TOSUN

GW2112 Product Manual

Version: V1.0 | English

tosunai.com

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 the principle of providing better services to users, Shanghai TOSUN Technology Ltd (hereinafter referred to as "TOSUN Technology") will present as much detailed and accurate product information as possible in this manual. However, due to the timeliness of the content in this manual, TOSUN Technology cannot fully guarantee the timeliness and applicability of this document at any time.

If there are any changes to the information and data in this manual, no separate notice will be given. To obtain the latest version of the information, please visit the official website of TOSUN Technology or contact the staff of TOSUN Technology. Thank you for your understanding and support!

Email: sales@tosunai.com

Why Need CAN (FD) Relay and Message Conversion?

The CAN network can be quite complex, especially in large vehicles or industrial systems. A relay can divide the network into multiple segments to reduce bus load and improve communication stability and reliability. CAN has different versions, such as Classic CAN and CAN FD (Flexible Data-rate). The message format and data transmission rates differ between Classic CAN and CAN FD. Therefore, when devices with different protocol versions exist in the network, message conversion is necessary to ensure compatibility and correct data transmission.

CAN FD offers higher data transmission rates and larger data payloads compared to Classic CAN. If Classic CAN and CAN FD devices need to be mixed on the same network, relays and message conversion can help facilitate data exchange between devices with different bandwidth requirements. In long-distance or high-interference environments, CAN signals may be affected. Relays can help amplify the signal, ensuring the integrity and reliability of data transmission. Through CAN relays, the CAN network can be extended over a larger range, supporting more devices and more complex system architectures. In modern systems, there may be devices from different brands and models, which might use various CAN standards or configurations. Relays and message conversion can help achieve seamless communication between these different devices.

In summary, CAN relays and message conversion are crucial for maintaining the stability, compatibility, and scalability of systems. They help resolve issues related to different devices, protocol versions, and network topology, enhancing the overall performance and reliability of the CAN network. The GW2112 offline gateway, launched by TOSUN, is a core product designed for CAN(FD) relaying and message conversion.

What Can the GW2112 Offline Gateway Do?

- Protocol Conversion

  The CAN/CAN FD gateway device can convert between different CAN protocols;
- Data Forwarding

  Forwarding and processing various signals on the CAN bus;
- Message Filtering

  Features ID filtering and conversion to reduce the load on the CAN bus and eliminate interference data;
- Rate and Data Length Adaptation

In networks where CAN and CAN FD coexist, the gateway can handle conflicts related to different rates and data lengths, such as forwarding 64-byte data from CAN FD to traditional CAN devices after packetizing it through a programmable CAN FD router.

- Relay Expansion

The gateway devices can increase load nodes and extend communication distances, achieving the function of network relay expansion.

- ...

Contents

1. About this User Manual

1.1 Warranty

This document is provided for reference only and does not constitute any form of guarantee or commitment from TOSUN. TOSUN Technology reserves the right to modify the content and data of the document without further notice. TOSUN Technology assumes no responsibility for the accuracy of the document or for any damages arising from the use of the document. TOSUN Technology greatly appreciates for pointing out errors or making suggestions for improvement, so that we can provide more efficient products in the future.

1.2 Copyright

2. GW2112

2.1 Overview

The GW2112 is a CAN (FD) offline gateway device launched by TOSUN. It can enhance the load capacity of the bus and extend communication distances, matching CAN (FD) networks with different baud rates, while also supports the conversion between CAN and CAN FD networks.

The GW2112 features two CAN (FD) channels, allowing it to forward CAN (FD) messages received on CAN1 (or CAN2) to the other CAN channel. It offers multiple conversion rules for users to choose from, which can be freely set using the accompanying software. The configured conversion rules can be permanently saved in the GW2112, ensuring they are not lost during power outages. Upon startup, the device will read the last stored conversion rules. Additionally, user-defined conversion rules can be saved as configuration files locally for easy import.

2.2 Features

- ✓ CAN channel DC 2500V isolation
- ✓ Adjustable CAN channel baud rate: 125Kbps—1Mbps;
- ✓ Adjustable CAN FD channel baud rate: 125Kbps—8Mbps;
- ✓ Built-in 120-ohm terminal resistor, with the resistance value configurable through software;
- ✓ CAN/CAN FD protocol conversion;
- ✓ CAN/CAN FD message filtering;
- ✓ CAN/CAN FD message data routing;
- ✓ Relay expansion;
- ✓ Supports for customized functions;
- ✓ Built-in configuration encryption with various encryption algorithms.

2.3 Technical Data

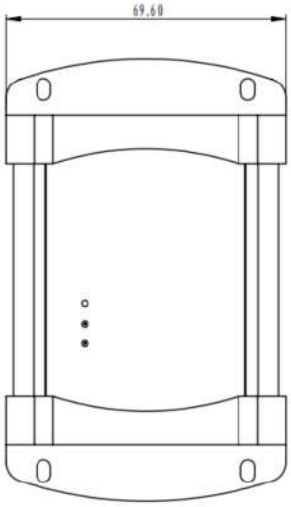
Channel	2 * CAN FD
PC Interface	USB 2.0
Driver	Driverless design for Windows system, offering excellent system compatibility
Software	TSMaster
CAN	Supports CAN 2.0 A and B protocols, compliant with the ISO 11898-1 standard, with baud rates from 125Kbps to 1Mbps
CAN FD	Supports CAN FD that complies with both ISO and non-ISO standards, with baud rates from 125Kbps to 8Mbps
Terminal Resistor	Built-in 120-ohm terminal resistor, with the resistance value configurable through software
Relay Type	Magnetic latching relay
Message Forwarding Capability	20,000 frames per second
Forwarding Latency	< 0.5ms
Isolation	CAN channel DC 2500V isolation
Power Supply	USB power supply, external DC power supply (9-36V)
Power Consumption	1W
Case Material	Metal
Dimension	Approx. 110*70*36mm
Weight	Approx. 150g (without packaging)/Approx. 368g (with packaging)

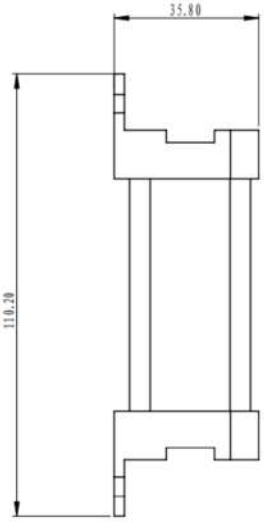
Operating Temperature	-40℃～80℃
Operating Humidity	10% ～ 90% (non-condensing)
Operating Environment	Keep away from corrosive gases

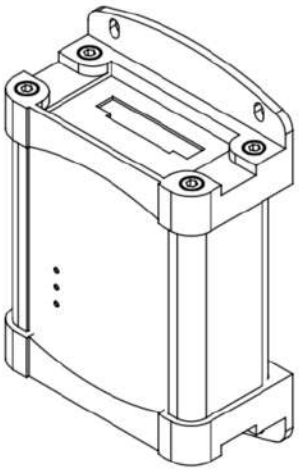
2.4 Electrical Data

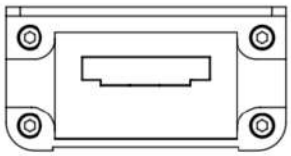
Parameter		Test Condition	Minimum Value	Typical Value	Maximum Value	Unit
Operating Voltage	USB power supply	CAN forwarding	--	5	--	V
	External DC power supply	CAN forwarding	9	12	36	V
Operating Current	USB power supply	CAN forwarding	--	0.16	--	A
	External DC power supply	CAN forwarding	--	0.06	--	A
Power Consumption	External DC power supply	CAN forwarding	--	1	--	W
CAN Interface	Bus pin voltage resistance	CANH、CAHL	-58	--	+58	V
	Isolation withstand voltage	Leakage current less than 1mA	2500	--	--	VDC

2.5 Mechanical Data









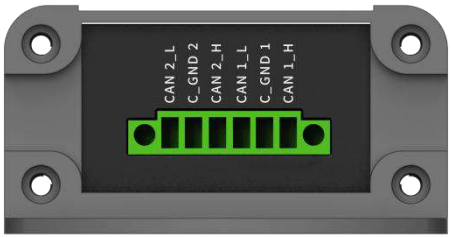
材质 Material		一般公差 General Tolerances IT12	表面处理 Surface Treatment	工艺tech.
A3		设计draw.	审核audi.	批准appr.
SCALE:1:1	SHEET 1 OF 1			
TOSUN				重量(g) Weight
上海同星智能科技有限公司 SHANGHAI TONGXING INTELLIGENT TECHNOLOGY CO.,LTD		GW2112尺寸图		版本 Rev. 00

2.6 Scope of Delivery

✓ Main device: GW2112

✓ USB cable

2.7 Hardware Interface



- USB 2.0 interface;
- DC power supply port;
- 6PIN phoenix terminal interface:

6PIN Phoenix Terminal Interface	PIN Number	Definition
CAN FD 1/2	PIN1	CAN2_L
	PIN2	C_GND2
	PIN3	CAN2_H
	PIN4	CAN1_L

	PIN5	C_GND1
	PIN6	CAN1_H

2.8 LED

Diagram of LED indicator:

Description of indicator:

Indicator	**Definition**
Config	Indicator for configuration
CAN FD 1	Indicator for CAN FD channel 1
CAN FD 2	Indicator for CAN FD channel 2

Description of LED color:

Color	**Description**
Config Green	Lights up after successful configuration
CAN FD 1 Green	CAN FD channel 1 data frame is sent or received correctly
CAN FD 2 Green	CAN FD channel 2 data frame is sent or received correctly

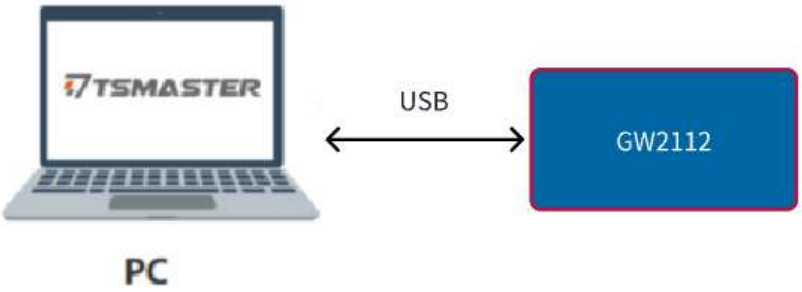
2.9 Optional Accessories

N/A.

3. Quick Start

3.1 System Connection

◆ Online configuration



When the users want to perform configuration for the device, simply connect it to a PC via USB. Once the Config light is solid, users can use the TSMaster software for the corresponding configuration functions.

