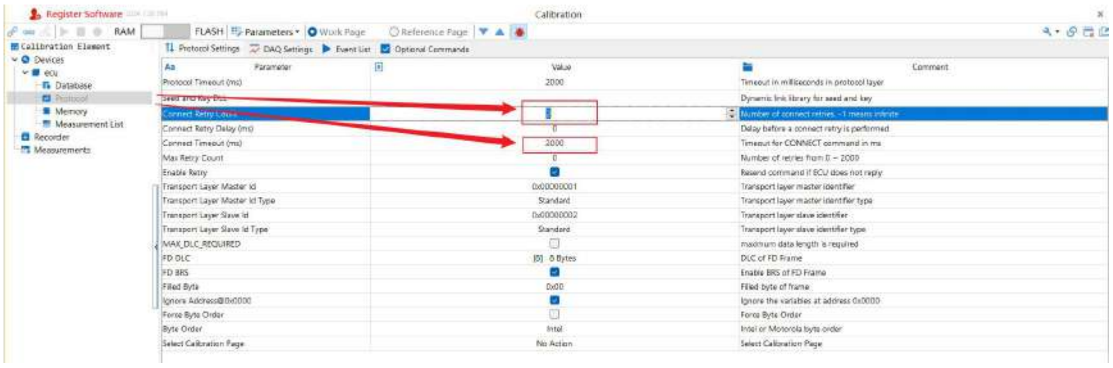
1.21.8.1. Set Reconnection to ECU

During the process of connecting to the ECU, this case may occur. After receiving the first frame of the Connect command, the ECU node does not respond, and only responds after receiving the second frame of the connection command, as shown in the figure below:



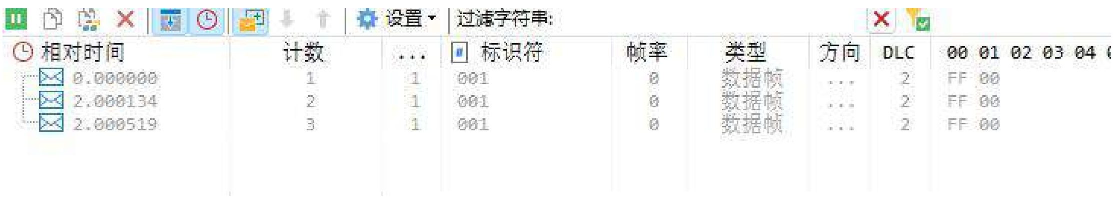
相对时间	计数	...	标识符	帧率	类型	方向	DLC	00 01 02 03 04 05 06 07
0.000000	1	1	18E15009	n.a.	FD	...	2	FF 00
2.000655	2	1	18E15009	n.a.	FD	...	2	FF 00
0.000173	3	1	18E15008	n.a.	FD	...	8	FF 05 00 40 40 00 01 01
0.000436	4	1	18E15009	n.a.	FD	...	1	FD
0.000163	5	1	18E15008	n.a.	FD	...	6	FF 20 00 40 00 00
0.000438	6	1	18E15009	n.a.	FD	...	1	DA
0.000171	7	1	18E15008	n.a.	FD	...	8	FF 43 00 00 0B 00 00 00
0.000524	8	1	18E15009	n.a.	FD	...	1	D9
0.000172	9	1	18E15008	n.a.	FD	...	8	FF 01 3F 01 3F 00 00 00
0.100852	10	1	18E15009	n.a.	FD	...	8	F4 01 00 00 08 E9 00 70
0.000143	11	1	18E15008	n.a.	FD	...	2	FF 55
0.000570	12	1	18E15009	n.a.	FD	...	8	F4 01 00 00 10 81 0E A0
0.000148	13	1	18E15008	n.a.	FD	...	2	FF 02

If the calibration has not configured a reconnection mechanism, it will cause the software to attempt only one connection and then return an error indicating a connection failure. The solution is to configure the reconnection mechanism for the ECU, from ecu -> Protocol -> Protocol Settings, as shown in the figure below:



Parameter	Value	Comment
Protocol Timeout (ms)	2000	Timeout in milliseconds in protocol layer
Seed and Key Dll		Dynamic link library for seed and key
Connect Retry Count	2	Number of connect retries, -1 means infinite
Connect Retry Delay (ms)	0	Delay before a connect retry is performed
Connect Timeout (ms)	2000	Timeout for CONNECT command in ms
Max Retry Count	0	Number of retries from 0 ~ 2000
Enable Retry	✓	Resend command if ECU does not reply
Transport Layer Master Id	0x00000001	Transport layer master identifier
Transport Layer Master Id Type	Standard	Transport layer master identifier type
Transport Layer Slave Id	0x00000002	Transport layer slave identifier
Transport Layer Slave Id Type	Standard	Transport layer slave identifier type
MAX_DLC_REQUIRED	☐	maximum data length is required
FD DLC	[8] 8 Bytes	DLC of FD Frame
FD BRS	✓	Enable BRS of FD Frame
Filled Byte	0x00	Filled byte of frame
Ignore Address@0x0000	✓	Ignore the variables at address 0x0000
Force Byte Order	☐	Force Byte Order
Byte Order	Intel	Intel or Motorola byte order
Select Calibration Page	No Action	Select Calibration Page

After the settings as shown in the above figure, the software will attempt to connect 2 more times, as shown in the figure below:



相对时间	计数	...	标识符	帧率	类型	方向	DLC	00 01 02 03 04
0.000000	1	1	001	0	数据帧	...	2	FF 00
2.000134	2	1	001	0	数据帧	...	2	FF 00
2.000519	3	1	001	0	数据帧	...	2	FF 00

1.21.9. Calibratin Data Management

1.21.9.1. Introduction

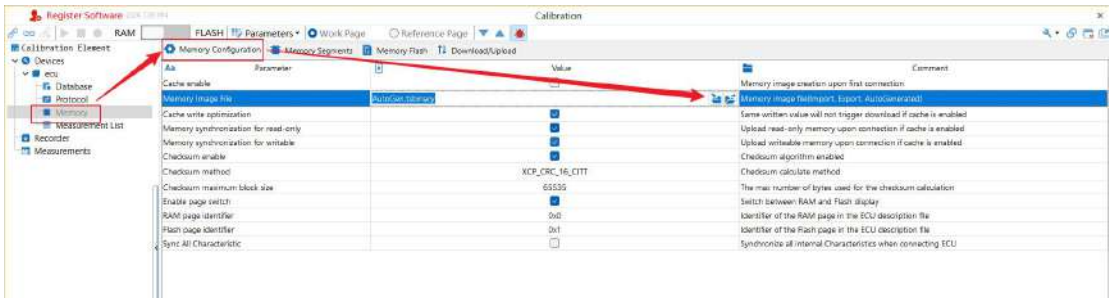
In the calibration module, the management of calibration data is also its core functionality. It mainly includes the following aspects:

- Loading of calibration data. Load existing calibration data into the calibration module.
- Exporting calibration data. Export the calibrated data from the calibration module as data files (s19, hex, mot) for management. This is used for the next calibration or for flashing to the target ECU.
- Flashing calibration data. Download and solidify the calibrated data files into the target ECU through the XCP Program/UDS protocol to make the calibration data effective.
- Flashing of companion applications. On the calibration site, it also supports users to directly download companion applications through the calibration module.

Below, these functions will be described in details.

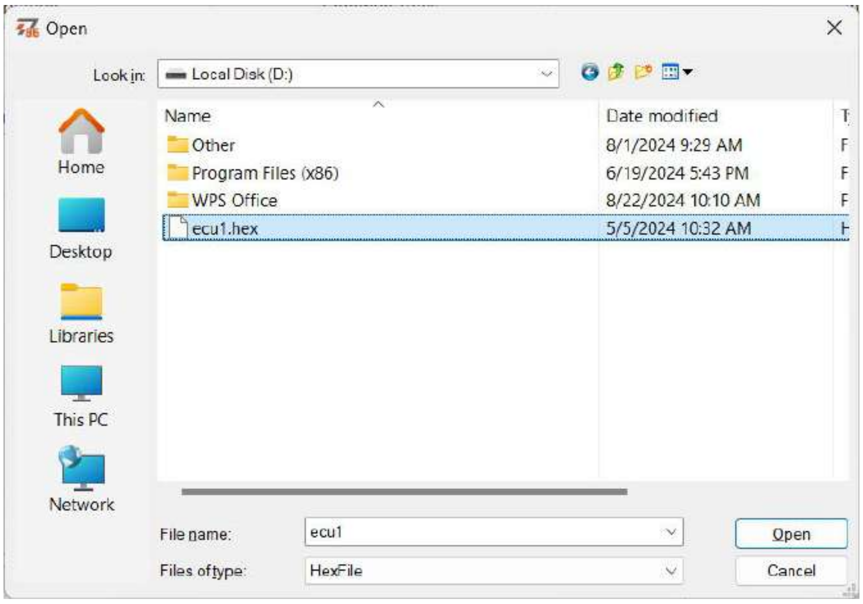
1.21.9.2. Load Calibration Data

The path for loading calibration data is as follows: select Target ECU -> Memory -> Memory Configuration -> Memory Image File -> Load button.



