USBCAN-I Pro

Industrial High-performance USB-CAN Interface Card

User Manual



Document Version: V4.46 (2024/12/26)

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

Revision History

Version	Date	Description
V1.00	2013/06/16	Document created
V2.01	2013/12/20	Device operating parameters corrected
V3.01	2014/10/22	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	2020/01/18	Document structure adjusted
V4.40	2022/06/06	Save-data file format added
V4.41	2022/07/06	Document corrected
V4.42	2022/09/05	Document corrected
V4.43	2022/09/16	Document corrected
V4.44	2023/01/05	Software updated
V4.45	2024/03/05	Contact information updated
V4.46	2024/12/26	Parameters adjusted

Contents

Revision History 2
Contents 3
1. Product Overview 5
1.1 Introduction 5
1.2 Features 5
1.3 Typical Applications 5
2. Device Installation 6
2.1 Device Dimensions 6
2.2 Interface Definition and Functions 6
2.3 Driver and Software Installation 7
2.3.1 Driver and Software Installation 7
2.3.2 Driver and Software Uninstallation 7
3. Device Usage 7
3.1 Connecting to the PC 7
3.1.1 USB Bus Power Supply Mode 8
3.2 Connecting to the CAN-bus 8
3.3 CAN-bus Terminal Resistor 9
3.4 System Status Indicators 9
4. GCANTools Software 11
4.1 Starting the Software 11
4.2 Auto Baud Rate Detection 13
4.3 Data Receive Functions 15
4.3.1 Save Data and Real-time Save 15
4.3.2 Pause Display 16
4.3.3 Display Mode 16
4.3.4 Clear 17
4.3.5 Filter Settings 17
4.3.6 Error Frame Display 18
4.3.7 Advanced Mask 19
4.4 Data Send Functions 19
4.4.1 Normal Mode Send 19
4.4.2 List Mode Send 19
4.4.3 File Send 20

4.5 Bus Diagnosis 21
4.6 OBD II Vehicle Diagnostics 22
4.7 CANopen Master Function 24
4.7.1 NMT Command 25
4.7.2 SYNC Message 25
4.7.3 EMERGENCY Message 26
4.7.4 PDO 26
4.7.5 SDO 27
4.7.6 NODE GUARDING 27
4.7.7 Servo Drive Control 28
4.8 Curve Function 28
4.9 Smart Decode 29
4.10 DBC Send 30
4.11 DBC Rec 31
5. Linux System Usage 32
6. Secondary Development 35
7. Technical Specifications 36
8. Frequently Asked Questions 37
9. Disclaimer 40
Copyright and Translation Notice 40
Product Information and User Responsibility 40
Interpretation 40

1. Product Overview

1.1 Introduction

The USBCAN-I Pro is a high-performance CAN-bus communication interface card integrating one CAN channel. Compatible with the USB 2.0 full-speed specification, it uses the USBCAN-I Pro high-performance CAN interface. A PC can be quickly connected to a CAN-bus network through the USB port, forming a CAN-bus network control node for data processing and acquisition in field-bus laboratories, industrial control, smart communities, automotive electronics networks and other CAN-bus applications.

The USBCAN-I Pro is a powerful tool for CAN-bus product development and 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 a CAN electrical-isolation protection module, which protects the device from transient over-current/over-voltage damage and enhances reliability in harsh environments.

The USBCAN-I Pro supports Windows XP/Win7/Win8/Win10 (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-I 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 USB 2.0 full-speed; compatible with USB 1.1 and USB 3.0
- Integrates one CAN-bus channel using a pluggable terminal-block wiring method
- Supports CAN 2.0A and CAN 2.0B frame formats; compliant with ISO/DIS 11898
- CAN-bus baud rate programmable from 5 kbps to 1 Mbps
- Powered from the USB bus (+5V DC, 130 mA)
- CAN-bus interface electrically isolated; isolation withstand voltage 3000V 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 throughput: 14000 fps
- CAN-side receive message timestamp accuracy up to 1 us
- Supports Windows XP, Win7, Win8 and Win10
- Supports Linux operating systems
- Supports the GCANTools test software
- 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

This chapter describes how to connect the USB-CAN interface card to a PC and the precautions for first-time use.

2.1 Device Dimensions

Device outline dimensions (length, including terminals) 96mm x (width) 58mm x (height) 24mm. The diagram is shown in Figure 2.1.