◆ Offline gateway

When using the device offline, connect the two CAN interfaces to the bus and provide power to the device. Upon powering on, the device will operate according to the existed configuration in the device.

3.2 Driver Installation

All TOSUN hardware adopts a driverless design, offering excellent system compatibility. The hardware allow for direct use on various operating systems (Windows 7/8/10/11) without the need to install drivers.

3.3 Software Overview

TSMaster is a powerful and comprehensive tool that can connect, configure, and control all TOSUN hardware tools and devices, enabling functions such as automotive bus embedded code generation, monitoring, simulation, development, UDS diagnostics, CCP/XCP calibration, ECU flashing, I/O control, test measurement, and so on.

TSMaster supports Matlab Simulink co-simulation and CarSim dynamic model ECU algorithm simulation testing (soft real-time HIL). It provides users with a series of convenient functions and editors, allowing them to directly execute ECU code within TSMaster and supports C script and Python script editing. At the same time, TSMaster also offers a mini-program function, enabling users to customize the simulation test panel, test process, test logic, and even the entire test system, and automatically generate reports. The code written by users based on TSMaster is hardware-independent, and can be easily shared, referenced, and used on different hardware platforms.

TSMaster supports multiple commonly used bus tool brands, including Vector, Kvaser, PEAK, IXXAT, as well as mainstream instruments in the market (such as oscilloscopes, waveform generators, and digital multimeters) and boards (such as AI, DI, DO, etc.). Its design concept is to perfectly integrate with the test system to achieve joint simulation and testing of multiple hardware and multiple channels. This enables TSMaster to meet the PV/DV test verification needs for various automotive electronic components and assemblies, as well as the inspection requirements for the production line.

3.4 Software Installation

TSMaster software download link:

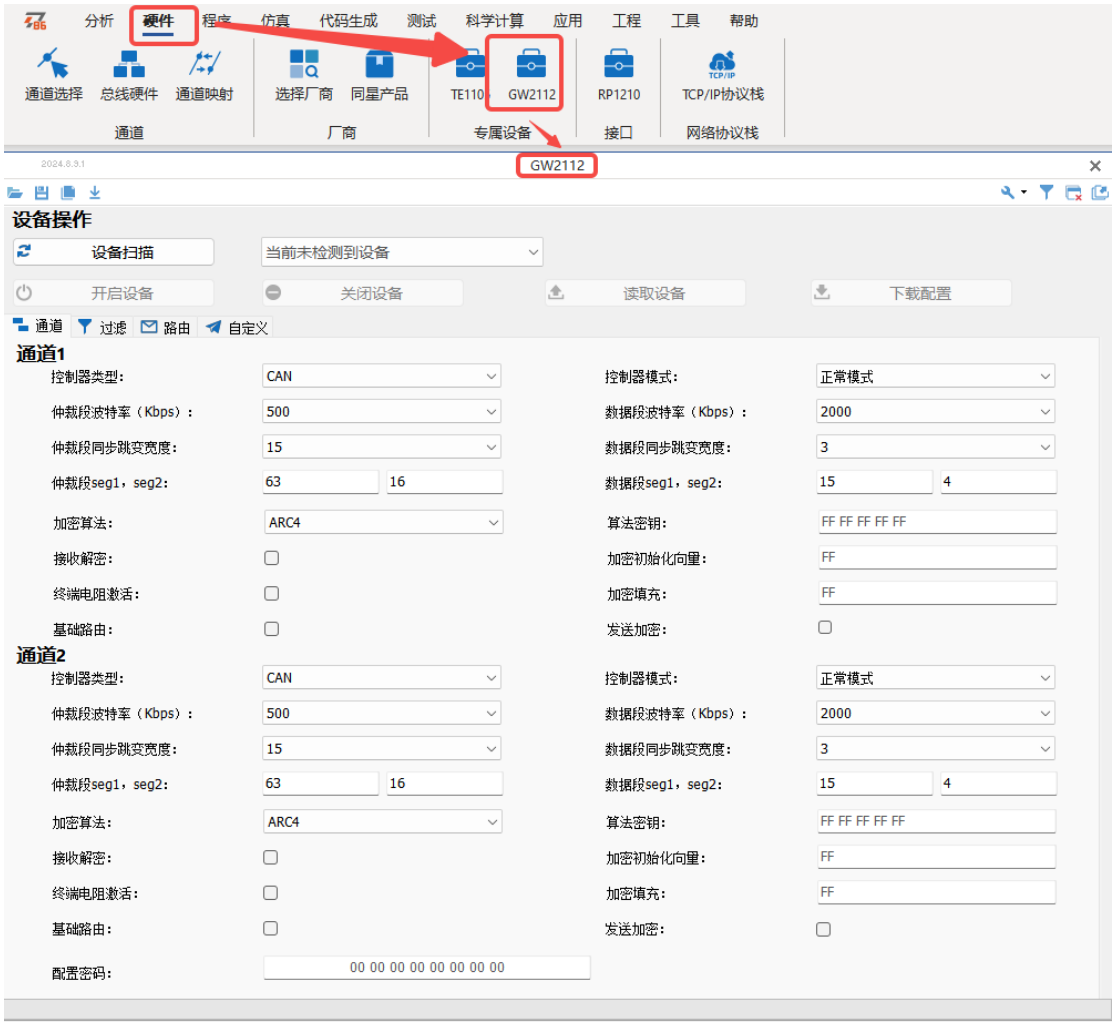
http://download.tosun.tech/TOSUNSoftware/TSMaster_Setup_beta.7z

If the link is not accessible, you can contact the corresponding sales personnel or visit the official TOSUN website to obtain the software. Meanwhile, you can scan the QR code to follow the TOSUN official account to get the download link.

After the installation, you can see the following software on the PC.

3.5 GW2112 Toolbox

Update the TSMaster software to the latest version, and then open the GW2112 configuration interface in TSMaster.



◆ File Import and Export

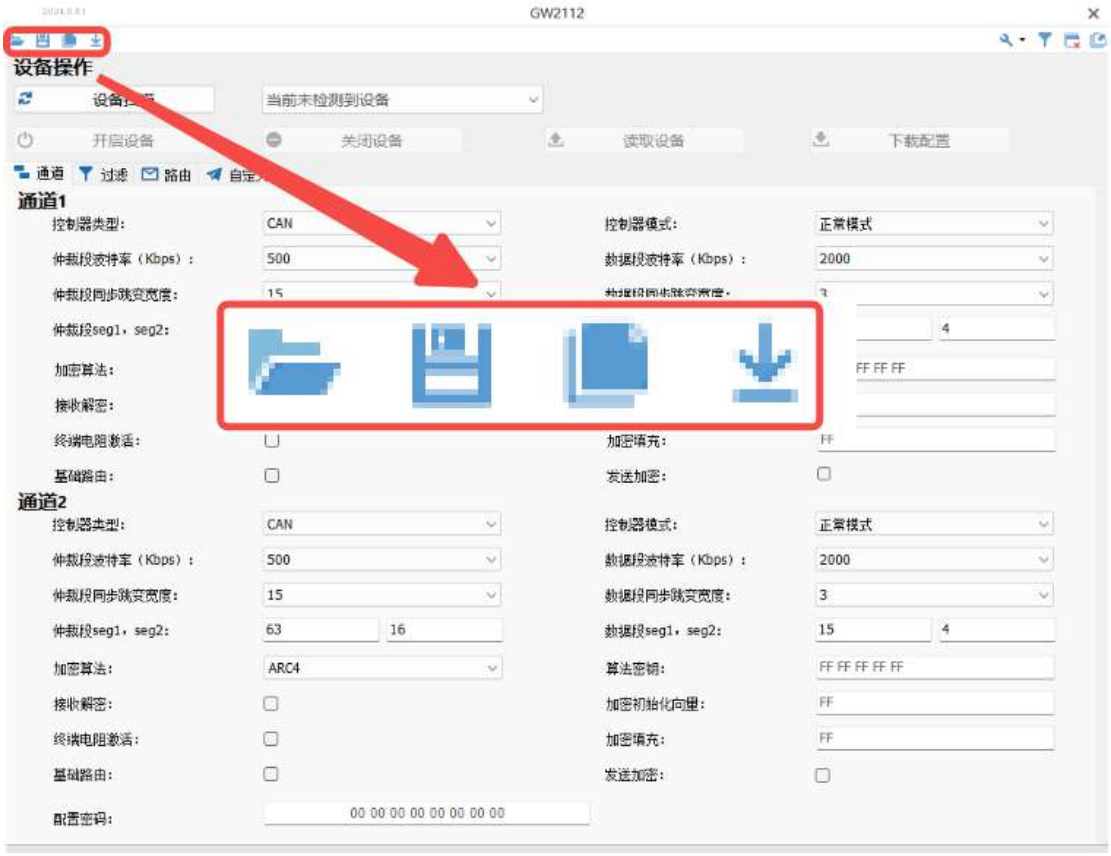
In the upper left corner of the configuration interface, there are the following four icons:

Load Configuration : load previously saved JSON configuration file;

Save Configuration : save the current configuration in a JSON file, so that the configuration can be directly imported next time by loading this file;

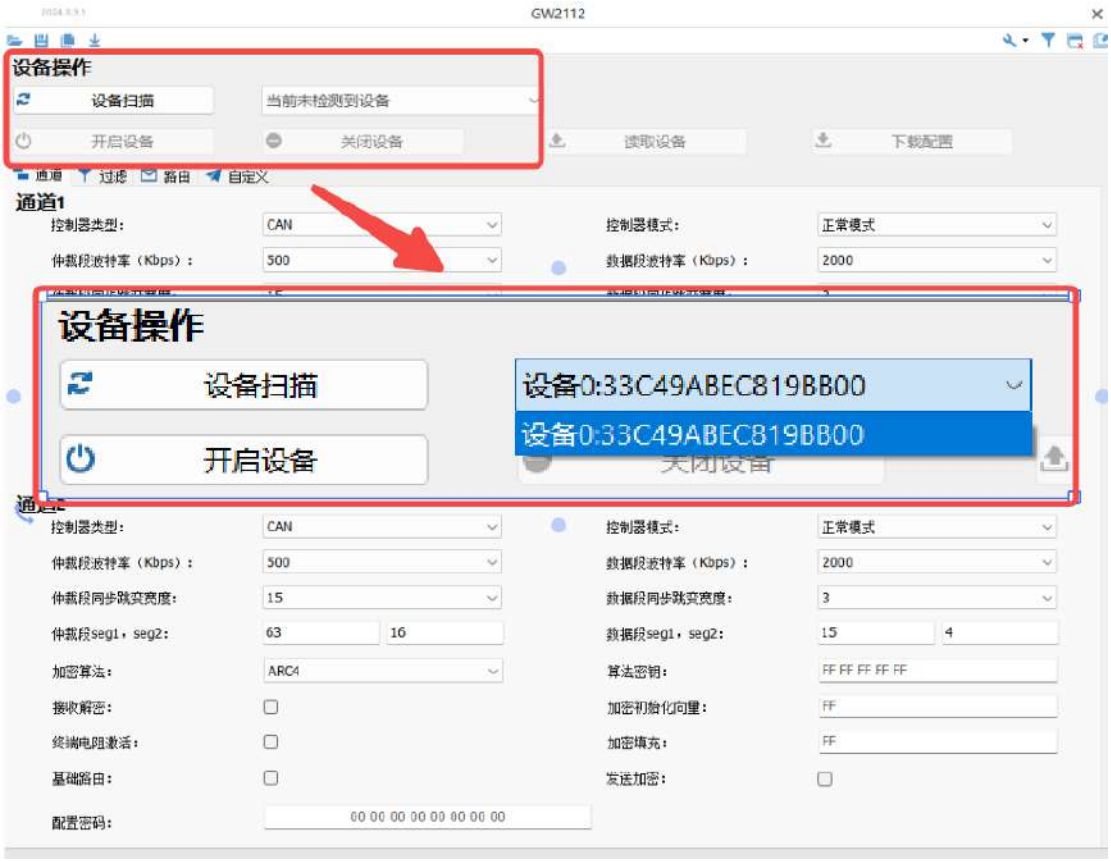
Load Custom BIN File : This button allows the users to import custom configuration BIN file;