Parameter	Value	Comment
Cache enable	☐	Memory image creation upon first connection
Memory image file	AutoGen.tsbinary	Memory image file(Import, Export, AutoGenerated)
Cache write optimization	✓	Same written value will not trigger download if cache is enabled
Memory synchronization for read-only	✓	Upload read-only memory upon connection if cache is enabled
Memory synchronization for writable	✓	Upload writeable memory upon connection if cache is enabled
Checksum enable	✓	Checksum algorithm enabled
Checksum method	XCP_CRC_16_CITT	Checksum calculate method
Checksum maximum block size	65535	The max number of bytes used for the checksum calculation
Enable page switch	✓	Switch between RAM and Flash display
RAM page identifier	0x0	Identifier of the RAM page in the ECU description file
Flash page identifier	0x1	Identifier of the Flash page in the ECU description file
Sync All Characteristic	☐	Synchronize all internal Characteristics when connecting ECU

This operation is equivalent to loading a hex, s19 file in competitors' software. When loading, it directly supports multiple formats. After loading, the file will automatically be converted and stored as (ECU Name + .hex) file. The file loading dialog box is as shown in the figure:



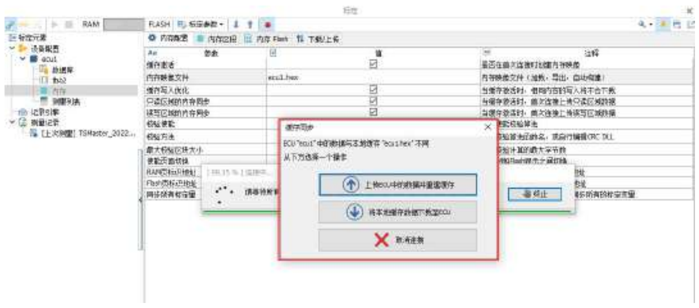
Name	Date modified
Other	8/1/2024 9:29 AM
Program Files (x86)	6/19/2024 5:43 PM
WPS Office	8/22/2024 10:10 AM
ecu1.hex	5/5/2024 10:32 AM

File name: ecu1
Files of type: HexFile

Support The Loading of Multiple Data Formats

1.21.9.3. Slow Connection Speeds Issue

When using the calibration module, some users have reported issues with slow connection speeds, such as taking more than a minute to complete an ECU connection process. This is because, when connecting to the ECU, the calibration module first checks whether the calibration data inside the ECU matches the calibration data in the calibration software. If there is a mismatch, it is necessary to synchronize the data from the calibration software to the ECU, or vice versa, as shown in the figure below:



参数		值	注释
缓存激活		☑	是否在首次连接时创建内存映像
内存映像文件	ecu1.hex		内存映像文件（加载，导出，自动创建）
缓存写入优化		☑	当缓存激活时，相同内容的写入将不会下载
只读区域的内存同步		☑	当缓存激活时，首次连接上传只读区域数据
读写区域的内存同步		☑	当缓存激活时，首次连接上传读写区域数据
校验使能			[illegible]
校验方法			[illegible]
最大校验区块大小			[illegible]
使能页面切换			[illegible]
RAM页标识符地址			[illegible]
Flash页标识符地址			[illegible]
同步所有标定量			[illegible]

缓存同步

ECU "ecu1" 中的数据与本地缓存 "ecu1.hex" 不同

从下方选择一个操作

上传ECU中的数据并重建缓存

将本地缓存数据下载至ECU

取消连接

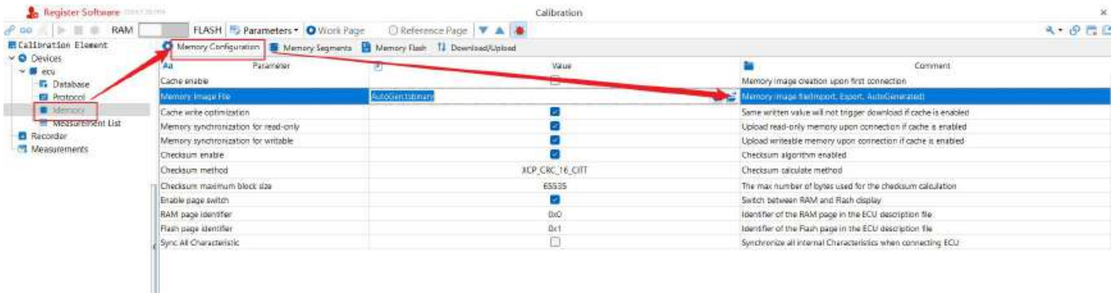
In more extreme cases, the calibration software may not have loaded a calibration file at all, so when connecting, it is necessary to read the calibration data from the ECU into the calibration software. When there is a large amount of calibration data, this reading process can be very time-consuming, which is why the connection process is slow.

Therefore, the solution is to load the calibration data file into the memory image before connecting. When the ECU is connected, if it is detected that the calibration data in the ECU is consistent with the calibration data in the software, there will be no process of synchronizing data, which can quickly complete the ECU connection process (actual testing indicates that the time is in seconds).

1.21.9.4. Export Calibration Data

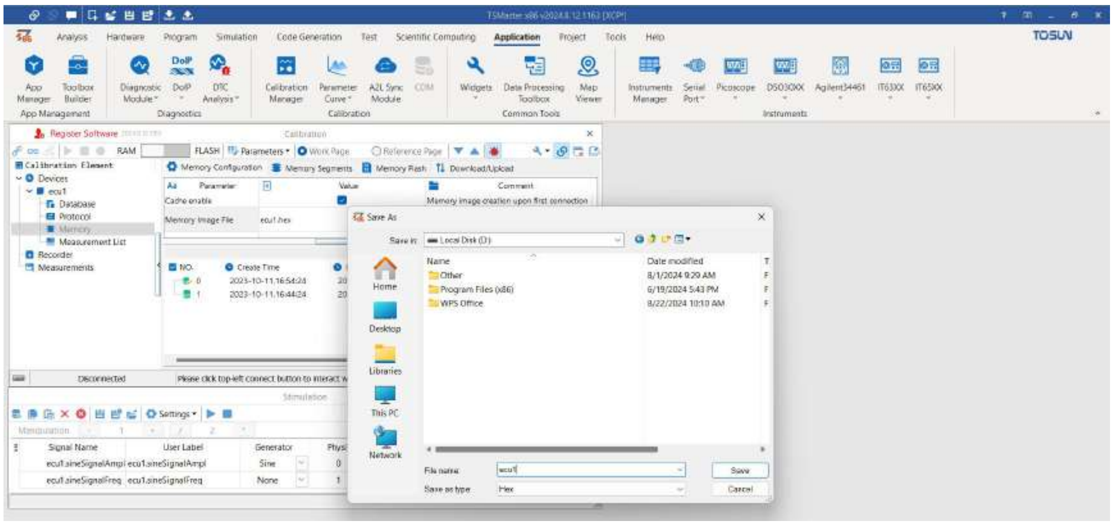
1.21.9.4.1. Directly Export The Existing Calibration File

The path to export data files from the existing calibration file is as follows: Select Target ECU -> Memory -> Memory Configuration -> Memory Image File -> Export button.