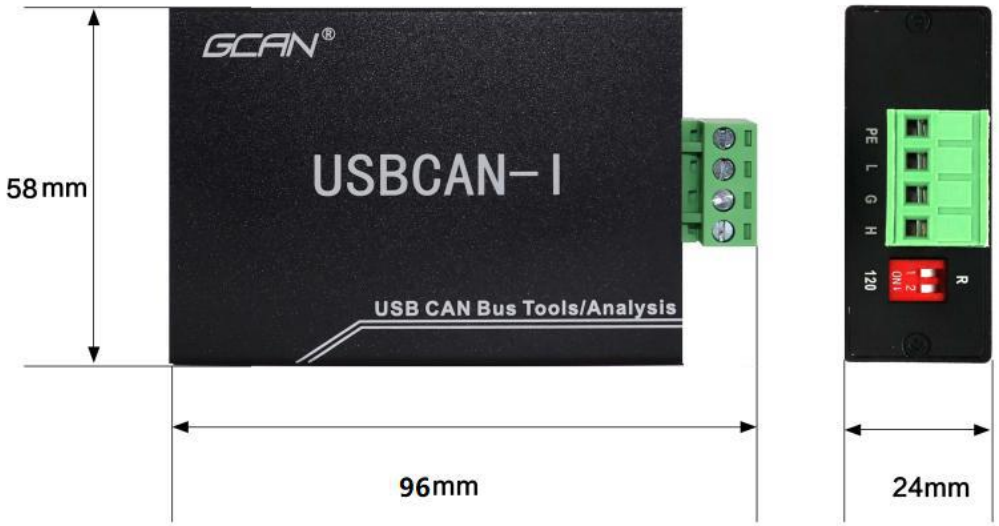

Figure 2.1 Outline dimensions of USBCAN-I Pro

2.2 Interface Definition and Functions

The USBCAN-I Pro integrates one USB port and one standard CAN-bus port. The CAN-bus port is brought out through a 4-pin pluggable terminal block, which can be used to connect one CAN-bus network or a CAN-bus device. 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 port

Pin (left to right)	Port	Name	Function
1	USB	USB	Powers the USBCAN; connects to the PC

Table 2.1 USB interface definition



Figure 2.3 Position of the CAN-bus port

Port	Name	Function
CAN	PE	Shield, connected to the case
	L	CAN_L signal line (CAN low)
	G	CAN_GND ground
	H	CAN_H signal line (CAN high)

Table 2.2 CAN-bus signal assignment of the USBCAN-I Pro

2.3 Driver and Software Installation

Before installing the driver and software, ensure that the Windows account you are logged in with has administrator privileges or the relevant permissions to install drivers and software; otherwise the installation may fail. To confirm the Windows account permissions: Control Panel - User Accounts.

2.3.1 Driver and Software Installation

You can install the driver and software together by directly installing the GCANTools software. To manually install the driver, open the "driver" folder on the CD and select the installation file matching your system (32/64-bit): DriverSetup.exe / DriverSetup64.exe.

2.3.2 Driver and Software Uninstallation

Run DriverSetup.exe / DriverSetup64.exe and click the "Uninstall" button to uninstall the installed device driver. You can uninstall the GCANTools software through "Add/Remove Programs".

3. Device Usage

3.1 Connecting to the PC

The USBCAN-I Pro is plug-and-play. You can power it directly from the PC's USB port. If USB power is insufficient, an external power supply is required (only the USBCAN-II Pro supports external power supply).

3.1.1 USB Bus Power Supply Mode

USB bus power supply is suitable for most applications, for example when the USBCAN-I Pro is the only device connected to the USB port. Connect the PC and the USBCAN-I Pro directly with the supplied USB cable; the cable provides +5V power to the card. The PWR and SYS indicators light up, indicating the device is working normally and in a standby state.

3.2 Connecting to the CAN-bus

To connect the USBCAN-I Pro to the CAN bus, 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 farthest endpoints on the bus must be fitted with 120Ω terminal resistors; if the number of nodes is greater than 2, intermediate nodes do not need 120Ω resistors. Branch connections should not exceed 3 m in length. The CAN-bus connection is shown in Figure 3.1.

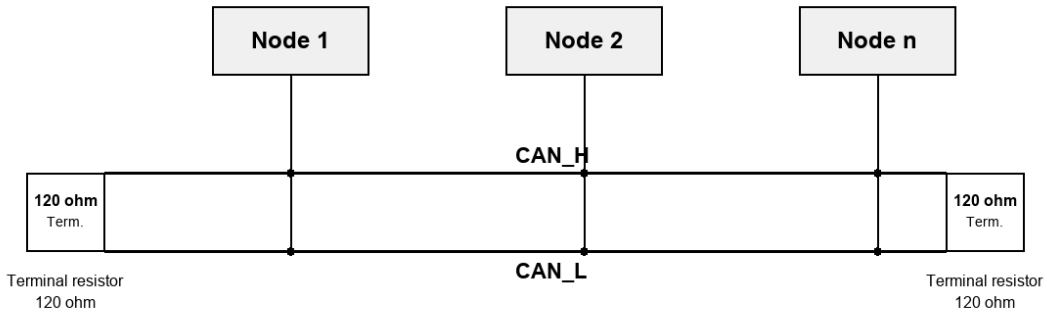

Figure 3.1 Topology of the CAN-bus network

Note: The CAN-bus cable can be ordinary twisted pair or shielded twisted pair. The theoretical maximum communication distance mainly depends on the baud rate; see Table 3.1 for the relationship between maximum bus length and baud rate. For communication distances over 1 km, ensure the conductor cross-section is greater than 1.0mm²; the specification should be selected according to the distance and appropriately increased as the distance lengthens.

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 vs. maximum bus length

3.3 CAN-bus Terminal Resistor

To enhance CAN communication reliability and eliminate terminal signal reflection, terminal matching resistors are usually added at the two farthest endpoints, as shown in Figure 3.2. The resistor value is determined by the characteristic impedance of the cable. For example, if the characteristic impedance of the twisted pair is 120Ω, the two endpoints should also integrate 120Ω terminal resistors.

