USBCAN-II Pro

Industrial USB-CAN Converter

User Manual



Document Version 4.45 (2025/01/08)

English Translation and Editorial Work by Shenzhen MoCAN Electric Co., Ltd.

Revision History

Version	Date	Reason
V1.00	2013/06/16	Document created
V2.01	2013/12/20	Device operating parameters corrected
V3.01	2015/04/22	Some parameters added
V3.50	2016/07/16	OBD II function added
V4.01	2017/01/13	CANopen function added
V4.20	2018/07/18	Document structure adjusted
V4.30	2020/09/18	Product appearance changed
V4.40	2022/06/06	Dual-channel merge and save-data file format added
V4.41	2022/07/06	Document corrected
V4.42	2023/01/05	Software updated
V4.43	2023/03/05	Contact information updated
V4.44	2024/12/16	Dimensions changed
V4.45	2025/01/08	Product appearance updated

Contents

Revision History 2
Contents 3
1. Function Introduction 5
1.1 Overview 5
1.2 Features 5
1.3 Typical Applications 6
2. Device Installation 6
2.1 Device Dimensions 6
2.2 Interface Definition and Functions 6
2.3 Driver and Software Installation 8
2.3.1 Driver and Software Installation 8
2.3.2 Driver and Software Uninstallation 8
3. Device Usage 8
3.1 Connection to PC 8
3.1.1 USB Bus Power Mode 8
3.1.2 External Power Supply Mode 8
3.2 Connection to CAN-bus 9
3.3 CAN-bus Terminating Resistors 9
3.4 System Status Indicators 10
4. Software Usage 11
4.1 Software Startup 11
4.2 Automatic Baud-rate Detection 13
4.3 Data Reception Functions 15
4.3.1 Save Data and Realtime Save Functions 16
4.3.2 Pause Display Function 17
4.3.3 Display Mode 17
4.3.4 Clear Function 18
4.3.5 Filter Settings 18
4.3.6 Error-frame Display Function 19
4.3.7 Advanced Mask Function 20
4.3.8 Dual-channel Merged Reception 21
4.3.9 Data Forwarding Function 22
4.4 Data Transmission Functions 24

4.4.1 Sending Data in Normal Mode 24
4.4.2 Sending Data in List Mode 25
4.4.3 File Sending Function 25
4.4.4 Relay Function 28
4.4.5 Intelligent Offline Mode Function 29
4.5 Bus Diagnosis Function 31
4.6 OBD II Automotive Electronics Interface 32
4.7 CANopen Master Function 35
4.7.1 NMT Commands 35
4.7.2 SYNC Message 36
4.7.3 EMERGENCY Message 37
4.7.4 PDO 37
4.7.5 SDO 38
4.7.6 NODE GUARDING 38
4.7.7 Servo Drive Control 40
4.8 Curver Function 40
4.9 Smart Decode Function 43
4.10 DBC Send Function 44
4.11 DBC Rec Function 45
5. Linux Operating System 46
6. Secondary Development 49
7. Technical Specifications 50
8. Frequently Asked Questions 51
9. Disclaimer 54
Copyright and Translation Notice 54
Product Information and User Responsibility 54
Interpretation 54

1. Function Introduction

1.1 Overview

The USBCAN-II Pro is a high-performance CAN-bus communication interface card integrating two CAN channels. Compatible with the USB 2.0 full-speed specification, the USBCAN-II Pro high-performance CAN interface card allows a PC to connect quickly to a CAN-bus network through the USB port, forming a CAN-bus network control node for data processing and data acquisition in field-bus laboratories, industrial control, smart communities, automotive electronics networks and other CAN-bus applications.

The USBCAN-II Pro high-performance CAN interface card is a powerful tool for CAN-bus product development and CAN-bus data analysis. Compact and plug-and-play, it is also the best choice for portable-system users. The card carries an on-board USB port and integrates an electrical-isolation protection module for the CAN interface, protecting the device from damage caused by transient over-current/over-voltage and enhancing system reliability in harsh environments.

The USBCAN-II Pro supports Windows XP, 7, 8 and 10 (32-bit/64-bit) and various Linux operating systems. A unified application programming interface and complete sample code are provided, including C#, LabVIEW, Python, Qt, VB.NET, VB and VC development examples, to facilitate application development.

The USBCAN-II Pro can be used with our independently developed GCANTools general-purpose test software to send, receive and monitor CAN-bus messages.

1.2 Features

- PC interface complies with the USB 2.0 full-speed specification; compatible with USB 1.1 and USB 3.0
- Integrates two CAN-bus channels using pluggable terminal-block wiring
- The two CAN-bus channels are individually isolated
- Supports CAN 2.0A and CAN 2.0B frame formats; compliant with ISO/DIS 11898
- Supports standard and extended frames, data frames and remote frames
- CAN-bus baud rate freely programmable from 5 kbps to 1 Mbps
- Automatic enhanced CAN-bus baud-rate detection
- Powered from the USB bus, or by an external power supply (9-24 V DC, 130 mA)
- CAN-bus interface electrically isolated; isolation withstand voltage 3000 V DC
- USB interface electrically isolated; isolation withstand voltage 3000 V DC
- ESD immunity: contact discharge ±4 kV, air discharge ±8 kV
- Electrical fast transient burst immunity: CAN interface ±2 kV
- Surge immunity: CAN interface ±4 kV
- Maximum receive data throughput: 14000 fps
- Transmit buffer 400 frames/channel; receive buffer 10000 frames/channel
- CAN-side received-message timestamp accuracy up to 1 us
- Supports Windows XP, 7, 8 and 10 operating systems
- Supports Linux operating systems
- Supports the GCANTools test software
- Supports simulated CANopen master card

- Supports secondary development
- Operating temperature range: -40°C to +85°C

1.3 Typical Applications

- CAN-bus network diagnosis and testing
- Automotive electronics applications
- Power communication networks
- Industrial control equipment
- High-speed, high-volume communication

2. Device Installation

2.1 Device Dimensions

The device outline dimensions are (length, including terminal block) 104.5 mm x (width) 64 mm x (height) 24 mm, as shown in Figure 2.1.



Figure 2.1 Outline dimensions of USBCAN-II Pro

2.2 Interface Definition and Functions

The USBCAN-II Pro interface card integrates one USB port, one 9-24 V DC auxiliary power port and two standard CAN-bus ports. The CAN-bus ports are brought out through one 6-pin pluggable terminal block, which can be used to connect two CAN-bus networks or CAN-bus devices. The positions and definitions of each interface are shown in Figure 2.2, Figure 2.3, Table 2.1 and Table 2.2.



Figure 2.2 Position of the USB and auxiliary power ports

Port	Name	Function
9-24 V DC	9-24 V DC	Auxiliary power; used in offline mode or when power is insufficient; normally not required
USB	USB	Powers the USBCAN and connects to the computer

Table 2.1 USB and auxiliary power port definitions



Figure 2.3 Position of the CAN-bus port

Port	Pin	Function
CAN1	L	CAN1_L signal line (CAN low)
	P	Shield
	H	CAN1_H signal line (CAN high)
CAN2	L	CAN2_L signal line (CAN low)
	P	Shield
	H	CAN2_H signal line (CAN high)
120 Ω	R1	CAN1 channel terminating resistor

	R2	CAN2 channel terminating resistor

Table 2.2 CAN-bus signal assignment of the USBCAN-II Pro interface card

Note: In addition to the internal terminating resistors, two 120-ohm resistors are supplied with the product and may be used as required.

2.3 Driver and Software Installation

Before installing the driver and software, please make sure that the Windows account you are logged into is an administrator account, or that the user account has the relevant permissions to install drivers and software; otherwise the installation may fail. The method for checking Windows account permissions is: Control Panel - User Accounts.

2.3.1 Driver and Software Installation

Users can complete the packaged installation of the driver and software by directly installing the GCANTools software. To install the driver manually, open the "Driver" folder on the CD, select the installation file matching the system (32/64-bit) (DriverSetup.exe / DriverSetup64.exe) and run it.

2.3.2 Driver and Software Uninstallation

After the driver and software are installed correctly, plug the device into the PC's USB port; a new USBCAN device will appear in the PC's Device Manager under the driver name "GCAN USBCAN Device". If there is no yellow exclamation mark or question mark, the device driver is working normally and the USBCAN device is correctly connected to the PC. Users can uninstall the driver by running DriverSetup.exe / DriverSetup64.exe and clicking the "Uninstall" button. The GCANTools software can be uninstalled from "Add/Remove Programs".

3. Device Usage

3.1 Connection to PC

The USB port of the USBCAN-II Pro complies with the USB 2.0 full-speed protocol and can communicate with PCs supporting USB 1.1, USB 2.0 and USB 3.0 standards.

3.1.1 USB Bus Power Mode

USB bus power mode is suitable for most applications, for example when the USBCAN-II Pro is the only device connected to the USB port. Connect the PC and the USBCAN-II Pro directly with the supplied USB cable; the USB cable provides +5 V power to the card. The PWR and SYS indicators light up, indicating that the device is working normally and is waiting for connection.

3.1.2 External Power Supply Mode

External power supply mode is suitable when the PC uses a USB hub, or when several USB devices are already connected so that the USB port cannot supply enough current to the USBCAN-II Pro. Connect an external power supply (+9 to 24 V DC, 200 mA) to the +9-24 V DC power socket of the USBCAN-II Pro; the PWR and SYS indicators light up. Then connect the PC and the USBCAN-II Pro with the supplied USB cable, and the card works normally.

3.2 Connection to CAN-bus

To connect the USBCAN-II Pro to a CAN bus, simply connect CAN_H to CAN_H and CAN_L to CAN_L to establish communication. The CAN-bus network uses a linear topology; the two most distant ends of the bus need 120-ohm terminating resistors. If there are more than two nodes, the intermediate nodes do not need 120-ohm terminating resistors. Branch connections should not exceed 3 meters. The CAN-bus connection is shown in Figure 3.1.

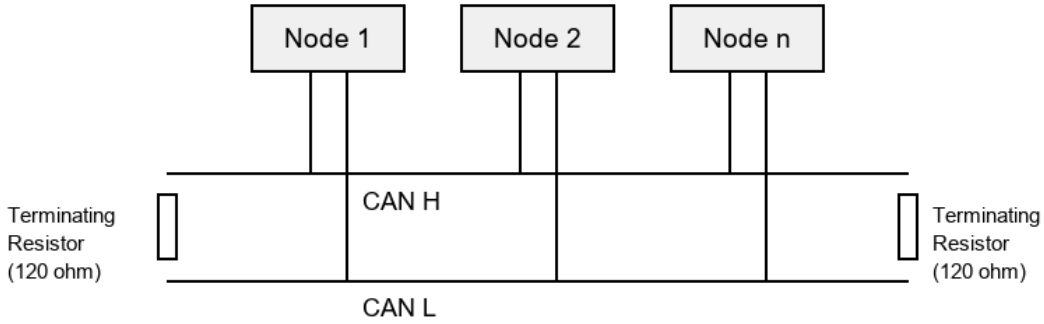


Figure 3.1 Topology of a CAN-bus network

Note: CAN-bus cables may be ordinary twisted pair or shielded twisted pair. The theoretical maximum communication distance depends mainly on the bus baud rate; see Table 3.1 for the relationship between maximum bus length and baud rate. If the communication distance exceeds 1 km, the wire cross-section should be larger than Φ1.0 mm²; the specific specification should be chosen according to the distance, generally increasing with distance.

Baud Rate	Bus Length
1 Mbit/s	25 m
500 kbit/s	100 m
250 kbit/s	250 m
125 kbit/s	500 m
50 kbit/s	1.0 km
20 kbit/s	2.5 km
10 kbit/s	5.0 km
5 kbit/s	13 km

Table 3.1 Baud rate versus maximum bus length

3.3 CAN-bus Terminating Resistors

To enhance the reliability of CAN communication and eliminate signal-reflection interference at the bus ends, terminating matching resistors are usually added at the two most distant ends of the CAN-bus network, as shown in Figure 3.2. The value of the terminating resistor is determined by the characteristic impedance of the transmission cable. For example, if the characteristic impedance of the twisted pair is 120 Ω, then a 120-ohm terminating resistor should be integrated at both ends of the bus.

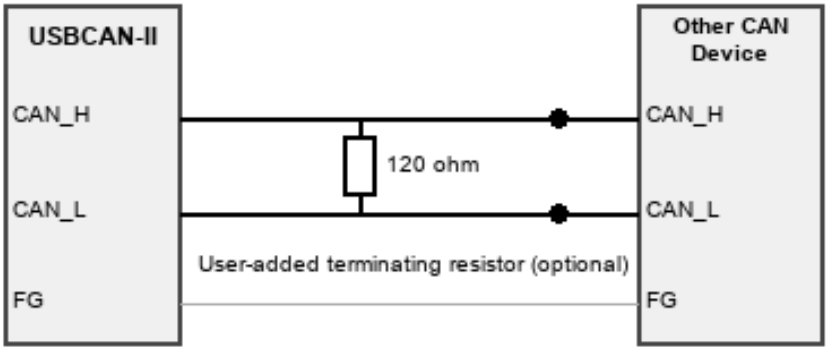


Figure 3.2 USBCAN-II Pro connected to other CAN node devices

Note: The USBCAN-II Pro has 120-ohm terminating resistors integrated internally; a DIP switch selects whether the resistors are connected to the bus. The DIP switch is next to the 6-pin terminal block; R1 and R2 correspond to the terminating resistors of CAN1 and CAN2 respectively. Switch to ON to enable the resistor.

3.4 System Status Indicators

The USBCAN-II Pro has one PWR indicator, one SYS indicator, one CAN1 indicator and one CAN2 indicator to show the operating status of the device. The functions of these four indicators are listed in Table 3.2, and the CAN-bus status corresponding to the various indicator states is shown in Table 3.3.

Indicator	Color	Indication
PWR	Green	Power indication
SYS	Green	System indication
CAN1	Green	CAN1 signal indication
CAN2	Green	CAN2 signal indication

Table 3.2 USBCAN-II Pro interface card indicators

After the USBCAN-II Pro is powered on, all four indicators light up simultaneously; then PWR and SYS stay on while CAN1 and CAN2 go off, indicating that the device is powered and the system has completed initialization. Otherwise, there is a system power fault or another fault, and you should contact our customer service.

After the USB interface is connected correctly, when PC software accesses the USBCAN device the USB signal indicator SYS will flash. At this time, when CAN1 or CAN2 sends or receives data, the corresponding CAN1 or CAN2 indicator will flash. If SYS flashes but CAN1 or CAN2 does not light up, there is no data on the CAN channel; check the wiring, communication baud rate and matching resistors.

Indicator	State	Indication
PWR	On	Power supply normal
	Off	Power supply fault
SYS	Steady on	Device initialization passed, standby state
	Off	Device initialization failed
	Flashing	PC software is accessing the device

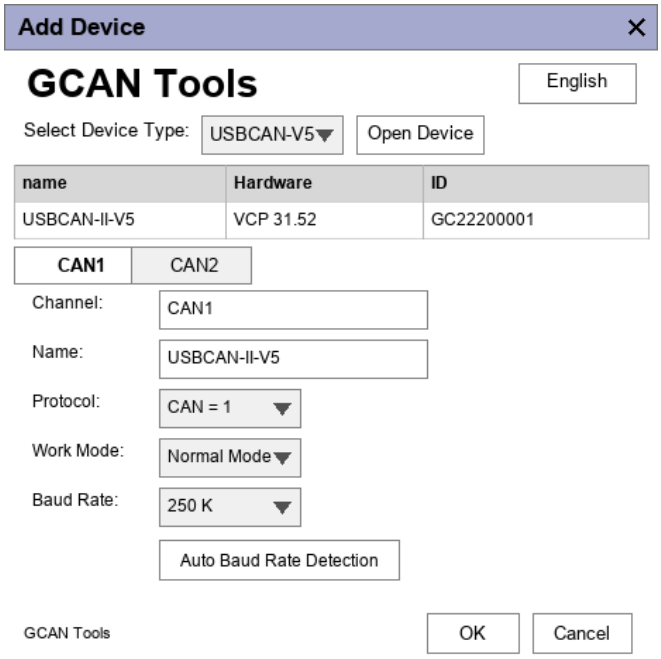
CAN1, CAN2	Off	No data transmission on CAN channel
	Flashing	Data transmission on the corresponding CAN channel

Table 3.3 USBCAN-II Pro indicator states

4. Software Usage

4.1 Software Startup

After installing the Windows driver, connect the CAN analyzer hardware and open the software; the setting interface shown below appears.



1) "Select Device Type": you need to select the version matching the hardware. V405/502 versions should select the V5 type; refer to the back of the device enclosure for the specific version and model. Do not select other device types. After selecting the correct device type, click Open Device to see the device information. "hardware" is our customized hardware version number and has no meaning for the user; "ID" is the hardware serial number (SN).

2) "Open Device" button: click this button to access the USBCAN device. If "USB device open error!" is displayed, check that the device type selected in step 1 is correct and that the driver in Device Manager is installed correctly.

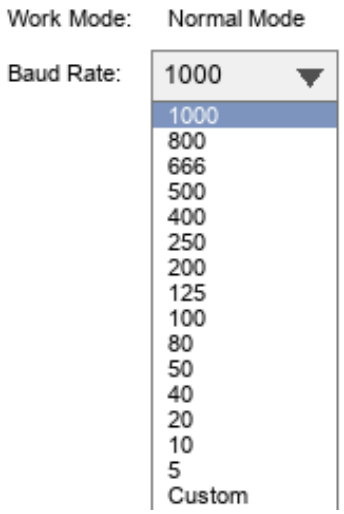
3) Device display window: this window shows the hardware number and serial number of the device.

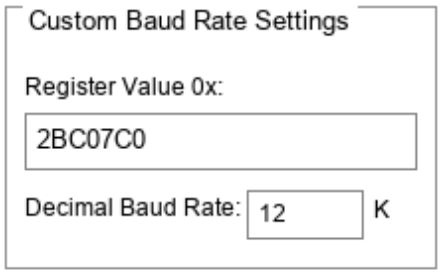
4) Channel selection area: here you can switch the setting window of the CAN channels.

5) Working mode: here you can select normal mode, listen-only mode, or self-transmit/self-receive mode. For normal use, select the default normal mode.