Manual download : download the GW2112 user manual;



◆ Device Connection

Clicking the "Scan Devices" button will detect the currently online devices, which may include multiple devices. Select the target device from the dropdown list. The "Open Device" button is used to connect to the device, while the "Close Device" button disconnects the device.



◆ Read Device and DownloadCfg

The "Read Device" button is used to read the configuration information of the currently connected device.;

The "DownloadCfg" button is used to download the configuration from the configuration interface into the device, replacing the existing configuration.

◆ Function Configuration

Channel: Configure parameters for the two CAN channels;

Filtration: Configure filters to filter unnecessary messages in the channel;

Route: Configure conversion relationships, mainly for flexible message conversion and management between two CAN (or CAN FD) networks;

Custom: Users can define custom routing rules, allowing them to customize the message conversion rules and processing logic of the GW2112 according to specific needs.

3.6 Function Configuration

Before using the GW2112, users need to perform the function configuration. After connecting the device to the computer via USB and ensuring the Config light on the device is solid, users can start the configuration.

Function configuration is divided into four parts: channel, filtration, route, and custom.

3.6.1 Device Connection

Refer to “3.5 GW2112 Toolbox”.

3.6.2 Read Configuration and Download Configuration

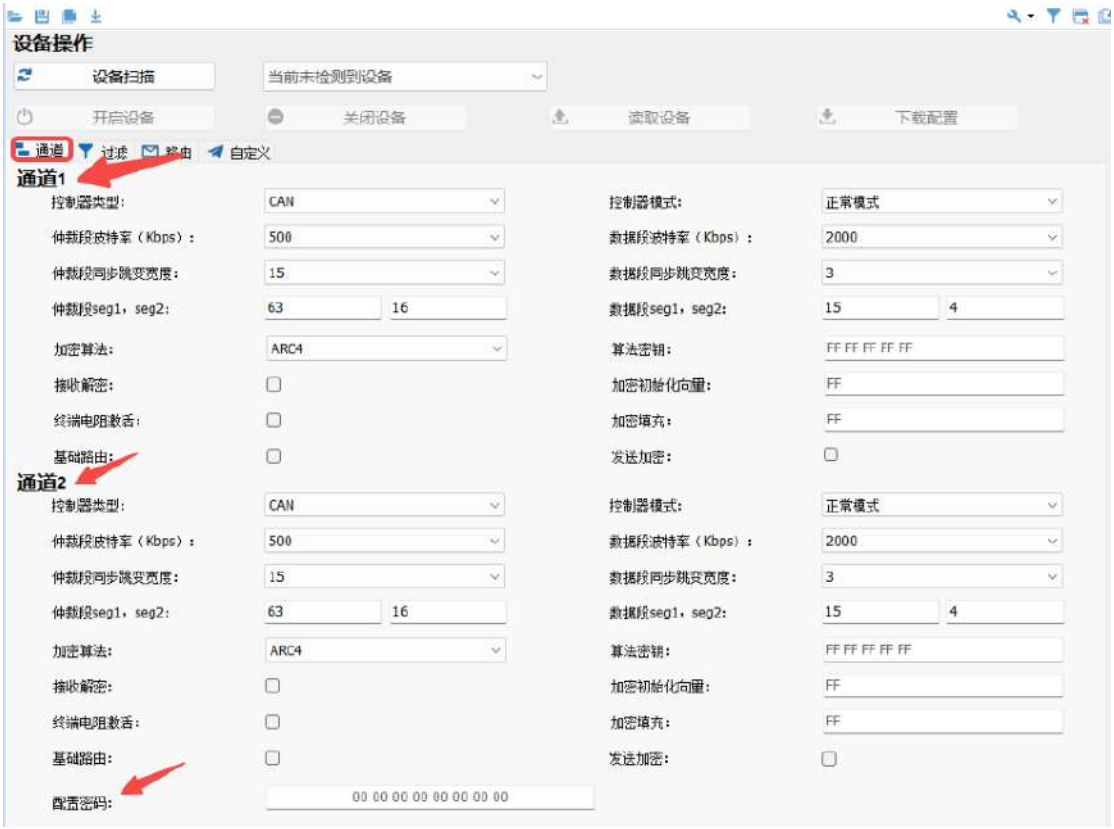
Refer to “3.5 GW2112 Toolbox”.

3.6.3 Channel Configuration

The channel configuration can set up two CAN/CAN FD channels. The upper part of the panel is for configuring channel 1 information, while the lower part is for configuring channel 2 information.

In addition to channel configuration, a password can also be set.

The panel layout for channel configuration is shown in the figure below:

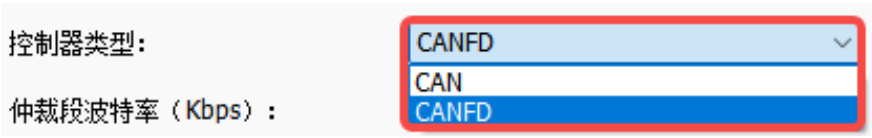


◆ Controller Type

Each channel can be configured to either CAN or CAN FD type.

Configured as CAN: Prevents CAN FD messages from being forwarded to the CAN bus;

Configured as CAN FD: Both CAN and CAN FD messages can be forwarded.

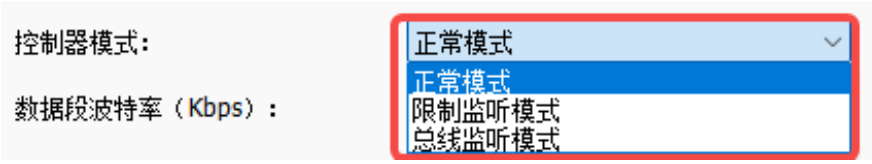


◆ Controller Mode

Normal Mode: No restrictions, normal forwarding.

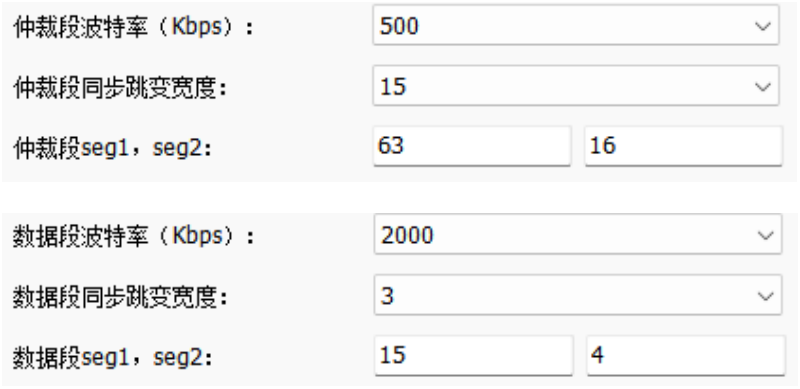
Restricted Listen Mode: In this mode, the controller can receive data frames and remote frames, acknowledge valid frames, but does not support sending data frames, remote frames, active error frames, or overload frames.

Bus Listen-Only Mode: In this mode, the controller can receive valid data frames and valid remote frames but does not support acknowledging valid frames.



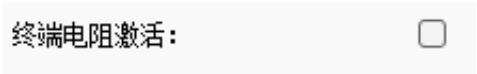
◆ Basic Parameters

Configure the baud rate, synchronization jump width, and segment as needed.

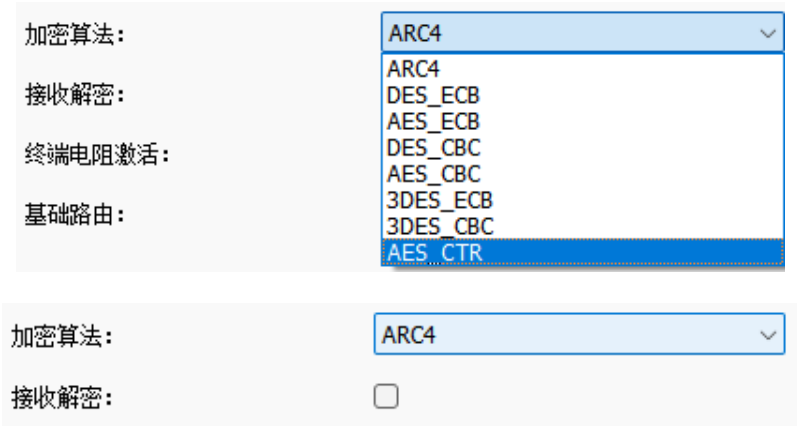


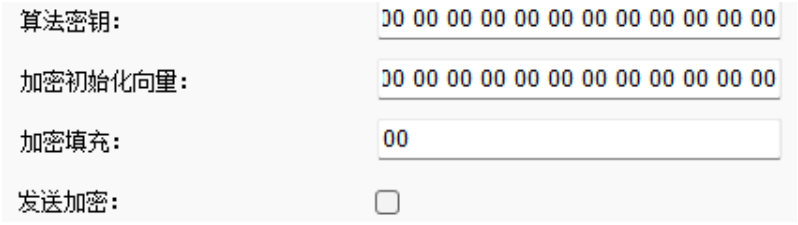
◆ Enable Terminal Resistor

When checked, the built-in 120Ω terminal resistor on the device will be enabled.