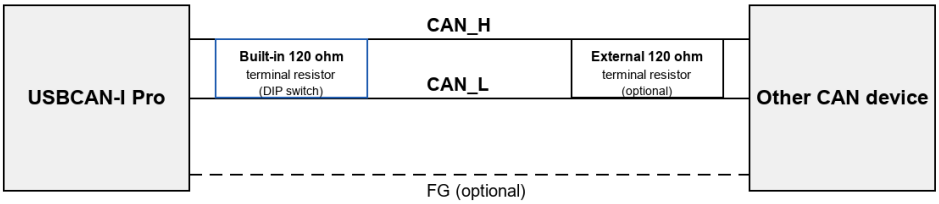

Figure 3.2 Connection of the USBCAN-I Pro with other CAN node devices

Note: A 120Ω terminal resistor is already built into the USBCAN-I Pro. A DIP switch next to the 4-pin terminal selects whether the resistor is connected to the bus. Switches R1 and R2 control the same resistor; operate R1 and R2 together and set them to ON to enable the resistor.

3.4 System Status Indicators

The USBCAN-I Pro has one PWR, one SYS, one RX and one TX indicator to show the operating status. The functions of these four indicators are listed in Table 3.2, and the CAN-bus states they indicate are listed in Table 3.3.

Indicator	Color	Indication
PWR	Green	Power indicator
SYS	Green	System indicator
RX	Green	CAN receive indicator
TX	Green	CAN transmit indicator

Table 3.2 Indicators of the USBCAN-I Pro

After power-on, all four indicators light up together; afterwards PWR and SYS remain steady on, indicating the device is powered and initialization is complete. Otherwise, there is a power or other fault; contact our customer service. When the USB interface is connected normally and PC software calls the USBCAN device, the SYS indicator blinks. When data is sent or received on the CAN bus, the corresponding RX and TX indicators blink.

Indicator	Status	Indication
PWR	On	Power supply normal
	Off	Power supply fault
SYS	Steady on	Initialization passed, standby

	Off	Initialization failed
	Blinking	PC software is calling the device
TX, RX	Off	No data on the CAN bus
	Blinking	Data transfer on the CAN bus

Table 3.3 Indicator status of the USBCAN-I Pro

4. GCANTools Software

GCANTools is dedicated debugging and analysis software for the Windows platform. With it, users can intuitively and quickly send and receive CAN-bus data. This chapter introduces the main functions of GCANTools.

4.1 Starting the Software

After the Windows driver is installed, connect the CAN analyzer hardware and open the software; the setting interface shown in Figure 4.1 appears.

1) "Select device type": select the version matching your hardware. For V405/502 versions select the V5 type (refer to the label on the back of the device for the exact version). Do not select other device types. After selecting the device type, click "Open Device" to see the device information. "Hardware" is our custom hardware version number and has no meaning for the user; "ID" is the hardware SN.

2) "Open Device" button: clicks it to open the USBCAN device. If "USB device open error!" is displayed, check whether the device selected in step 1 is correct and whether the driver in Device Manager is installed correctly.

3) Device display window: shows the device hardware number and SN.

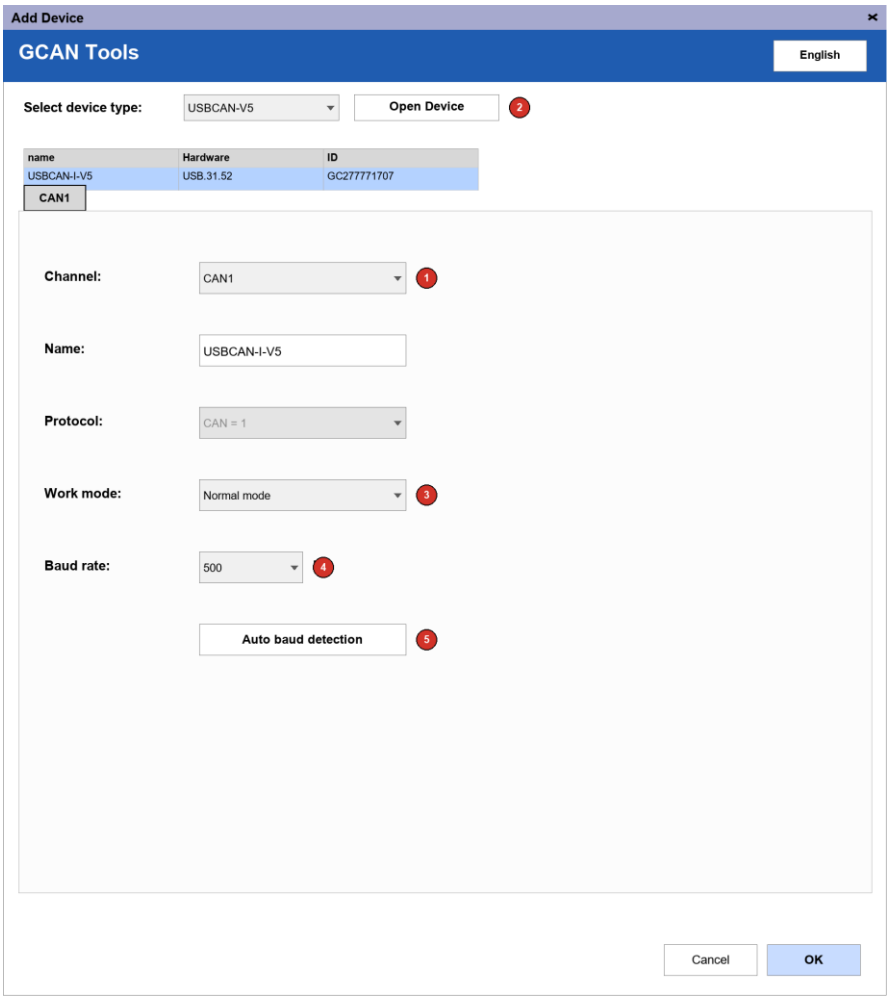

Figure 4.1 Add-device window of GCANTools

4) Work mode: select Normal mode, Listen-only mode or Self-send/self-receive mode. For normal use, select the default Normal mode.