6) Baud rate selection: here you can select the baud rate. The baud rate is critical for CAN-bus communication; before communicating you need to determine the baud rate of the target device or target bus.

After selecting the device, you can set the working mode and baud rate below. If you have a dual-channel device, you can set CAN1 and CAN2 separately using the tabs. Setting the baud rate is particularly important; here the baud rate cannot be entered manually but must be selected from a drop-down menu. Our products support the standard baud rates shown below.





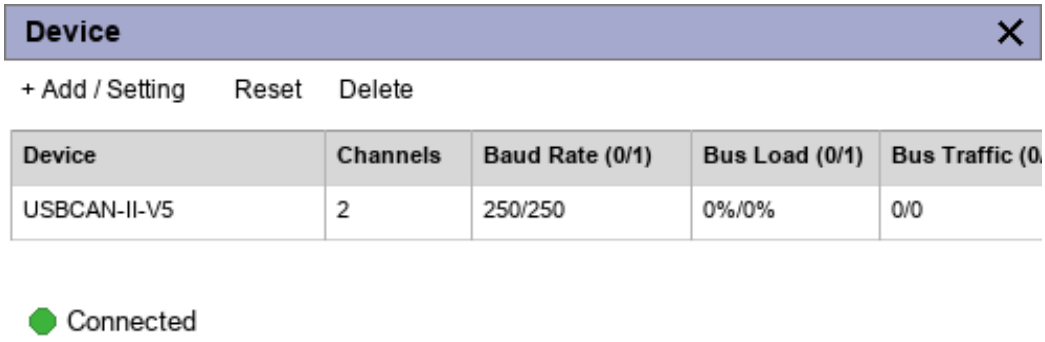
If you use a special baud rate, click the Custom button; you then need to enter a hexadecimal code. The table below lists some special baud-rate values. If you need other baud rates, please contact us and we will help you calculate the register setting value.

Baud Rate	Parameter
83.3 kbps	27C017
66.6 kbps	2DC013
60 kbps	2BC018
35 kbps	2EC023
33.3 kbps	2EC025
25 kbps	2FC02F
13.3 kbps	16C0B3
12 kbps	2BC07C

7) Automatic baud-rate detection button: if you do not know the baud rate of the target bus or device, you can select "Auto Baud Rate Detection". Conditions for successful detection: the device under test is powered on and its CAN side is active.

Please note that setting the baud rate is especially important. Many customers report that after connecting the device there is no data or there are bus errors, when in fact they clicked OK to open the device without setting the baud rate. Whether you use our device as a master or a slave, as long as you connect it to the CAN bus, you must set its baud rate to match that of the target device for it to work properly.

8) OK button: after selecting the baud rate, click OK to enter the software. If the hardware starts correctly, the SYS indicator changes from steady on to flashing (flashing twice continuously). After a successful connection the interface shown below appears.



4.2 Automatic Baud-rate Detection

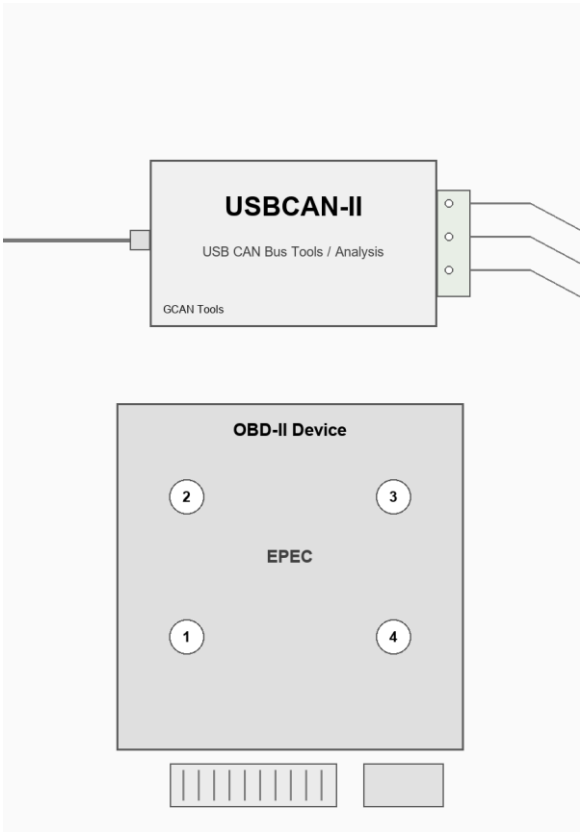
If you do not know the baud rate of the target device, you can use the automatic baud-rate detection function. Two modes are available: "Normal Silent Mode Detection" and "Enhanced Mode Detection". Normal Silent Mode Detection requires at least two devices under test on the CAN bus that can communicate with each other normally, and requires active CAN data on the bus. Enhanced Mode Detection requires the device under test to be powered on and its CAN channel to work normally, but does not require the device to actively send CAN data or require active CAN data on the bus. Before auto-detection, make sure the wiring is correct and two 120-ohm resistors are connected to the bus. You can use a multimeter to measure the resistance between CANL and CANH without powering the devices on the CAN bus; it should be about 60 ohms.

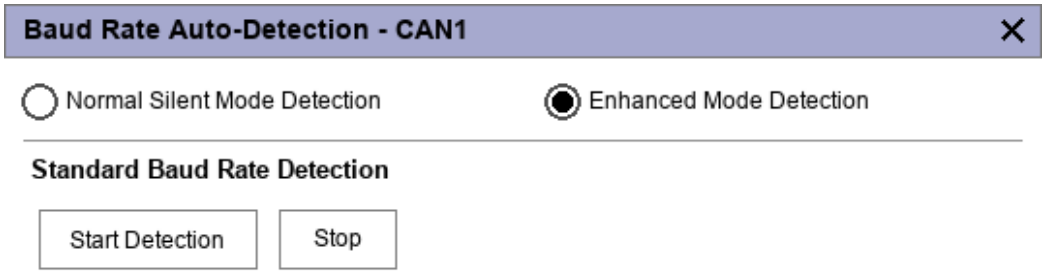
Please note that for automotive CAN-bus systems, select "Normal Silent Mode Detection" to detect the baud rate. Using "Enhanced Mode Detection" may cause instrument errors in the vehicle.

Here a Guangcheng USBCAN-II Pro dual-channel device (pink) is used to detect the unknown baud rate of another CAN node. The wiring method is CANH to CANH and CANL to CANL. The DIP switch for the terminating resistor of channel 1 of the USBCAN-II Pro analyzer is switched off, and a 120-ohm resistor is added between CANL and CANH of the other CAN node. After setting up the environment, open the GCANTools software.

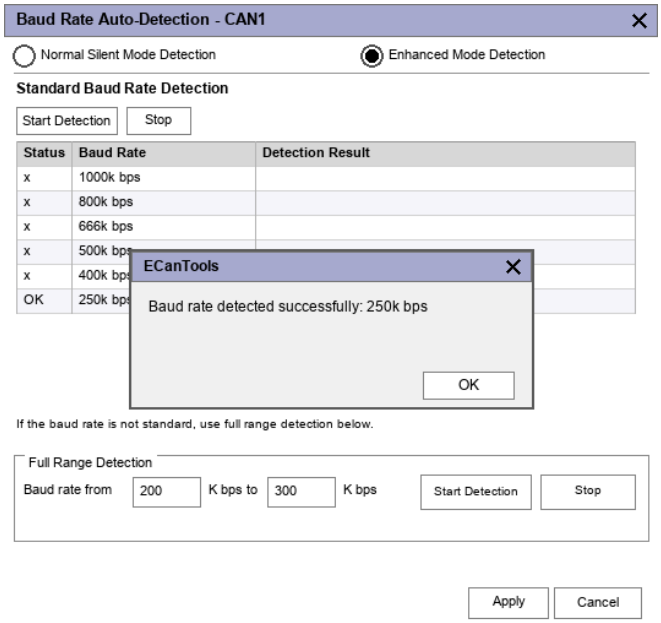
The automatic baud-rate detection function can be divided into two modes according to the search range: A. Standard baud-rate detection (identifies standard CAN baud rates one by one), and B. Full-range baud-rate detection (manually enter a detection range; the software comprehensively matches baud rates within the range). For example, if the baud rate of a CAN-bus system is unknown and standard baud-rate detection shows "No suitable baud rate found", you can set the baud rate from 5 kbps to 1000 kbps in the "Full-Range Detection" area to detect the required baud rate.

A. Standard baud-rate detection screenshot:





B. Full-range baud-rate detection screenshot:

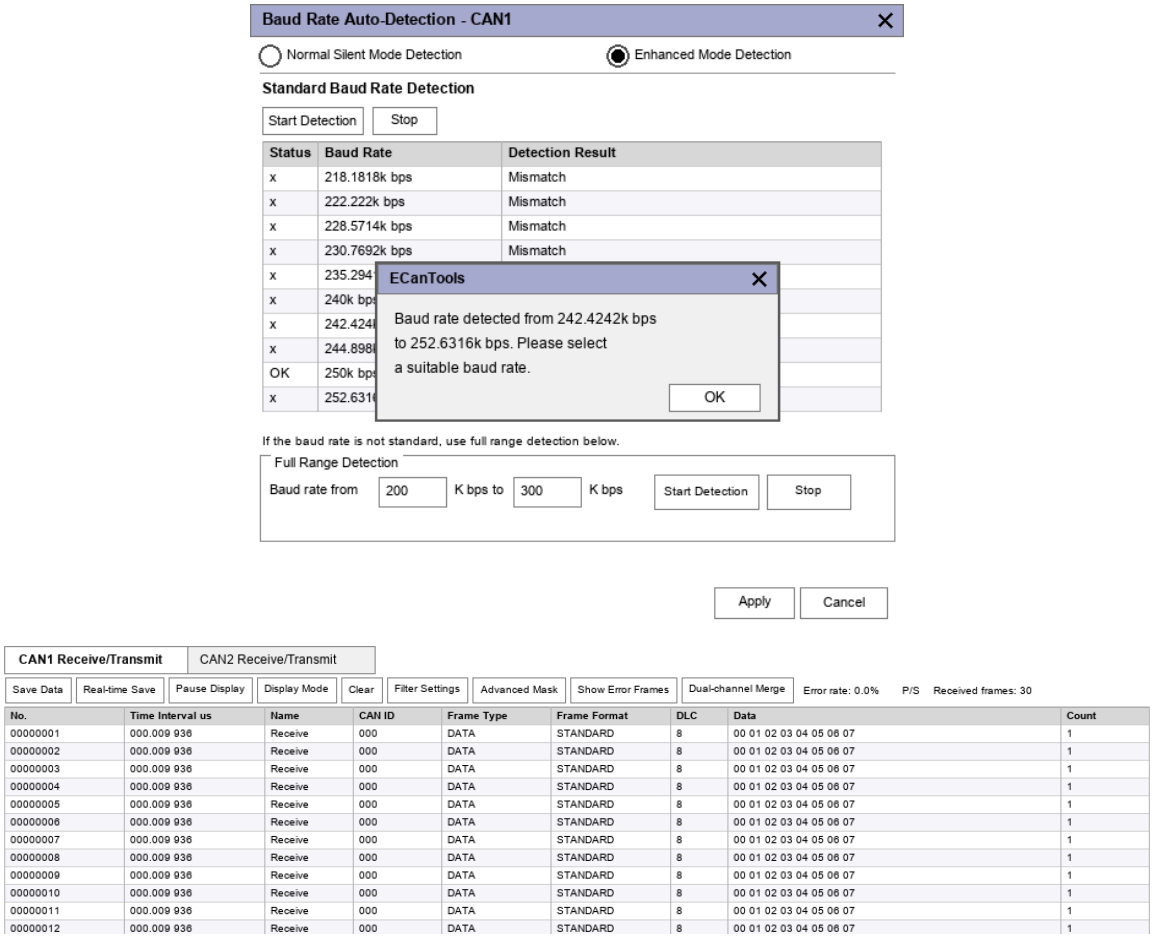




If standard baud-rate detection cannot find the baud rate of an unknown node, you can use full-range baud-rate detection for further identification. This function helps engineers correct baud-rate deviations in newly developed devices.

4.3 Data Reception Functions

After the device parameters are set, the software enters the working state. If there is data on the bus, the receive-data window will display the data. The receive window is shown below.

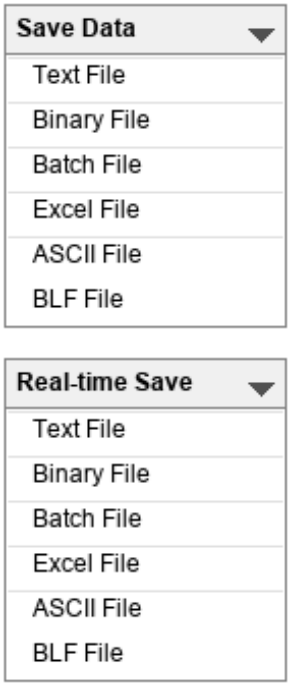


4.3.1 Save Data and Realtime Save Functions

Users can save all data in the current send/receive list to local storage; the save formats are listed below.

File Type	File Format	Editor
Text file	.txt	Notepad
Binary file	.dat	Notepad
Batch file	.can	Notepad
Excel file	.csv	WPS Excel
ASC file	.ASC	CANoe
BLF file	.blf	CANoe

Text files are convenient for data storage and later analysis. Batch files can be modified with Notepad and sent back to the bus. For a description of the batch-file format, see section 4.4.3.

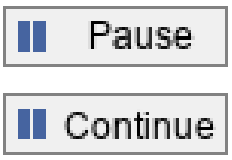


Click "Realtime Save" on the toolbar, set the text type and file name for realtime saving, and the realtime save function starts (i.e., save point A is set). Clicking again (i.e., setting save point B) stops saving and writes all data from the start (A) to the end (B) into the save file.

Please note that the system does not write data in real time. During realtime saving, the save file contains no data.

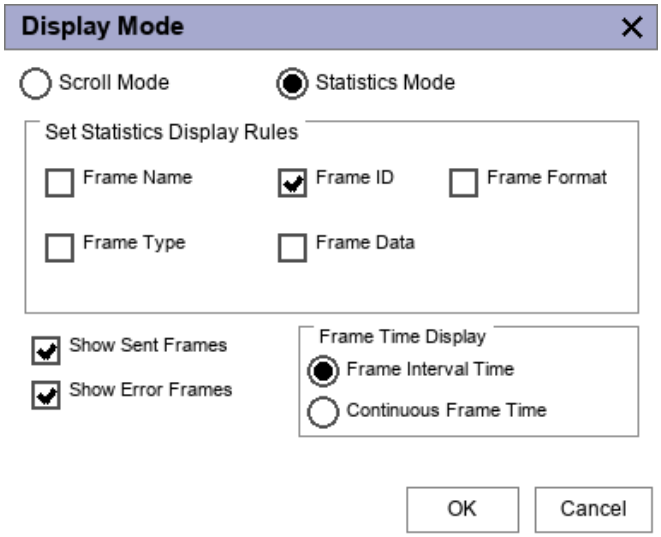
4.3.2 Pause Display Function

You can click Pause to freeze the scrolling data window. While paused, the device and software still receive data normally; only the data window does not refresh. Click Continue Display to resume scrolling.



4.3.3 Display Mode

Display modes include scrolling mode and statistics mode. Scrolling display continuously scrolls received data downwards in the receive list, so the current window shows the latest data. The statistics list displays data grouped according to set rules; for example, packets with the same ID can be displayed together with a count of the number of packets. Statistics mode makes it convenient for engineers to capture newly generated or changed data on the bus.



No.	Frame Interval us	Name	CAN ID	Frame Type	Frame Format	DLC	Data	Count
00000000	0.000 937	Receive	000	DATA	STANDARD	8	00 01 02 03 04 05 06 07	38

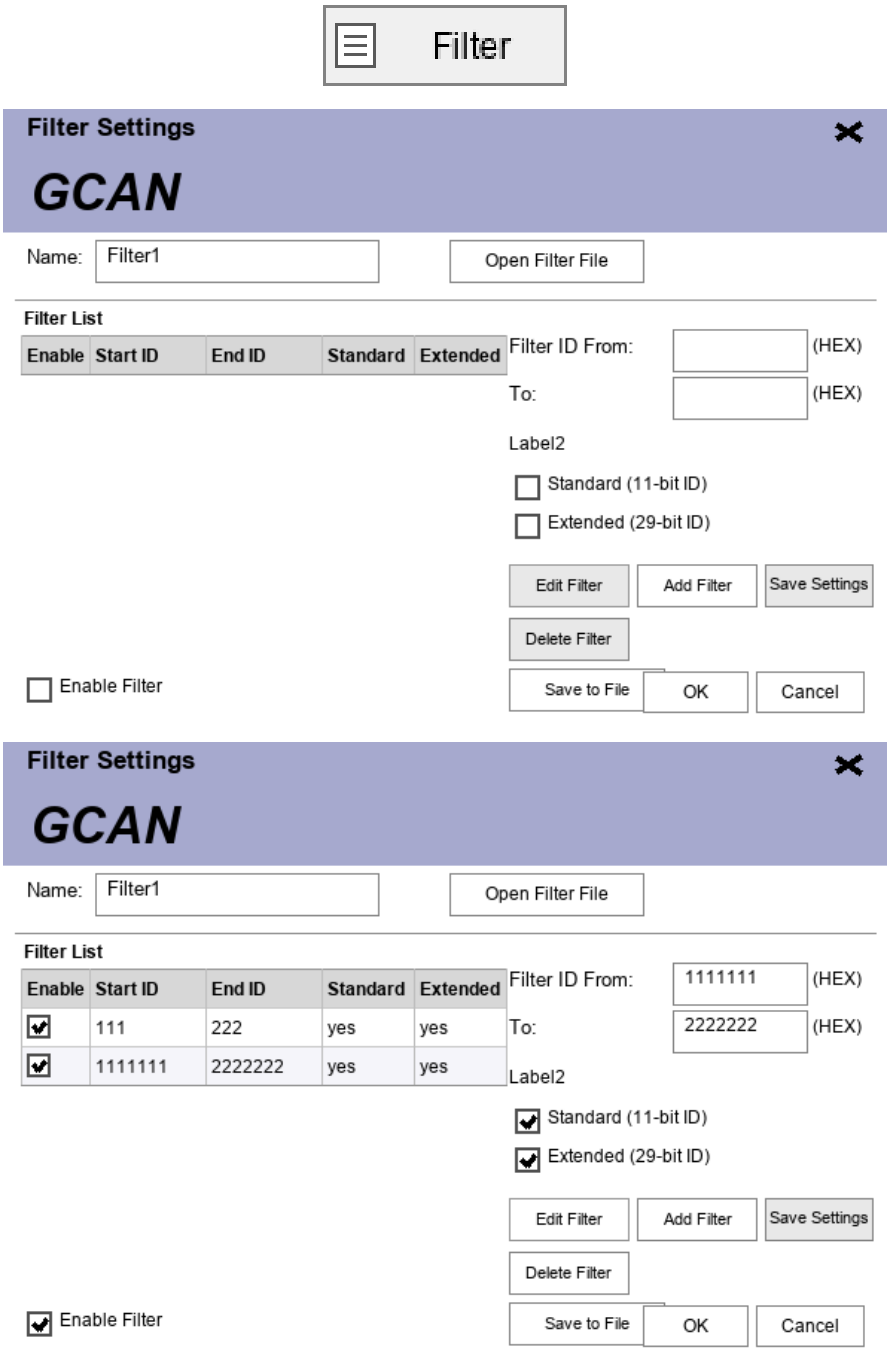
4.3.4 Clear Function

You can clear the data in the receive/send windows and the data in the buffer.



