TOSUN-TX1000
User Manual

tosunai.com

Product Features & Interface Overview

Product Name	Channel
TX1000	CAN FD * 2 Differential Cable * 1 POD Board * 1

Copyright Information

Shanghai TOSUN Technology Ltd.

No. 9 Building, 1288 Jiasong North Road, Jiading District, Shanghai (Headquarters)

Buildings 14-17, Lane 4849 Cao'an Highway (Shanghai Research Institute)

In an effort to provide users with the best possible service, Shanghai TOSUN Technology Ltd. (hereinafter referred to as "TOSUN Technology") has made every attempt to present accurate and detailed product information as possible in this manual. However, due to the time-sensitive nature of the content, TOSUN Technology cannot guarantee the timeliness and applicability of the information at all times.

The information and data contained in this manual are subject to change without prior notice. For the latest updates, please visit the official website of TOSUN Technology or contact our support team directly. We appreciate your understanding and continued support!

No part of this manual may be reproduced in any form or by any means without prior written permission from TOSUN Technology.

@ Copyright 2024-2025, Shanghai TOSUN Technology Ltd. All rights reserved.

Contents

1. Introduction

The TX1000 is a high-speed calibration device that connects to a PC via an RJ45 Ethernet interface. Featuring a driverless design for Windows systems, it offers excellent system compatibility. When used with the powerful TSMaster software, TX1000 supports connections to ARM and AURIX series MCUs. The supported debugging protocol is DAP (JTAG/SWD are not currently supported and will be added in future releases).

TX1000 supports operations such as modifying the ECU program counter (PC) for jump control, program halt, and resume. Through the debug interface, it enables high-speed memory access to ARM/AURIX-based ECUs without affecting the normal execution of ECU code. In addition, TX1000 can also be used for ECU calibration based on the XCP protocol.

Included resources:

- CAN FD monitoring software TSMaster
- Cross-platform secondary development library (with a dedicated programming manual)

1.1. Technical Specifications

➢ Device Specifications

Parameter	Description
PC Interface	USB 2.0, RJ45 Ethernet
Timestamp Precision	Microsecond-level high-precision timestamps
Driver	Cross-platform, driver-free design
Connector	Standard D-Sub, 9-pin
License	Supports all TSMaster paid licenses
Relay Type	Magnetic latching relay
Power Supply	USB-powered, external DC power (9-36 V)
Power Consumption	4 W
ESD Protection	±4 kV contact discharge, ±8 kV air charge
Enclosure Material	Metal

Dimensions	Approx. 178 * 113* 38 mm
Weight	Approx. 420 g (without package)/775 g (with package)
Operating Temperature	-40℃ to +80℃
Operating Humidity	10% ~ 90% RH (no condensing)

➢ **CAN Specifications**

Parameter	Description
Connection Standard	High-speed CAN (ISO 11898-2 compliant)
Supported Protocols	Full support for CAN and CAN FD (ISO 11898-1 compliant)
CAN Baud Rate	125 kbps ~ 1 Mbps
CAN Frame Data Length	Up to 8 bytes
CAN FD Baud Rate	125 kbps ~ 8 Mbps
CAN FD Frame Data Length	Up to 64 bytes; supports BRS frames
Max Frame Rate	Transmit: 20,000 frames/s; Receive: 20,000 frames/s (single channel, 1Mbps, remote frame, 0 data bytes)

1.2. Electrical Specifications

➢ **Power Characteristics**

Parameter	Condition	Min	Typ.	Max	Unit
Operating Voltage	USB power supply	4.8	5.0	--	V
Operating Current	USB power supply	--	0.6	--	A
Power Consumption	USB power supply	--	3	--	W

➢ CAN Interface Characteristics

Parameter	Condition	Min	Typ.	Max	Unit
Bus Pin Tolerance Voltage	CAN_H, CAN_L to GND	-58	--	58	V
Isolation Voltage	Leakage ＜ 1mA	2500	--	--	VDC

➢ EMC Performance

Test Item	Standard	Condition	Level	Unit
ESD	IEC 61000-4-2	Contact discharge	±4	kV
		Air discharge	±8	kV
EFT/Burst	IEC 61000-4-4	Electrical fast transient	±2	kV

➢ Mechanical Dimensions

Unit: mm

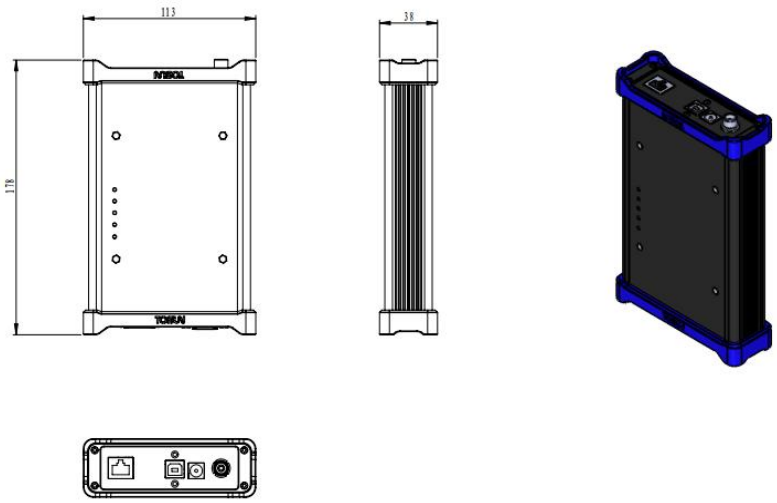


Figure 1-1 Mechanical Dimensions

1.3. Pin Definition

Figure 1-2 Hardware Interface

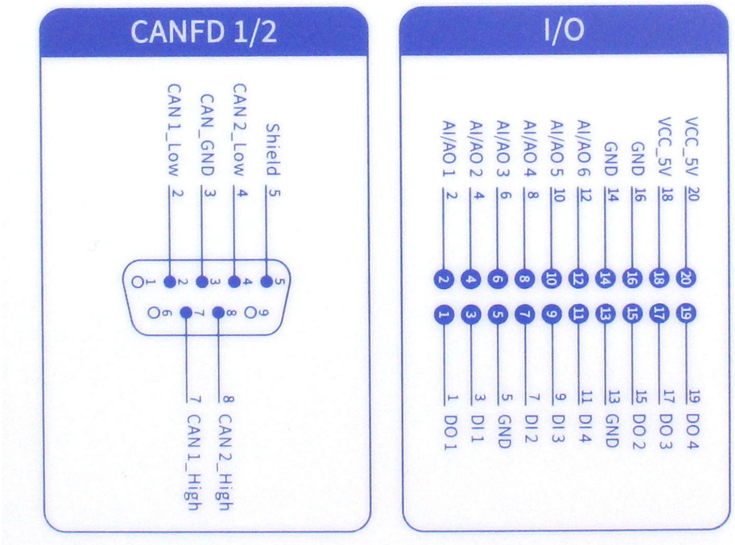


Figure 1-3 Pin Assignment

1.4. LED Indicators

Figure 1-4 Front Panel Layout

➢ LED Definitions

Indicator	Description
Power	Status of power indicator
MCD	Status of high-speed calibration indicator
Link	Hardware connection indicator
CAN FD 1	Status of CAN FD channel 1
CAN FD 2	Status of CAN FD channel 2

➢ LED Color Description

Indicator	Color	Description
CAN FD	Green	Normal frame transmission/reception

	Red	Frame transmission/reception error — configuration, protocol, or wiring fault
Link	Green	Device connected successfully

1.5. System Requirements

➢ PC Requirements

- Operating System: Windows or Linux
- One available USB port (USB 2.0 or higher), or a powered USB hub

➢ Driver Installation

- The TX1000 features a driver-free design, ensuring outstanding system compatibility. It can be used directly on Windows or Linux without manual driver installation.

➢ Downloads

- TSMaster software
- PDF user manual
- Programming library (for secondary development)

> The download link is available on the official website of Shanghai TOSUN Technology Ltd.: https://www.tosunai.com/

1.6. Packing List

Item	Qty.	Illustration	Standard/Optional

TX1000 Main Device	**1**		**Standard**
USB Cable	**1**		**Standard**
DB9 Female-Dual Male Signal Cable (CAN)	**2**		**Standard**
Category 6 Gigabit Ethernet Cable	**1**		**Standard**
Differential Cable	**1**		**Standard**
POD Board	**1**		**Standard**

12V 2A Adapter	**1**		**Standard**
CAN Surge Protector	**--**		**Optional**

For details on the paid accessory “CAN Surge Protector”, please refer to the appendix.

2. Application Example in Windows

2.1. Hardware Connection

2.1.1. CAN

Use the included DB9 Female–Dual Male Signal Cable (CAN) to access two independent channels via two D-Sub 9-pin connectors.