5) Baud rate: select the baud rate here. The baud rate is critical for CAN-bus communication; determine the baud rate of the target device or bus before communicating. After selecting the device, set the work mode and baud rate below. For a dual-channel device, set CAN1 and CAN2 separately via the tabs. The baud rate cannot be entered manually; it must be selected from the drop-down menu. The standard baud rates supported are shown in Figure 4.2.

For a special baud rate, click the Custom button and enter a hexadecimal code. Some special baud-rate values are listed in the table below. For other baud rates, contact us and we will calculate the register setting.

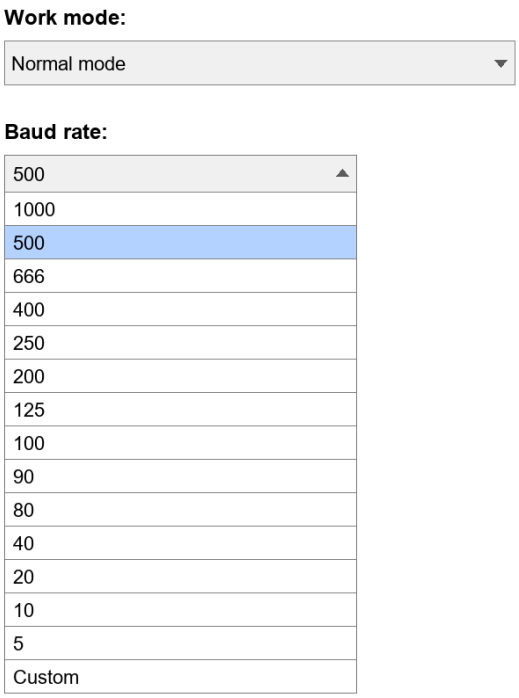

Figure 4.2 Work mode and baud-rate selection

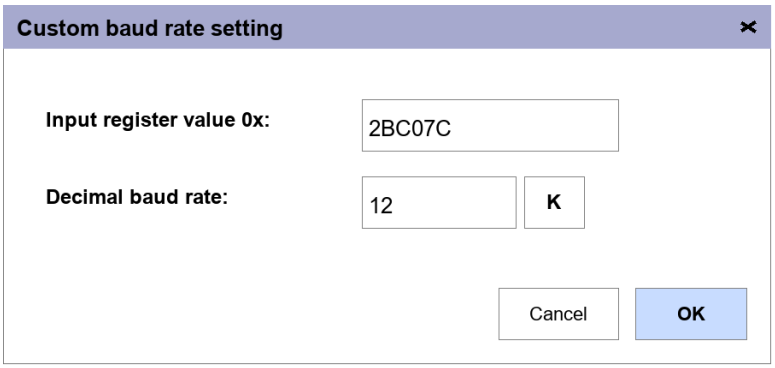

Figure 4.3 Custom baud-rate setting

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

6) Auto baud detection button: if you do not know the baud rate of the target bus or device, you can select "Auto baud detection". The success condition for detection: the device under test is powered on and the CAN side is active.

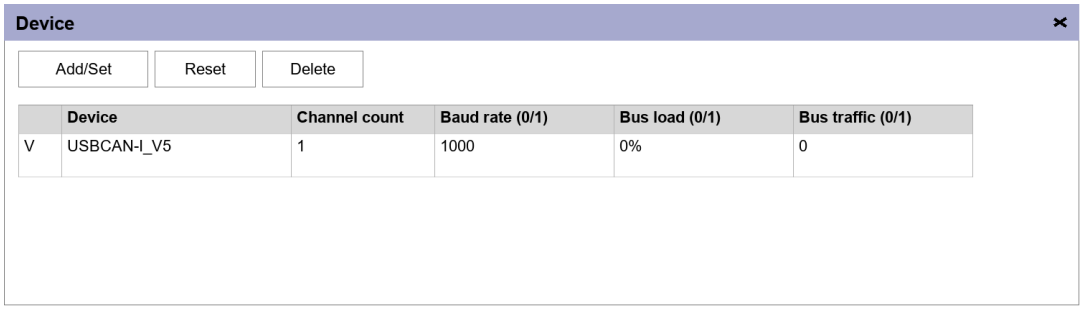

Figure 4.4 Device management window

7) 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 blinking (two consecutive blinks). The interface after a successful connection is shown in Figure 4.8.

Note: setting the baud rate is critical. Many users report no data or bus errors after connecting because they clicked OK to open the device without setting the baud rate. Whether you use the device as master or slave, once connected to the CAN bus you must set the device baud rate to match the target device.