4.3.5 Filter Settings

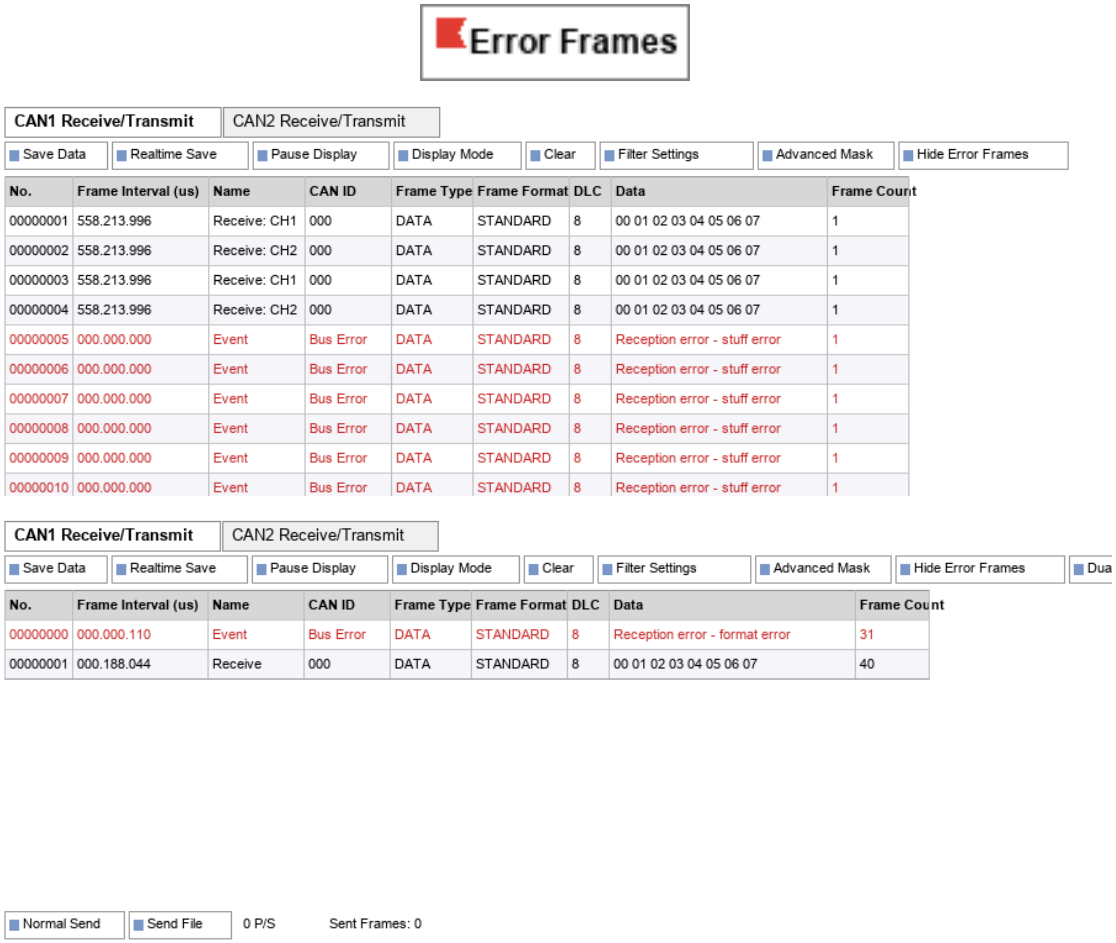
Receive filter settings can set a filter ID or ID range. If a filter is set, the software displays only the set filter ID (range); IDs outside the filter range are filtered out. Click "Edit Filter" to open the filter setting window, where you can set the filter range.



Click "Add Filter" to start editing. Enter the ID or ID range you want to filter and click "Save Settings" to add the filter content to the left side. Several filter entries can be added to the left list; check the corresponding filter segment to enable it. The software can enable multiple filters at the same time. Please note that you need to check "Enable Filter" for the filters to take effect. Filter files can be saved and loaded through "Open Filter File".

4.3.6 Error-frame Display Function

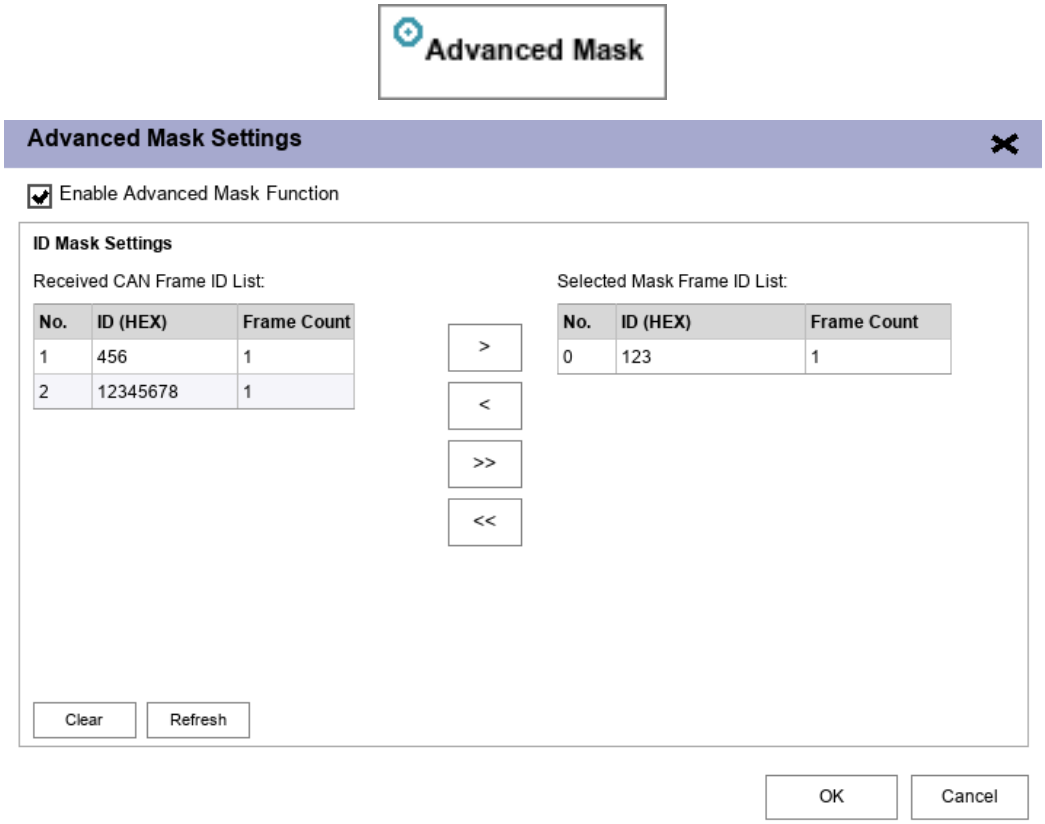
The software can capture error frames on the bus; when an error frame is received, it is displayed in red in the receive list.



Error frames can also be displayed merged by setting statistics mode.

4.3.7 Advanced Mask Function

The GCANTools software can mask specified received CAN frame data. It groups received data by frame ID and displays it merged in the "to-be-masked" list on the left. From the left list you can select the frame IDs to mask and click the right arrow to add the selected frame IDs to the mask list. The software automatically masks all data starting with those frame IDs in the right list. Multiple mask frame IDs can be set; remember to check Enable Filter.

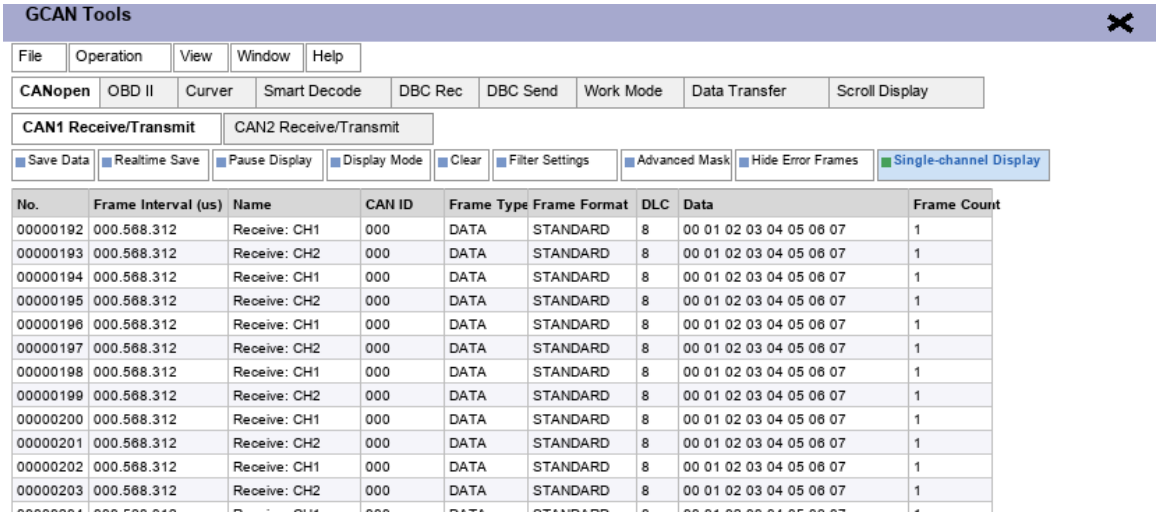


4.3.8 Dual-channel Merged Reception

For dual-channel devices, when CAN1 and CAN2 receive data simultaneously, click "Dual-channel Merge" to receive data from both channels in the current interface, distinguished by name. When saving data, the merged dual-channel data is saved in the same file; when sending files, both channels send simultaneously according to the number of channels in the saved file (see section 4.4.3). If "Dual-channel Merge" is not clicked, each interface normally receives its own channel's data.

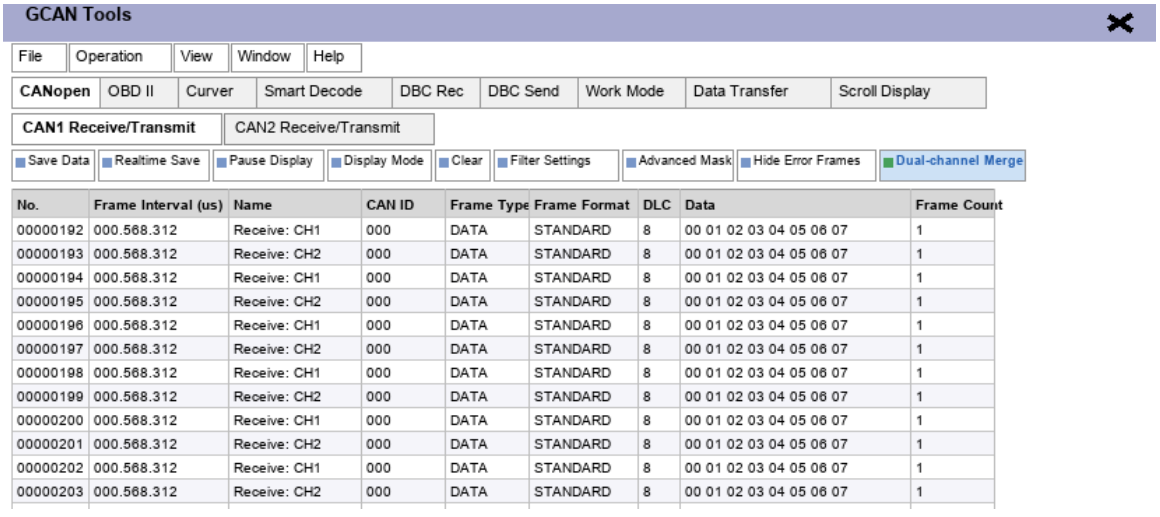






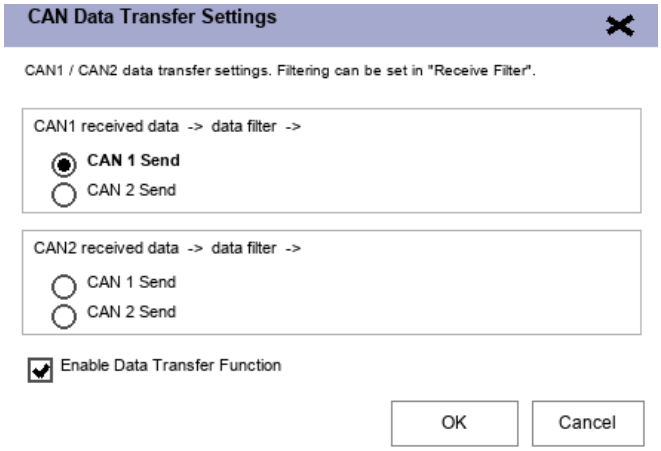
4.3.9 Data Forwarding Function

1) Data received by CAN1 - sent by CAN1: data received by CAN1 is immediately sent out from CAN1, equivalent to a data reflection function.





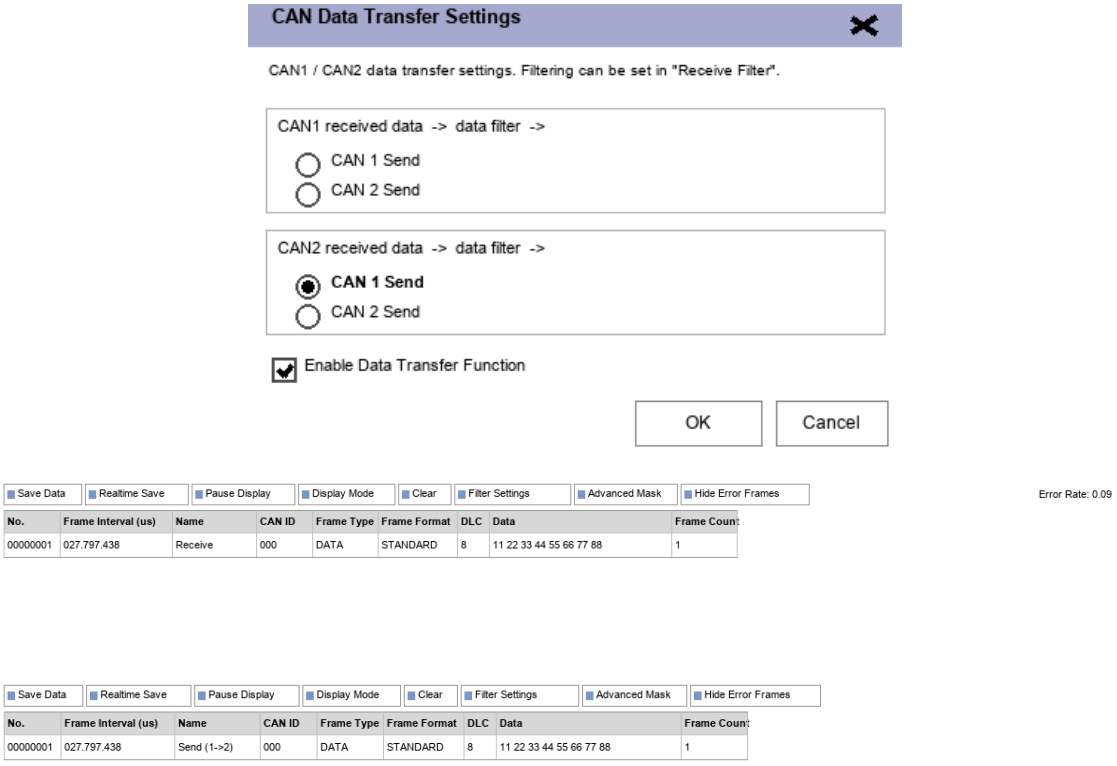
2) Data received by CAN1 - sent by CAN2: after data is received on the CAN1 channel it is filtered and sent out from the CAN2 channel; CAN1 and CAN2 can be set to different baud rates.



Save Data | Realtime Save | Pause Display | Display Mode | Clear | Filter Settings | Advanced Mask | Hide

No.	Frame Interval (us)	Name	CAN ID	Frame Type	Frame Format	DLC	Data	Frame Co
00000001	009.358.782	Receive	000	DATA	STANDARD	8	00 01 02 03 04 05 06 07	1
00000002	009.622.762	Send (1->1)	000	DATA	STANDARD	8	00 01 02 03 04 05 06 07	1
00000003	007.678.775	Receive	000	DATA	STANDARD	8	11 22 33 44 55 66 77 88	1
00000004	007.358.775	Send (1->1)	000	DATA	STANDARD	8	11 22 33 44 55 66 77 88	1

3) Data received by CAN2 - sent by CAN1: after data is received on the CAN2 channel it is filtered and sent out from the CAN1 channel; CAN1 and CAN2 can be set to different baud rates.