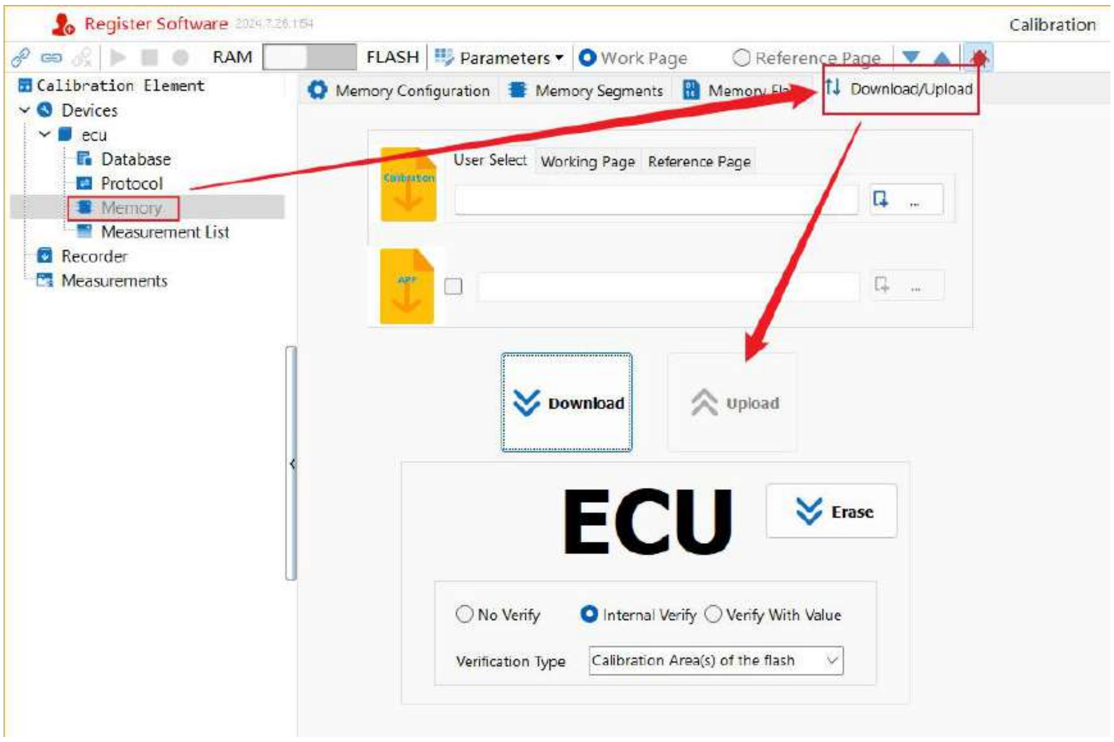
Parameter		Value	Comment
Cache enable		☐	Memory image creation upon first connection
Memory Image File	AutoGen.tsbinary		Memory image file(Import, Export, AutoGenerated)
Cache write optimization		☑	Same written value will not trigger download if cache is enabled
Memory synchronization for read-only		☑	Upload read-only memory upon connection if cache is enabled
Memory synchronization for writable		☑	Upload writeable memory upon connection if cache is enabled
Checksum enable		☑	Checksum algorithm enabled
Checksum method		XCP_CRC_16_CITT	Checksum calculate method
Checksum maximum block size		65535	The max number of bytes used for the checksum calculation
Enable page switch		☑	Switch between RAM and Flash display
RAM page identifier		0x0	Identifier of the RAM page in the ECU description file
Flash page identifier		0x1	Identifier of the Flash page in the ECU description file
Sync All Characteristic		☐	Synchronize all internal Characteristics when connecting ECU

Calibration data supports to save in formats such as s19, hex, bin, etc. After clicking the Data Export button, the format selection for export data is as shown below:



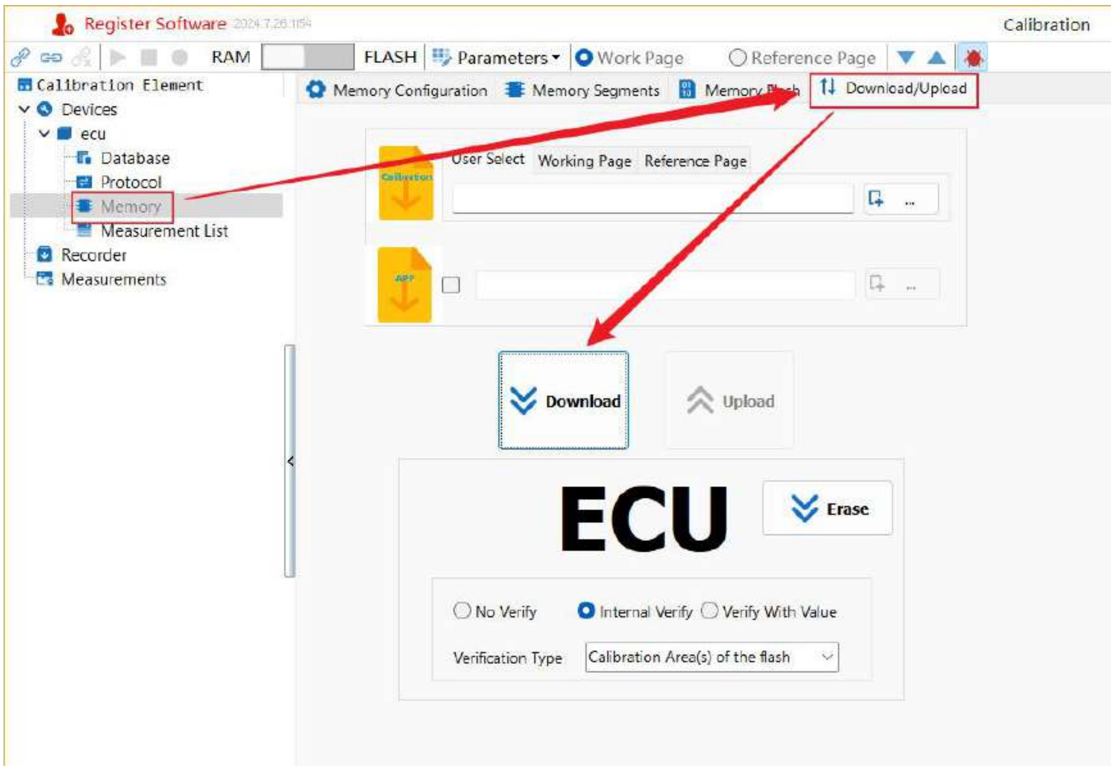
1.21.9.4.2. Read and Export from ECU

Before reading and exporting data file from the ECU, user needs to first complete the connection to the ECU. The operation path is: Connect ECU -> Memory -> Download/Upload -> Upload.



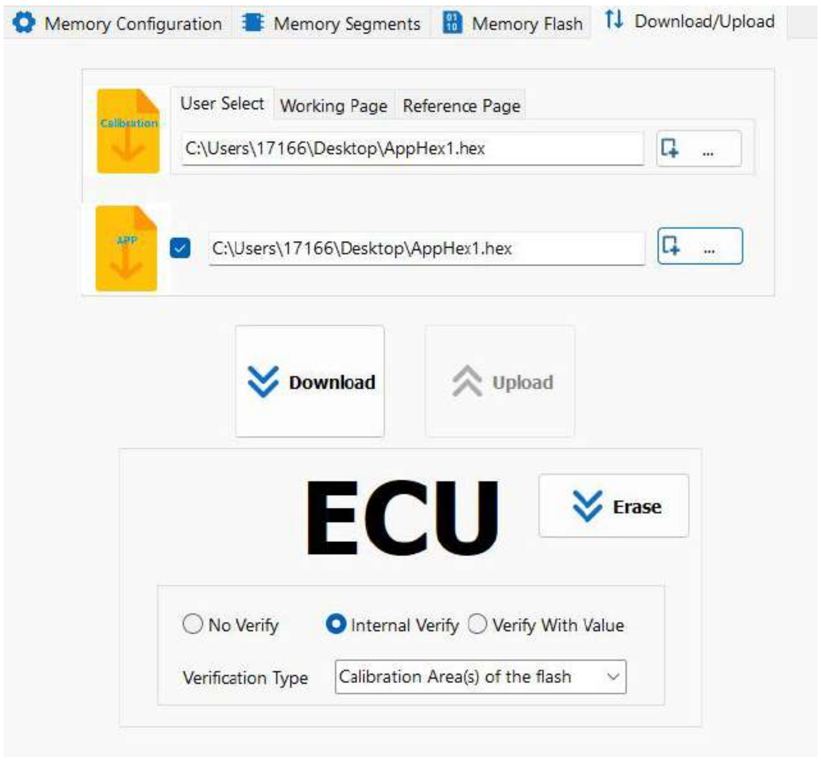
1.21.9.5. Calibration Data/Application Flashing

Using the XCP Program protocol (UDS protocol will be explained separately), the calibrated data file is downloaded and solidified into the target ECU to make the calibration data effective. The operation path is as follows: Select ECU -> Memory -> Download/Upload -> Download.