4.2 Auto Baud Rate Detection

If you do not know the target device's baud rate, use the auto baud-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 normally with each other and active CAN data on the bus. "Enhanced-mode detection" requires the device under test to be powered on with the CAN channel working normally, but does not require the device to actively send CAN data or active CAN data on the bus. Before detection, ensure the wiring is correct and that two 120Ω resistors are connected on the bus. You can measure the resistance between CAN_L and CAN_H with a multimeter without powering the devices on the bus; it should be about 60Ω.

Note: for the CAN-bus system of a vehicle, select "Normal silent-mode detection". Using "Enhanced-mode detection" may cause the vehicle instrument cluster to report errors.

The auto baud-detection function has two modes by search range: A. Standard baud-rate detection (identifies standard CAN baud rates one by one); B. Full-range baud-rate detection (manually enter a range and the software matches all baud rates within it). For example, if

the baud rate of a CAN-bus system is unknown and standard detection shows "No suitable baud rate found", set the full-range detection from 5 kbps to 1000 kbps to find the required baud rate.

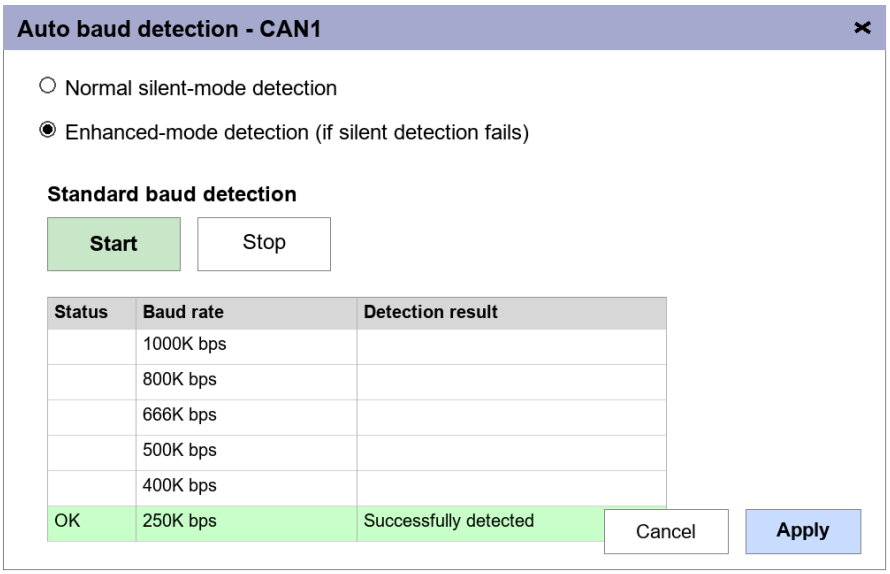

Figure 4.5 Standard baud-rate detection

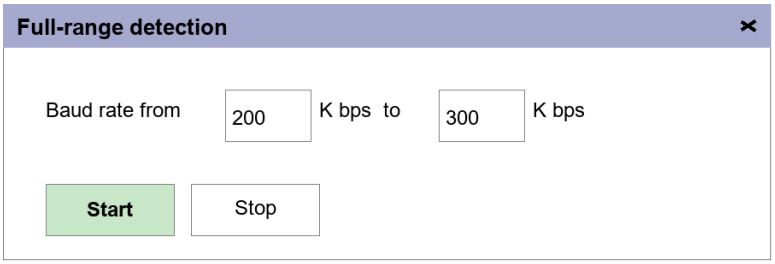