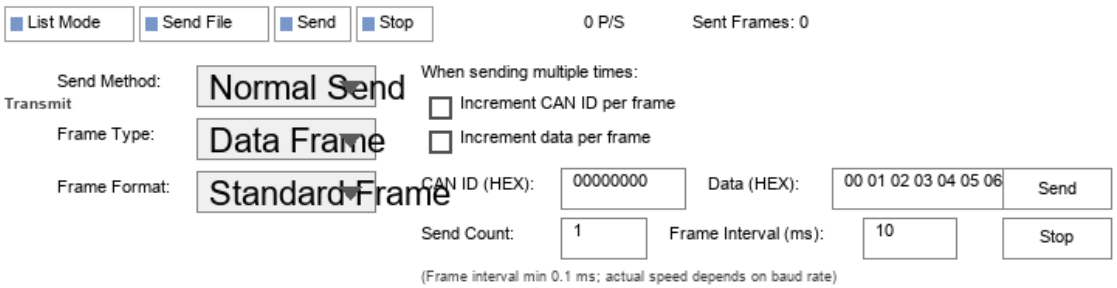
4) Data received by CAN2 - sent by CAN2: data received by CAN2 is immediately sent out from CAN2, equivalent to a data reflection function.

Note: single-channel devices can only implement function 1). Data forwarding can be used together with the filter function; after setting a filter on the receiving channel, only filtered data is forwarded. See section 4.3.5 for filter settings.

4.4 Data Transmission Functions

4.4.1 Sending Data in Normal Mode

Normal mode allows you to edit the frame data to be sent very intuitively and to set special functions such as cyclic sending.

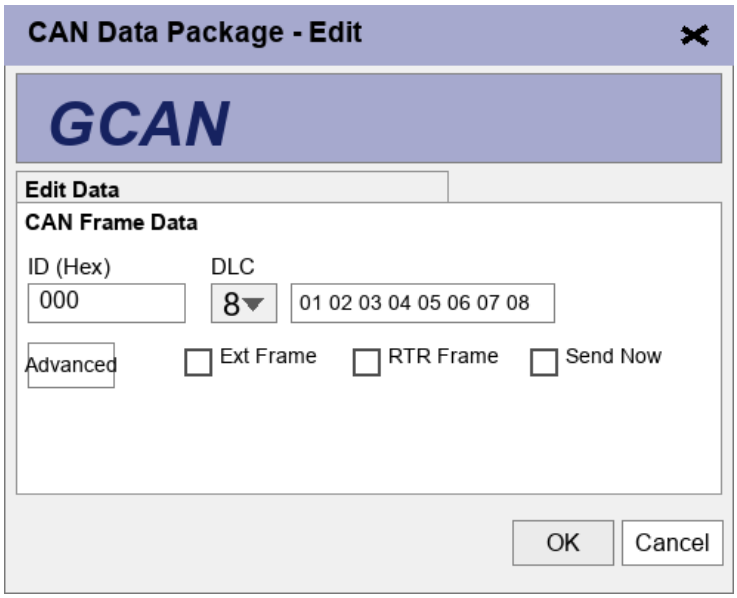


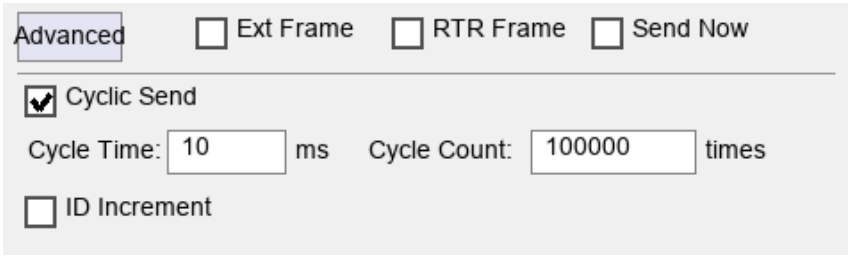
Normal mode is simple and frame editing is very intuitive. Note that when entering data, a space must be entered between each byte; otherwise the "Data format error!" message appears.

4.4.2 Sending Data in List Mode

After clicking List Mode, the original position switches to Normal Mode. Click New Frame to edit the data frame you want to send. In the edit-data interface you can set the frame ID, frame type, frame format, frame length and so on; in Advanced Options you can set the cyclic sending method, cyclic sending interval, number of cyclic sends, ID increment, and other options.



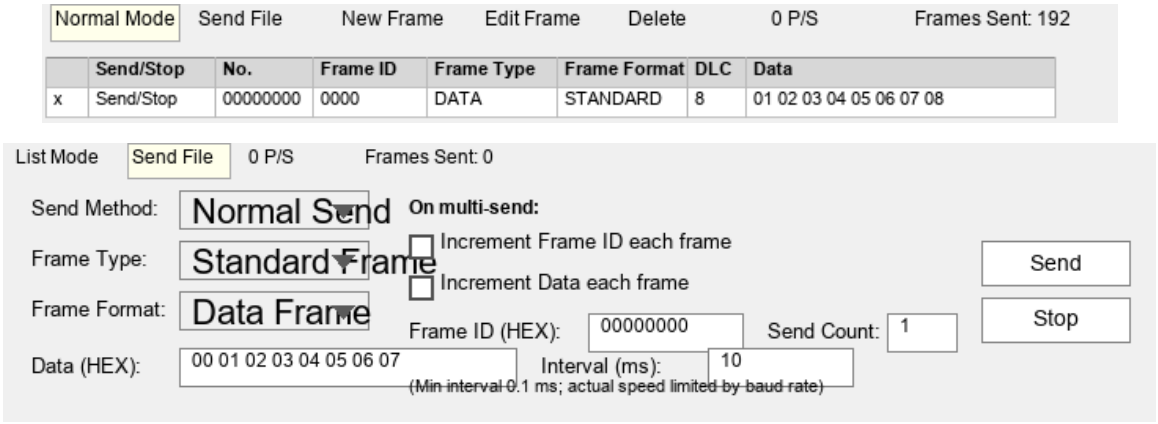




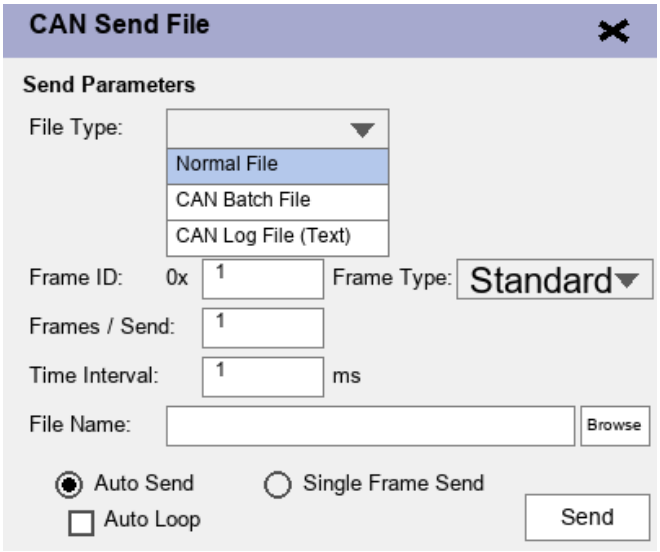
Once set, the CAN message can be added to the send queue: click the checkbox to the left of "send" to start or stop sending. When sending finishes, the checkbox resets automatically, and the number of successfully sent packets is shown in "Frames Sent".

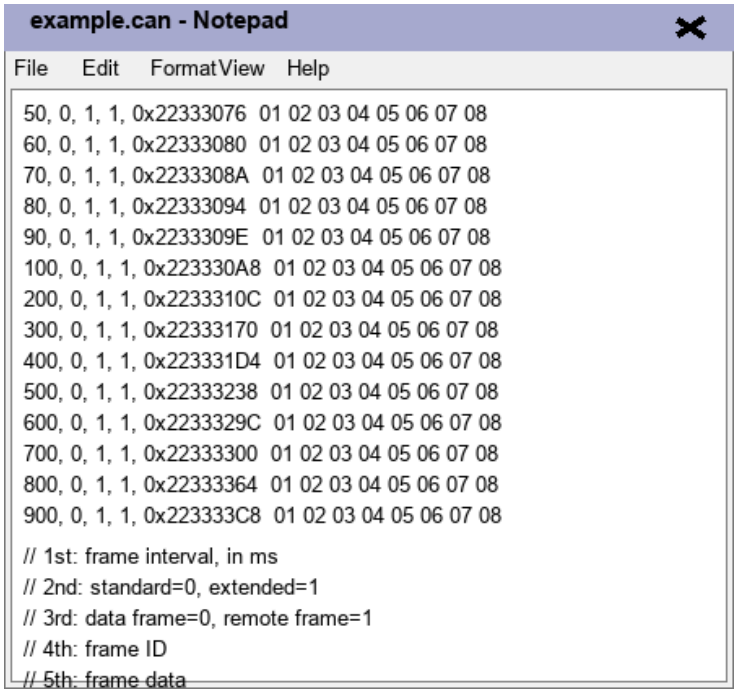
4.4.3 File Sending Function

Select "Send File" to open the send-file dialog. The file type can be a normal file or a batch file (.can). Normal files are used to program CAN-bus devices and require you to develop the programming software yourself. The following focuses on batch-file sending.



The GCANTools software can save received data as a batch file (.can). The batch file can be opened with Notepad, where you will see the saved frame data content and format. The "example.can" file shown below can be found in the installation directory of the GCANTools software.

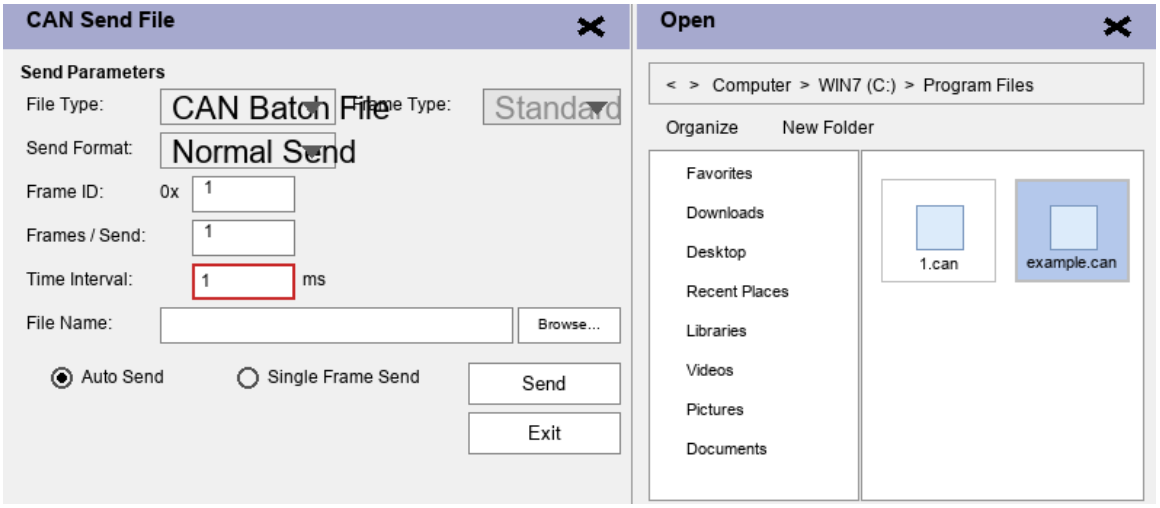




After opening the batch file (.can), you can clearly see the saved data parameters. Users can modify and add to these parameters; the definition of each position is given in the figure above. Note that no non-standard data of any kind may appear in the batch file; the comments added in the figure above are for explanation only and should not be added in actual editing.

Field	1st	2nd	3rd	4th	5th
Description	Frame interval (ms)	Std/Ext	Data/Remote	Frame ID	Frame data
Example	50,	1,	1,	0D223344,	01 02 03 04 05 06 07 08

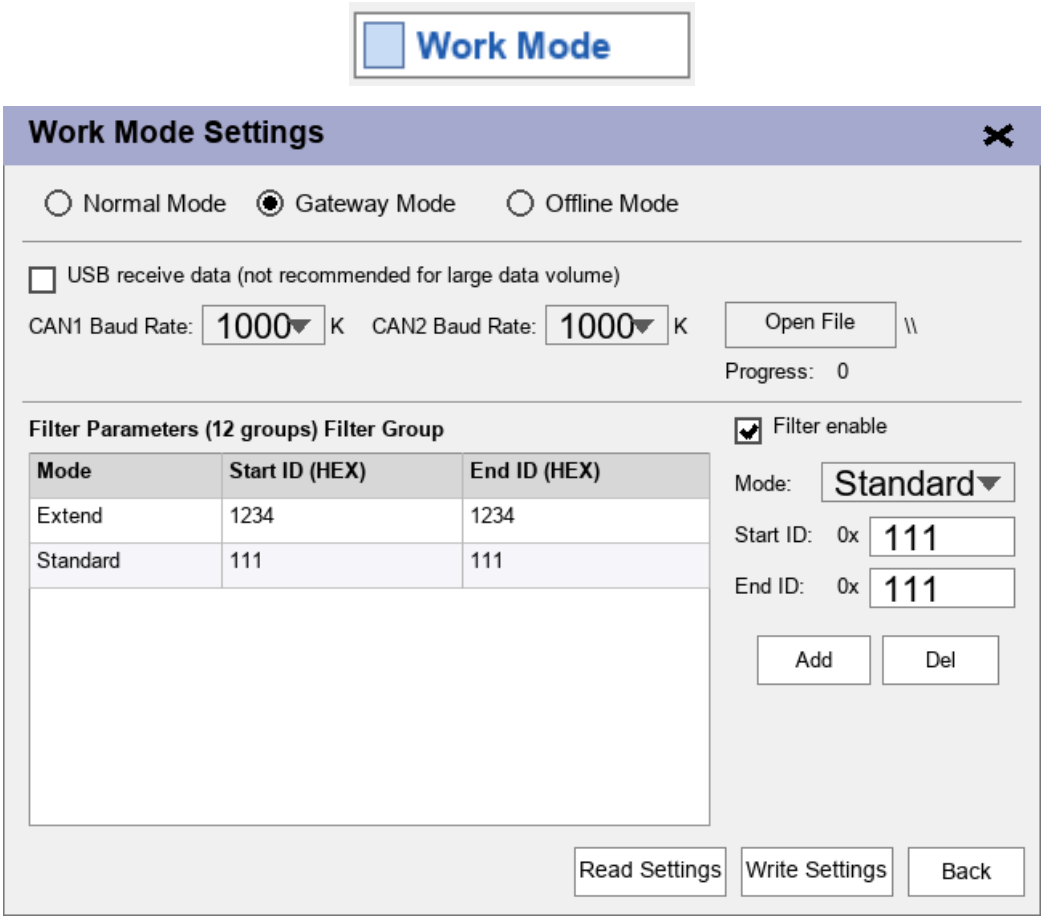
Please note that you may find a frame interval of 0 in a batch file (.can). In this case GCANTools sends at the minimum interval for that baud rate, and multiple frames will not be sent at the same time.



Afterwards select the saved batch file to send the file. The time interval defaults to 1 and should not be changed; if you need to set a time interval, change it in the batch file.

4.4.4 Relay Function

The working mode of the device can be selected above the CAN1/2 channel tabs; both channels of the dual-channel device support CAN relay mode. By setting the baud rates of CAN1 and CAN2 separately, data can be relayed between two buses with different baud rates. After clicking "Write Settings", close the software and power the device on again to implement offline relay. The hardware provides an external 5 V DC standard industrial field power plug for convenient industrial use.

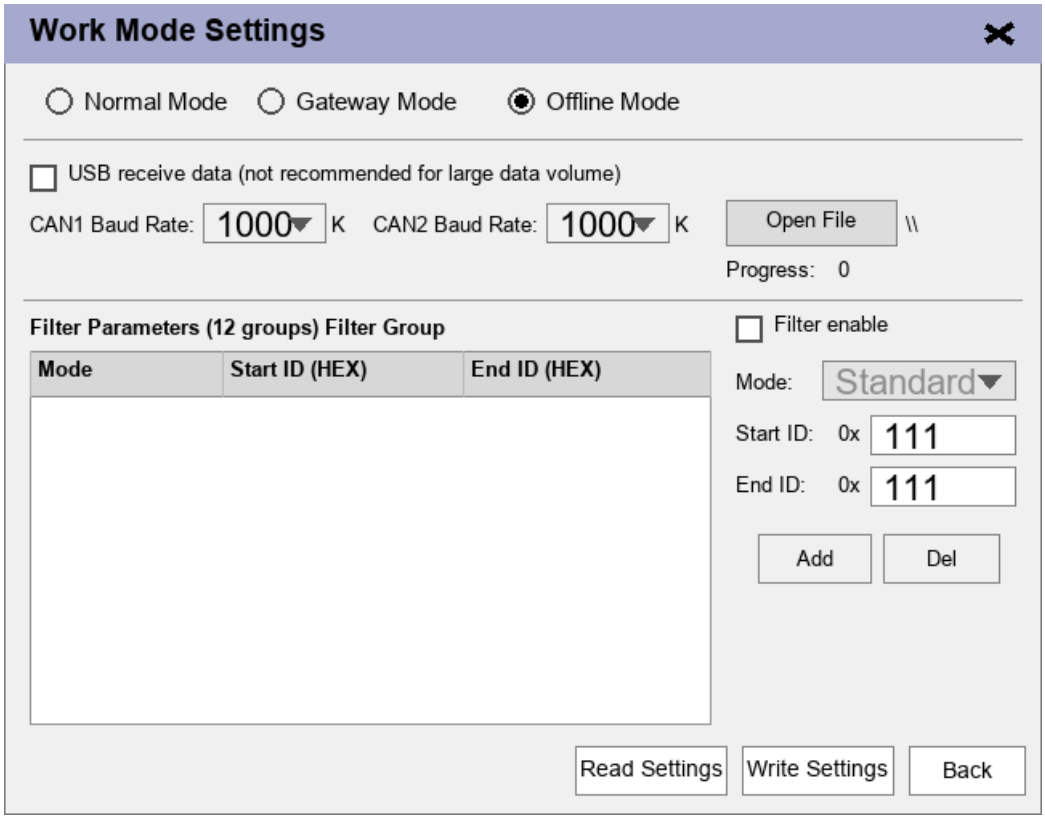


After checking "USB Receive Data", you can use the GCANTools software to monitor the relayed data of both channels and display the relay data flow. This function is useful for capturing CAN-bus data packets, especially for capturing handshake protocols.