1.21.9.5.1. Basic Configuration

The related configurations mainly include selecting the calibration file, enabling/selecting the application file, and choosing the verification type, as shown below:



C:\Users\17166\Desktop\AppHex1.hex

C:\Users\17166\Desktop\AppHex1.hex

No Verify [✓ Internal Verify] Verify With Value

Verification Type: Calibration Area(s) of the flash

1.21.9.5.2. Whether to Select Application Data

Configuration 2 (Enable Application File) allows users to choose whether to download application data simultaneously. Under normal circumstances, the application data of the ECU only needs to be reloaded after a new version is released. After users have made modifications to the calibration data, they only need to download the calibration data alone. At this point, configuration 2 should be set to not load the application data, which can greatly reduce the amount of data being overwritten and save flashing time.

When developers release a new version of the application data, checkbox for configuration 2 should be checked, and the application data should be loaded into the configuration. This way, during downloading, the XCP protocol can simultaneously download both application data and calibration data to the ECU.

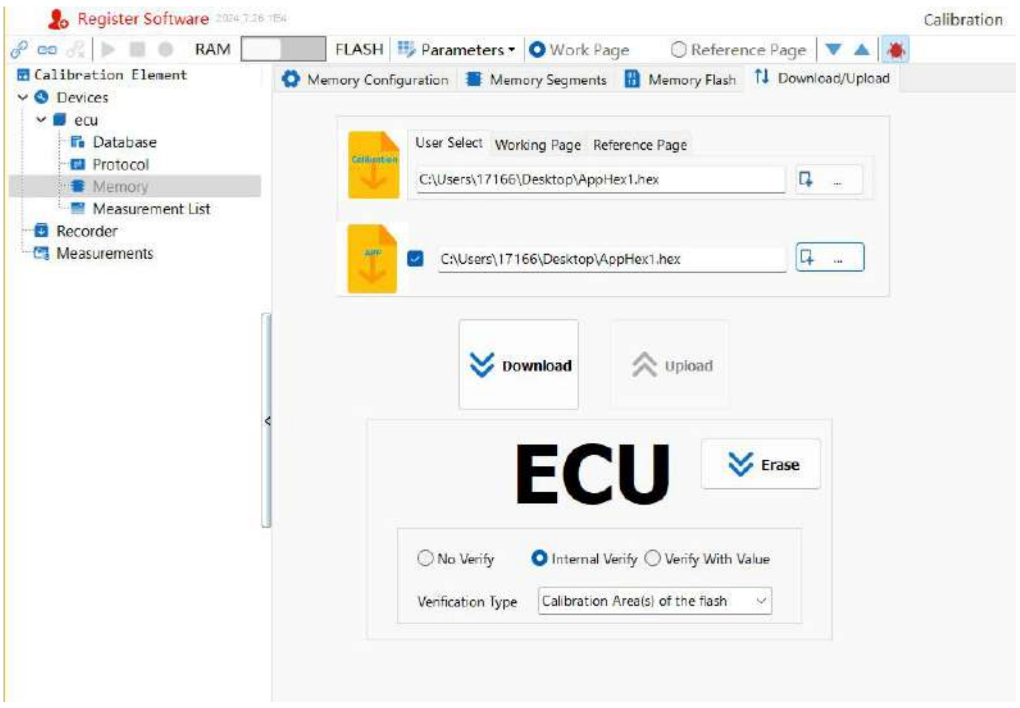
1.21.9.5.3. Difference in Download Speeds

In actual testing of a controller from an automotive manufacturer in East China, the following differences were found:

- When the APP (application) in the ECU is ready, the download mode is non-block download mode, which is a query-response download method;

- When the APP in the ECU is erased, the download mode is block download mode.

The speed difference between these two download modes can be as much as 5 to 10 times. Therefore, TSMaster specifically provides an erase mode to clear the internal APP program in the ECU. As shown below:



C:\Users\17166\Desktop\AppHex1.hex

C:\Users\17166\Desktop\AppHex1.hex

( ) No Verify [✓ Internal Verify] ( ) Verify With Value

Verification Type: Calibration Area(s) of the flash

That is to say, after completing the configuration of the download parameters, the recommended download method is:

1. Erase the internal data of the ECU.

2. Execute the download process.

By adopting this method, the flashing speed will be much faster than directly executing the download process. Data of 0x3C0000 (2359296 bytes) can be downloaded within 1 minute. Users can conduct a test to verify this

1.21.9.6. Common Errors

1.21.9.6.1. ECU Can not Operate Normally after the Download

- **Description:**

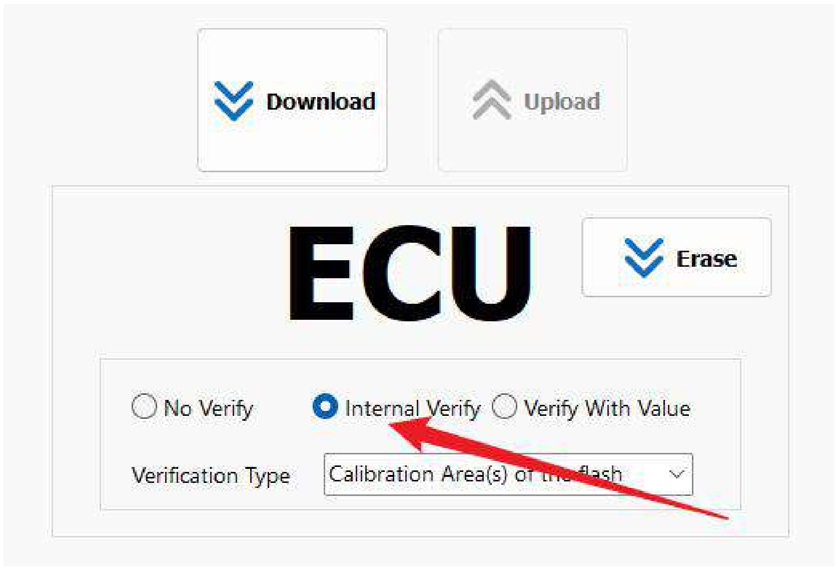
Customers found that after the calibration data and application data were downloaded to the ECU, the ECU could not operate normally. By comparing the messages, it was confirmed that all data were correctly downloaded to the correct addresses in the ECU, but the ECU still worked abnormally after starting up.

➢ **Cause Analysis:**

After investigation, it was found that the verification type was not checked in the download module. For this ECU's download process, it is stipulated that after downloading the data, an internal verification must be carried out to confirm that the data file is correct. If there is no internal verification, the ECU dare not to start up, which means that the data within the ECU is considered invalid.

➢ **Solution:**

Check the verification option for the ECU. As follows:



[✓ Internal Verify]

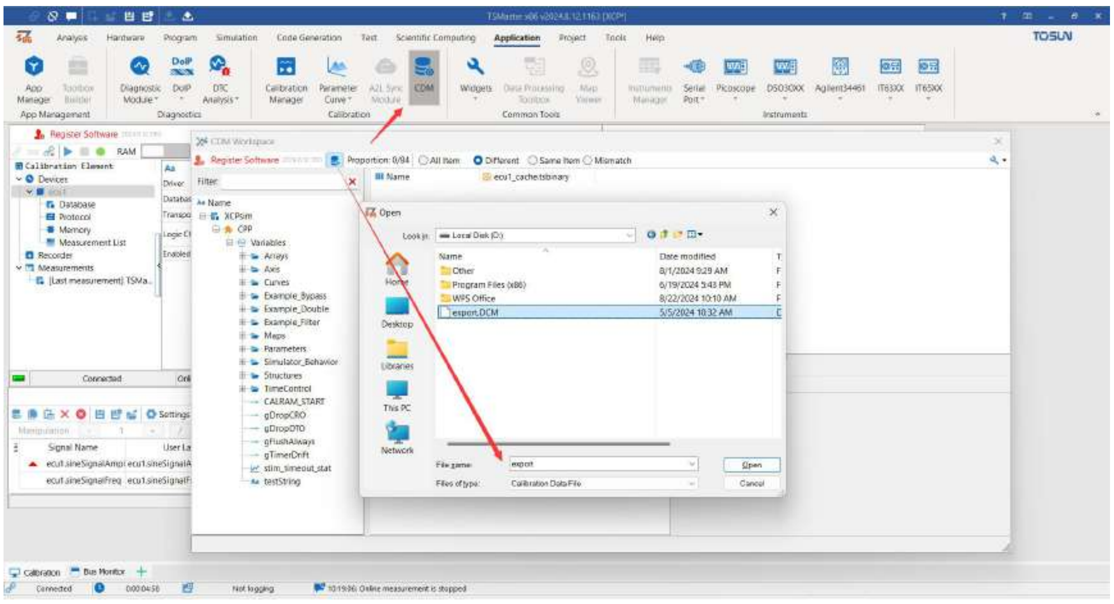
Verification Type: Calibration Area(s) of the flash