The following diagram shows the pin mapping of the “DB9 Female–Dual Male Signal Cable (CAN)” harness:

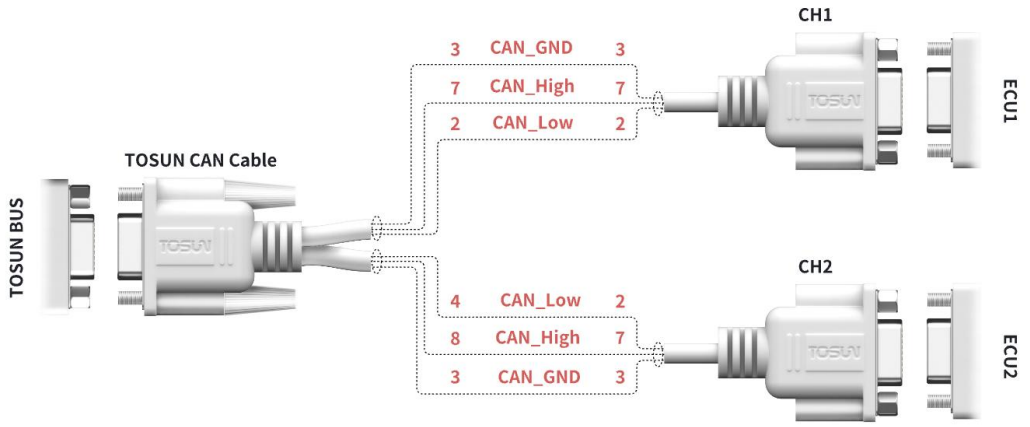


Figure 2-1 DB9 Female-Dual Male Signal Cable (CAN)

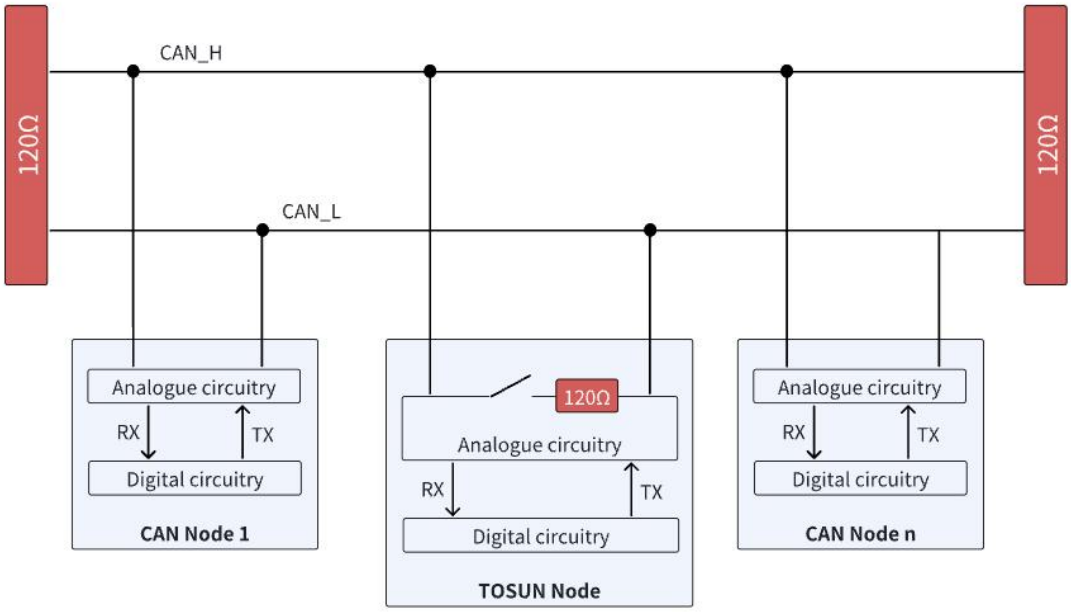


Figure 2-2 Connecting to the CAN Bus

2.1.2. System Connection

The TX1000 device is connected to the PC via Ethernet (RJ45). The TX1000 is connected to the POD board through a differential cable, while the ECU is connected to the POD board using JTAG / SWD / DAP debugging cables.

On the PC side, when used together with the powerful TSMaster software, the TX1000 device can be controlled to perform XCP measurement, memory read operations, and more.

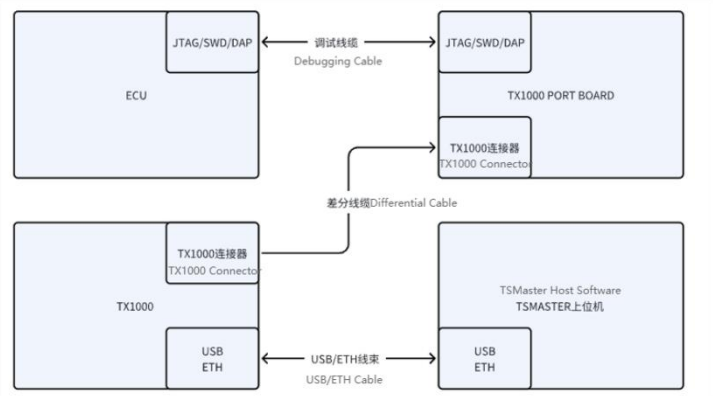


Figure 2-3 System Connection Diagram

Figure 2-4 Physical System Connection

2.2. TX1000 XCP Function Usage Example

The TX1000 XCP functionality supports two operating modes: XCP ON ECU and XCP ON TX1000.

- **XCP ON ECU mode**

In this mode, TOSUN's XCP protocol stack code is embedded into the customer's ECU and compiled together. Protocol stack parameters, events, and related configurations are set within the ECU.
This mode is relatively stable and can effectively guarantee data causality.

- **XCP ON TX1000 mode**

In this mode, the XCP protocol stack runs internally on the TX1000 device. The main advantage is that no code needs to be embedded into the customer's ECU; XCP calibration and measurement can be completed entirely within the TX1000.
To enable this mode, the xcp_config.ini file must be edited according to the A2L file, or the configuration can be directly programmed into the TX1000.
Since the XCP code runs inside the TX1000, line delays and memory access latency may, in some cases, result in discontinuous data or unstable periods.

2.2.1. TX1000 STATION APP

The TX1000 STATION APP is a STATION example application implemented by calling the TOSUN TSDEV device DLL. It converts TX1000 XCP messages into XCP ON ETH messages and uses TCP as the XCP transport layer.
After the STATION program is launched, the ECU can be directly connected using XCP ON ETH in TSMaster.

```
SI_Error err = SI_FAIL;
err = ini.LoadFile("xcp_config.ini");
if (err != SI_OK) {
    printf("Failed to load config file! Error: %d, use default config value\n", err);

    for (int i = 0; i < 6; ++i) {
        event_cycle_ms[i] = 0;
    }
    xcp_pars.addr_bits = 0;
    xcp_pars.checksum_type = 0;
    xcp_pars.daq_timestamp_unit = 0;
    xcp_pars.timestamp_size = 4;
    xcp_pars.enable_cal_pag = 1;
    xcp_pars.enable_daq = 1;
    xcp_pars.enable_get_comm_mode_info = 1;
    xcp_pars.enable_pgm = 1;
    xcp_pars.enable_slave_block_mode = 1;
    xcp_pars.enable_stim = 1;
    xcp_pars.id_type = 1;
    xcp_pars.max_cto_len = 255;
    xcp_pars.max_dto_len = 0xfff;
    xcp_pars.max_daq_cnt = 6;
    xcp_pars.max_event = 6;
    xcp_pars.max_odt_cnt = 20;
    xcp_pars.max_unlock_attempts = 3;
    xcp_pars.timestamp_size = 4;
}
else {
    for (int i = 0; i < xcp_pars.max_event; i++) {
```

Figure 2-5 XCP ON TX1000 Configuration Parameter Demo

First, the initialization stage requires loading the .ini file. If loading fails, default settings will be used. Note that these defaults correspond to the configuration provided in the Demo project. If the .ini file is not available, manual configuration is required.

```
uint32_t addr = (uint32_t)0x70004a64;
if (tsdev_err_ok == tsdev_api_tx1000_init(handle_p, 200000)) {
    printf("tx1000 init success\r\n");
    if (tsdev_err_ok == tsdev_api_tx1000_target_set_sync(handle_p, 0,
        tsdev_TC39x,50000000, tsdev_DapProtocol, tsdev_DapStandardMode)
        )
    {
        puts("target set ok");
    }
    if (tsdev_err_ok == tsdev_api_tx1000_connect_sync(handle_p, 0))
    {
        puts("DebugPort Connect Finish");
    }
    if (1)
    {
        if (tsdev_err_ok == tsdev_api_tx1000_set_xcp_host_mode(handle_p, event_cycle_ms, &xcp_pars))
        {
            puts("DebugPort Connect Finish");
        }
    }
    else
    {
        uint32_t version;
        if (tsdev_err_ok == tsdev_api_xcp_set_mcu_ring_addr_sync(handle_p, 0x7000412c, &version))
        {
            printf("DebugPort Connect Finish version %x\n", version);
        }
    }
    if (tsdev_err_ok == tsdev_api_xcp_rx_event_register(handle_p, tsdev_xcp_rx_cmd_data_func))
    {
        puts("rx reg Finish");
    }
```

Figure 2-6 Initializing the TX1000 Device as XCP ON ECU / XCP ON TX1000

After the TSdev device scan is successful, configure the target as TC397, set the clock frequency

to 50 MHz, and select the DAP standard mode. Then, use the connect API to connect to the TC397.