The relay function can enable a filter to implement relay filtering. The filter is set as a CAN-ID range, with up to 12 entries. The filter must be enabled by checking Filter Enable, and takes effect only after the device is powered on again.

4.4.5 Intelligent Offline Mode Function

Users can program the device offline by loading a TXT document. After the offline program is written, the device can intelligently send data and other functions; after writing, no computer connection is needed and it runs as soon as it is powered on. For the specific TXT document writing rules and examples, see the "Offline Mode User Guide" on the supplied CD.



No.	Description
1	Waiting time (unit: ms)
2	Mode: 1=CAN1 transmit/receive; 2=CAN2 transmit/receive; 3=CAN1 receives then CAN2 forwards; 4=CAN2 receives then CAN1 forwards
3	Trigger ID: 40000000 means no trigger; 11111111 means forward only data starting with this frame ID
4	Replace mask: hexadecimal converted to binary; 0 = do not replace, 1 = replace
5	Send ID: if a trigger ID is set, this is used only when triggered; otherwise it is the normal frame ID to send
6	Frame format/type: 0=standard data frame, 1=standard remote frame, 2=extended data frame, 3=extended remote frame
7	Data length
8	Frame data: spaces must be placed between bytes
9	Number of sends: 0 means unlimited
10	Send interval (unit: ms)

Example 1: 10,1,40000000,0000,144,0,8,01 02 03 04 05 06 07 08,100,500

This data means: after power-on wait 10 ms, initialize CAN1 to transmit and receive data, no trigger set, no ID or data replacement, send ID 144, standard frame, data length 8 bytes, send data 01 02 ... 08, send 100 times, with an interval of 500 ms between sends.

Example 2: 10,3,11111111,0020,123,0,8,01 02 03 04 05 06 07 08,1,500

This data means: after power-on wait 10 ms, initialize so that CAN1 receives data and CAN2 forwards it; trigger ID 11111111 means forward only data starting with this ID; the replace

mask 0020 replaces the third byte of data with 00; send ID 123 (meaningless if no trigger is set); standard frame; data length 8 bytes; send once; interval 500 ms.

Example 3: 10,3,00000222,0000,123,0,8,01 02 03 04 05 06 07 08,1,500

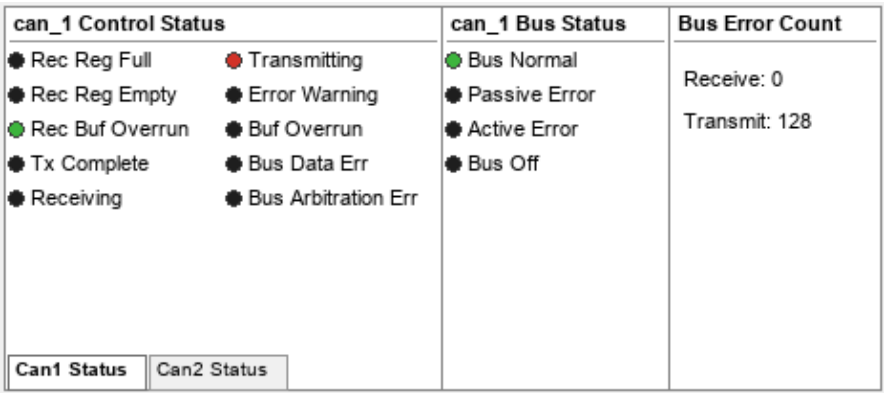
This data means: after power-on wait 10 ms, initialize so that CAN1 receives data and CAN2 forwards it; trigger ID 00000222 means forward only data starting with this ID; the replace mask 0000 means no data is replaced; send ID 123; standard frame; send once; interval 500 ms. This data realizes unidirectional relay: all data on CAN1 starting with frame ID 00000222 is forwarded unchanged to CAN2.

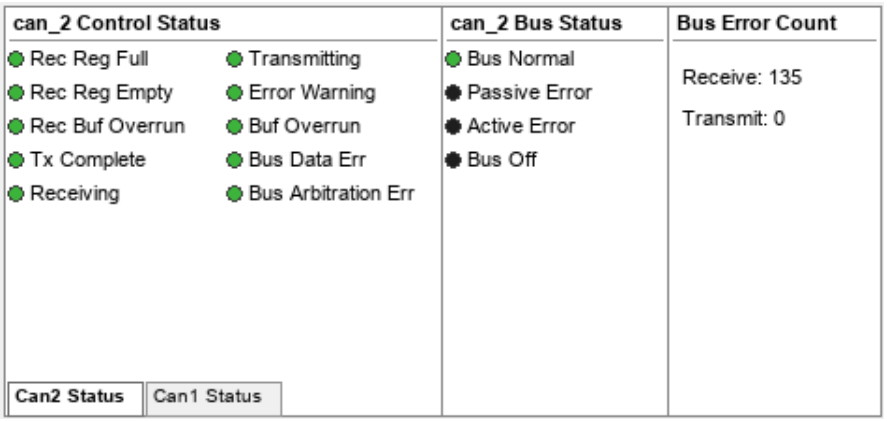
4.5 Bus Diagnosis Function

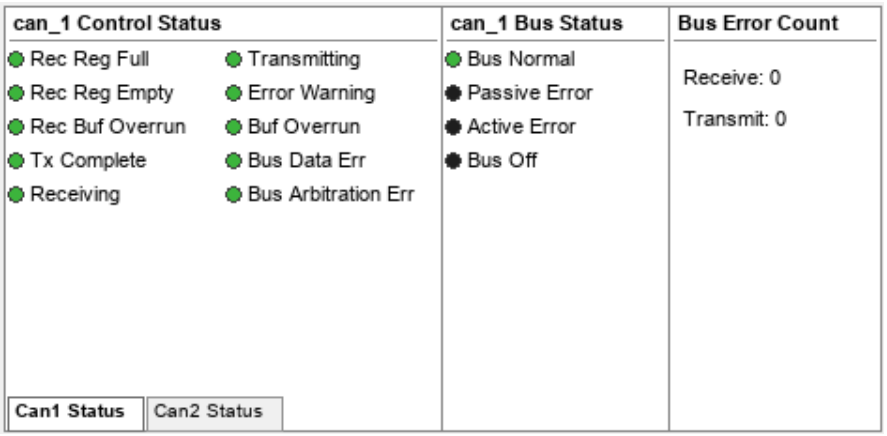
Using the GCANTools software, users can read out whether the bus is normal from the status in the lower-right corner of the software. Typical examples:

1) Active error or passive error: indicates an incorrect baud-rate setting, a wiring error, an empty bus or no bus response.

2) All green lights but no data: indicates the bus is empty and there is no data to receive.







4.6 OBD II Automotive Electronics Interface

Users can use the automotive signal-analysis function of the GCANTools software. Connecting this device to a vehicle's OBD port allows some actual sensor values inside the vehicle to be resolved. The signal-analysis function supports only the ISO15765 protocol for passenger gasoline vehicles. Vehicle speed, engine speed and coolant temperature can be displayed on the software dashboard.

Currently this function supports only the CAN1 channel. Users can intuitively see the current real-time vehicle speed, engine speed and fuel level on this interface, making it convenient to verify the accuracy of the vehicle's instrument panel. The GCANTools software can also read, resolve and clear vehicle fault codes, and resolve vehicle sensor data specified in the ISO15765 protocol, including engine speed, coolant temperature, vehicle speed, voltage, intake manifold pressure, intake air temperature, air flow rate, throttle position, oxygen-

sensor voltage, fuel pressure, and so on. Changes in all these values can be saved to the computer in real time.

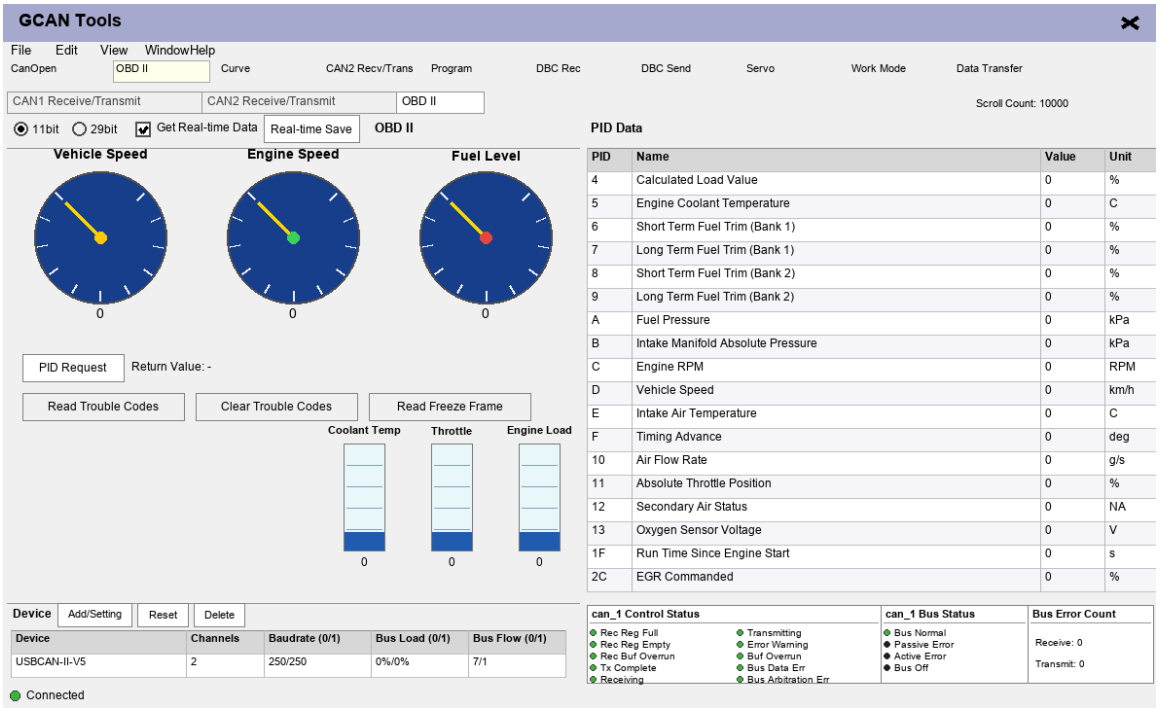
When using, first select the 11Bit button and check "Cyclic Acquire Real-time Data". If the USBCAN analyzer is connected to the vehicle powertrain bus, parameters such as vehicle speed, engine speed and coolant temperature will be returned. If there is no data, switch to 29Bit and try again. If neither mode responds, check the communication baud rate and wiring.

Manual PID acquisition function: you can manually acquire various vehicle parameters by entering a PID value. Note that you need to first select the correct send-command type, i.e. 11Bit or 29Bit. Detailed PID values (decimal) are listed in the table below.

PID	Data Header	Min	Max	Unit
01	Number of current fault codes	0	127	count
04	Calculated load value	0	100	%
05	Engine coolant temperature	-40	215	°C
06	Short-term fuel trim (bank 1 and 3)	-100	99.22	%
07	Long-term fuel trim (bank 1 and 3)	-100	99.22	%
08	Short-term fuel trim (bank 2 and 4)	-100	99.22	%
09	Long-term fuel trim (bank 2 and 4)	-100	99.22	%
010	Fuel pressure	0	765	kPa
011	Intake manifold absolute pressure	0	255	kPa
012	Engine speed	0	9999	rpm
013	Vehicle speed	0	255	km/h
014	Ignition timing advance for cylinder 1 (excluding mechanical advance)	-64	63.5	°
015	Intake air temperature	-40	215	°C
016	Air flow rate from mass air flow sensor	0	655.35	g/s
017	Absolute throttle position	0	100	%
031	Time since engine start	0	65535	s
033	Distance travelled with MIL activated	0	65535	km
034	Fuel rail pressure relative to manifold vacuum	0	5177.265	kPa
035	Fuel rail pressure relative to atmospheric pressure	0	655350	kPa
044	EGR commanded opening	0	100	%
045	EGR opening error (actual-commanded)/commanded	-100	99.22	%
046	Evaporative purge control command	0	100	%
047	Fuel level input	0	100	%
048	Number of warm-up cycles since fault codes cleared	0	255	N/A
049	Distance travelled since fault codes cleared	0	65535	km
050	Evaporative system vapor pressure	-8192	8192	Pa
051	Atmospheric pressure	0	255	kPa

060	Catalyst temperature B1S1	-40	6513.5	°C
061	Catalyst temperature B2S1	-40	6513.5	°C
062	Catalyst temperature B1S2	-40	6513.5	°C
063	Catalyst temperature B2S2	-40	6513.5	°C
066	Control module voltage	0	65.535	V
067	Absolute load value	0	25700	%
068	Commanded equivalence ratio	0	2	N/A
069	Relative throttle position	0	100	%
070	Ambient air temperature	-40	215	°C
071	Absolute throttle position B	0	100	%
072	Absolute throttle position C	0	100	%
073	Accelerator pedal position D	0	100	%
074	Accelerator pedal position E	0	100	%
075	Accelerator pedal position F	0	100	%
076	Throttle actuator control command	0	100	%
077	Engine run time with MIL activated	0	65535	min
078	Time since fault codes cleared	0	65535	min

Table 4.1 OBD II PID data



PID	Name	Value	Unit
4	Calculated Load Value	0	%
5	Engine Coolant Temperature	0	C
6	Short Term Fuel Trim (Bank 1)	0	%
7	Long Term Fuel Trim (Bank 1)	0	%
8	Short Term Fuel Trim (Bank 2)	0	%
9	Long Term Fuel Trim (Bank 2)	0	%
A	Fuel Pressure	0	kPa
B	Intake Manifold Absolute Pressure	0	kPa
C	Engine RPM	0	RPM
D	Vehicle Speed	0	km/h
E	Intake Air Temperature	0	C
F	Timing Advance	0	deg
10	Air Flow Rate	0	g/s
11	Absolute Throttle Position	0	%
12	Secondary Air Status	0	NA
13	Oxygen Sensor Voltage	0	V
1F	Run Time Since Engine Start	0	s
2C	EGR Commanded	0	%

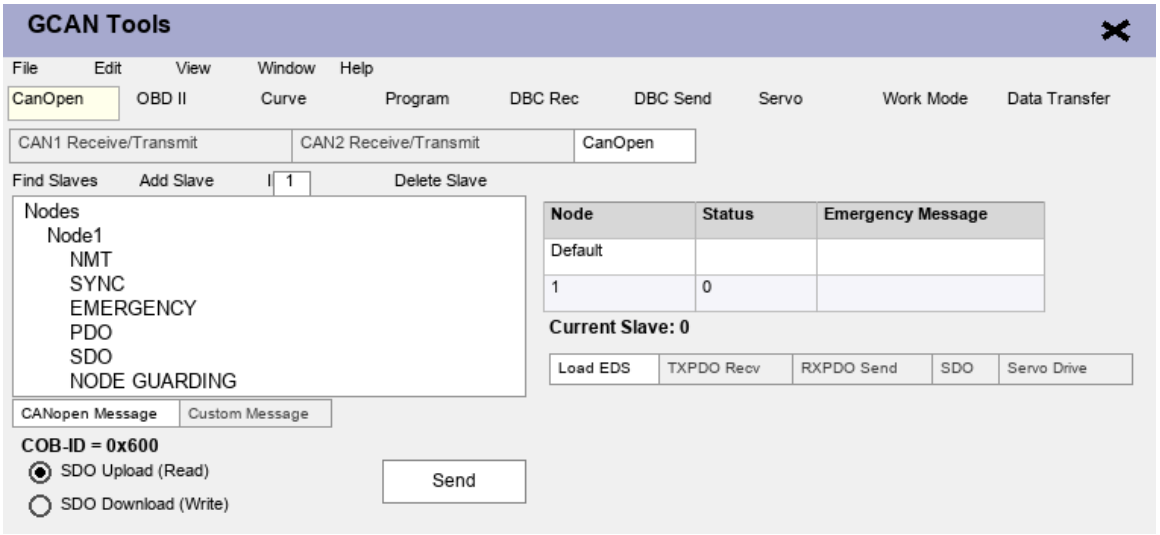
Realtime TXT file save function: through this function users can intuitively see the specific values of some sensors inside the vehicle and diagnose whether each sensor is working normally based on these data. Users can also intuitively compare the variation patterns of certain sensor values and compare values between multiple sensors.

4.7 CANopen Master Function

Switch to the "CANopen" tab of the GCANTools software to enter the CANopen master page. Users can click "Find Slaves" on the left to scan all CANopen slave nodes on the current CAN bus, or manually add CANopen slaves by entering the node number. Scanned or added slave nodes are displayed below; users can double-click a specified node number to activate the communication interface with that slave.

The CANopen master function of GCANTools can turn the USBCAN device into a CANopen master card connected to a CANopen network. Using a computer together with the USBCAN, applications such as control, management and configuration of CANopen slaves can be completed. GCANTools can perform many common CANopen master functions, such as NMT (Network Management), SYNC (synchronization protocol), EMERGENCY (emergency message), PDO (Process Data Object), SDO (Service Data Object) and NODE GUARDING.

For the application-layer description of the CANopen protocol, device sub-protocols and interface specifications, you may also refer to the book "Design and Application of Fieldbus CANopen". The table below lists the function codes and frame ID ranges corresponding to common CANopen message types.