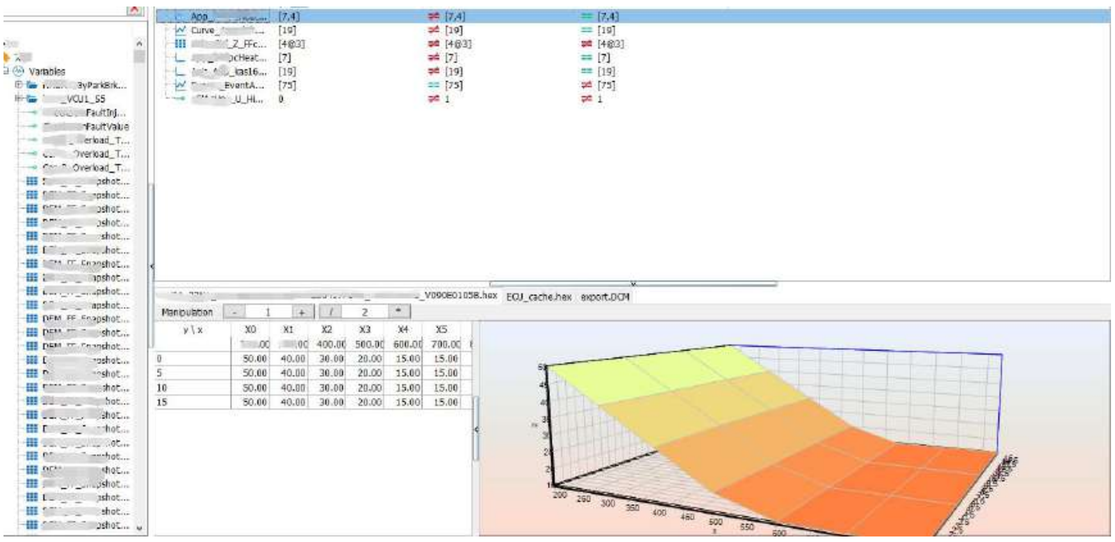
1.21.10. Calibration Data Manager

In the TSMaster calibration module, to manage calibration process data, a dedicated Calibration Data Manager (Calibration Data Manager Studio) is provided for calibration personnel to handle tasks such as calibration data management, data comparison, quick modifications, and trimming. Roughly, it mainly includes the following types of operations:

1.21.10.1. Supports Loading Various Data Types

This includes data files such as Hex, S19, bin, as well as calibration data management files like Par, DCM. The import window is shown in the figure below:



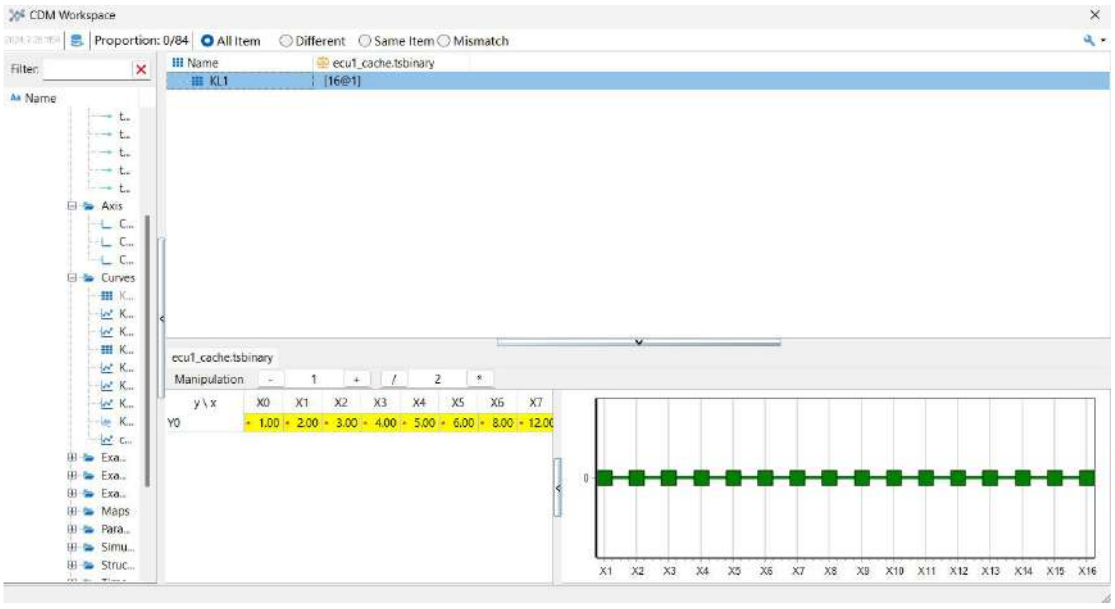


y \ x	X0	X1	X2	X3	X4	X5
	[redacted].00	[redacted].00	400.00	500.00	600.00	700.00
0	50.00	40.00	30.00	20.00	15.00	15.00
5	50.00	40.00	30.00	20.00	15.00	15.00
10	50.00	40.00	30.00	20.00	15.00	15.00
15	50.00	40.00	30.00	20.00	15.00	15.00

1.21.10.2. Analysis and Comparison of Multiple Calibration Data

TSMaster's CDM module offers filtering options that primarily include:

- All Item: displaly all variables.
- Different: only display variables that are different.
- Same Items: only display variables that have not changed.
- Mismatched Items: variables that appear in the loaded data file but are not defined in the A2L.

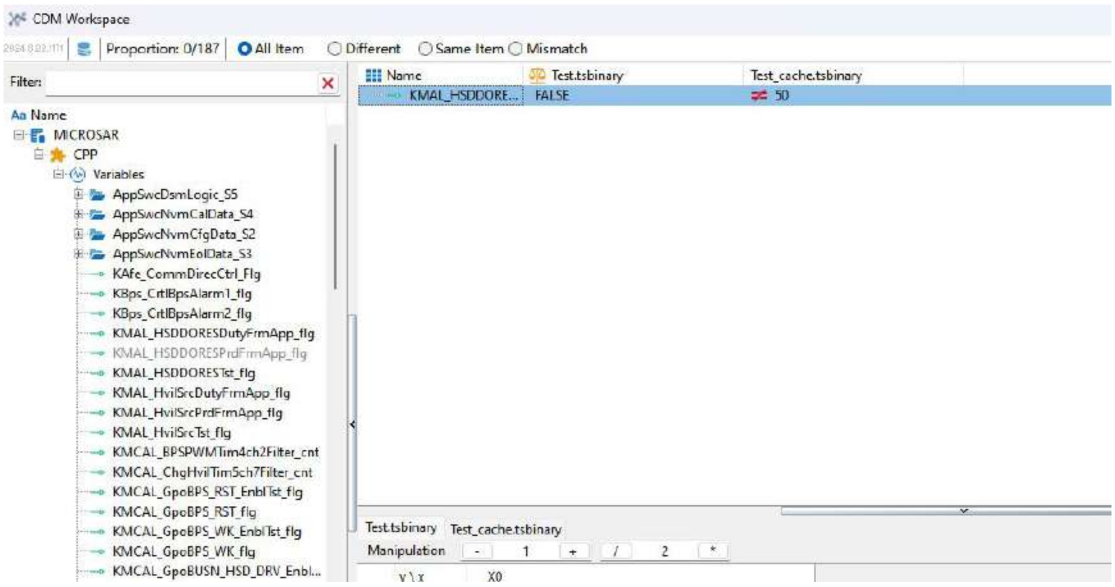


Name	ecu1_cache.tsbinary
KL1	[16@1]

ecu1_cache.tsbinary

y \ x	X0	X1	X2	X3	X4	X5	X6	X7
Y0	1.00	2.00	3.00	4.00	5.00	6.00	8.00	12.00

Users can choose a specified file as the reference file. After switching the reference file, the corresponding comparison and analysis results will be refreshed in real time, as shown in the figure below:



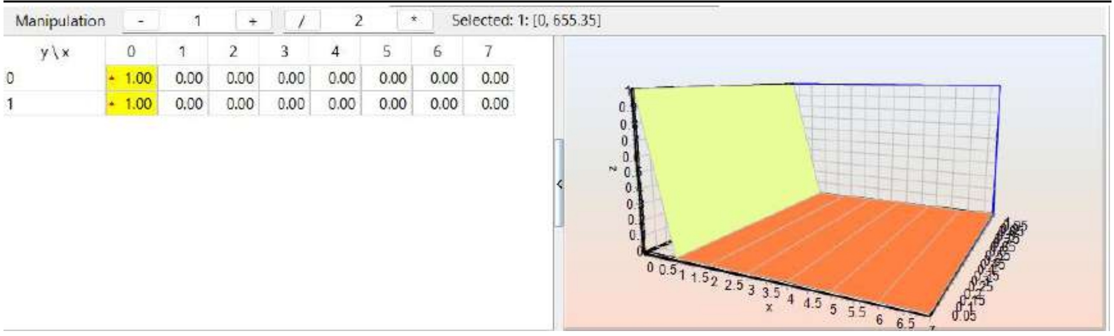
MICROSAR
CPP
Variables
AppSwcDsmLogic_S5
AppSwcNvmCalData_S4
AppSwcNvmCfgData_S2
AppSwcNvmEolData_S3
KAfe_CommDirecCtrl_Flg
KBps_CrtlBpsAlarm1_flg
KBps_CrtlBpsAlarm2_flg
KMAL_HSDDORESDutyFrmApp_flg
KMAL_HSDDORESPrdFrmApp_flg
KMAL_HSDDORESTst_flg
KMAL_HvilSrcDutyFrmApp_flg
KMAL_HvilSrcPrdFrmApp_flg
KMAL_HvilSrcTst_flg
KMCAL_BPSPWMTim4ch2Filter_cnt
KMCAL_ChgHvilTim5ch7Filter_cnt
KMCAL_GpoBPS_RST_EnblTst_flg
KMCAL_GpoBPS_RST_flg
KMCAL_GpoBPS_WK_EnblTst_flg
KMCAL_GpoBPS_WK_flg
KMCAL_GpoBUSN_HSD_DRV_Enbl...

Name	Test.tsbinary	Test_cache.tsbinary
KMAL_HSDDORE...	FALSE	50

Test.tsbinary Test_cache.tsbinary

y \ x X0

Selecting a specific data item allows the user to view the detailed change status of that data across different data source files. As shown in the figure below, data with a yellow background indicates that it has been modified.

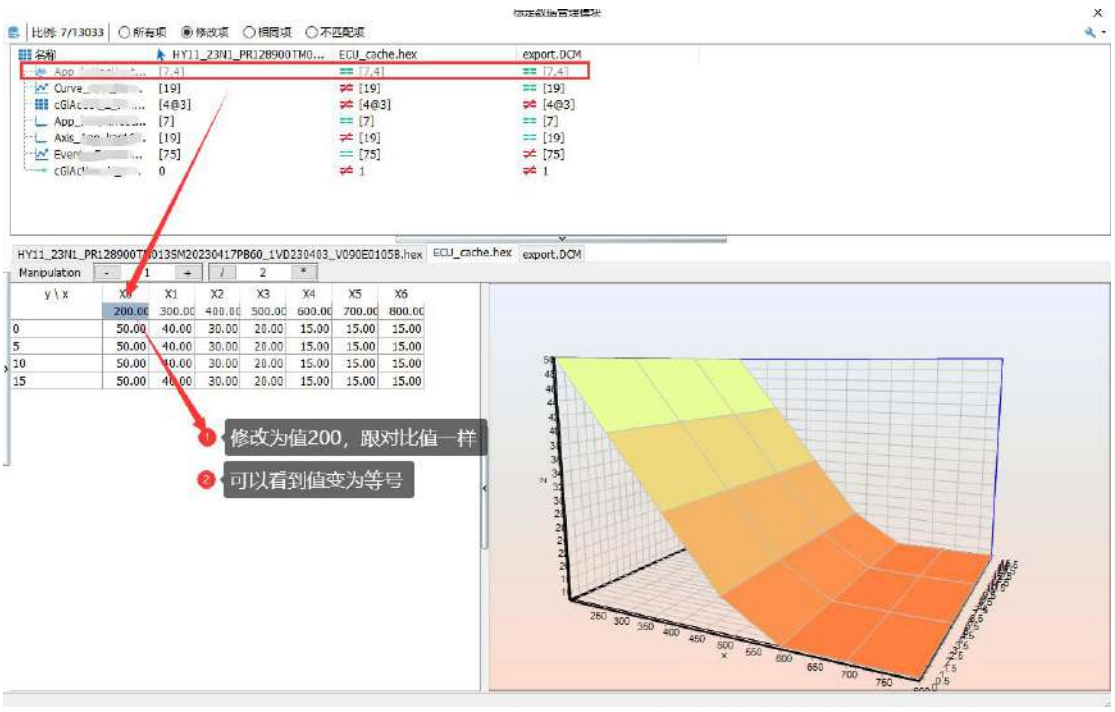


Manipulation | - | 1 | + | / | 2 | * | Selected: 1: [0, 655.35]

y \ x	0	1	2	3	4	5	6	7
0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

1.21.10.3. Offline Modification of Calibration Parameters

In the CDM module, users can directly modify calibration data files, with the operation process being similar to static calibration. After completing the operations, users can export data to data files such as hex/s19 or calibration data management files such as cdm/par.



标定数据管理模块

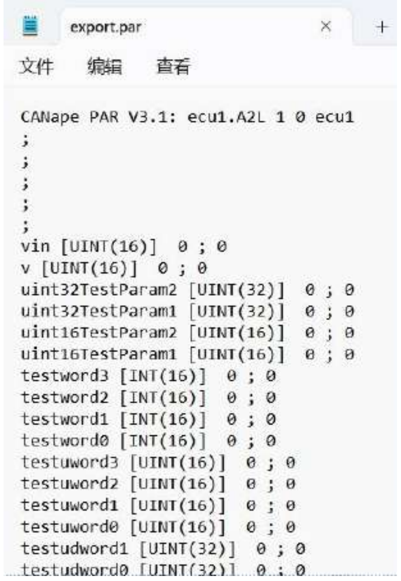
比例: 7/13033 | ○所有项 [✓ ◉修改项] ○相同项 ○不匹配项

名称	HY11_23N1_PR128900TM0...	ECU_cache.hex	export.DCM
App [redacted]	[7,4]	== [7,4]	== [7,4]
Curve_[redacted]	[19]	≠ [19]	== [19]
cGlAc[redacted]	[4@3]	≠ [4@3]	≠ [4@3]
App_[redacted]	[7]	== [7]	== [7]
Axis_[redacted]	[19]	≠ [19]	== [19]
Even[redacted]	[75]	== [75]	≠ [75]
cGlAc[redacted]	0	≠ 1	≠ 1

HY11_23N1_PR128900TM013SM20230417PB60_1VD230403_V090E0105B.hex | ECU_cache.hex | export.DCM

Manipulation | - | 1 | + | / | 2 | *

y \ x	X0	X1	X2	X3	X4	X5	X6
	200.00	300.00	400.00	500.00	600.00	700.00	800.00
0	50.00	40.00	30.00	20.00	15.00	15.00	15.00
5	50.00	40.00	30.00	20.00	15.00	15.00	15.00
10	50.00	40.00	30.00	20.00	15.00	15.00	15.00
15	50.00	40.00	30.00	20.00	15.00	15.00	15.00



```
CANape PAR V3.1: ecu1.A2L 1 0 ecu1
;
;
;
;
;
vin [UINT(16)]  0 ; 0
v [UINT(16)]  0 ; 0
uint32TestParam2 [UINT(32)]  0 ; 0
uint32TestParam1 [UINT(32)]  0 ; 0
uint16TestParam2 [UINT(16)]  0 ; 0
uint16TestParam1 [UINT(16)]  0 ; 0
testword3 [INT(16)]  0 ; 0
testword2 [INT(16)]  0 ; 0
testword1 [INT(16)]  0 ; 0
testword0 [INT(16)]  0 ; 0
testuword3 [UINT(16)]  0 ; 0
testuword2 [UINT(16)]  0 ; 0
testuword1 [UINT(16)]  0 ; 0
testuword0 [UINT(16)]  0 ; 0
testudword1 [UINT(32)]  0 ; 0
testudword0 [UINT(32)]  0 ; 0
```

1.22. J2534

1.22.1. Temporary Installation Instrucations

->Run startup.bat as administrator user.