◆ Encryption





Encryption algorithm: The device offers various encryption algorithms that can be selected in the encryption algorithm dropdown list. Different algorithms have specific requirements for encryption and decryption lengths.

Algorithm	Length
ARC4	No restrictions
DES_ECB	Multiples of 16
AES_ECB	Multiples of 8
DES_CBC	Multiples of 16
AES_CBC	Multiples of 8
3DES_ECB	Multiples of 16
3DES_CBC	Multiples of 16
AES_CTR	Multiples of 8

Algorithm key: Set the corresponding key based on the algorithm.

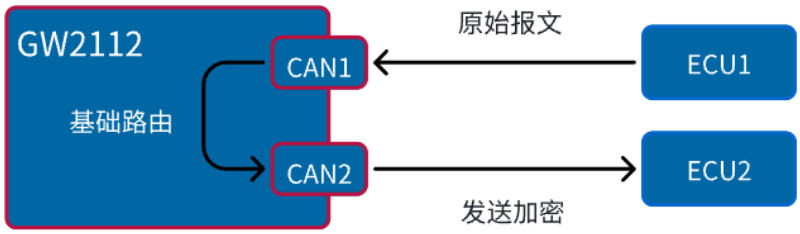
Receive decryption: Check this option when it is necessary to decrypt the received message.

Send encryption: Check this option when it is necessary to encrypt the message being sent.

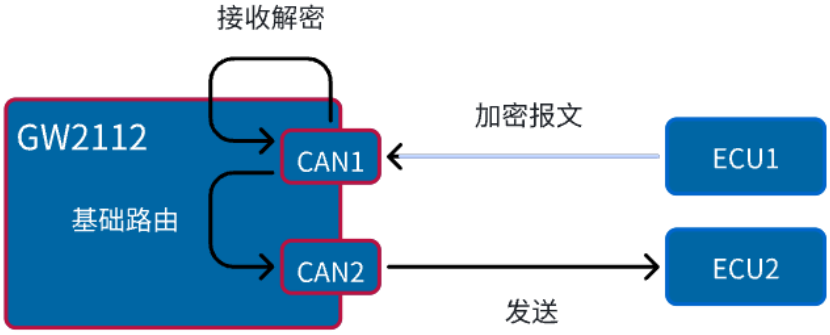
Encryption initialization vector: Used to increase the randomness of encrypted data, ensuring that the same data produces different ciphertexts in different encryption operations;

Encryption padding: When the plaintext length is insufficient to meet the block size requirements of the encryption algorithm, padding data will be added to the plaintext to make its length compliant.

Encryption process:



Decryption process:



◆ Basic routing

When basic routing is selected, data received from one channel will be forwarded unchanged if no processing is applied. This feature is mostly used for testing.

Configuration password:

It is an 8-byte hexadecimal number, and the configured hexadecimal password must be entered when reading the device.

3.6.4 Configuration Filtering

The filter is used to block messages that do not need to be processed by devices in the channel. Users can add a filtering item by clicking "Add." Each filtering item needs to specify which channel it applies to, the filtering format (standard frame or extended frame), the filter ID,

and the mask ID. The filtering method uses mask filtering.

In most cases, the filter is used to filter a single message. For example: channel CAN1, filtering format as standard frame, filter ID is 0x1, and mask ID is 0x7FF. This filtering item indicates that when the channel receives a standard frame, only messages with an ID of 0x1 are allowed through. Multiple filtering items are in a union relationship.

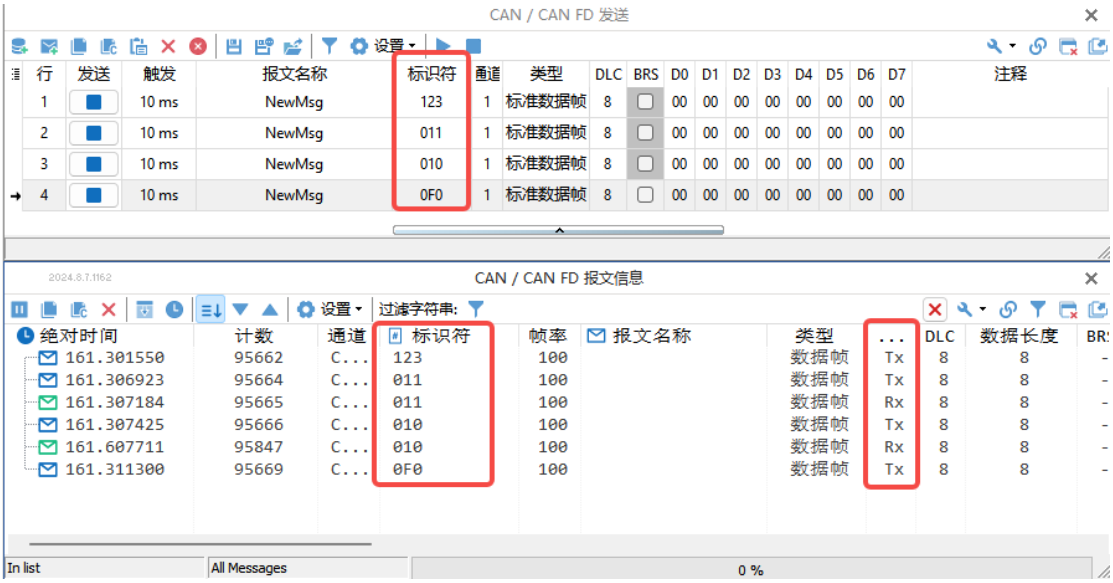
◆ Example

Configure the channel as CAN1, filtering format as standard frame ID, filter ID as 0x11, and mask ID as 0xF0.

The result is as shown in below figure:

Using TSMaster's message sending and message information functions, messages with IDs 0x123, 0x011, 0x010, and 0x0F0 were sent through the CAN1 channel. It can be observed that only the messages with IDs 0x11 and 0x10 were received, indicating that GW2112 filtered out messages with IDs other than 0x11 and 0x10 and those outside the mask ID.

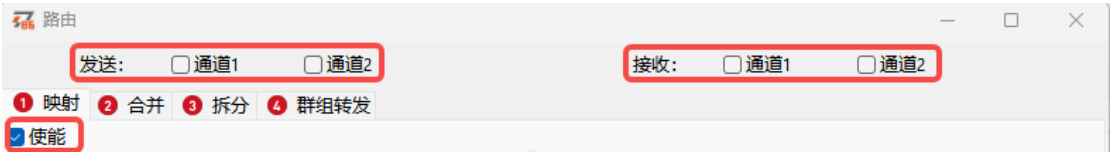
(0x11 & 0xF0 = 0x10)



3.6.5 Routing Configuration

Routing configuration is used to configure conversion relationships, which include four types: mapping, splitting, merging, and group forwarding. Users can add a relationship by clicking "Add".

Note: When adding any type of relationship, users must first check the "Enable" option at the top, as well as the "Receive" and "Send" options, to determine which channel's messages the rule will process and through which channel the processed messages will be sent. Users can only begin the configuration after checking "Enable".



The routing configuration feature comes with built-in routing, so to make the routing configuration effective, the basic routing option in the channel configuration must be turned off. This is because basic routing forwards data without any processing.

◆ Map

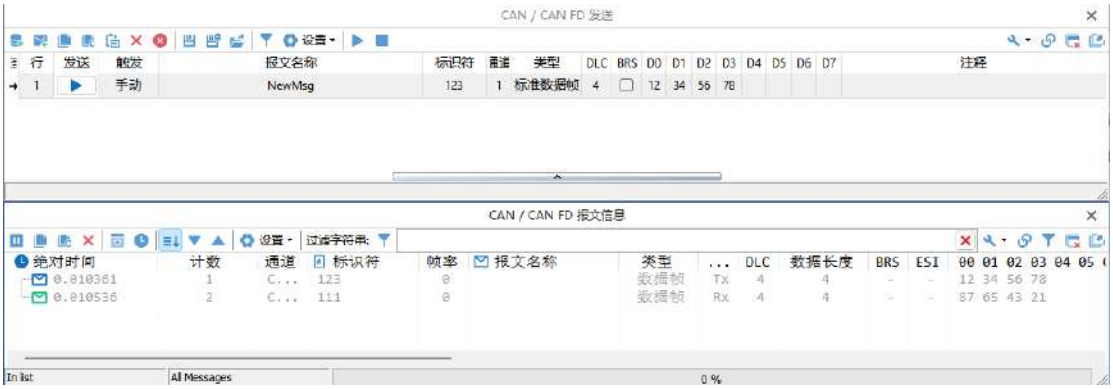
When a specified message is received, it will be transformed into another pre-configured message, as shown in the figure below. When a source message that matches the left side is received, it will be converted into the target message on the right side. Note that only the checked items on the left side will be transformed; the unchecked parts will retain the original message format.

Note: The "BRS" option refers to the BRS bit, which is not present in CAN, so this option should be set to "No Switch". The other configurations should also comply with the standard specifications.

➢ Example

The configuration is as shown below:

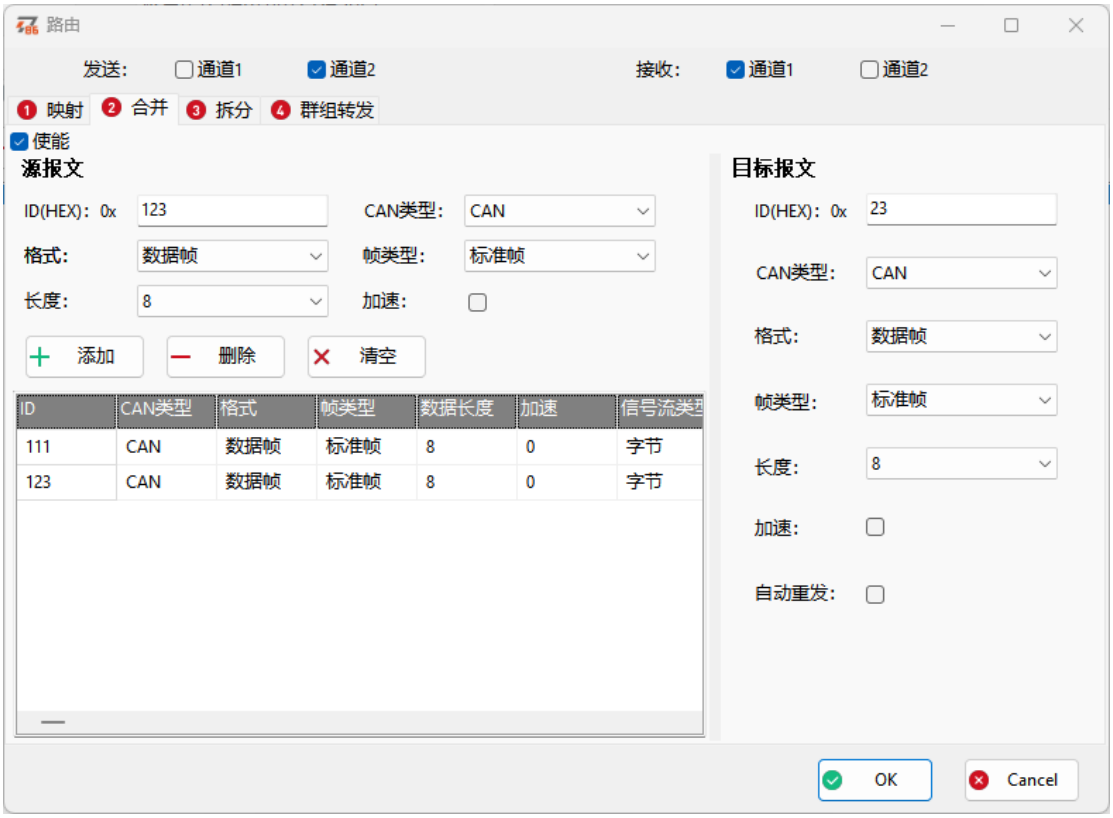
Demo: As shown in the figure below, the source message was sent with an ID of 0x123 and data of 0x12345678. After being converted by the GW2112, it was transformed into a message with an ID of 0x111 and data of 0x87654321.