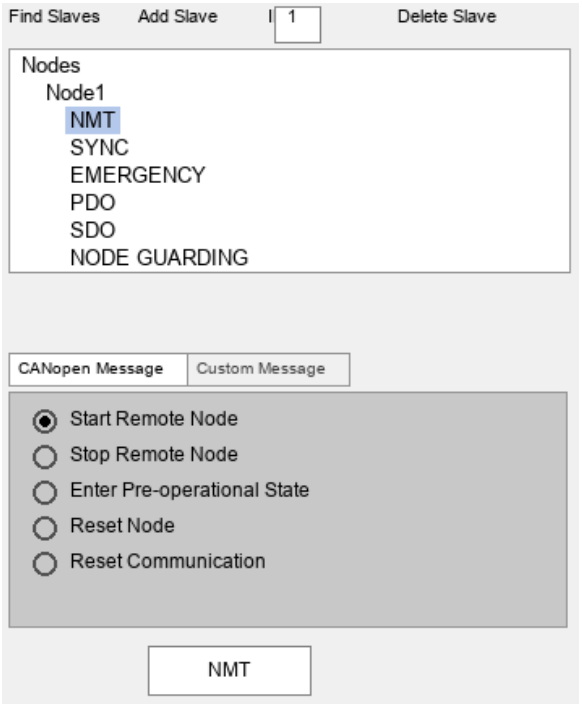


4.7.1 NMT Commands

NMT commands (Network Management) provide network-management services such as initializing, starting and stopping nodes, and detecting failed nodes. This service is implemented using the master-slave communication model (so there is only one NMT master node). Users can use NMT commands to change the running state of slave nodes:

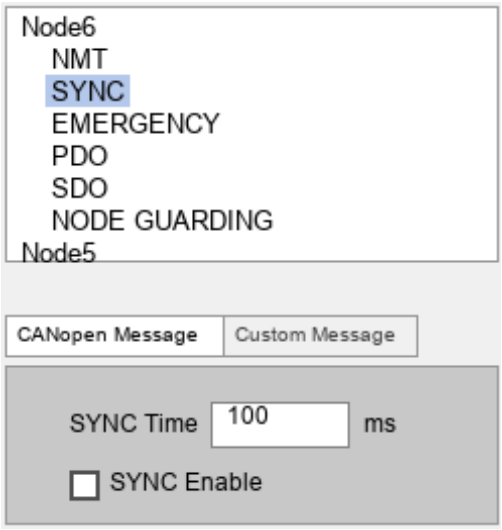
- Start remote node - set the node to the operational state
- Stop remote node - set the node to the stopped state
- Enter pre-operational state - set the node to the pre-operational state
- Reset node - reset the node
- Reset communication - reset the node communication

Message Type	Function Code	COB-ID Range (Hex)
NMT	0000	000h
SYNC	0001	080h
EMERGENCY	0001	081h~0FFh
TIME	0010	100h
PDO1 (Transmit)	0011	181h~1FFh
PDO1 (Receive)	0100	201h~27Fh
PDO2 (Transmit)	0101	281h~2FFh
PDO2 (Receive)	0110	301h~37Fh
PDO3 (Transmit)	0111	381h~3FFh
PDO3 (Receive)	1000	401h~47Fh
PDO4 (Transmit)	1001	481h~4FFh
PDO4 (Receive)	1010	501h~57Fh
SDO (Transmit)	1011	581h~5FFh
SDO (Receive)	1100	601h~67Fh
NMT Error Control	1110	701h~77Fh



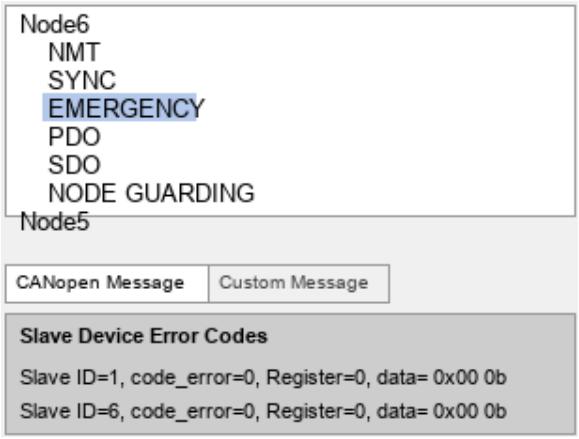
4.7.2 SYNC Message

SYNC is a synchronization message sent by the master; all slaves configured with synchronous PDOs communicate with the master according to the SYNC message. Users can send SYNC messages cyclically through GCANTools, and all synchronous PDOs will be sent according to the SYNC message.



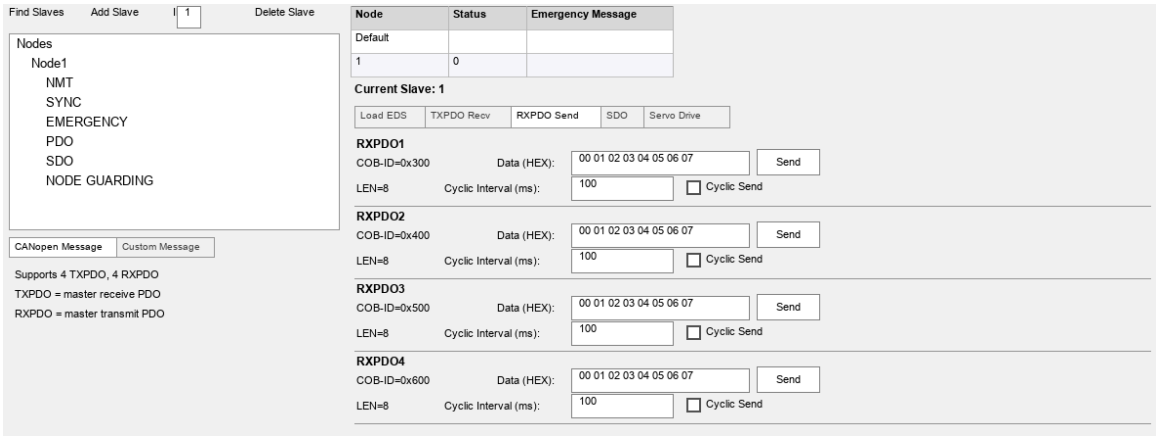
4.7.3 EMERGENCY Message

An emergency indication message is triggered by a fatal error occurring inside a device and is sent by the relevant application device to other devices at the highest priority. It is suitable for interrupt-type error alarm signals. GCANTools can receive EMERGENCY messages sent by slave nodes; these messages are sent by slaves in an emergency state, and the CANopen master function of GCANTools can receive and process them.



4.7.4 PDO

CANopen nodes use PDOs for data transmission, mainly for transmitting real-time data. A Process Data Object (PDO), unlike an SDO, belongs to process data, is transmitted unidirectionally, and requires no CAN message response from the receiving node for confirmation; it belongs to the "producer-consumer" model. GCANTools provides four pairs of transmit and receive PDOs by default, namely TPDO/RPDO.



4.7.5 SDO

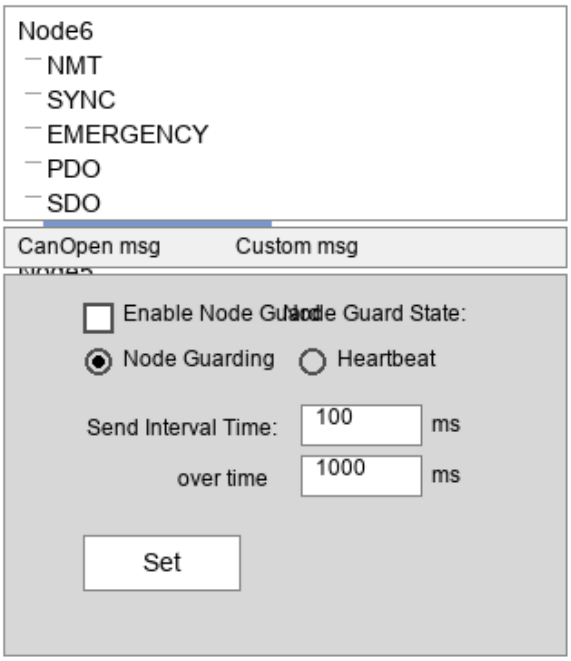
The SDO function of the CANopen master module is divided into sending SDO and receiving SDO, used to read and write the object dictionary. An SDO (Service Data Object) has a specified address of the receiving node (Node ID) and requires the specified receiving node to respond with a CAN message to confirm reception; if no confirmation is received within the timeout, the sending node resends the original message. This communication method belongs to the common "server-client" communication model, i.e. the so-called polling type. The SDO function in GCANTools can exchange information with slaves by editing the index and sub-index, realizing read and write access to entries in the object dictionary.

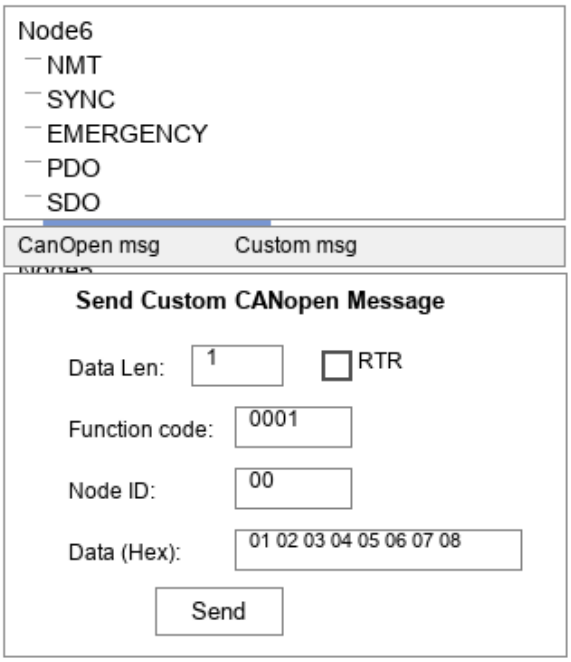
Index and Object (OD) Table:

Index	Object
0000h	Reserved
0001h~001Fh	Basic data types
0020h~003Fh	Complex data types
0040h~005Fh	Manufacturer-specific data types
0060h~007Fh	Basic data types for device profile
0080h~009Fh	Complex data types for device profile
00A0h~0FFFh	Reserved
1000h~1FFFh	Communication parameters
2000h~5FFFh	Manufacturer-specific device profile
6000h~9FFFh	Standard device profile
A000h~BFFFh	Standard interface profile
C000h~FFFFh	Reserved

4.7.6 NODE GUARDING

The Node Guarding function in GCANTools has two implementation methods: the master actively sends commands to query the state of a slave node (Node Guarding), and the slave node periodically transmits its state by means of a heartbeat (Heartbeat). GCANTools can also simulate CANopen data manually by editing custom messages; users can send data directly to CANopen slaves by writing the function code, slave node ID and data.





The functions corresponding to the function codes are listed below:

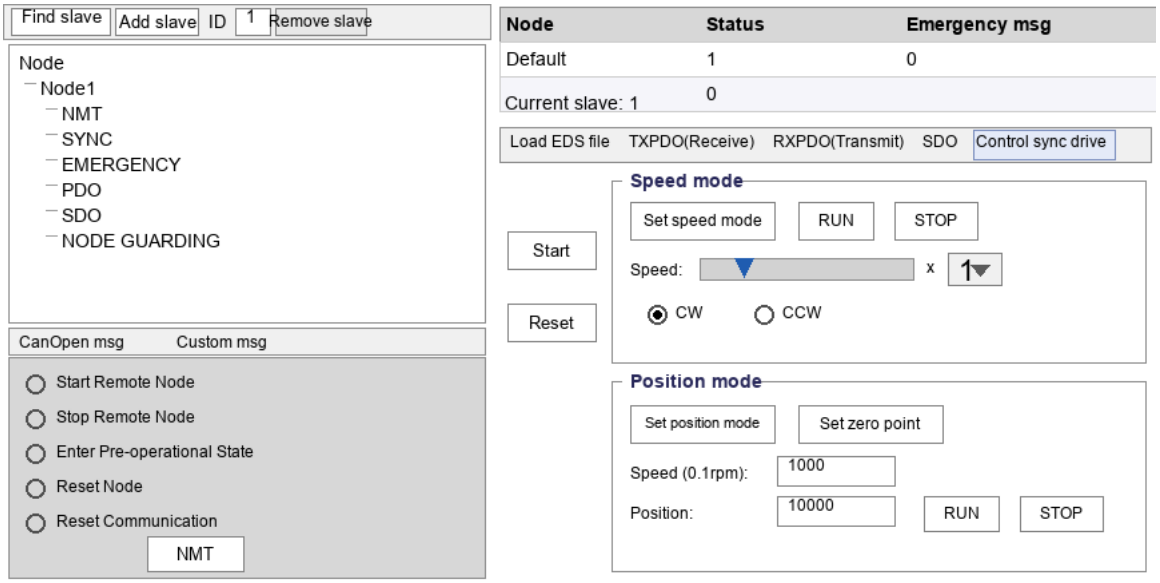
- Configure PDO 0x1800+n - its COB-ID is 0x387; this PDO is always trigger-transmitted
- It must contain the data: data X (2 bytes), data Y (4 bytes), in the following order
- data X is defined at index 0x6000, sub-index 03
- data Y is defined at index 0x2010, sub-index 21
- 1) Index 1800+n, sub-index 01: write COB-ID (4 bytes)
- 2) Index 1800+n, sub-index 02: write transmission byte <t> (1 byte): t=1 to 0xF0 means the PDO is transmitted after every <t> SYNCs; t=FD means transmit after receiving a PDO request (RTR=1); t=FF means transmit on event, node sends PDO spontaneously
- 3) Index 1A00+n: define the mapping of the n-th data: sub-index 0 writes the number of data items embedded in the PDO (1 byte), in this example 2; sub-index 1 defines where to find the first embedded data and its size (8 bytes), in the format index (2 bytes) -

subindex (1 byte) - size in bits (1 byte), in this example 60000310; sub-index 2 defines the second embedded data, in this example 20102120

- To configure PDO 1802 of node 5 to transmit every 3 SYNCs, the SDOs sent should be:
- 605 23 02 18 01 00 00 87 03
- 605 2F 02 18 02 03 00 00 00
- 605 2F 02 1A 00 02 00 00 00
- 605 23 02 1A 01 10 03 00 60
- 605 23 02 1A 02 20 21 10 20

4.7.7 Servo Drive Control

This function supports only servo motor devices conforming to the CANopen 402 standard; please confirm in advance when using it. At present only "Set Speed Mode" is available for simple control of servo motor devices. After connecting the CAN bus of the servo motor device, click "Find Slaves" - "Start" - "Set Speed Mode" - "RUN" in sequence to drive the servo motor.



4.8 Curver Function

Data value changes can also be displayed in real time as curves, as shown below. Users can select up to four variables to display simultaneously in one interface. Users can select up to four required variables through the curve function and understand the change of each data in real time by observing the curves; they can also intuitively compare the relative changes between multiple values. This can be used to adjust the engine idle speed, torque and so on.

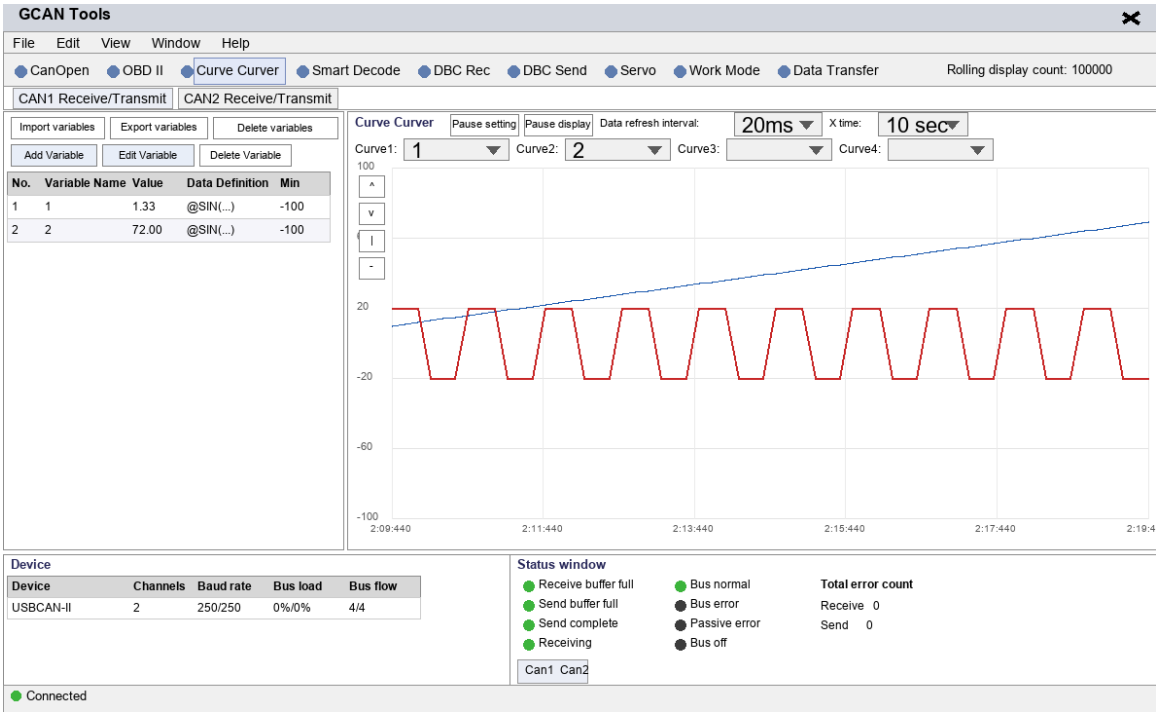
When using the curve function, please set the variables first and then receive data; variables created during data reception cannot be displayed in the curve.

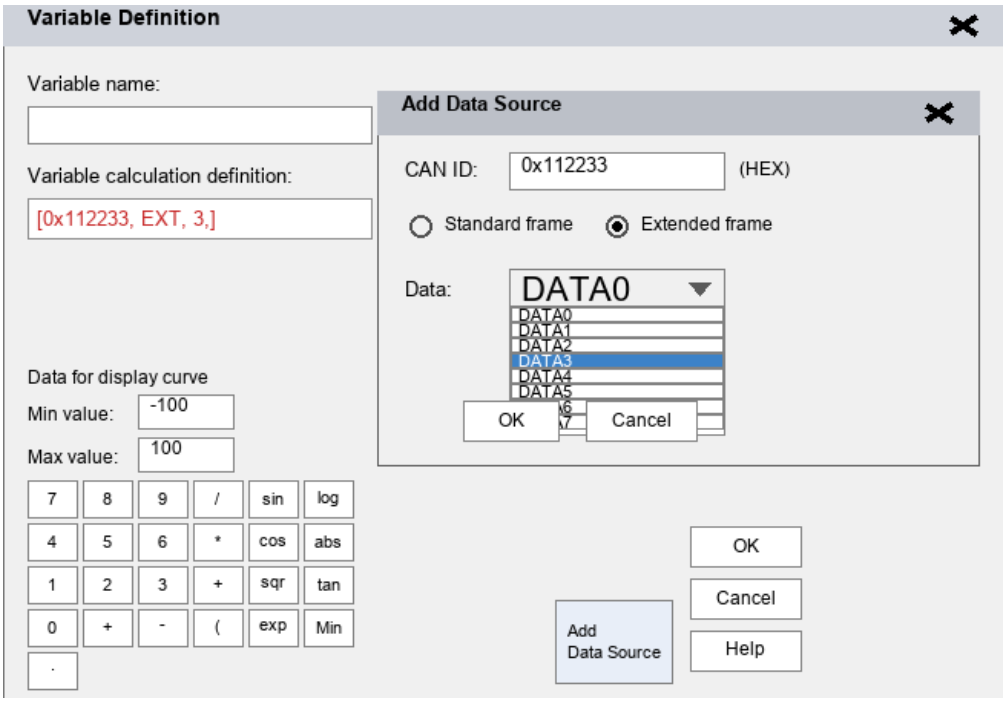
Click "Add Variable" and "Edit Variable" to add and edit variables. These variables can perform simple mathematical operations on the raw CAN-bus data received and display the results as curves. Multiple groups of variables can be added; variable names may contain letters, numbers, Chinese characters, spaces and underscores for easy identification.