This will register the WIN system registry according to the requirements of J2534-1. The .dll file will be copied to the system folder. Omit this step if no need to check the registry. Simply place all the .dll files in the folder to a location where they can be called (which varies depending on the programming language.

1.22.2. Introduction

The provided DLLs are all 32-bit DLLs, among which the J2534 DLL is the one called by the user. The DLLs need to be called in the form of the WINAPI.

Supported protocols are as follows: ISO J2534-1 2004，ISO J2534-2 2019，GMW 17753.

The supported version of J2534 is **04.04**.

1.22.3. Function Description

1.22.3.1. PassThruOpen

The J2534 version 04.04 only supports the use of one CAN channel at a time. By default, the "open" function will connect to channel 1 of the TOSUN device that comes with the shipment. When the Open function is called, the software will connect with the hardware and may directly initiate communication. This is because the TOSUN API can only obtain the hardware's handle after the connection is established.

If users wish to specify a channel, they can make the call in the following manner. The call below will connect to channel 2 (but still cannot specify the hardware):

```
unsigned long deviceID;
PassThruOpen("J2534-2:TOSUN CH2", &deviceID);
```

1.22.3.2. PassThruClose

Disconnect the hardware and release the resources in the DLL.

1.22.3.3. PassThruConnect

Select the protocol to be used. If a non-CAN FD protocol is specified, the hardware's CAN baud rate will be directly modified. If a CAN FD protocol is specified, the CAN baud rate will be cached. It will only be truly applied when the PassThruIoctl is called to set the data field baud rate.

The input variable Flags will not be checked, and the hardware will definitely support both the 11-bit CAN ID standard frame and the 29-bit CAN ID extended frame.

Supported protocols are as follows:

CAN， CAN_PS， FD_CAN_PS， CAN_FD_PS， ISO15765， ISO15765_PS， FD_ISO15765_PS，ISO15765_FD_PS.

It should be noted that due to the lack of support for pin selection, there is no practical difference between protocols with and without PS.

1.22.3.4. PassThruDisconnect

All settings made after the connect are deleted. However, this does not disconnect or stop the hardware (for example, the hardware will still respond with ACKs). When connect again, the J2534 API will clean up the messages received in between, and these messages will not be read again.

1.22.3.5. PassThruReadMsgs

Read J2534 messages. The input Timeout parameter will not be processed.

The J2534 API maintains a receive buffer, and messages are placed into the buffer upon reception. The Read function retrieves data from this buffer.

Supported RxStatus:

CAN_29BIT_ID: whether to use extended frames.

TX_MSG_TYPE: whether it is a message that has been echoed back.

FD_CAN_BRS and CAN_FD_BRS: whether to use bit rate switching for the data field.

FD_CAN_FORMAT and CAN_FD_FORMAT: whether it is a CAN FD frame.

FD_CAN_ESI and CAN_FD_ESI: whether the ESI bit of the message is set to 1.

1.22.3.6. PassThruWriteMsgs

Send J2534 messages. All supported types can be sent, and it does not necessarily have to be consistent with the protocol set in Connect. The Timeout parameter will not be processed.

If a user chooses to send ISO15765 messages, but the CAN ID has not been pre-registered using PassThruStartMsgFilter, the function will automatically register a TP layer address mapping with a request ID as this ID, but without a response CAN ID. This is used to trigger a functional addressing request. However, without additional configuration of the filter, the user will not receive any response messages.

Supported TxFlag:

CAN_29BIT_ID: whether to use extended frames.

FD_CAN_BRS and CAN_FD_BRS: whether to use bit rate switching for the data field.

FD_CAN_FORMAT and CAN_FD_FORMAT: whether it is a CAN FD frame.

1.22.3.7. PassThruStartPeriodicMsg

Periodically send messages. Up to 10 periodic messages are supported.

CAN and CAN FD messages, as well as ISO15765 messages that are determined to be single-frame, can be sent periodically. Other messages cannot be sent periodically.

1.22.3.8. PassThruStopPeriodicMsg

Stop periodic sending of messages.

1.22.3.9. PassThruStartMsgFilter

Set filters. Note: According to the J2534 protocol, if a user does not set filters, the user will not receive any messages. The API will discard all received messages.

PASS_FILTER and BLOCK_FILTER can be set under any circumstances, while FLOW_CONTROL_FILTER can only be set when the ISO15765-related protocol is selected during the Connect.

PASS_FILTER and BLOCK_FILTER together support up to 10, and FLOW_CONTROL_FILTER supports up to 64.

For FLOW_CONTROL_FILTER, the function does not verify the input PASSTHRU_MSG. As long as the data is greater than 4, the first 4 bytes will be treated as CAN ID, and the rest of the content will not be processed.

The batch mapping for 29-bit ID provided by GMW 17753 is not supported by the API.

1.22.3.10. PassThruStopMsgFilter

Delete the specified filter.

1.22.3.11. PassThruSetProgrammingVoltage

Not supported, the function will always return ERR_NOT_SUPPORTED.

1.22.3.12. PassThruReadVersion

Can obtain the correct PassThru version number. However, the other two cannot obtain the correct version number.

1.22.3.13. PassThruGetLastError

Get a detailed description of the last error. If it is an error from J2534, it will be described according to J2534. If it is originally an error returned by TOSUN API, error descriptions from TOSUN API will be provided. The protocol specifies that the string length is 80, and error descriptions exceeding this limit will be forcibly truncated.

1.22.3.14. PassThruIoctl

Supported parameters:

GET_CONFIG and SET_CONFIG: they will be explained at the end.

CLEAR_TX_BUFFER: not supported, but it will return success directly.

CLEAR_RX_BUFFER: clear the received buffer.

CLEAR_PERIODIC_MSGS: stop all periodic message transmissions.

CLEAR_MSG_FILTERS: delete all filters.

GET_DEVICE_INFO: get the protocols supported by the API.

GET_PROTOCOL_INFO: obtain the specific content supported by the protocol.

WRITE_MSG_EXTENSION and READ_MSG_EXTENSION: if the handling of CAN FD ISO15765 messages that exceed the PASSTHRU_MSG byte limit is specified in GMW 17753, all such handling methods are supported. If the CAN FD ISO15765 is not as prescribed in GMW 17753, messages that exceed the length will be discarded.

Supported CONFIG parameters:

DATA_RATE: modify the arbitration segment bit rate.

LOOPBACK: whether to support echo reading. Note that echoed messages are not affected by filters.

BIT_SAMPLE_POINT: not supported, but it will return success.

SYNC_JUMP_WIDTH: not supported, but it will return success.

ISO15765_BS: Set the content of the flow control frame that the TOSUN hardware responds with.

ISO15765_STMIN: Set the content of the flow control frame that the TOSUN hardware responds with.

ISO15765_BS_TX: not supported, but it will return success.

ISO15765_STMIN_TX: not supported, but it will return success.

ISO15765_WFT_MAX: supported.

CAN_MIXED_FORMAT: supported. The API supports simultaneous transmission and reception of standard messages of ISO15765 and corresponding protocols.

J1962_PINS: this parameter will take no action and will directly return success. Even if the pin is not set, other functions will not report an error as a result.

FD_CAN_DATA_PHASE_RATE and CAN_FD_DATA_PHASE_RATE: both J2534-2 and GMW 17753 define this parameter, but with different values. Therefore, only the corresponding protocol can set these parameters. It is used to change the data field bit rate of CAN FD. For CAN FD protocols, the CAN FD mode can only be entered after this parameter is set.

FD_ISO15765_TX_DATA_LENGTH and CAN_FD_TC_DATA_LENGTH: both J2534-2 and GMW 17753 define this parameter, but with different values. Therefore, only the corresponding protocols can set these parameters. It is used to set the maximum DLC for single-frame ISO 15765 messages in CAN FD mode.

HS_CAN_TERMINATION and CAN_FD_TERMINATION: both J2534-2 and GMW 17753 define this parameter, but with different values. Therefore, only the corresponding protocols can set these parameters. It is used to set whether to use the terminal resistors on the TOSUN hardware. Protocols defined in J2534-1 use the parameter values defined in J2534-2.

N_CR_MAX: both J2534-2 and GMW 17753 define this parameter with the same name but different values. Currently, this parameter is not supported, but it will directly return success.