For XCP ON TX1000 mode, call the function
`tsdev_api_tx1000_set_xcp_host_mode`
to configure the internal XCP protocol stack of the TX1000.

For XCP ON ECU mode, call the function
`tsdev_api_xcp_set_mcu_ring_addr_sync`
to read the version number of the TOSUN XCP protocol stack inside the ECU. Once a valid version number is obtained, calibration and measurement can begin.

Next, register the receive callback function, which has already been implemented.

```
std::cout << "客户端已连接: " << inet_ntoa(clientAddr.sin_addr) << ":" << ntohs(clientAddr.sin_port) << std::endl;

struct _XCP_TRANS_LAYER_t data;
uint8_t* buffer = (uint8_t*)malloc(65536);
if(buffer != NULL)
{
    while (1) {
        iResult = sread(clientSocket, &data, 4);
        if (iResult != 1) {
            break;
        }
        iResult = sread(clientSocket, buffer, data.len);
        if (iResult != 1) {
            break;
        }
        else {
            {
                if (tsdev_err_ok != tsdev_api_xcp_tx_cmd_data(handle_p, 0, buffer, data.len))
                {
                    printf("tx xcp cmd failed\n");
                }
                else {
                    printf("tx xcp cmd %x successful\n", buffer[0]);
                }
            }
        }
    }
}
```

Figure 2-7 XCP ON ETH Message Parsing and Forwarding

Finally, once the TCP connection is successfully established, XCP commands can be transparently forwarded to the TX1000.

2.2.2. X1000 STATION GUI APP

The TX1000 STATION GUI APP is a sample application that exposes certain configuration items of the TX1000 STATION APP through a graphical interface using an .ini file.
XCP configuration parameters can be set via the .ini file to ensure consistency with the A2L file (effective only in XCP ON TX1000 mode).

The TX1000 host application primarily functions as a Socket communication server, monitoring the behavior of the TSMaster client, while also acting as the host application for the TX1000 by

issuing commands and configuration parameters.

The software automatically scans network adapters. After the user selects the appropriate network adapter and connects to the device, the server can be started. Once TSMaster is launched on the client side, communication will be established automatically.

2.2.2.1. Device Connection

Enter the device connection interface, select the corresponding network adapter, and click “Connect Device” to connect to the TX1000 device.

Ensure that the host PC and the TX1000 are on the same subnet. For example, if the TX1000 IP address is 192.168.1.10 with a subnet mask of 255.255.255.0, the PC IP address should be 192.168.1.x (x ranging from 2 to 255, excluding 10).

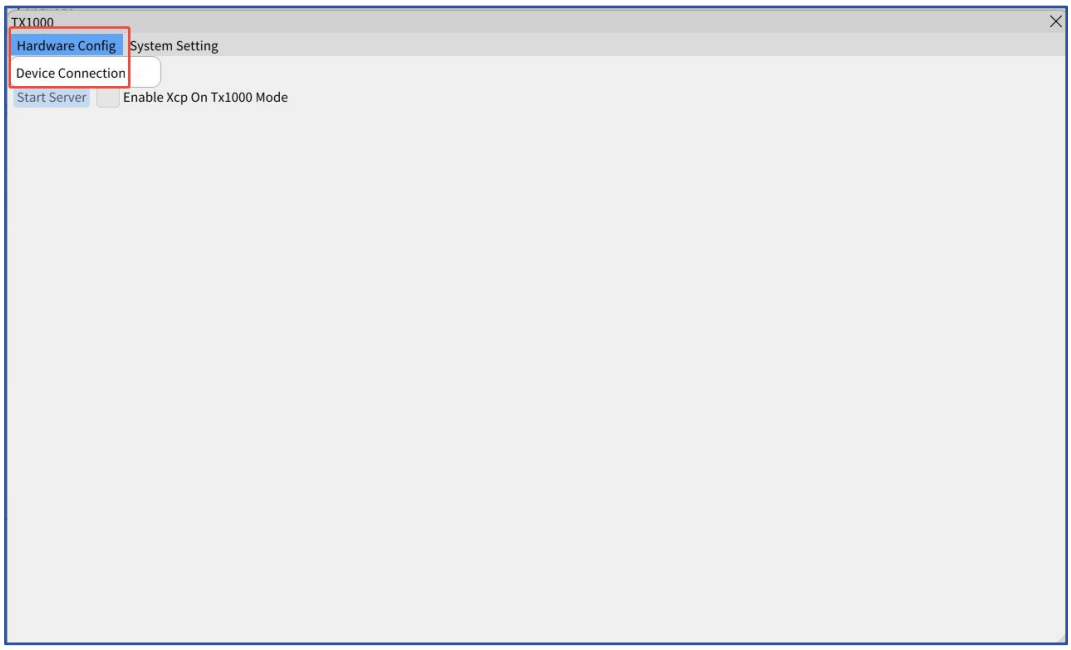


Figure 2-8 TX1000 Main Interface

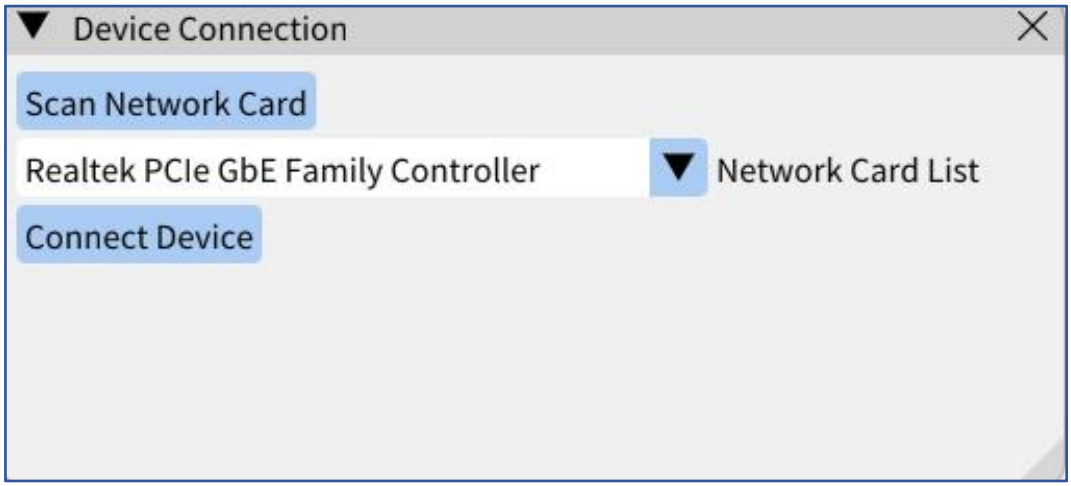


Figure 2-9 TX1000 Device Connection Interface

On the main interface, the device connection status can be viewed. Click “Disconnect Device” to disconnect from the device.

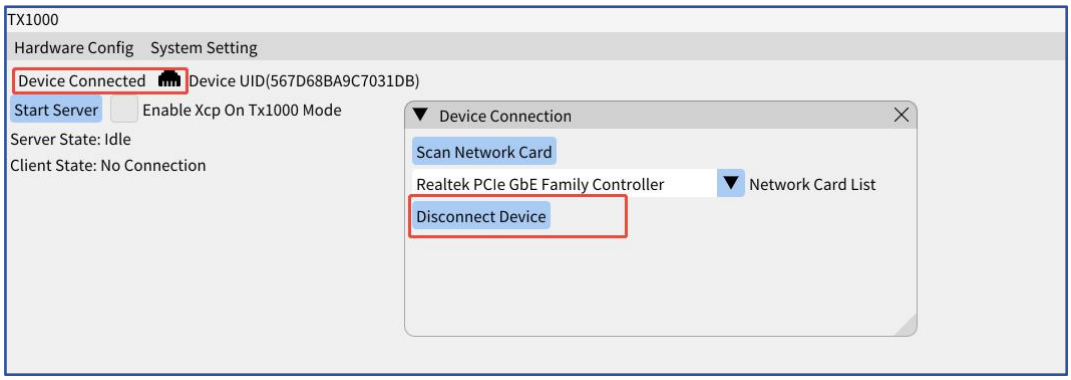


Figure 2-10 TX1000 Device Connected

2.2.2.2. Parameter Configuration

Enter the parameter configuration interface to set various parameters. After the server is started, these parameters will be sent to the TX1000 device.