The special operations provided by the curve function are listed below; please write them strictly in the specified form. When using a special operator, at least one operand must be

inside the parentheses. For example, @SIN(data source) is incorrect and will prompt a non-numeric error; the correct form is @SIN(data source*1).

Operator	Function	Writing Format (example: sine)
sin/cos/tan	Sine / cosine / tangent	@SIN(source*1)
sqr	Arithmetic square root of a non-negative real number	@SQR(source*1)
log	Natural logarithm of a number (ln x)	@LOG(source*1)
abs	Absolute value of a number	@ABS(source*1)
exp	Exponential function with base e (e^x)	@EXP(source*1)
Max	Current maximum value of the variable	@MAX(source*1)
Min	Current minimum value of the variable	@MIN(source*1)





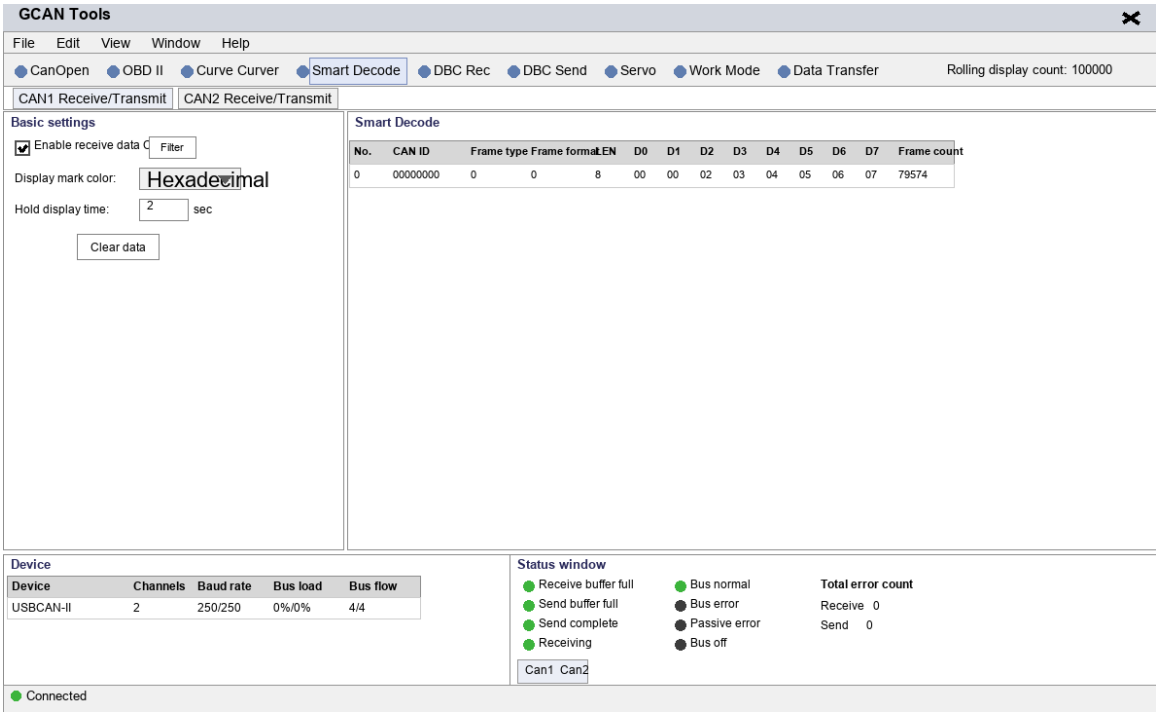
"Add Data Source" adds a certain byte of a certain frame in the received raw CAN-bus data as a data source to the variable calculation definition. First click "Add Data Source", then fill in "CAN ID", "Standard Frame (Extended Frame)", "Data" and so on in sequence; after clicking OK, a corresponding statement is generated in "Variable Calculation Definition". This statement can also be written manually, in which case please follow this format.

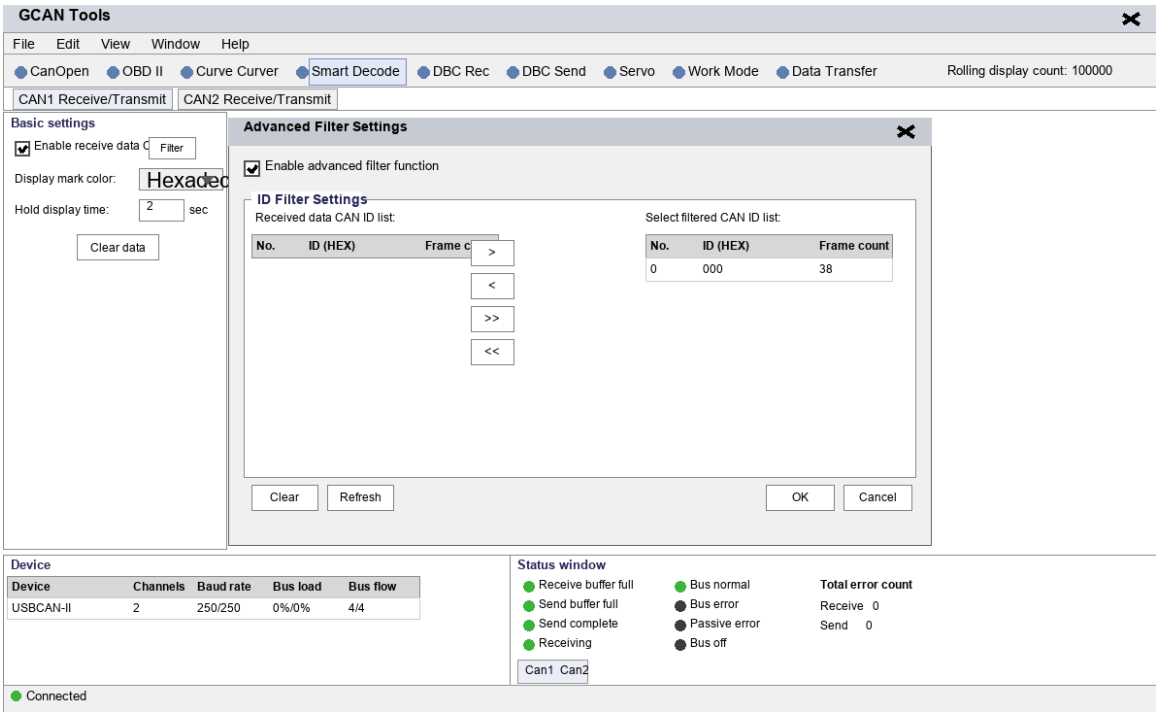
In the curve interface, the up/down/left/right arrow buttons switch to the previous curve, the next curve, or move the current color curve's axis up or down respectively.



4.9 Smart Decode Function

The smart decode function currently supports only the display of the change frequency of frame data for a specific frame ID on the CAN1 channel. The color changes with the change frequency: the faster the change, the darker the color. It can be displayed in hexadecimal, decimal or binary. If data filtering is needed, click "Filter" to set the relevant filter parameters. When using, please check "Enable Receive Data CAN1".





4.10 DBC Send Function

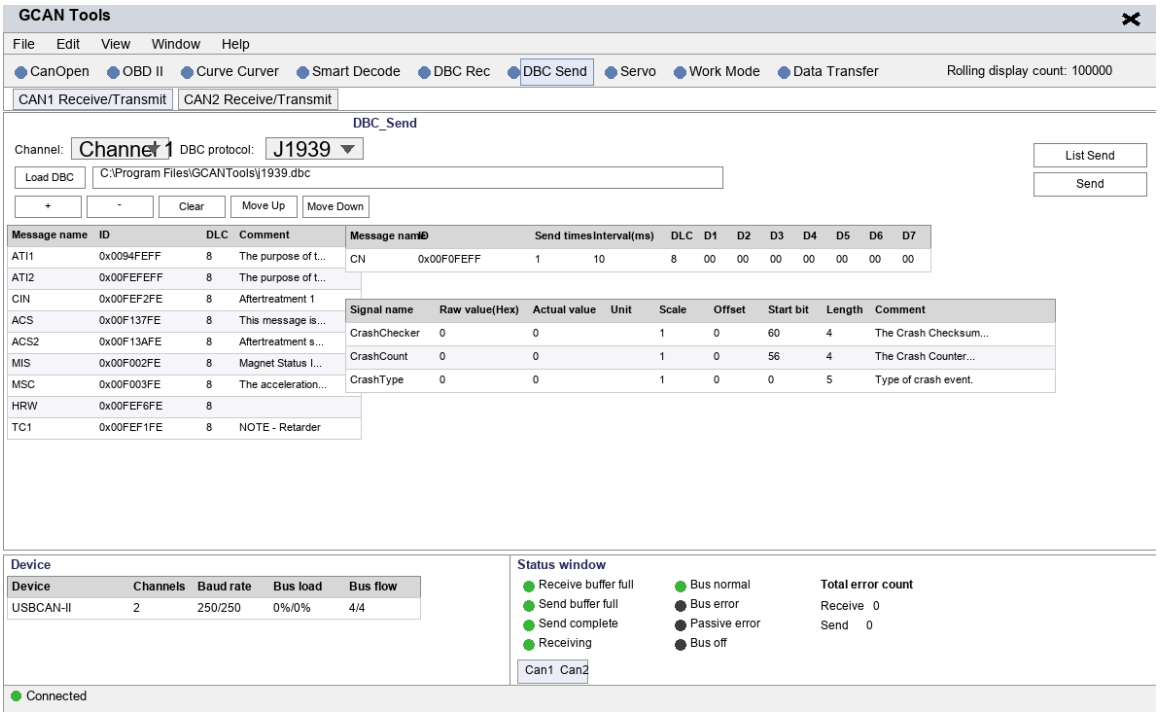
Click DBC Send on the toolbar. This tab is used to send DBC messages and supports sending each message independently without interference. In the sending interface shown below, the table on the left is the DBC message selection list, the upper table on the right is the list of messages ready to send, and the lower table on the right is the signal list of the currently selected message.

Loading DBC: to load a DBC protocol, first select the DBC protocol, then click the "Load DBC" button to select the DBC file. The DBC message selection list on the left shows all

messages in the currently loaded DBC file; after selecting one, click the [+] button to add it to the DBC message sending list in the upper right.

DBC message editing: the DBC message sending list in the upper right shows the messages to be sent; double-click to edit the number of sends and the sending interval. The message signal list in the lower right shows all signals of the currently selected message in the sending list; double-click to edit the actual signal value, and the data field of that message changes synchronously after editing.

- Add: click [+] to add a message from the DBC message selection list to the sending list
- Remove: click [-] to remove the currently selected message from the DBC message sending list
- Clear: remove all messages from the DBC message sending list
- Move Up: move the currently selected highlighted message up one position in the sending list
- Move Down: move the currently selected highlighted message down one position in the sending list

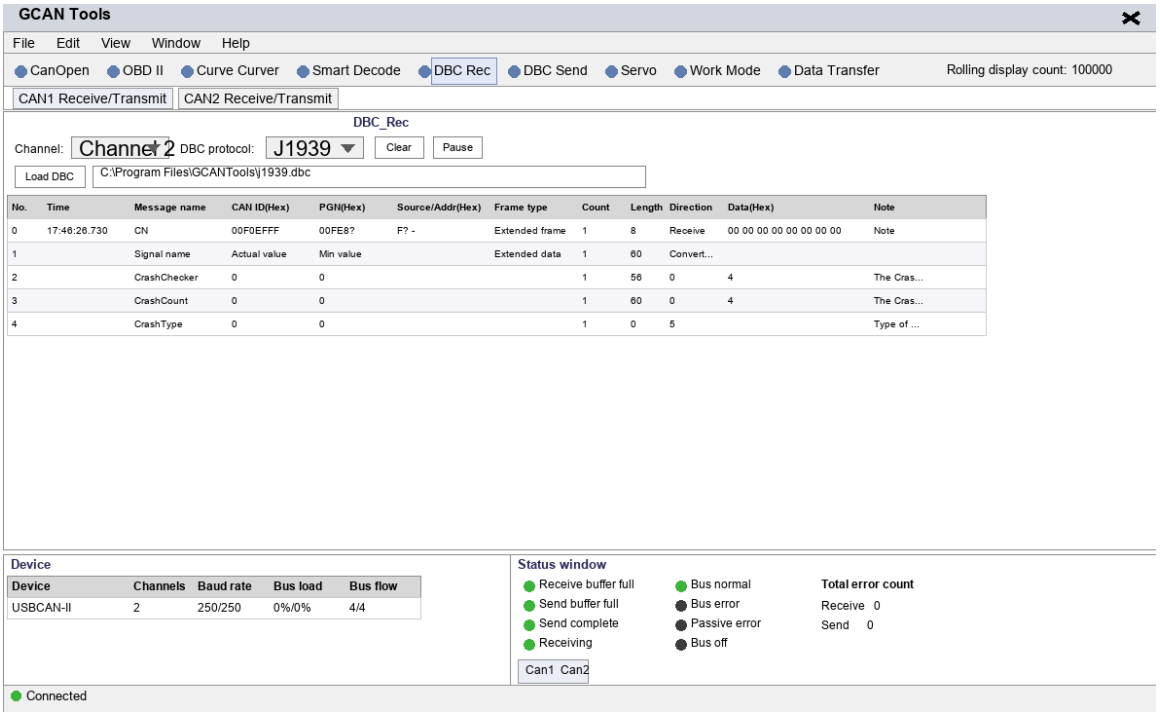


DBC sending: Channel Selection - select the device channel used for DBC message sending in the "Channel" drop-down box at the upper left. List Send / Stop Send - start or stop sending all messages in the list. Send - start sending the currently highlighted message.

4.11 DBC Rec Function

Click DBC Rec on the toolbar. This tab shows the result of parsing raw messages according to the DBC specification, as shown below. Channel Selection: select from the drop-down the channel whose data you want to view. DBC Protocol: the protocol on which DBC parsing is based. J1939: matches the PGN part of the frame ID; refer to the J1939 specification for PGNs; matching succeeds when the PGN of the DBC message ID equals the PGN of the message ID to be parsed. Normal: matching succeeds when the DBC message ID equals the message ID to be parsed. Load DBC: click [Load DBC] to select a DBC file. Triangle:

click to view all signals of the current message. Clear: clear the list. Pause: pause or resume receiving messages.



5. Linux Operating System

The Guangcheng USBCAN analyzer series supports various Linux operating systems. We provide 32/64-bit Linux drivers and secondary-development documents to customers, and users can develop and use the device in Linux themselves.

The general method of using our devices in Linux is: 1) obtain system administrator privileges; 2) copy the necessary files to the system GCC compilation directory; 3) switch to the USBCAN driver folder to compile; 4) run the test program. The specific operations are as follows.

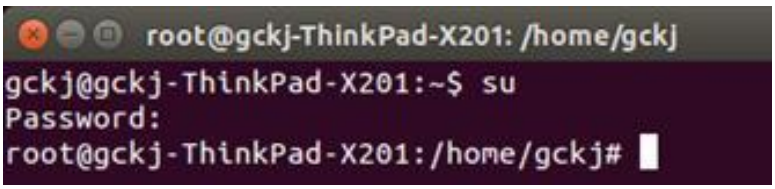
1. Query the Linux version number and confirm the system type (32/64-bit). Enter: uname -a



2. After determining the Linux system type, copy the corresponding USBCAN driver file to the system. (In this example we put the driver file on the system desktop.)

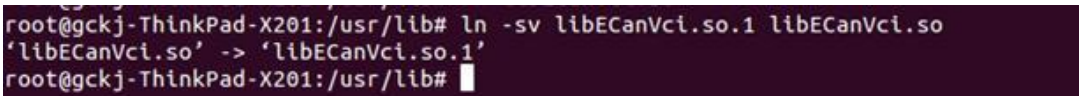


3. Obtain administrator privileges for subsequent driver installation and other operations. Enter: su (after entering the su command you will be asked for the administrator password; entering the correct password grants administrator privileges).