◆ Merge

When a specified message is received, part of its content is cached. When the trigger message (the last message involved in the merging) is received, the merged message is sent.

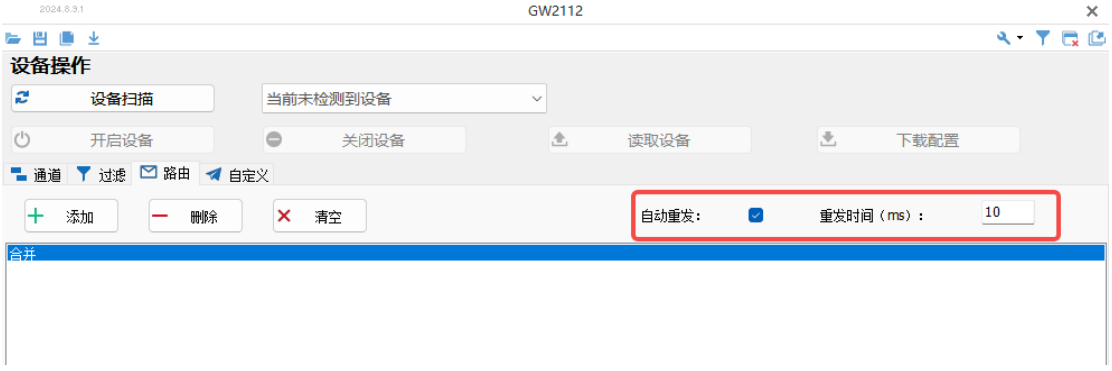
As shown in the figure below, the routing merge configuration panel is divided into two parts: Src message and Dst message.



➢ Dst message

Configuring the target message is straightforward. Simply configure it directly, and the GW2112 will process the source message accordingly to match the target message format.

Automatic retry: If the "Automatic retry" option is selected for the target message, it indicates that after the target message is triggered for sending, it will automatically resend. The automatic retry feature for the target message also needs to be used in conjunction with the automatic retry setting on the previous panel. If automatic retry is required, both options must be checked, and the retransmission interval should be configured, with the unit being milliseconds (ms).



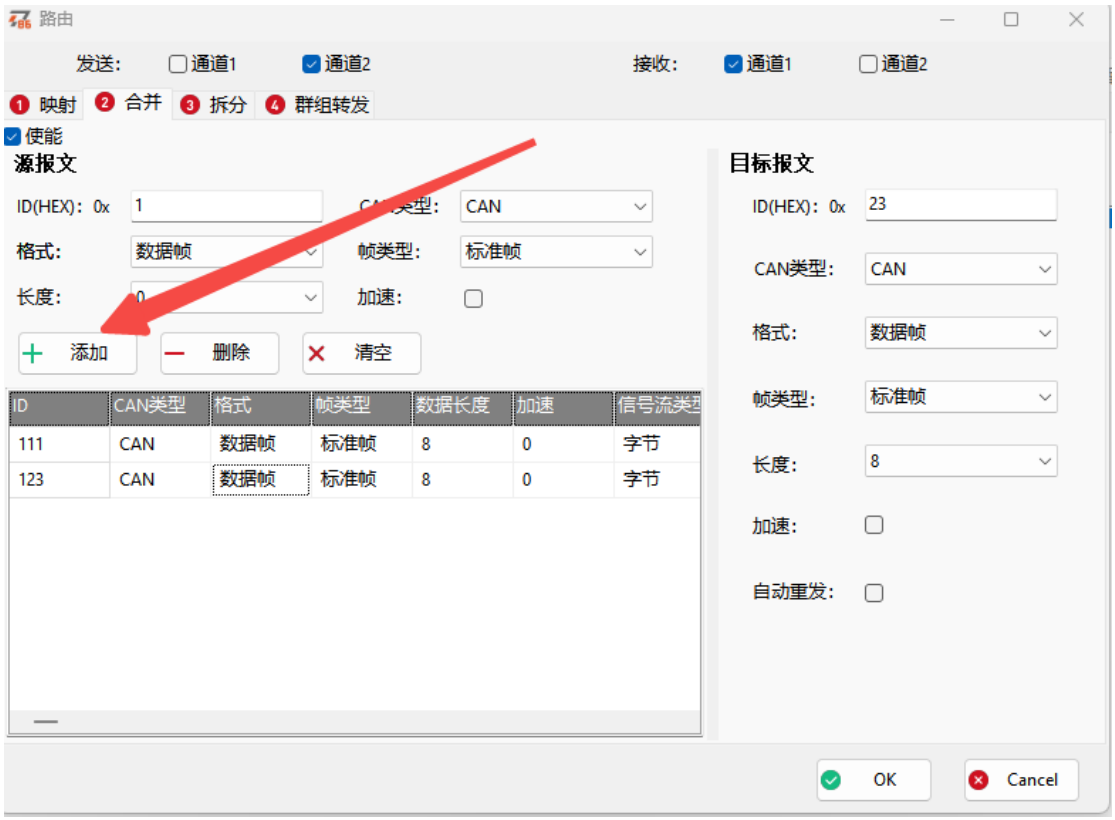
➢ Src message

Check "Enable" to start the configuration. Multiple different IDs can be configured, and each ID needs to have a corresponding merge rule added. When the GW2112 receives a message configured here, it will cache it until the last configured message is received. The last configured message acts as the trigger message. Once the GW2112 receives this trigger message, it will merge it with the previously cached messages to form the data segment of the target message and then send it out.

After setting up the ID and other basic information, clicking the "Add" button will open the merge rule configuration window.

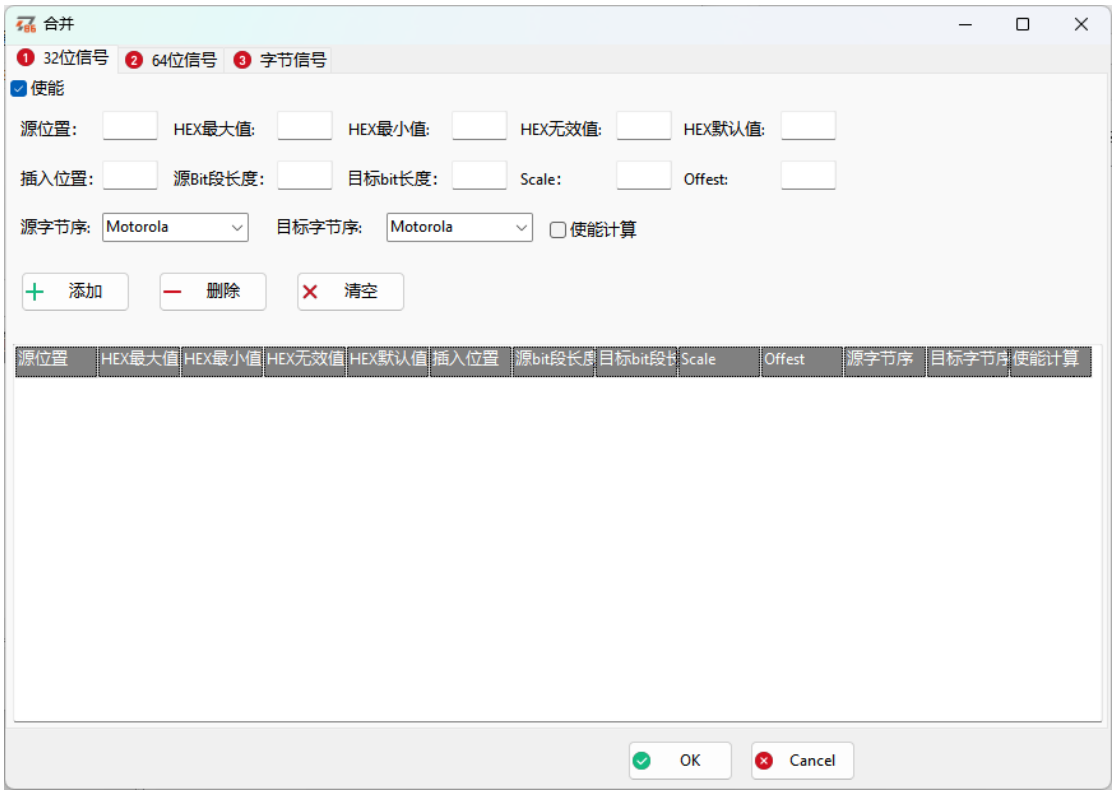
The "Delete" button, when clicked, will remove the last message configuration.

The "Clear" button will clear all message configurations at once



➢ 32-bit Signal

As shown in the figure below, the 32-bit signal has multiple parameters for configuration.



Src Position, Src Bit Segment, Src Endian are used to confirm the source of the data.

For endian, Motorola represents big-endian, while Intel represents little-endian.

Src Position the starting position (bit) from which the signal is extracted in the original message. Along with the length and byte order, this can yield an actual physical value (which the device calculates internally). If "Enable Calcula" is checked, this value will be multiplied by the Scale and then the Offset will be added to obtain a new value.

The HEXMin, HEXInvalid, and HEXDefault values are used to constrain the significance of this physical value. If the physical value is less than the minimum value, greater than the maximum value, or equal to the invalid value, it will be changed to the default value.