Note 1: Once the server has been started, modifying parameters during operation will not affect the already running configuration.

Note 2: When the software is launched for the first time, it automatically loads configuration parameters from the `xcp_config.ini` file. If the file does not exist, system default parameters are used.

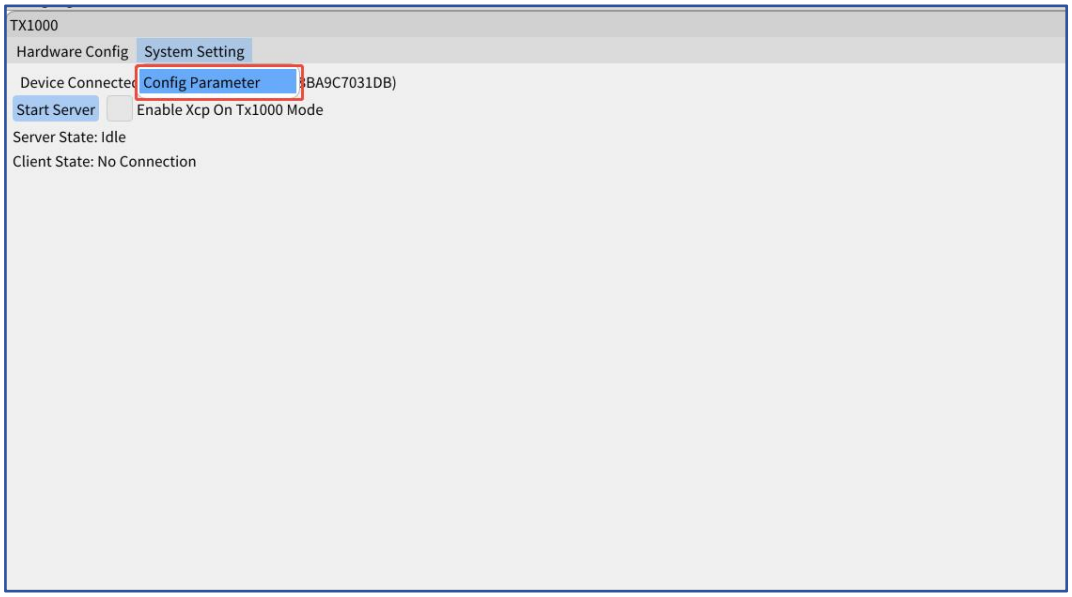


Figure 2-11 Parameter Configuration Interface

This interface provides three buttons with the following functions:

- Load configuration file parameters: loads parameters from the `xcp_config.ini` file into the interface.
- Save configuration parameters to file: saves the parameters currently shown in the interface to the xcp_config.ini file.
- Reset to default parameters: resets all parameters in the interface to system default values, without automatically saving them to the file.

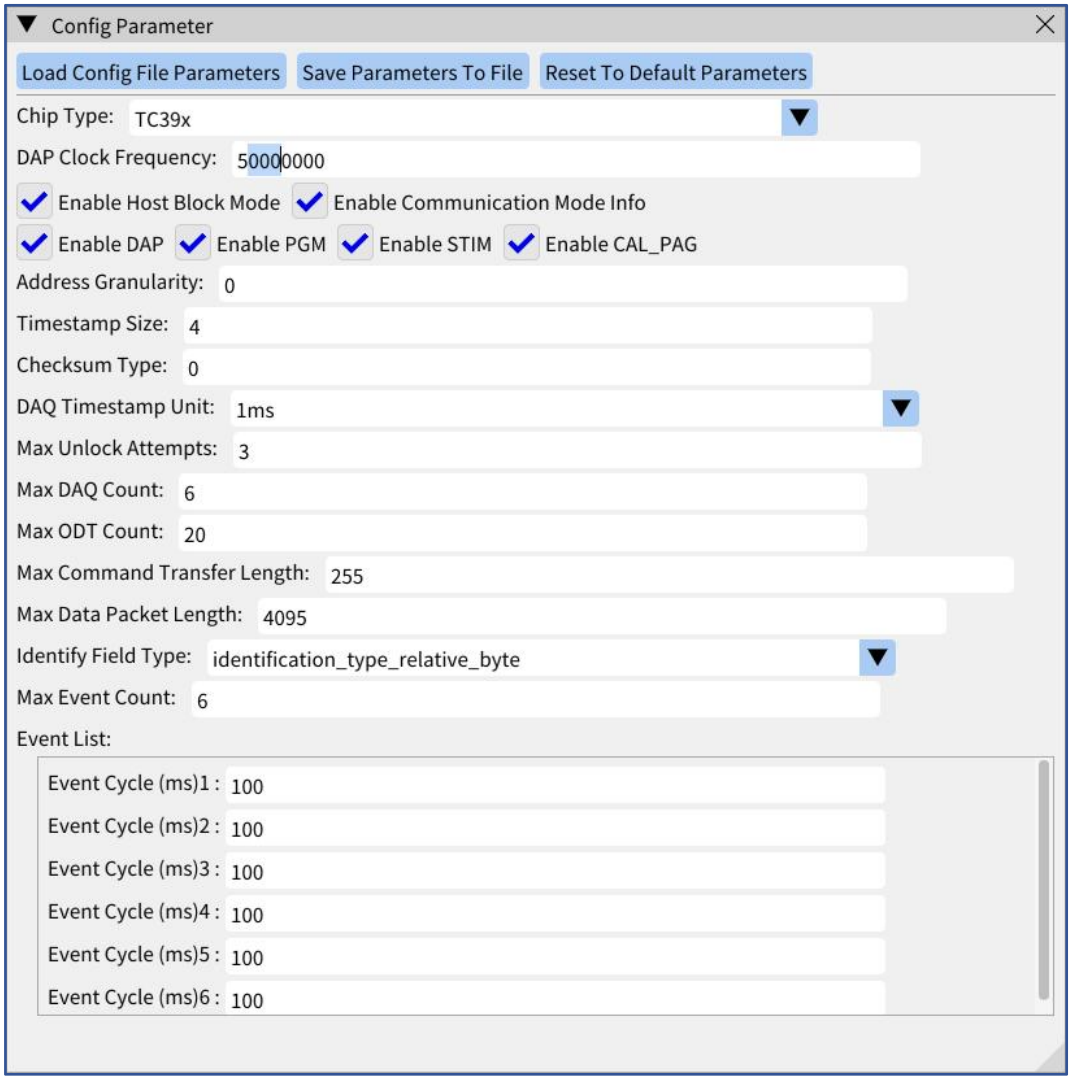


Figure 2-12 Configure XCP ON T1000 Parameters

2.2.2.3. Start Server

Click “Start Server” to download the configuration parameters to the TX1000 device and begin listening for client connections. The main interface will display the connection status of both the server and the client.

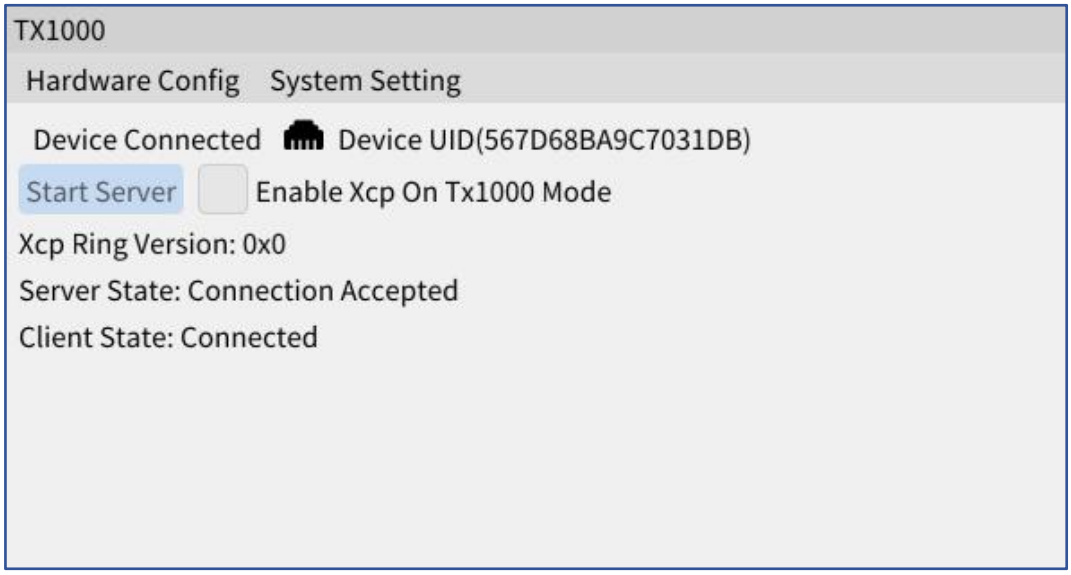


Figure 2-13 Start XCP Server

2.2.3. Using TX1000 with TSMaster

1. Before using TSMaster, the STATION APP or STATION GUI APP must be started first. After startup, open the TX1000 Demo project in TSMaster.