Figure 4.6 Full-range baud-rate detection

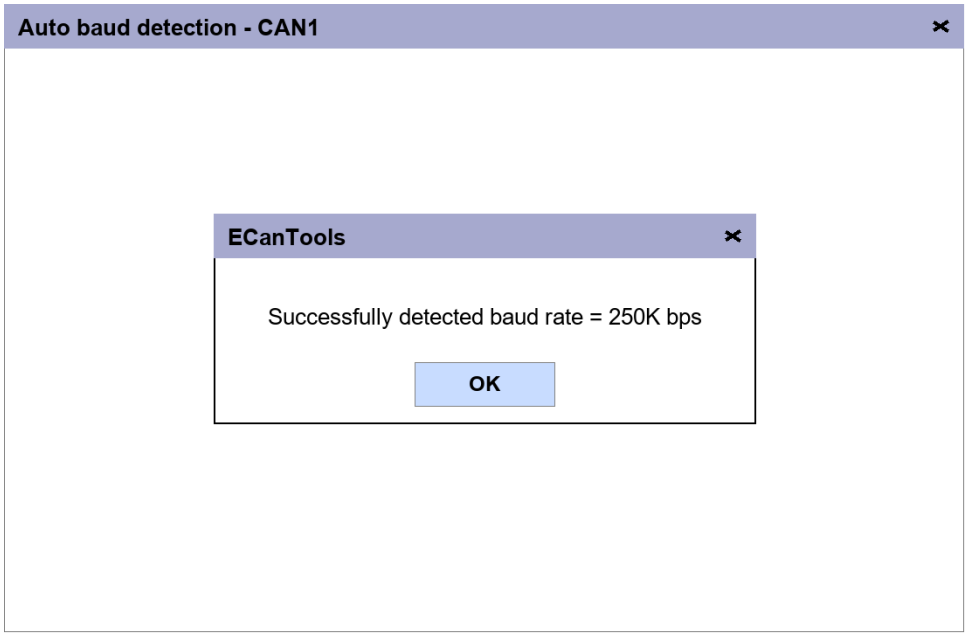

Figure 4.7 Auto baud-rate detection result

4.3 Data Receive 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 it. The receive window is shown in Figure 4.8.

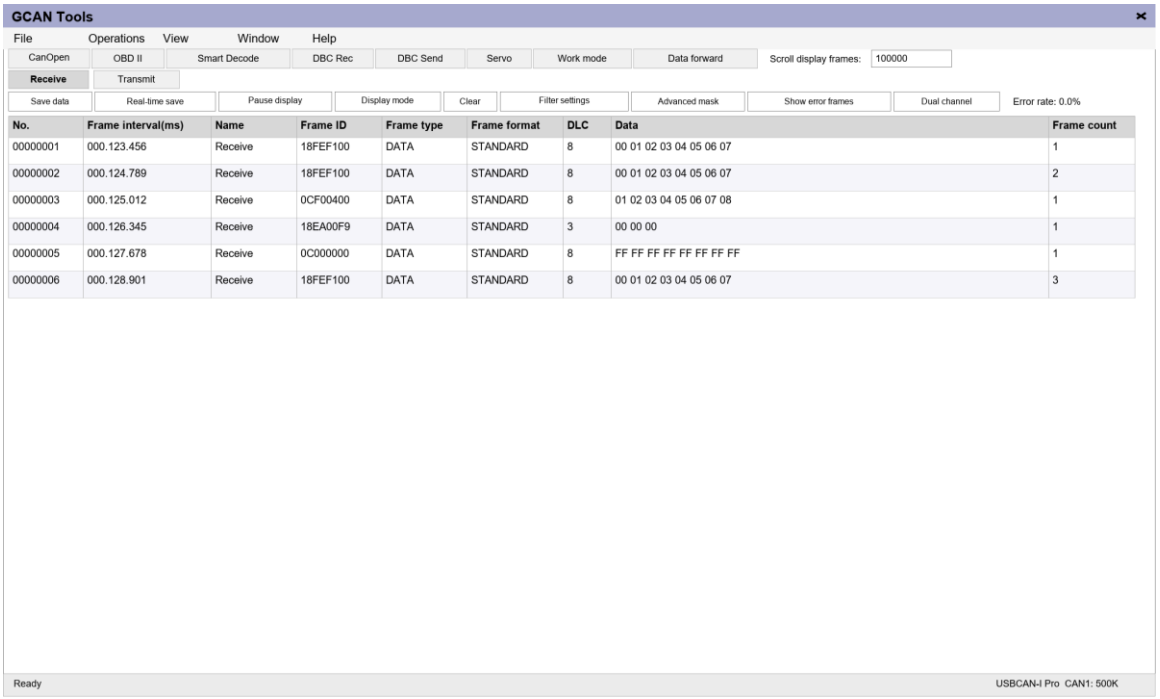

Figure 4.8 Main window (receive/transmit)

4.3.1 Save Data and Real-time Save

You can save all data in the current send/receive list locally. The save formats are listed in the table below.

Save data
Text file
Binary file
Batch file
Excel file
ASC file
BLF file

Figure 4.9 Save-data formats

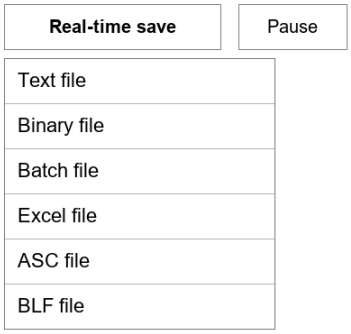

Figure 4.10 Real-time save formats

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 in Notepad and re-sent to the bus. For the batch-file format, see section 4.4.3.

Click "Real-time save" on the toolbar, set the text type and file name, and real-time saving begins (sets save point A). Click again (sets save point B) and the system stops saving and writes all data from A to B into the file. Note: the system does not write data in real time; during real-time saving, the saved file will not contain any data.

4.3.2 Pause Display

Click Pause to pause the scrolling data window. While paused, the device and software still receive data normally; only the data window does not refresh. Click "Resume display" to resume scrolling.

4.3.3 Display Mode

Display modes include Scroll mode and Statistics mode.