The insert Position, DstBitLength, and DstEndian are used to determine the position of the signal within the data segment of the split sub-item, meaning that the resulting HEX value will be converted into bits and inserted into the corresponding position of the split sub-item.

- 64-bit Signal

Same as 32-bit signal. The only difference is the the upper limit of the bit segment length.

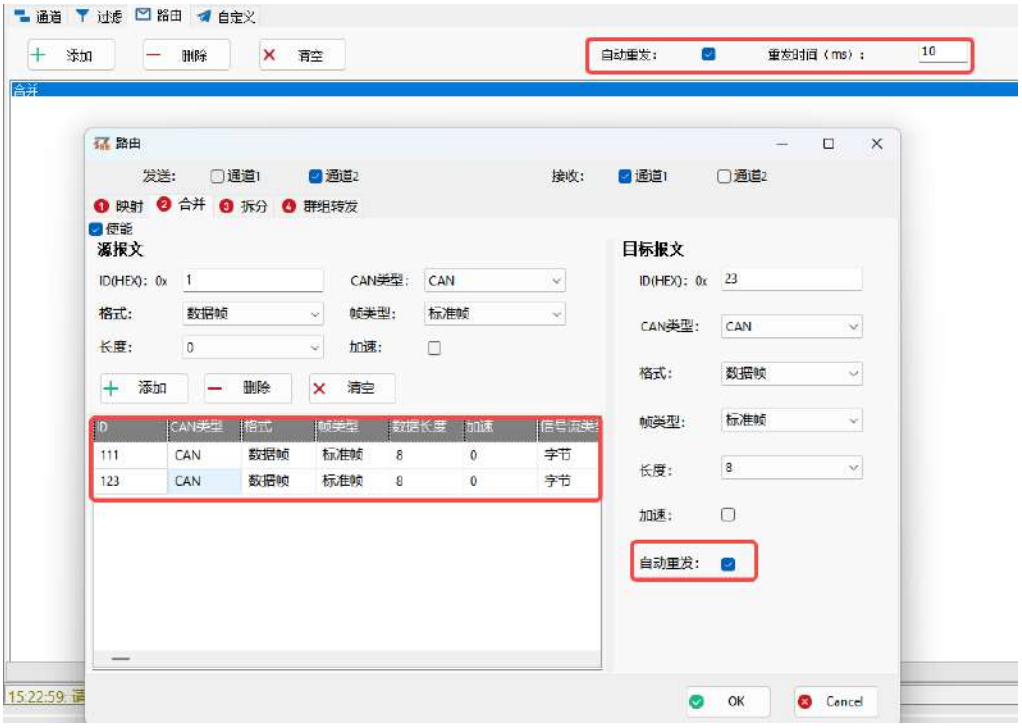
When the signal length exceeds 32 bits, select 64-bit signal.

- Byte Signal

In most cases, byte signal is used. Users only need to specify the Src Start Endian, DLC, and Dst Start Endian. The data from the source message, starting at the source starting byte (Src Start Endian) and for the specified length (DLC), will be placed into the split sub-item.

- Example

Configure messages with IDs 0x111 and 0x123, with the merge rule set to "Byte Signal". The Dst Message ID is set to 0x23, and "Automatic Retry" is set to enabled, with a time interval of 10 ms:



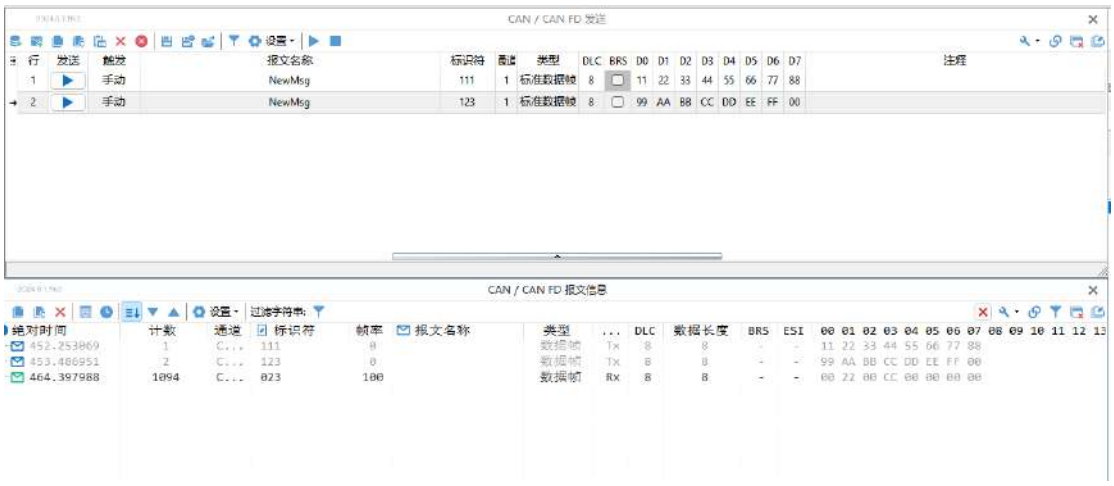
The configuration for the message with ID 0x111 is as follows:

目标起始字节	长度	源起始字节
1	1	1

The configuration for the message with ID 0x123 is as follows:

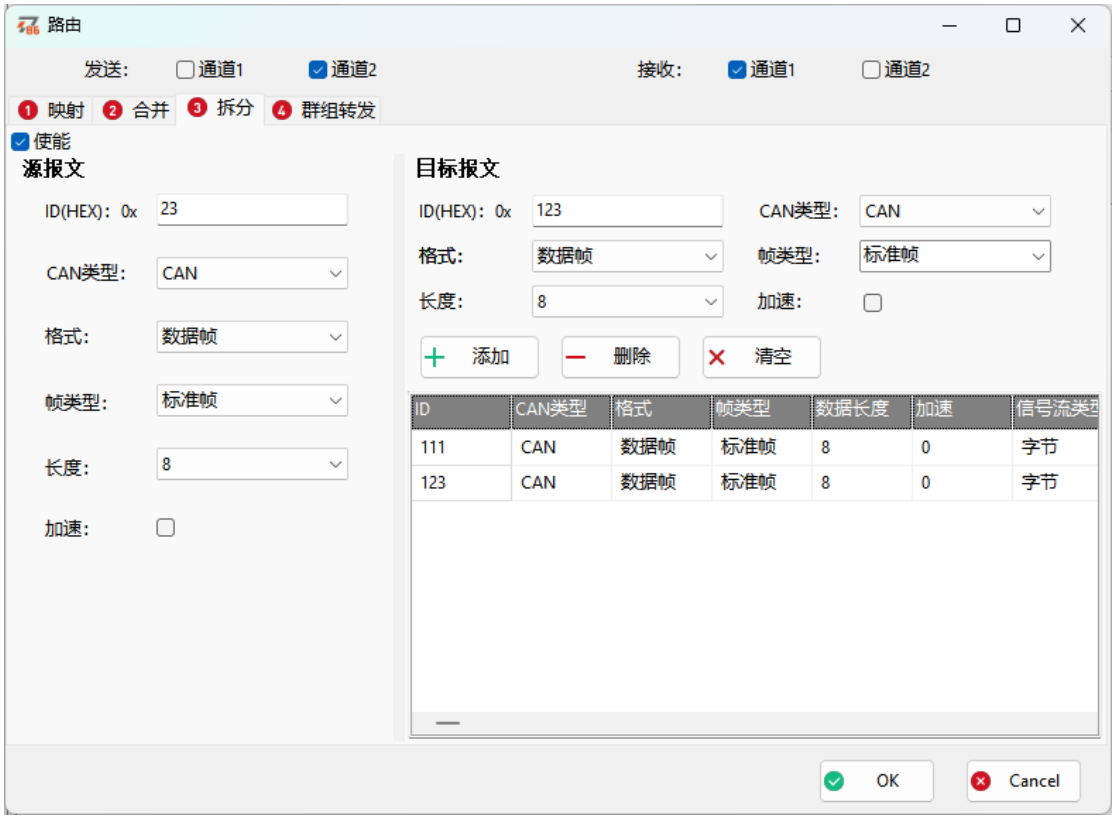
目标起始字节	长度	源起始字节
3	1	3

The result is as follows: First, send a message with ID 0x111, followed by a message with ID 0x123. After sending the message with ID 0x123, a message with ID 0x23 will be received. This message corresponds to the target message configured in the GW2112. It can be observed that the data segment of the target message with ID 0x23 is merged according to the bit positions and lengths specified in the configuration rules for the two source IDs. The remaining bits are padded with zeros. The target message is received at intervals of 10 ms.



◆ Split

Refer to “Merge”. Split refers to dividing a single message into multiple messages.

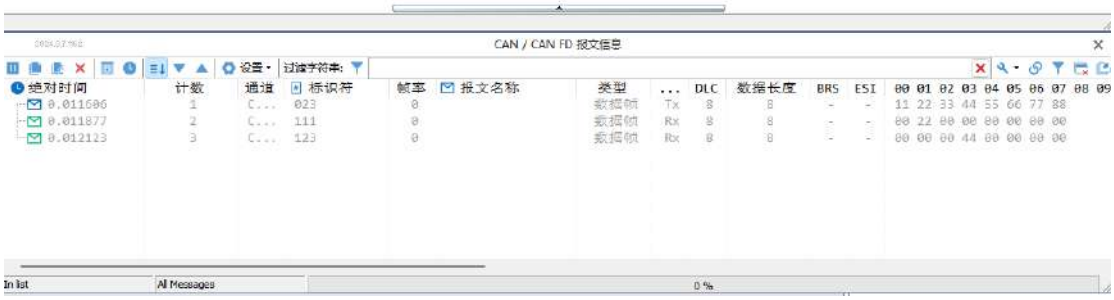


➢ Example

Configure the source message ID as 0x23, with target message IDs as 0x111 and 0x123. The target starting bytes are set to 1 and 3 respectively, both with a length of 1, and the source starting bytes are also set to 1 and 3 respectively. The result is shown in the figure below:

After being forwarded by the GW2112, the message with ID 0x23 is split into two messages,

and the data segments are split according to the configuration rules.