2. To test XCP in TSMaster:

- Click “Applications” in TSMaster, then open the Calibration Manager.

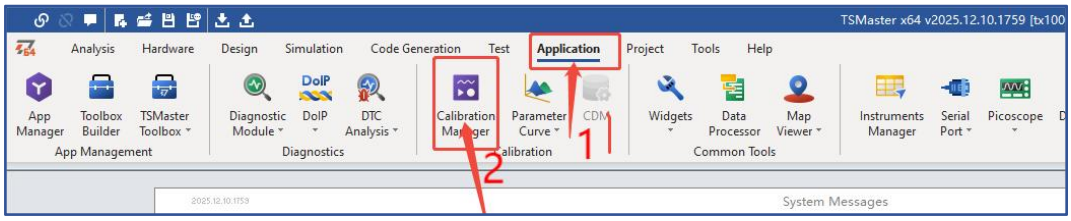


Figure 2-14 TSMaster Calibration Manager

- Import the A2L file and configure the transport layer as `ETH: ON TCP/IP STACK`.

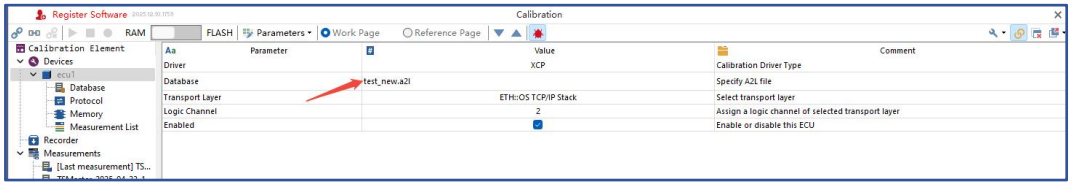


Figure 2-15 Configure XCP ON ETH

- Configure optional commands. It is recommended to uncheck commands that are not used (unchecked commands are pending adaptation and can be prioritized if necessary).

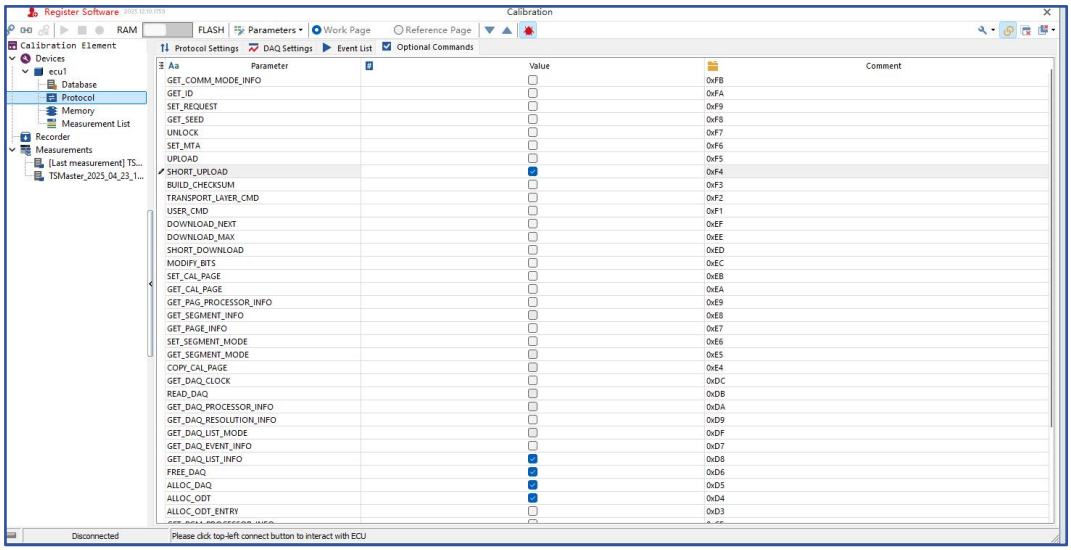


Figure 2-16 Configure optional commands

- Select the signals to be measured from the database and add them to the measurement list. Signals can also be added to graphs for waveform observation.

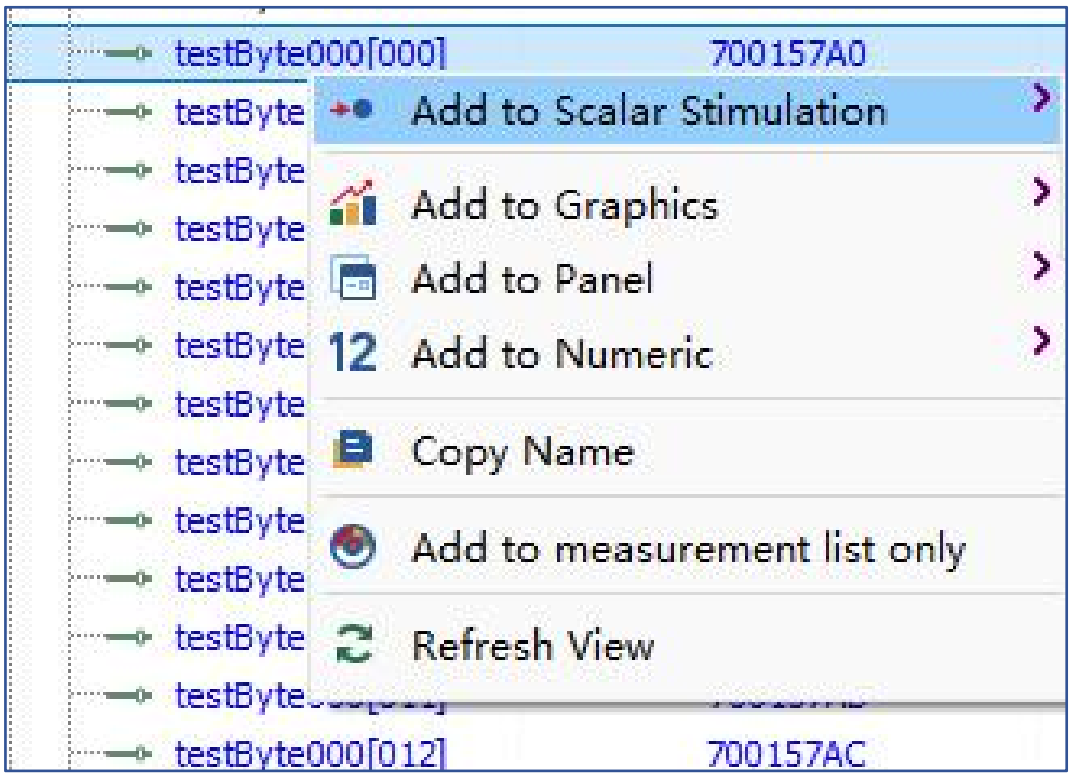


Figure 2-17 Add Signals to Measurement List

- In the measurement list, configure the selected signals as DAQ mode.

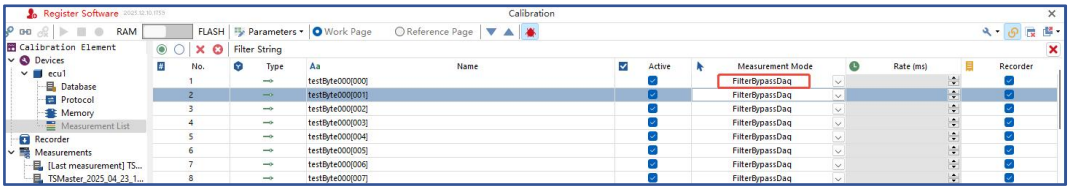


Figure 2-18 Configure Signals as DAQ Mode

- Start TSMaster to automatically execute XCP measurement. Signals added to graphs will display real-time changes.