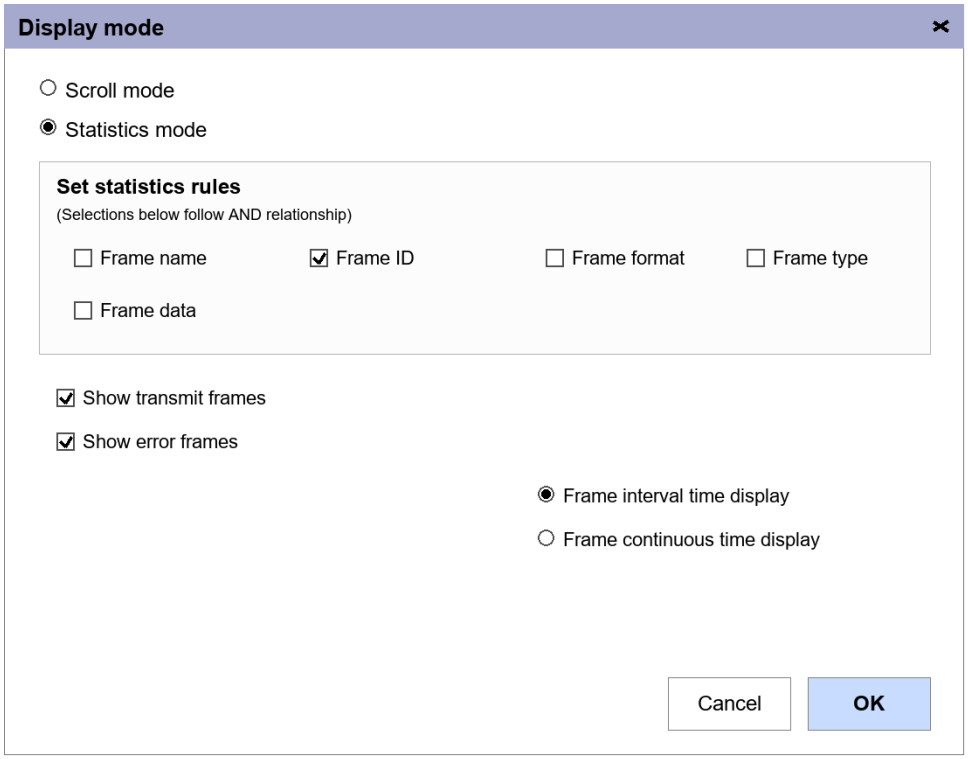

Figure 4.11 Display-mode settings

In Scroll mode the received data scrolls continuously downward in the receive list; the current window shows the latest data. In Statistics mode data is grouped and displayed according to set rules, for example grouping packets with the same ID together and showing the packet count. Statistics mode helps engineers capture newly generated or changed data on the bus.

4.3.4 Clear

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

4.3.5 Filter Settings

The receive filter can set a filter ID or ID range. When a filter is set, the software displays only the filtered ID (range); IDs outside the filter range are filtered out. Click "Filter settings" to open the filter settings window shown in Figure 4.12.

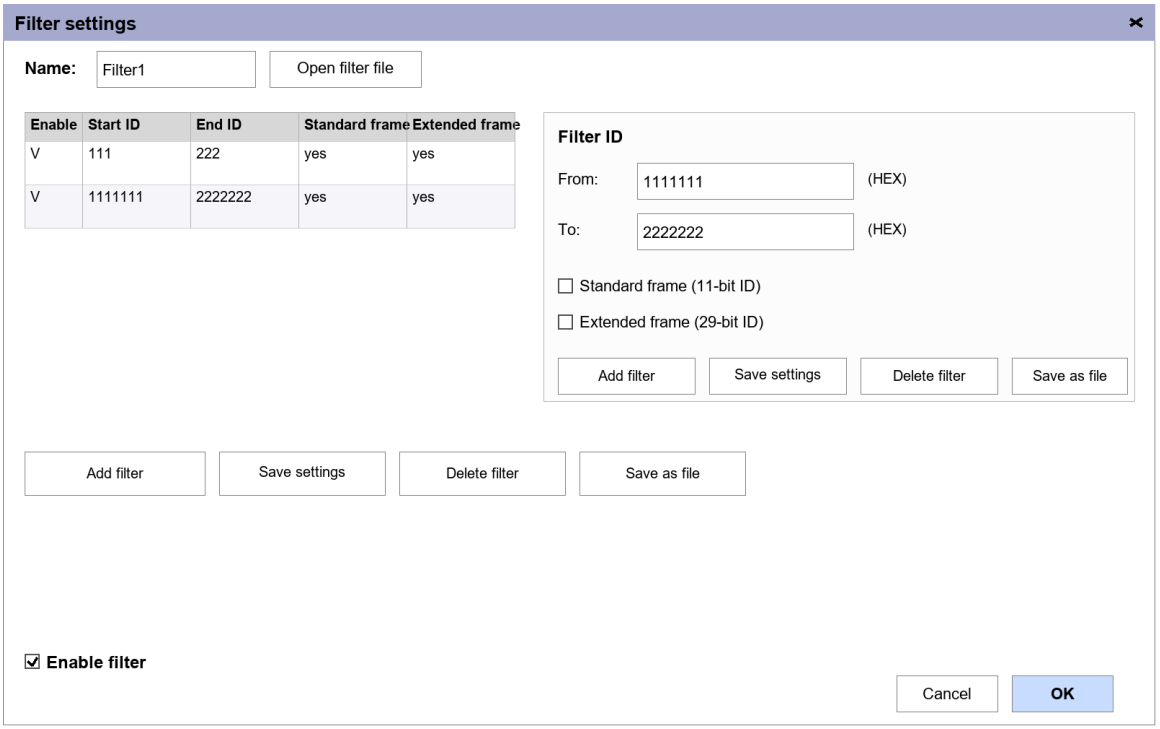

Figure 4.12 Filter settings

Click "Add filter" to start editing. Enter the ID or ID range to filter and click "Save settings" to add it to the list on the left. The filter list on the left can hold multiple entries; tick a filter range to enable it. Multiple filters can be enabled at the same time. Note: you must tick "Enable filter" for the filter to take effect. Filter files can be saved and loaded via "Open filter file".

4.3.6 Error Frame Display

The software can capture error frames on the bus. When an error frame is received, it is displayed in red in the receive list, as shown in Figure 4.13.