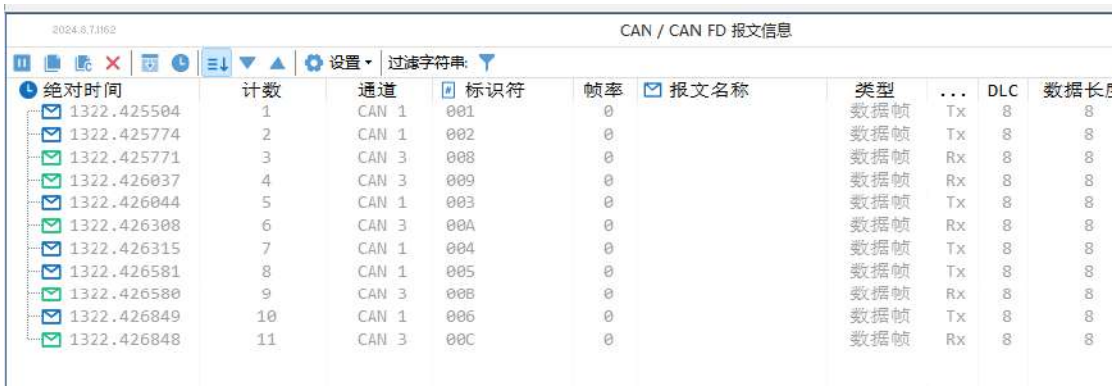
◆ Group Forwarding

As shown in the figure below, group forwarding has three configuration items, and these items only take effect for ID.

Next, let's explain these three configuration items in conjunction with a case:

The srcMsgID is configured as 1, the dstMsgID is configured as 8, and the ID Offset Length is set to 4. The result is as follows: Using the TSMaster message sending tool, six consecutive messages with IDs starting from 1 were sent. The message information tool indicates that the received (RX) message IDs start from 8, and a total of five messages from 0x8 to 0xC were received. However, a total of six messages were sent out. This is because the ID Offset Length is set to 4, meaning that starting from the initial ID (including the starting ID), a total of 4+1 messages will be forwarded by the GW2112 with a starting message ID of 8. Therefore, in this

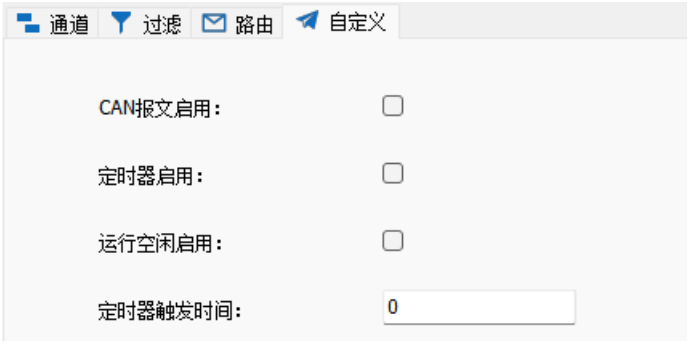
case, the message with ID 6 is not forwarded.



3.6.6 Custom configuration

Users can customize configuration rules. In the custom configuration, all previously mentioned rules can be configured, allowing for more detailed and precise configuration.

◆ Panel Description



CAN Message Enable: Check "CAN Message Enable" to indicate the activation of an event upon receiving a CAN message.

Timer enable and timer trigger time: Check "Timer enable" and set the "Timer trigger time" (in milliseconds). This indicates that the timer start event is triggered at the specified millisecond interval.

Run idle enable: This represents an event that is executed every time in the main loop program of the device.

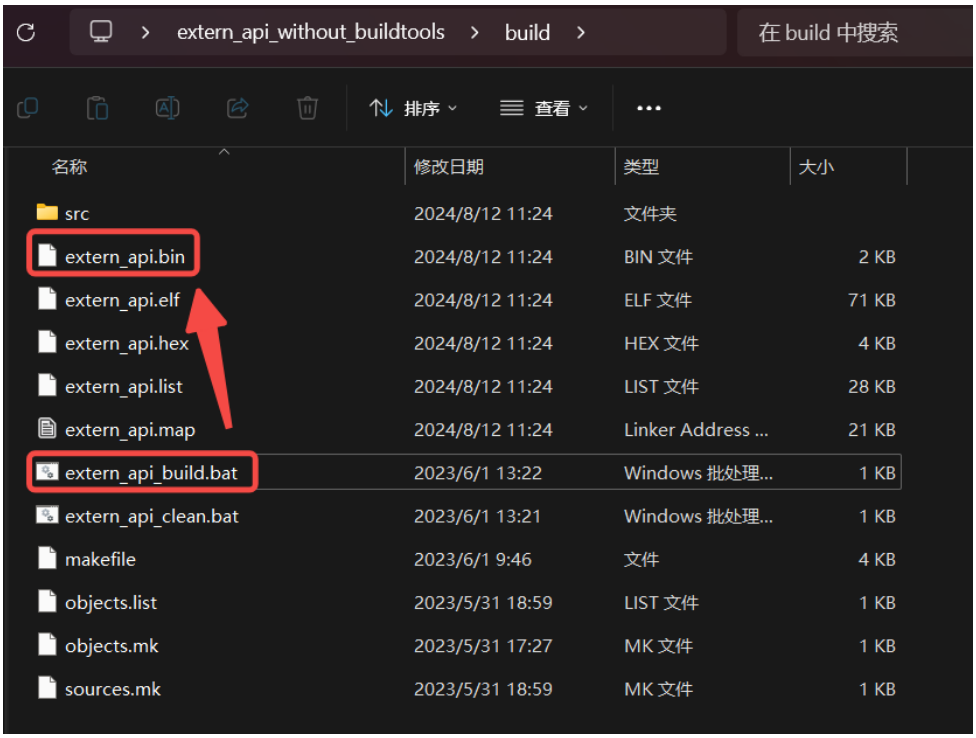
All three options can be enabled simultaneously.

◆ Compile and Download

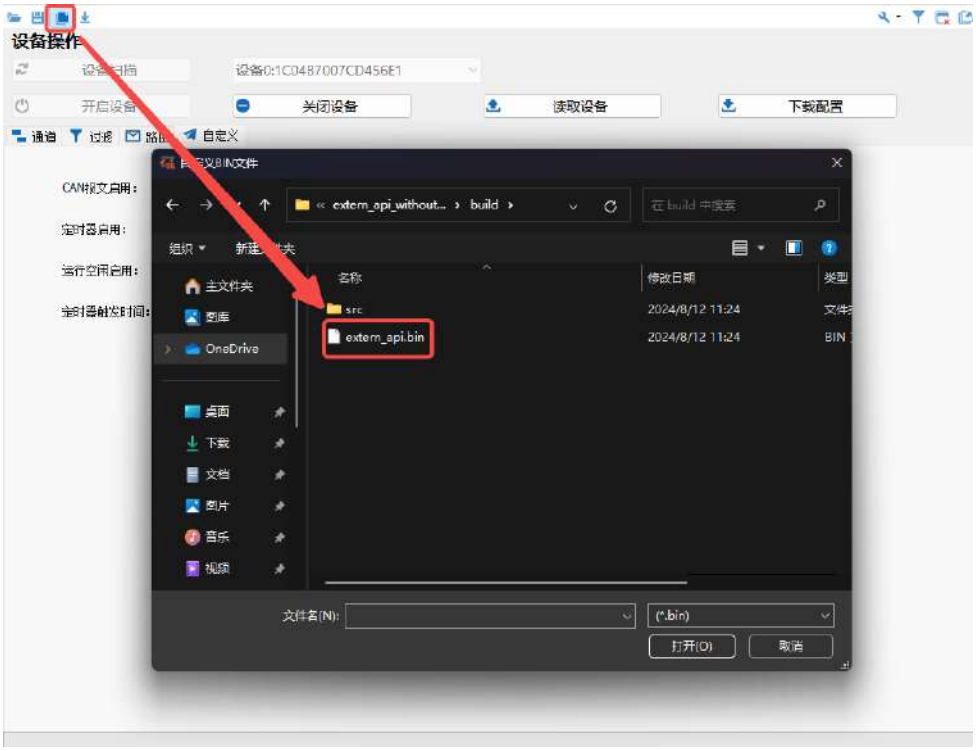
GW2112 custom functions can be compiled by the user or use a pre-compiled bin file from others, which can then be downloaded to the device through the host computer for use.

Open the attachment: Note that the directory containing the attachment should not have a Chinese path.

Enter the "build" directory and double-click the "extern_api_build.bat" file. This will generate/update a bin file in the same directory. This bin file is what needs to be downloaded to the device.



Open TSMaster on the host computer, select the GW2112 device, and enter the configuration panel to turn on the device. Then click the “Load Custom BIN File” button, and select the corresponding bin file to load it.



◆ Custom function source codes

Open the "extern_api.c" file in the src directory of the attachment, and locate the "Event_Handle" function. This function is the body of the custom function.

Function Definition:

```
uint32_t Event_Handle(const Event *Event_p, const Action_Source* Action_p);
void Event_Runtime(const Action_Source* Action_p);
```

Function Parameters:

➢ Event_p

```
typedef struct {
    Event_Type Event_Source;
    Event_Argument Argument;
} Event;
```

Event_Source: indicates the reason for triggering the custom function. During the operation of the device, there are various conditions that can lead to entering this custom function. Entering this custom function is due to the occurrence of a specific situation.. This Event_Source helps to determine the reason for the current entering of the custom function.

```
typedef enum {
    RECEIVE_CAN = 0x00,
    TIMER_IRQ = 0x01,
    START_IRQ = 0x02
} Event_Type;
```

Currently, there are three conditions: "RECEIVE_CAN" indicates that a CAN message has been received, triggering the entry into this custom function. "TIMER_IRQ" means that the entry into this custom function was triggered by a timer, and "START_IRQ" indicates that the device has just been powered on, leading to the entry into this custom function. Each time the custom function is entered, there can only be one triggering reason, and different handling is done for different triggering conditions.

```
typedef union {
    Can_Receive_Event Receive_Can;
    Timer_Irq_Event Irq_Timer;
    Start_Irq_Event On_Start;
} Event_Argument;
```

Argument: This structure is used to provide the resources corresponding to the triggering of this custom function. If the custom function is triggered by receiving a CAN message, then the

only resource available here is Receive_Can:

```
//触发事件，触发原因，操作外设的接口，指令需不需要向上层汇报，用户的空间
typedef struct {
    uint32_t Id;
    uint8_t SourcePort; //来自哪个通道 0为1通道，1为2通道
    uint8_t FrameType; //0为远程帧，1为数据帧
    uint8_t IdType; //0为标准帧，1为扩展帧
    uint8_t CanType; //0为经典CAN，1为FDCAN
    uint8_t BRS; //0为关闭，1为开启
    uint8_t DataLength;
    uint8_t Free[2];
    uint8_t Data[64];
} Can_Receive_Event;
```

Here, users can get all the information about the received message that triggered this custom function.