ISO15765_PAD_VALUE: all padding bits used by ISO15765 can be utilized.

CAN_FD_TYPE: support for using non-ISO CAN FD.

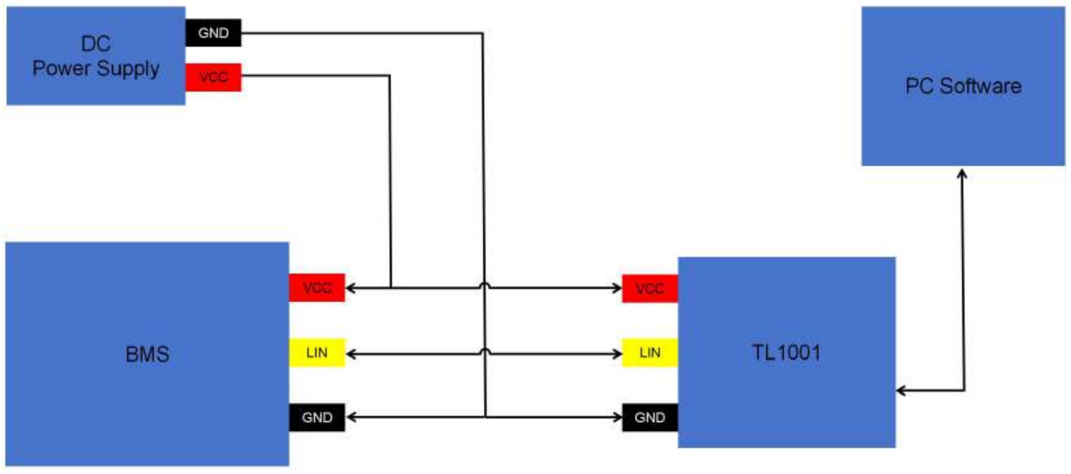
All other parameter functions will return as unsupported.

2. TOSUN Hardware

2.1. LIN Bus Devices

2.1.1. TSLIN Wiring Diagram

2.1.1.1. External Power Supply (Open-Drain Design)



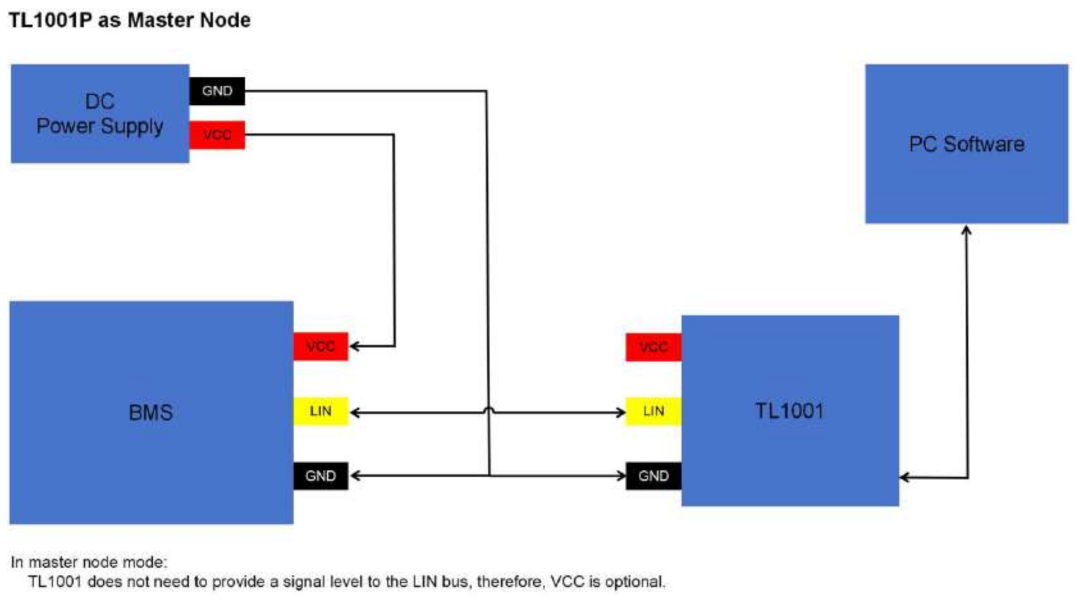
In master node mode:
TL1001 is required to provide power supply for the LIN bus. From a rigorous perspective, in order to maintain the LIN bus level signal consistent with the device under test, the power supply of TL1001 adopts an open-drain design, with VCC using the same power source as the device under test to ensure the LIN bus level signal consistentency.

Devices such as TL1001, TC1016, and TC1012 use external power supply for their level signals, which have the following advantages and disadvantages:

➢ Advantages: because an external power supply is used, the LIN signals can utilize a wider range of signal levels. For example, the LIN signal level for passenger cars is 12V, and for commercial vehicles, it is 24V. By connecting different voltage levels to VCC, LIN nodes with different operating voltages can be supported.

➢ Disadvantages: 1. An external power supply is required, which can be inconvenient for use in actual vehicles. 2. The power is supplied externally, and if the same external power source is also used to supply loads such as electric motors, the operation of these loads can cause significant voltage interference, thereby affecting the quality of the LIN bus signals.

2.1.1.2.Internal Power Supply， P-Series

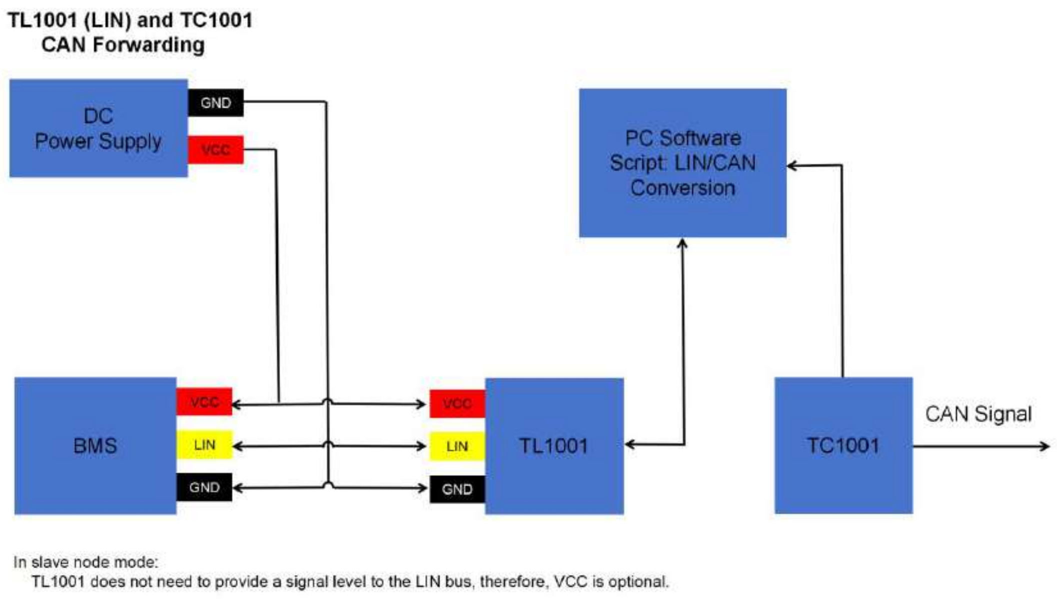


Devices such as TL1001P, TC1016P, and TC1012P use internal power supply to provide a fixed 12V signal level, and compared to devices with external power supply, they have the following advantages and disadvantages:

- Advantages: 1. With an internal power supply, it can operate by simply connecting to a computer via USB, which is very convenient for use in actual vehicles and other settings. 2. The voltage is generated internally and is not affected by the working voltage of external test ECUs.

- Disadvantages: because it uses an internally fixed 12V power supply, if there is a situation with commercial vehicle 24V working voltage and signal, this kind of LIN devices cannot be used.

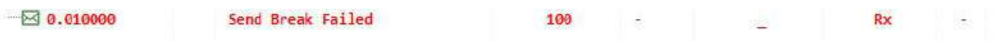
2.1.1.3.Network Node



2.2. Common Errors

2.2.1. Send Break Failed

In certain situations, the Trace interface indicates that a "Send Break Failed" has occurred, as shown below:



```
0.010000    Send Break Failed    100    -    _    Rx    -
```

This is caused by insufficient power supply from the USB device, which affects the normal operation of the LIN device's signal isolators, transceivers, etc. The solution is to check whether the USB is directly connected to the motherboard. If connected through a USB hub, it is easy to cause a situation of insufficient power supply.