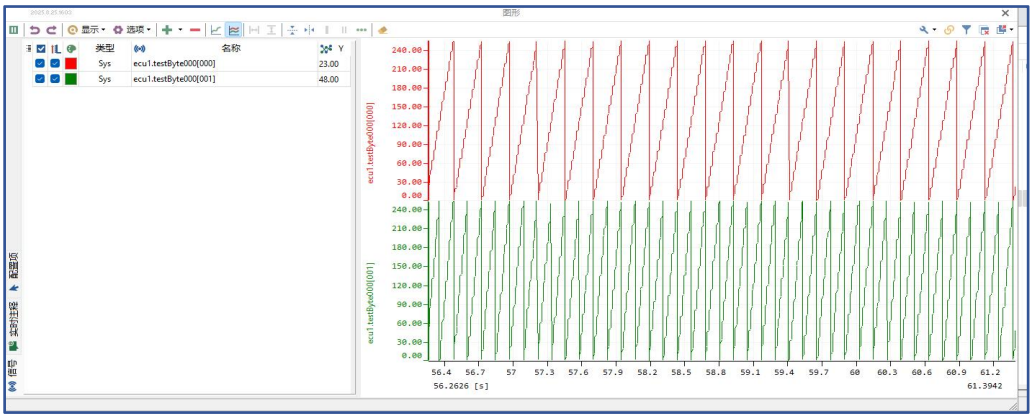


Figure 2-19 Graphical Display of DAQ Signal Changes

2.3. TX1000 XCP ON ECU Mode Code Adaptation

To help users quickly get started with the TX1000 device, detailed usage instructions are provided below:

- First, add the `01_embedded_source` directory from the provided code package to the ECU compilation path.

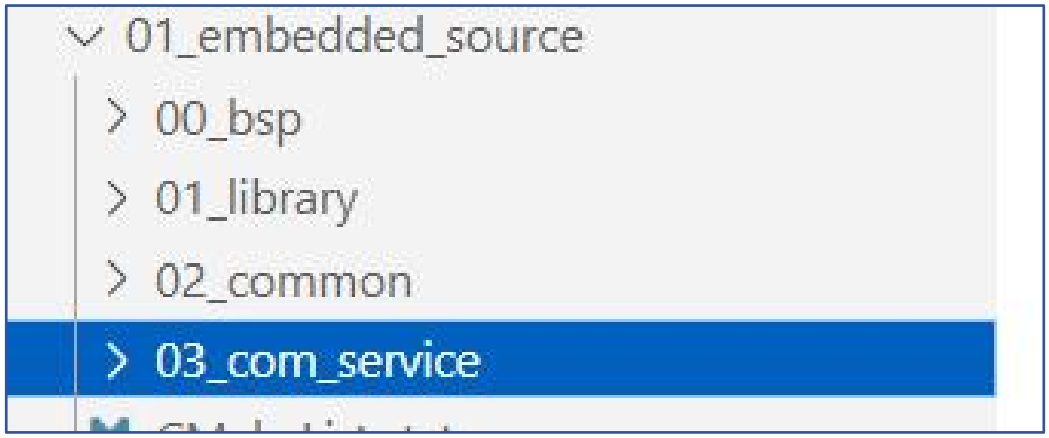


Figure 2-20 XCP Code Directory

- Then, refer to the code in `xcp_init.c` and modify the initialization configuration:

  - Modify the parameters of the `xcp_ring` structure, including the XCP version and XCP Tx/Rx buffer parameters.
  - Configure parameters used by the XCP protocol stack, mainly including memory usage (allocate two contiguous memory regions: one for the protocol stack and one for DAQ), memory configuration, Seed and Key algorithm parameters.
  - The `updata_odt_entry` function is provided for viewing data during upload (optional).

```
uint8_t tx_ring_buff[4096];
uint8_t rx_ring_buff[4096];
struct tosun_xcp_ring_t xcp_ring = {
        .info ={
                .xcp_version = 0x20252025,
                .xcp_tx_buff_len = sizeof(tx_ring_buff),
                .xcp_rx_buff_len = sizeof(rx_ring_buff),
                .xcp_rx_ring_buff_addr = (uint32_t)rx_ring_buff,
                .xcp_tx_ring_buff_addr = (uint32_t)tx_ring_buff
        },
        .var ={
         0,0,0,0
        }
};
struct tosun_xcp_init_pars_t xcp_pars={
    .tosun_xcp_ring_addr = (uint32_t)&xcp_ring,
    .event_pars = event_pars,
    .daq_max_use_mem = sizeof(daq_mem_pool_0),
    .daq_use_mem = daq_mem_pool_0,
    .ag = xcp_if_addr_granularity_8,
    .get_daq_timestamp = NULL,
    .max_event = 6,
    .xcp_use_mem = mem_pool_0,
    .xcp_max_use_mem = sizeof(mem_pool_0),
    .tosun_xcp_mem_read = tosun_xcp_mem_read,
    .tosun_xcp_mem_write = tosun_xcp_mem_write,
    .get_key = get_key,
    .get_seed = get_seed,
    .max_odt_cnt = 20,
    .max_cto_len = 0xff,
    .max_daq_cnt = 8,
    .max_dto_len = 0xff,
    .update_odt_entry = update_odt_entry
};
```

Figure 2-21 XCP Initialization Parameter Demo

- Pass the initialized parameters to the `tosun_xcp_setup` function in `tosun_xcp_api.h`. If the function returns a non-null value, initialization is considered complete. The returned pointer is used as the handler for `tosun_xcp_event_handle`.

```
};
void* tosun_xcp_setup(struct tosun_xcp_init_pars_t* init_pars_p);
void tosun_xcp_handle();
void tosun_xcp_event_handle(void* xcp_p, uint16_t event_idx);
#ifdef __cplusplus
```

Figure 2-22 XCP API

- After initialization, calibration and measurement can begin. According to application requirements, place the `tosun_xcp_handle()` function in a periodic task. The scheduling period can be adjusted based on the calibration and measurement signals. For XCP events, insert the `event_handle function` at the corresponding measurement points to implement DAQ functionality. A simplified pseudo-code example is shown below:

```
void*xcp_p = NULL;
struct tosun_xcp_init_pars_t init;
void task_10ms(){
    uint16_t event_idx=0;
    uint16_t daq_val=0;
    xcp_p = tosun_xcp_setup(&init);
    {
        while(1){
            delay_ms(10);
            tosun_xcp_handle();
            daq_val++;
            tosun_xcp_event_handle(xcp_p,event_idx);
        }
    }
}
```

Figure 2-23 Simplified Pseudo-code Example

3. Appendix

3.1. CAN Surge Protector

Figure 3-1 CAN Surge Protector

The CAN surge protector safeguards the CAN bus system from transient over-voltage or surge current.

It requires no external power, uses a DB9 interface, and is plug-and-play without affecting communication quality.

For devices without built-in surge protection, the optional TCA00011 surge protector can be used.

➢ CAN Surge Protector

Parameter	Value
Dimensions	Approx. 76 * 38 * 25 mm
Weight	Approx. 71 g
Surge Protection Level	±2 kV

3.2. Software Installation

The section describes the steps for installing the TSMaster software on a Windows PC.

➢ TSMaster Software Download

https://www.tosunai.com/downloads/

If the site is unavailable, contact your sales representative or visit the TOSUN official website.

You may also scan the QR code below to follow the official WeChat account and obtain download links.