If it is triggered by a timer, then the only resource available here is Irq_Timer:

```
typedef struct {
    uint32_t Time;  //Timer记时，叠加，单位为ms

} Timer_Irq_Event;
```

- Action_p

This structure also includes two parts: a sending function and a user-defined space:

```
typedef struct {
    void(*Can_Transmit)(Transmit_CanTypedef SendCan);

    uint8_t* user_array;
} Action_Source;
```

Sending function Can_Transmit: Use this function to send a frame of messages from the device. The parameters for SendCan are as follows:

```
typedef struct {
    uint32_t Id;
    uint8_t TargetPort; //1为1通道，2为2通道
    uint8_t FrameType; //0为远程帧，1为数据帧
    uint8_t IdType; //0为标准帧，1为扩展帧
    uint8_t CanType; //0为经典CAN，1为FDCAN
    uint8_t BRS; //0为关闭，1为开启
    uint8_t DataLength;
    uint8_t Free[2];
    uint8_t Data[64];
} Transmit_CanTypedef;
```

According to the requirements, modify the parameters for SendCan, and then call

Can_Transmit to send a frame of the custom message.

The user_array is a pointer that points to a dedicated custom space. General data is typically released after the corresponding trigger function call ends, so the next time the custom function is entered, it will contain new data. If users need to retain some data from this custom processing for use when re-entering the custom function, users can use user_array to store that data. The data preserved in the user-defined space will remain until manually overwritten in the custom function, and it will persist until power is lost.

- Run the idle enable function

```
/*
示例: 用户自定义函数
*/
__attribute__((section(".EXTERN_FUNC")))
void Event_Runtime(const Action_Source* Action_p)
{

}
```

The implementation of this function is defined by the user. It will be called when the "Run Idle Enable" option is checked in the host computer. This function will be invoked during each iteration of the device's main program loop.

Example:

The attachment includes a very detailed example, and here is a comprehensive introduction to the example:

```
Transmit_CanTypedef TxCAN = {0};
uint32_t Counter = 0;
```

First, two variables are defined: TxCAN has the parameters for the sending function, and Counter is used for counting. The two variables will be used later.

```
if (Event_p->Event_Source == ON_START) {
    TxCAN.Id = 0x1CCAB21;
    TxCAN.FrameType = 1;
    TxCAN.IdType = 1;
    TxCAN.CanType = 1;
    TxCAN.BRS = 1;
    TxCAN.DataLength = 8;
    TxCAN.TargetPort = 0x02;
    for (int i = 0; i < 8; i++) {
        TxCAN.Data[i] = (i + 2);
    }
    Action_p->Can_Transmit(TxCAN);
}
```

Here, it indicates that when the device is powered on, channel 2 will send a CAN FD message with an ID of 0x1CCAB21, a length of 8, and data with contents 2 to 9.

First, let's take a look at the CAN_RECEIVE part, and ignore the timer trigger part for now:

```
else if (Event_p->Event_Source == CAN_RECEIVE) {        //接收报文事件
    if (Event_p->Argument.Receive_Can.SourcePort == 0x00) {
        if (Event_p->Argument.Receive_Can.Id == 0xA1) {
            Action_p->user_array[4] = (Event_p->Argument.Receive_Can.Id >
```

Here, it indicates that this is valid only when channel 1 receives a message.

```
if (Event_p->Argument.Receive_Can.Id == 0xA1) {
    Action_p->user_array[4] = (Event_p->Argument.Receive_Can.Id >> 24) & 0xFF;
    Action_p->user_array[5] = (Event_p->Argument.Receive_Can.Id >> 16) & 0xFF;
    Action_p->user_array[6] = (Event_p->Argument.Receive_Can.Id >> 8) & 0xFF;
    Action_p->user_array[7] = (Event_p->Argument.Receive_Can.Id) & 0xFF;
    Action_p->user_array[8] = (Event_p->Argument.Receive_Can.FrameType);
    Action_p->user_array[9] = (Event_p->Argument.Receive_Can.IdType);
    Action_p->user_array[10] = (Event_p->Argument.Receive_Can.CanType);
    Action_p->user_array[11] = (Event_p->Argument.Receive_Can.BRS);
    Action_p->user_array[12] = (Event_p->Argument.Receive_Can.DataLength);
    for (int i = 0; i < 64; i++) {
        Action_p->user_array[13 + i] = (Event_p->Argument.Receive_Can.Data[i]);
    }
}
```

When channel 1 receives a message with an ID of A1, the content of the message will be stored in the user-defined space. Pay attention to the store position in the user-defined space, the user has full control over the content of each byte in the this space.

```
else if (Event_p->Argument.Receive_Can.Id == 0xA2) {
    Action_p->user_array[4 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.Id >> 24) & 0xFF;
    Action_p->user_array[5 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.Id >> 16) & 0xFF;
    Action_p->user_array[6 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.Id >> 8) & 0xFF;
    Action_p->user_array[7 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.Id) & 0xFF;
    Action_p->user_array[8 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.FrameType);
    Action_p->user_array[9 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.IdType);
    Action_p->user_array[10 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.CanType);
    Action_p->user_array[11 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.BRS);
    Action_p->user_array[12 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.DataLength);
    for (int i = 0; i < 64; i++) {
        Action_p->user_array[13 + i + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.Data[i]);
    }
}
```

When channel 1 receives a message with an ID of A2, the message will be stored in the user-defined space at a different position than A1. CAN_OCCUPY represents the size of the content occupied by one CAN message.

```
else if (Event_p->Argument.Receive_Can.Id == 0xA3) {
    Counter++;
    Action_p->user_array[0] = (Counter >> 24) & 0xFF;
    Action_p->user_array[1] = (Counter >> 16) & 0xFF;
    Action_p->user_array[2] = (Counter >> 8) & 0xFF;
    Action_p->user_array[3] = Counter & 0xFF;
}
```

When channel 1 receives a message with an ID of A3, the Counter will be incremented by 1, and the value of Counter will be stored in the user-defined space to keep track of the count.

Next, let's look at the value of Counter when the function is first entered.

```
Counter = ((Action_p->user_array[0] << 24) |
          (Action_p->user_array[1] << 16) |
          (Action_p->user_array[2] << 8) |
          (Action_p->user_array[3]));
```

Here, it indicates that each time the function is entered, the value of Counter will be retrieved to obtain the count value.

Next, let's look at the timer handling part.

```
else if (Event_p->Event_Source == TIMER_MS) {       //定时器触发
    if (Counter != 0) {
        TxCAN.Id = ((Action_p->user_array[4] << 24) |
                    (Action_p->user_array[5] << 16) |
                    (Action_p->user_array[6] << 8) |
                    (Action_p->user_array[7]));
        TxCAN.FrameType = Action_p->user_array[8];
        TxCAN.IdType = Action_p->user_array[9];
        TxCAN.CanType = Action_p->user_array[10];
        TxCAN.BRS = Action_p->user_array[11];
        TxCAN.DataLength = Action_p->user_array[12];
        TxCAN.TargetPort = 0x02;
        for (int i = 0; i < 64; i++) {
            TxCAN.Data[i] = Action_p->user_array[13 + i];
        }
        Action_p->Can_Transmit(TxCAN);
        if (Counter > 1) {
            TxCAN.Id = ((Action_p->user_array[4 + CAN_OCCUPY] << 24) |
                        (Action_p->user_array[5 + CAN_OCCUPY] << 16) |
                        (Action_p->user_array[6 + CAN_OCCUPY] << 8) |
                        (Action_p->user_array[7 + CAN_OCCUPY]));
            TxCAN.FrameType = Action_p->user_array[8 + CAN_OCCUPY];
            TxCAN.IdType = Action_p->user_array[9 + CAN_OCCUPY];
            TxCAN.CanType = Action_p->user_array[10 + CAN_OCCUPY];
            TxCAN.BRS = Action_p->user_array[11 + CAN_OCCUPY];
            TxCAN.DataLength = Action_p->user_array[12 + CAN_OCCUPY];
            TxCAN.TargetPort = 0x02;
            for (int i = 0; i < 64; i++) {
                TxCAN.Data[i] = Action_p->user_array[13 + i + CAN_OCCUPY];
            }
            Action_p->Can_Transmit(TxCAN);
        }
        TxCAN.Id = 0x1EAC882E;
        TxCAN.FrameType = 1;
        TxCAN.IdType = 1;
        TxCAN.CanType = 1;
        TxCAN.BRS = 1;
        TxCAN.DataLength = 8;
        TxCAN.TargetPort = 0x02;
        for (int i = 0; i < 8; i++) {
            TxCAN.Data[i] = (Action_p->user_array[13 + i] + Action_p->user_array[13 + i + CAN_OCCUPY]);
        }
        Action_p->Can_Transmit(TxCAN);
    }
}
```

When Counter is not zero, it will periodically send the received A1 and A3 messages through Channel 2 based on the timer trigger frequency. The content of the A3 message will be the sum of A1 and A2. When Counter is greater than 1, it will send the A1 and A3 messages along with the A2 message.

4. Inspection and Maintenance

The main electrical components of the GW2112 products are semiconductor components. Although the equipment have a long service life, they may also accelerate aging and significantly reduce their service life under an incorrect environment. Therefore, during the use of the

equipment, periodic inspection should be carried out to ensure that the use environment maintains the required conditions.

It is recommended to conduct inspections at least once every 6 months to 1 year. Under improper environmental, more frequent inspections should be conducted. As shown in the table below, if you encounter problems during maintenance, please read the following content to find the possible causes of the problem. If the problem still cannot be solved, please contact Shanghai TOSUN Technology Ltd.

Item	Inspection	Standard	Action
Power Supply	Inspect for voltage fluctuations at the power supply end	Power supply port +12V DC	Use a voltage meter to check the power input end. Take necessary actions to keep the voltage fluctuations within the acceptable range.
Surrounding Environment	Check the ambient temperature of the surrounding environment. (Including the internal temperature of enclosed environments)	-40℃~+80℃	Use a thermometer to check the temperature and ensure that the ambient temperature within in the acceptable range.
	Check the ambient humidity. (Including the internal humidity of enclosed environments)	The relative humidity must be within the range of 10% to 90%	Use a hygrometer to check the humidity and ensure that the ambient humidity within the acceptable range.
	Check for the accumulation of dust, powder, salt, and metal shavings	No accumulation	Clean and protect the equipment.
	Check for any	No contact	Clean and protect the equipment if

	contact with water, oil, or chemical sprays on the equipment		necessary.
	Check for the presence of corrosive or flammable gases in the equipment area	No presence	Inspect by the smell, or using a sensor.
	Check for levels of vibration and shock	Vibration and shock are within the acceptable range	Install padding or other shock-absorbing devices if necessary.
	Check for noise sources near the equipment	No significant noise source	Isolate the equipment from noise sources or protect the equipment.
Wiring Installation	Check the crimped connectors in the external wiring	Ensure enough space between the connectors	Visually inspect and adjust if necessary.
	Check for damage in the external wiring	No damage	Visually inspect and replace the wiring 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