4. Enter the USBCAN driver folder and copy libusb.so, libusb-1.0.so and libECanVci.so.1 to the GCC compilation library directory (default path /usr/lib). Enter: cp libusb.so libusb-1.0.so libECanVci.so.1 /usr/lib



5. Enter the GCC compilation library folder and link libECanVci.so.1 and libECanVci.so together. Enter: ln -sv libECanVci.so.1 libECanVci.so



6. Enter the USBCAN driver folder again and compile. Enter: make

```
root@gckj-ThinkPad-X201:/home/gckj/Desktop/linux64# ./test
test [DevType] [DevIdx] [ChMask] [Baud] [TxType] [TxSleep] [TxFrames]
    example: test 16 0 3 0x1400 0 1 1000
                  |  | | |      | | |
                  |  | | |      | | |1000 frames / channel
                  |  | | |      | |
                  |  | | |      | |tx > sleep(3ms) > tx > sleep(3ms) ....
                  |  | | |      |
                  |  | | |      |0-normal, 1-single, 2-self_test, 3-single_self_
test, 4-single_no_wait....
                  |  | | |
                  |  | | |0x1400-1M, 0x1c03-125K, ....
                  |  | |
                  |  | |bit0-CAN1, bit1-CAN2, bit2-CAN3, bit3-CAN4, 3=CAN1+CAN2,
 7=CAN1+CAN2+CAN3
                  |  |
                  |  |Card0
                  |
                  |1-usbcan,  ....
root@gckj-ThinkPad-X201:/home/gckj/Desktop/linux64#
```

7. Run the test program to test USBCAN transmit and receive. Enter: ./test

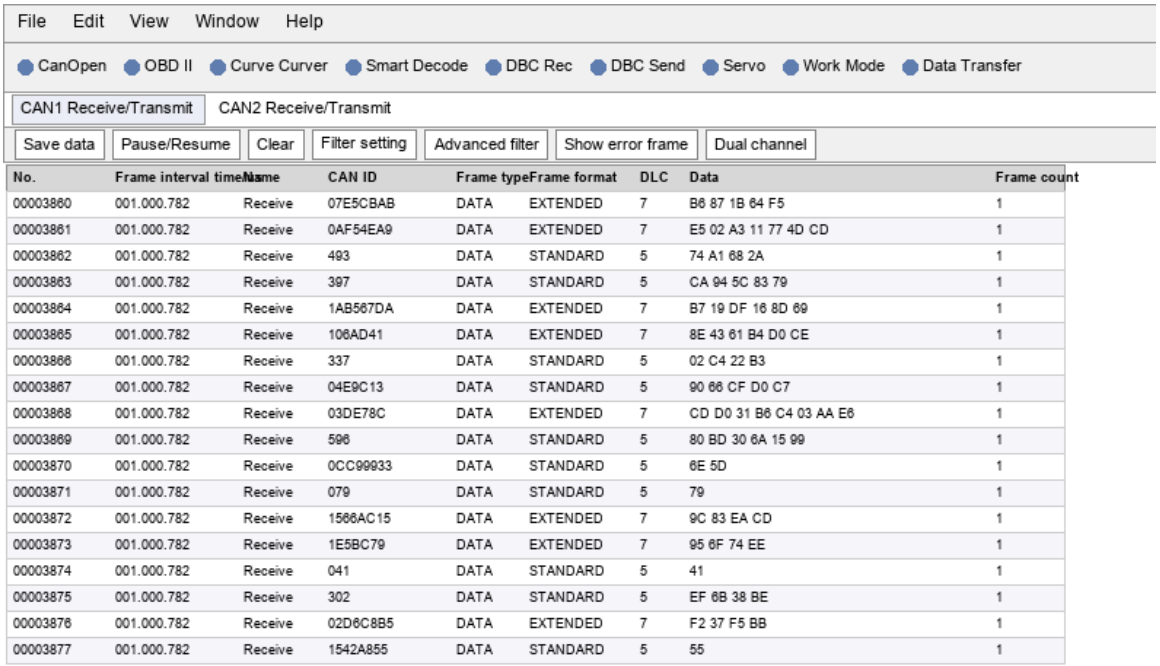
```
root@gckj-ThinkPad-X201: /lib/test/linux64
Receive returned: time ~= 2 seconds
Receive returned: time ~= 3 seconds
<ENTER> to start TX: 1000 frames/channel, baud: t0=0x00, t1=0x14...
1
TX stopped, <ENTER> to terminate RX-threads...
CAN0 RX thread terminated, 0 frames received & verified: no error
CAN1 RX thread terminated, 0 frames received & verified: no error

 ***** 1000 frames/channel transferred, 2 seconds elapsed *****
       performance: 500 frames/channel/second
CloseDevice
root@gckj-ThinkPad-X201:/lib/test/linux64# ./test 4 0 3 0x1400 0 1 1000
DevType=4, DevIdx=0, ChMask=0x3, Baud=0x1400, TxType=0, TxSleep=1, TxFrames=0x00
0003e8(1000)
OpenDevice succeeded
InitCAN(0) succeeded
StartCAN(0) succeeded
InitCAN(1) succeeded
StartCAN(1) succeeded
Receive returned: time ~= 1 seconds
Receive returned: time ~= 2 seconds
Receive returned: time ~= 3 seconds
<ENTER> to start TX: 1000 frames/channel, baud: t0=0x00, t1=0x14...
```

After entering ./test, the system prompt and examples appear, in which:

- 1st value (16): device type; enter 3 for a single-channel device, 4 for a dual-channel device
- 2nd value (0): device index; 0 when only one USBCAN is connected
- 3rd value (3): which CAN channel to open; enter 1 for CAN1, 2 for CAN2, 3 for both CAN1 and CAN2
- 4th value (0x1400): CAN-bus baud rate; 0x1400 means 1000 kbit/s; see the "EcanVCI Dynamic Library User Manual" for other baud-rate values
- 5th value (0): working mode; 0 is normal mode; see the "EcanVCI Dynamic Library User Manual" for other modes
- 6th value (1): send interval, in ms
- 7th value (1000): number of sends

8. After running the test program, you can use another USBCAN device to receive the data it sends.

The test results are shown below:



No.	Frame interval time	Name	CAN ID	Frame type	Frame format	DLC	Data	Frame count
00003860	001.000.782	Receive	07E5CBAB	DATA	EXTENDED	7	B6 87 1B 64 F5	1
00003861	001.000.782	Receive	0AF54EA9	DATA	EXTENDED	7	E5 02 A3 11 77 4D CD	1
00003862	001.000.782	Receive	493	DATA	STANDARD	5	74 A1 68 2A	1
00003863	001.000.782	Receive	397	DATA	STANDARD	5	CA 94 5C 83 79	1
00003864	001.000.782	Receive	1AB567DA	DATA	EXTENDED	7	B7 19 DF 16 8D 69	1
00003865	001.000.782	Receive	106AD41	DATA	EXTENDED	7	8E 43 61 B4 D0 CE	1
00003866	001.000.782	Receive	337	DATA	STANDARD	5	02 C4 22 B3	1
00003867	001.000.782	Receive	04E9C13	DATA	STANDARD	5	90 66 CF D0 C7	1
00003868	001.000.782	Receive	03DE78C	DATA	EXTENDED	7	CD D0 31 B6 C4 03 AA E6	1
00003869	001.000.782	Receive	596	DATA	STANDARD	5	80 BD 30 6A 15 99	1
00003870	001.000.782	Receive	0CC99933	DATA	STANDARD	5	6E 5D	1
00003871	001.000.782	Receive	079	DATA	STANDARD	5	79	1
00003872	001.000.782	Receive	1566AC15	DATA	EXTENDED	7	9C 83 EA CD	1
00003873	001.000.782	Receive	1E5BC79	DATA	EXTENDED	7	95 6F 74 EE	1
00003874	001.000.782	Receive	041	DATA	STANDARD	5	41	1
00003875	001.000.782	Receive	302	DATA	STANDARD	5	EF 6B 38 BE	1
00003876	001.000.782	Receive	02D6C8B5	DATA	EXTENDED	7	F2 37 F5 BB	1
00003877	001.000.782	Receive	1542A855	DATA	STANDARD	5	55	1

6. Secondary Development

We provide a standard interface function library for secondary-development users, including ECANVCI.h, ECANVCI.lib and ECANVCI.dll. These libraries are all in standard format; users can declare and call these interface functions in programming environments such as VC, VB and LabVIEW. For specific usage, see the "EcanVCI Dynamic Library User Manual". Figure 6.1 shows the common structure names and the function-library call flow.

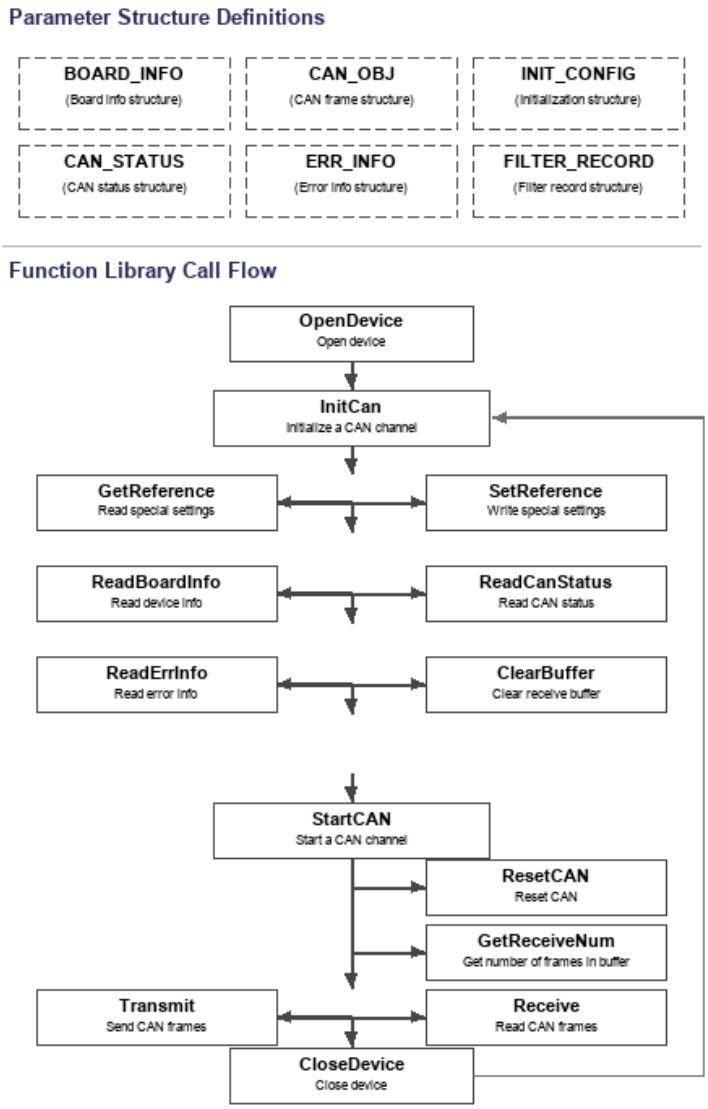


Figure 6.1 Secondary-development function calls

7. Technical Specifications

Category	Item	Specification
Connection	PC side	USB port, Type A
	CAN side	OPEN6 pluggable terminal
Interface features	USB interface	USB 2.0 full-speed, compatible with USB 1.1 and USB 3.0
	CAN interface	Compliant with ISO 11898, supports CAN 2.0A/B; two CAN channels individually isolated
	CAN baud rate	5 kbit/s to 1 Mbit/s
	Electrical isolation	3000 V DC
	CAN terminating resistor	Integrated; enabled via DIP switch

Power supply	Supply voltage	+5 V DC (USB port)
	Supply current	Maximum 130 mA
Environmental	Operating temperature	-40°C to +85°C
	Operating humidity	15% to 90% RH, non-condensing
EMC test	Standard	EN 55024:2011-09; EN 55022:2011-12
Protection	Ingress rating	IP 20
General	Dimensions	104.5 mm x 64 mm x 24 mm (including terminal block)
	Weight	120 g

Table 7.1 Technical specifications

8. Frequently Asked Questions

1. Why does the GCANTools test software display "Device open error"?

Possible causes are: the wrong device type was selected, or the USBCAN device driver was not installed correctly. Check the USBCAN device properties in the PC's Device Manager to see whether there is a "!" or "?" in front of the USBCAN device; if so, check for hardware/software conflicts and reinstall the USBCAN device driver.

2. Is a 120-ohm terminating resistor always necessary?

A 120-ohm terminating resistor is recommended to absorb end reflections and provide a stable physical link. For a single-node self-transmit/self-receive test, the 120-ohm terminating resistor must be connected to form a loop; otherwise the test cannot be performed. The USBCAN high-performance CAN interface card already has a 120-ohm terminating resistor connected internally.

3. Can multiple USBCAN interface cards be installed on one computer?

Older interfaces do not support multi-card operation, but current USBCAN interface cards support up to eight USBCAN cards of the same model operating simultaneously.

4. What is the maximum data conversion rate of the USBCAN-II Pro interface card?

A single CAN channel of the USBCAN interface card supports a maximum of 8000 fps of CAN-bus data conversion. The frames mentioned here are standard data frames with 8 data bytes; frames with fewer than 8 bytes or remote frames may be faster. In addition, the maximum data throughput is limited by PC performance.

5. Why is the CAN status indicator not lit?

Because all operations of the USBCAN interface card are controlled by the PC, the CAN status indicator is meaningful only after the PC has sent a command to start CAN communication.

6. Why does the system report an illegal operation when an interface function is called?

First, when using the interface functions, carefully read the function descriptions to ensure the input parameters are legal, paying special attention to the passing of pointers

(addresses), or refer to the provided example programs. If the problem is still not solved, please contact our technical support.

7. How is the communication baud rate of the USBCAN interface card set?

The device provides a set of commonly used baud-rate values. To use other baud rates, please contact Guangcheng customer service for calculation. Note that the CAN controller of the USBCAN interface card uses a 24 MHz clock; user-defined baud rates must be calculated based on this clock frequency.

8. Does the system entering standby or sleep mode affect reception?

Yes. All processing will stop, which may at worst cause a hardware receive-buffer overflow error. If a program has opened the device, it will try to prevent the system from entering standby or sleep mode to ensure normal operation. When using the USBCAN interface card, please disable the system's standby and sleep functions.

9. How should application errors be handled?

Errors are mainly divided into function-call errors and CAN-bus communication errors. Function-call errors are generally caused by parameter errors, such as a device number out of range or an incorrect type number; the error number returned by the Win32 function GetLastError is 87. Other cases involve operating on a device that has not been opened, i.e. operating on an illegal handle; corresponding Win32 standard error codes are provided according to the specific function call, and users can use GetLastError for error analysis. This debugging work should generally be completed during the design stage.

CAN-bus communication errors are generally caused by the CAN network, but may also be caused by improper user settings, such as inconsistent baud rates or calling the send function without starting the CAN controller. Most errors are handled simply in the device driver; for deeper error analysis and handling, the ReadCANStatus function can be called.

Another point to note is the data-overflow interrupt error, which can occur in two cases: (1) software receive-buffer overflow, which means the application cannot process received data in time; in this case users should optimize the application or change the communication strategy; (2) hardware receive-buffer overflow, caused by excessive PC interrupt latency. This can only be solved by improving computer performance or coordinating other nodes to appropriately reduce the sending speed.

10. What should be noted when opening and closing the device?

The USBCAN-II Pro interface card provides two CAN ports, and users can operate different ports in the same program. The card does not allow the device to be opened in shared mode; the same device cannot be opened multiple times by different processes calling OpenDevice. The OpenDevice and CloseDevice functions generally need to be called only once at application initialization and exit. When closing the device, if the current port is no longer in use, the ResetCAN function should be called first to detach the port from the CAN bus; the device driver automatically calls ResetCAN to exit the CAN-bus connection only when the last device handle is closed.

11. How should the Transmit sending function be used effectively?

The driver of the USBCAN interface card provides a transmit FIFO of about 128 frames, and each Transmit call can send up to about 128 frames. The sending speed is determined by

the current computer's hardware and software performance; generally the continuous sending speed is about 2000 fps (standard data frame, 11 bytes, 1 Mbps). If the sending speed is too fast, the remote receiving device may overflow and stop responding; in this case users can add an appropriate delay in the application to reduce the sending speed.

Each frame has a timeout limit during sending: for a single frame the timeout is about 2 seconds; when sending multiple frames at once the last frame has a 2-second timeout and the others 1 second. Send timeouts generally occur when the CAN bus is busy and the current node has a low priority; they are not function-call or communication errors, and users can implement retransmission in their programs (send timeouts are rare in low- and medium-speed networks). Therefore, when designing a system, note that CAN-bus occupancy should not exceed 60-70% of the bus capacity.

12. How should the Receive function be used effectively?

The device driver provides a 100000-frame software receive buffer, giving application programmers sufficient reaction and processing time. When the software receive buffer overflows, the driver calls ResetCAN to reset the CAN bus and sets the data-overflow interrupt flag in the CAN status. Note that both software buffer overflow and CAN-controller hardware buffer overflow use this flag.

When Wait is 0, the function returns immediately with the number of frames successfully read, or 0 if the receive buffer is empty. When Wait is non-zero, if data is already in the receive buffer the function returns immediately with the number of frames read; if the buffer is empty, the function waits for a specified timeout or until data is received before returning the number of frames successfully received. When Wait is 0xFFFFFFFF it waits indefinitely until data is received; it is advisable not to set Wait too large, and indefinite waiting should be used with even more caution.

When nFrames equals 0, the function is actually a notification-message return and does not read the receive buffer; this is a special, skillful usage. Note: if the Receive function is called in the main thread with Wait non-zero, the application may temporarily stop responding. For polled reception, Wait should generally be set to 0.

9. Disclaimer

Thank you for purchasing GCAN series hardware and software products.

1. Copyright and Translation Notice

The original Chinese-language version of this document was prepared by Shenyang Guangcheng Technology Co., Ltd., which retains the copyright to the original document and its contents.

This English version has been independently translated and prepared by Shenzhen MoCAN Electric Co., Ltd. based on its understanding of the original Chinese documentation. It is intended solely for overseas customer reference and technical support purposes. It has not been officially reviewed, verified or approved by Shenyang Guangcheng Technology Co., Ltd., and should not be regarded as an official or authoritative English version of the original documentation.

The English translation, editorial work and original layout elements created by Shenzhen MoCAN Electric Co., Ltd. are protected by applicable copyright laws. No part of this English version may be reproduced, redistributed, republished or used for commercial purposes without prior written permission from Shenzhen MoCAN Electric Co., Ltd., except as permitted by applicable law.

GCAN is a registered trademark of Shenyang Guangcheng Technology Co., Ltd.

2. Product Information and User Responsibility

Product specifications, technical parameters, functions and other information provided in this document are for reference purposes and may be subject to change without prior notice.

Users are responsible for evaluating the suitability of the products for their specific applications and ensuring proper installation, configuration and operation in accordance with the relevant technical documentation and applicable safety requirements.

For product selection, technical clarification or application-specific requirements, users are advised to consult the manufacturer or an authorized supplier before making purchasing or implementation decisions.

3. Interpretation

This English document is provided as a reference translation of the original Chinese-language documentation. While reasonable efforts have been made to ensure the accuracy and consistency of the translation, differences in technical terminology, interpretation or expression may occur.

In the event of any discrepancy, ambiguity or inconsistency between this English version and the original Chinese-language documentation, the original Chinese-language version shall prevail.

For technical questions or clarification regarding the product specifications and intended functions, users are encouraged to contact Shenyang Guangcheng Technology Co., Ltd. or their authorized supplier.