Figure 3-2 TOSUN Official WeChat Account

➢ Software Installation

1. Double-click the TSMaster installer and select the installation language.

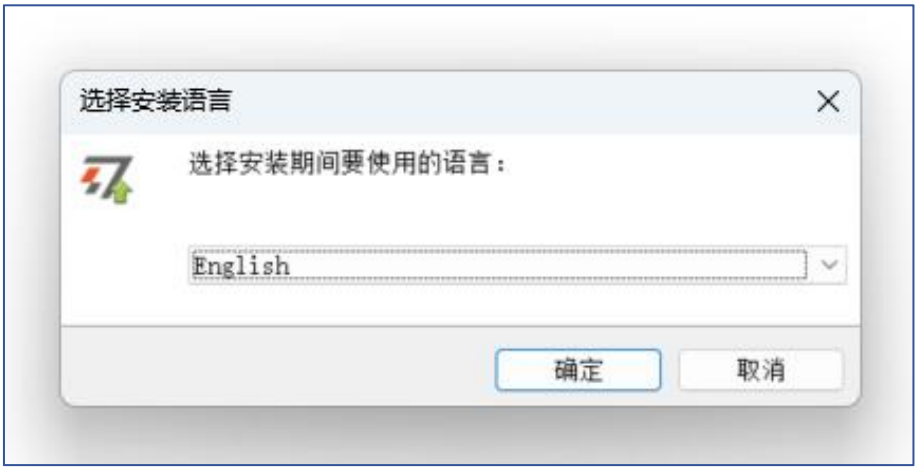


Figure 3-3 TSMaster Installation

2. Accept the license agreement and click “Next”.

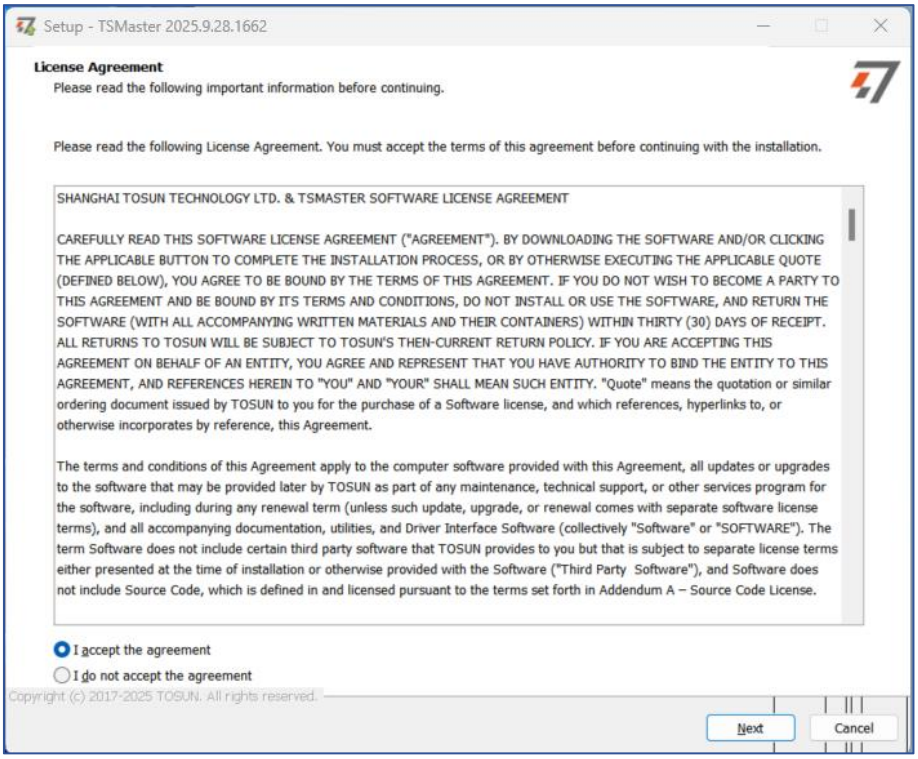


Figure 3-4 TSMaster Installation

3. Choose an installation directory and click “Next “.

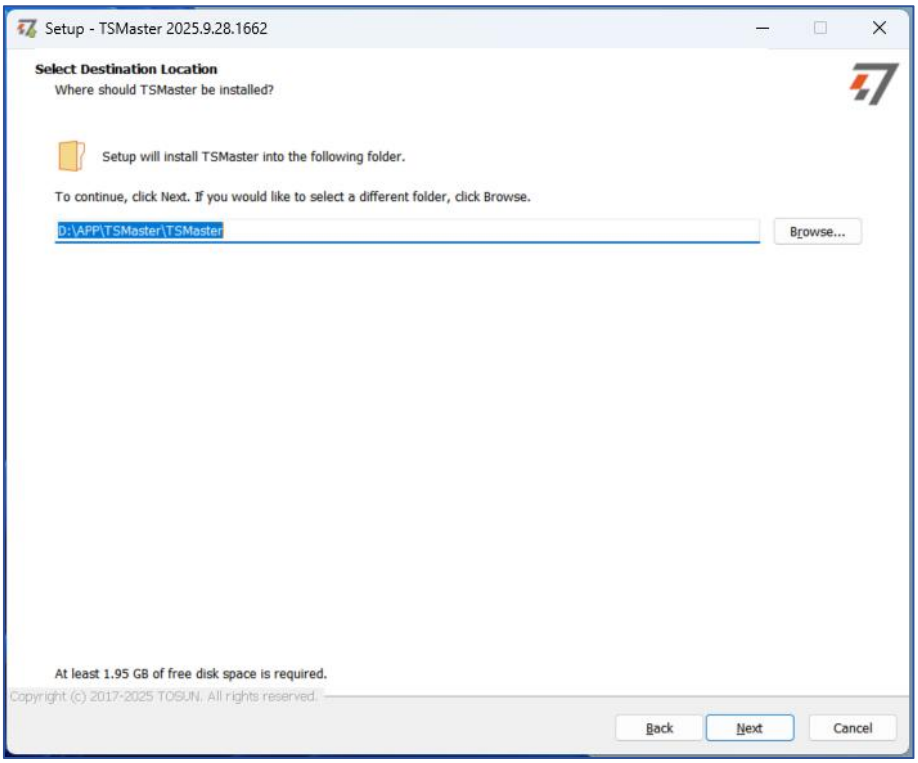


Figure 3-5 TSMaster Installation

4. Select additional tasks as needed and click “Next”.

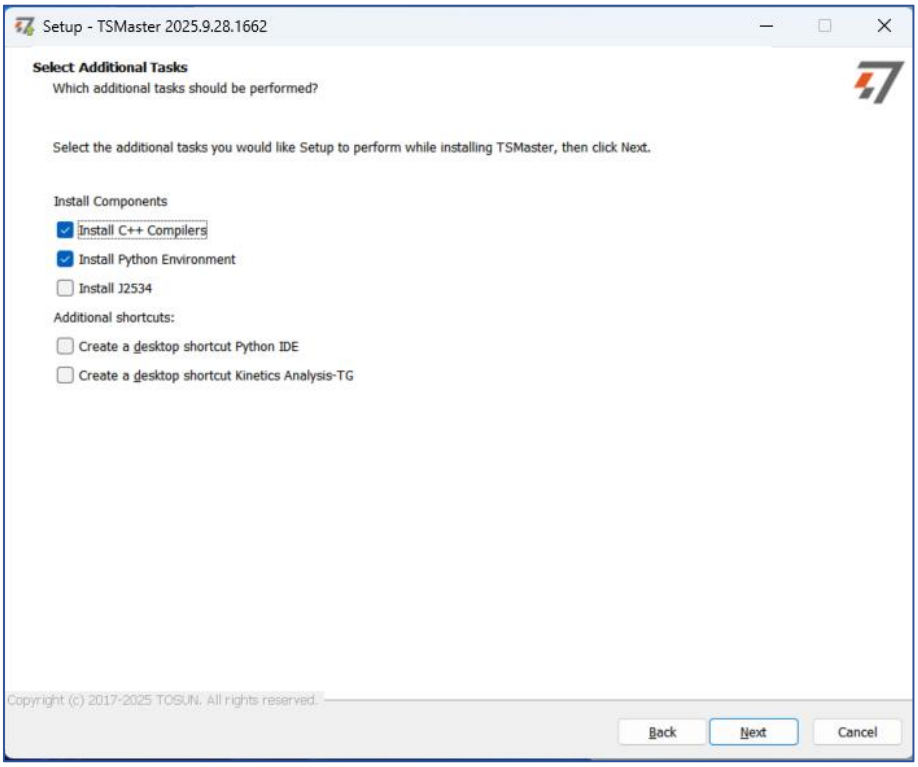


Figure 3-6 TSMaster Installation

5. Click “Install” and wait for completion.

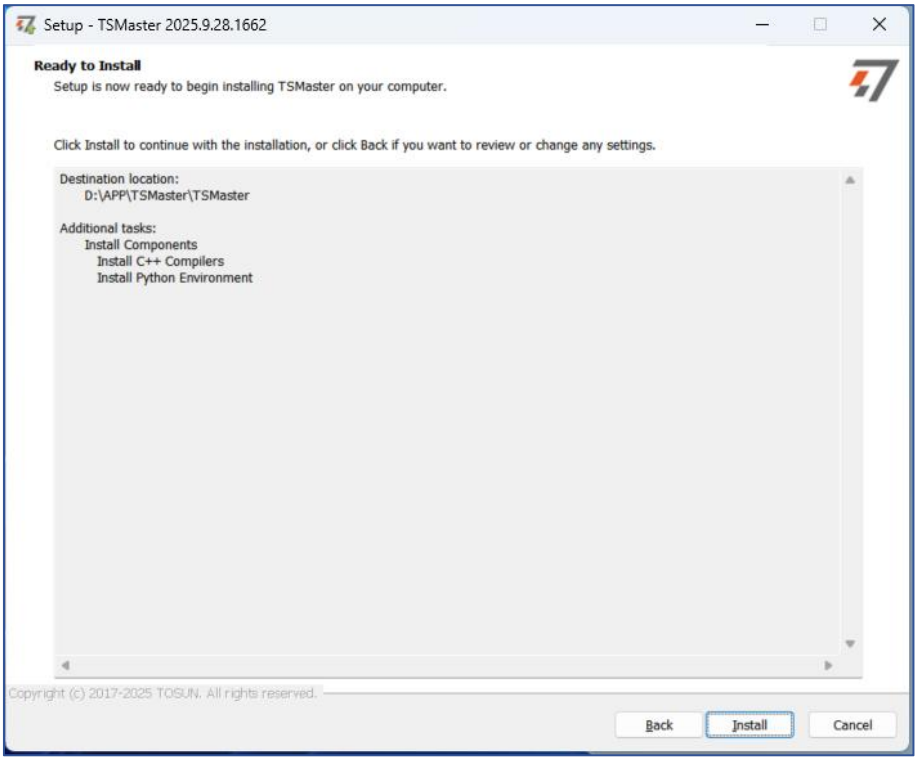


Figure 3-7 TSMaster Installation

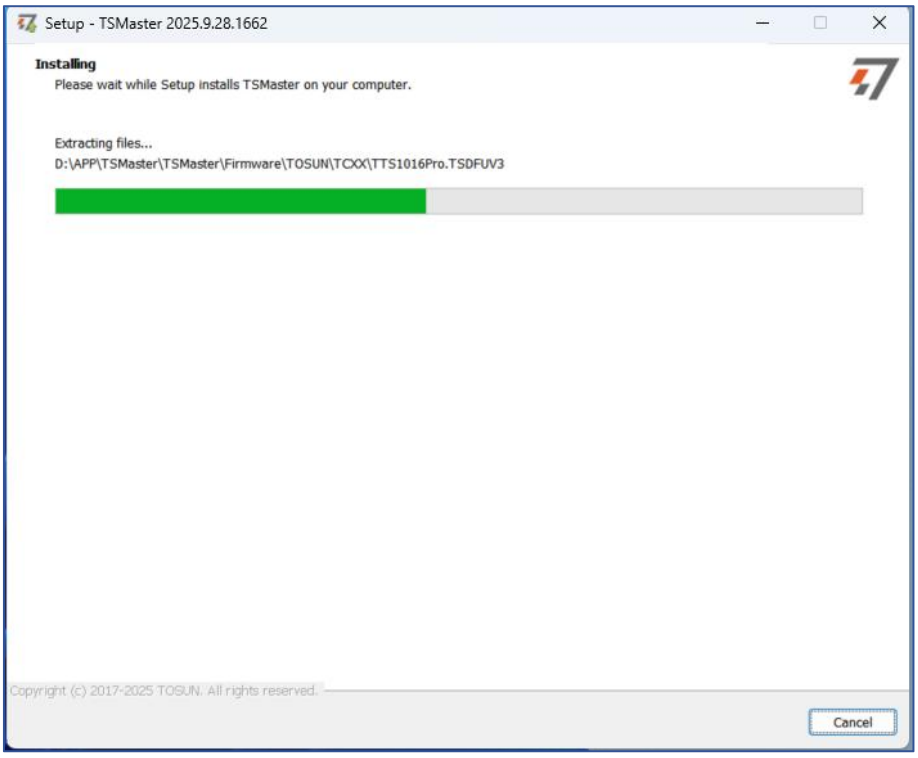


Figure 3-8 TSMaster Installation

6. Click “Finish” to complete installation.

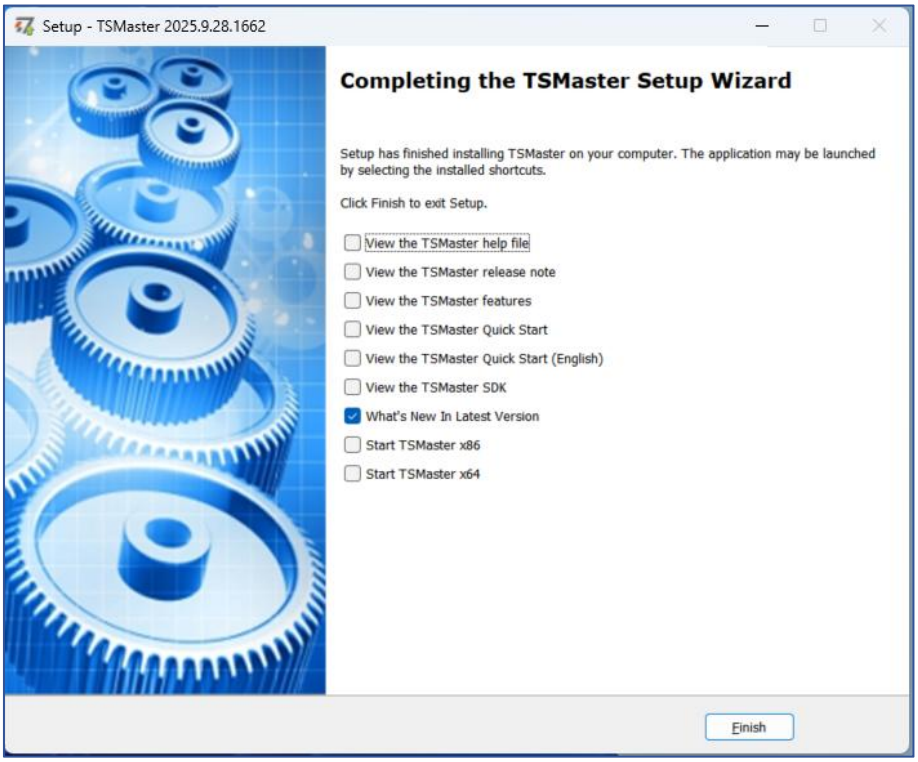


Figure 3-9 TSMaster Installation

4. Inspection and Maintenance

The TX1000 primarily contains semiconductor components, which typically have a long service life. However, adverse environmental conditions may accelerate aging and degrade performance. To ensure proper operation, regular inspections are recommended to maintain the required environmental conditions.

It is recommended to inspect the device at least once every 6 to 12 months. In harsher environments, inspections should be performed more frequently. Refer to the table below for inspection criteria and recommended actions. If issues persist, please contact Shanghai TOSUN Technology Ltd.

➢ Power Environment Inspection

Item	Check Content	Standard/ Range	Action/Measure
Power Supply	Check voltage fluctuation at power input	USB port:+5V DC Power port: +12V DC	Use a power meter or voltmeter at the input; ensure voltage fluctuation is within range
Ambient Conditions	Check ambient temperature (including internal temperature within enclosures)	-40℃ ~ +80℃	Use a thermometer to ensure temperature is within specified range
	Check the ambient humidity (including internal humidity within enclosures)	10% ~ 90% RH	Use a hygrometer to ensure humidity is within specified range

➢ Contamination & Protection Check

Item	Check Content	Standard/ Range	Action/Measure
Contamination	Check for accumulation of dust, powder, salt, and metal debris	None	Clean the device and prevent future contamination
	Check for exposure to water, oil, or chemicals	None	Clean and shield if necessary
Hazardous Gases	Check for corrosive or flammable gases	None	Use sensors or odor detection to verify

➢ Mechanical Stress & EMI Check

Item	Check Content	Standard/ Range	Action/Measure
Mechanical Stress	Check vibration and shock levels	Within specified limits	Install padding or vibration isolation measures if necessary
Electromagnetic Environment	Check for noise sources near the device	No significant noise sources	Isolate or shield the device from noise sources

➢ Installation & Wiring Check

Item	Check Content	Standard/ Range	Action/Measure
Wiring	Check crimped connectors in external wiring	Adequate clearance between connectors	Visually inspect and adjust as needed
	Check for damage to external wiring	No damage	Visually inspect and replace damaged cables if necessary

TOSUN

Engineer Everything !

Software

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

UDS diagnostics/ECU flashing/CCP/XCP calibration

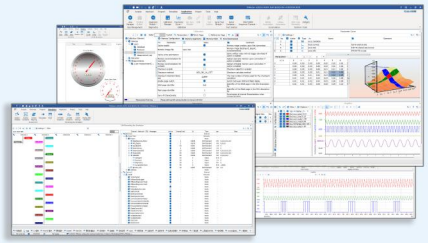
Embedded code generation/Application builder

Encrypted release/Logging and bus replay

Graphical programming/Residual bus simulation

C and Python scripting

Bus monitoring/Transmiting/Automated testing



TSMASTER

Hardware

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

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

Multi channel FlexRay/CAN FD to USB/PCIe device

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

Automotive Ethernet media conversion device (T1 to Tx)

Multi-channel CAN FD/Ethernet/LIN datalogger

TTS test systems

-CAN FD/CAN/FlexRay/LIN communication boards

-Relay and fault injection boards

-Resistors for sensor simulation

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

CAN CAN FD LIN FlexRay OPEN ALLIANCE

Solutions

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

About TOSUN

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

International Organization

ASAM CiA

Quality Assurance

ISO9001:2015

CE Certification

CE

Contact Us :

+86 21-5956 0506

sales@tosunai.com

website :

www.tosunai.com

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