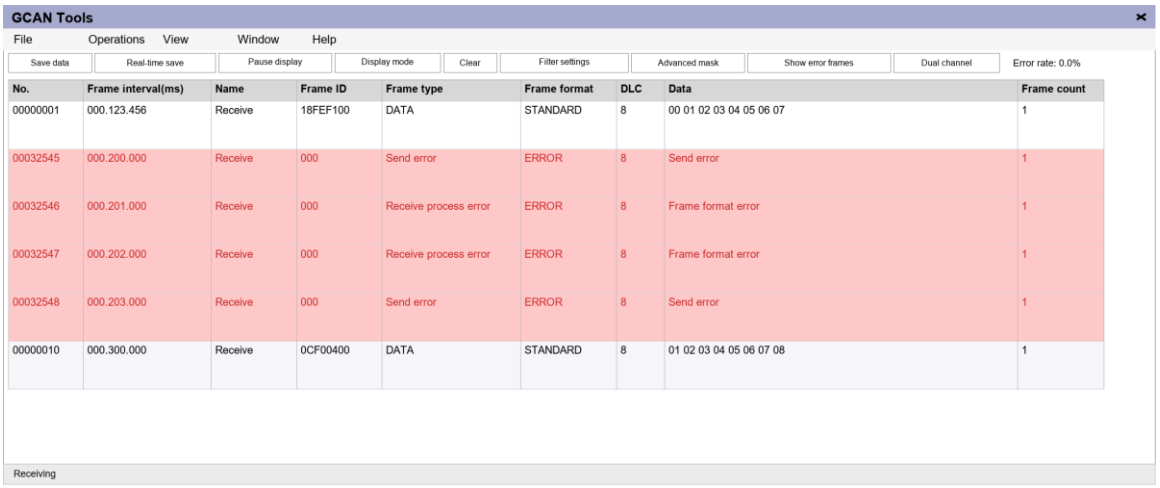

Figure 4.13 Error-frame display

Error frames can also be merged for display by setting Statistics mode.

4.3.7 Advanced Mask

GCANTools can mask specified received CAN frames. It groups received data by frame ID and merges it into a to-be-masked list on the left. Select the frame ID to mask and click the right arrow to add it to the mask list. The software then automatically masks all data beginning with that frame ID. Multiple mask frame IDs can be set.

4.4 Data Send Functions

4.4.1 Normal Mode Send

Normal mode lets you intuitively edit the frame data to send and supports special functions such as cyclic sending. Editing frame information is straightforward. Note: when entering data, separate each byte with a space, otherwise "Data format error!" will be reported.

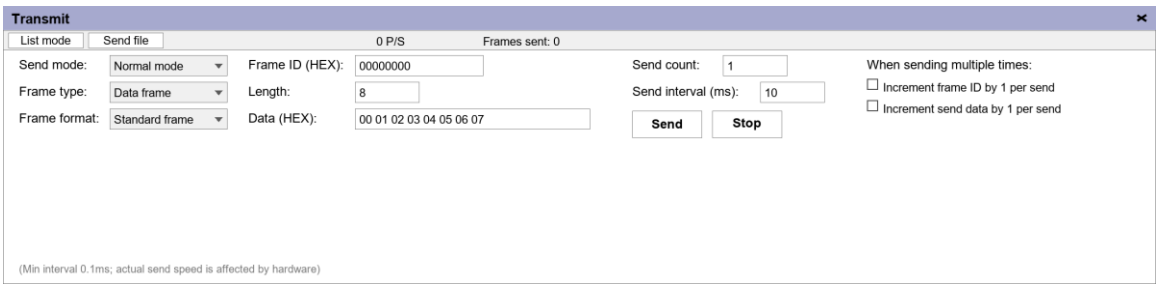

Figure 4.14 Normal-mode send panel

4.4.2 List Mode Send

Click List mode to switch the panel to list mode (click again to return to Normal mode). Click "New frame" to edit the data frame to send.

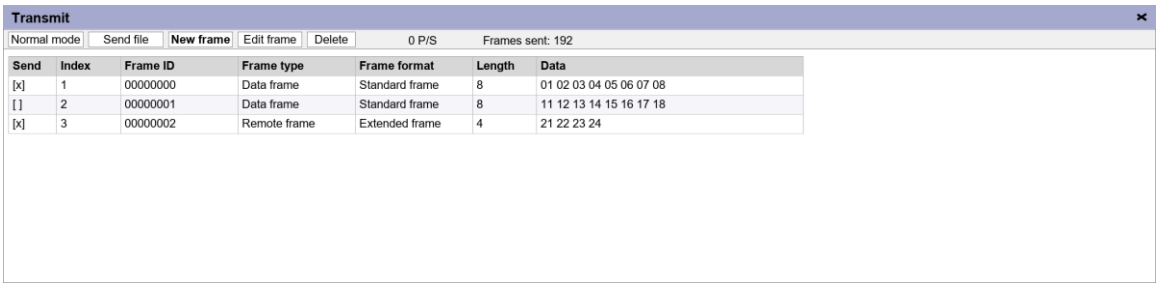

Figure 4.15 List-mode send panel

In the edit-data interface you can set the frame ID, frame type, frame format, frame length and so on. In the advanced options you can set the cyclic send method, send interval, send count, and ID increment.

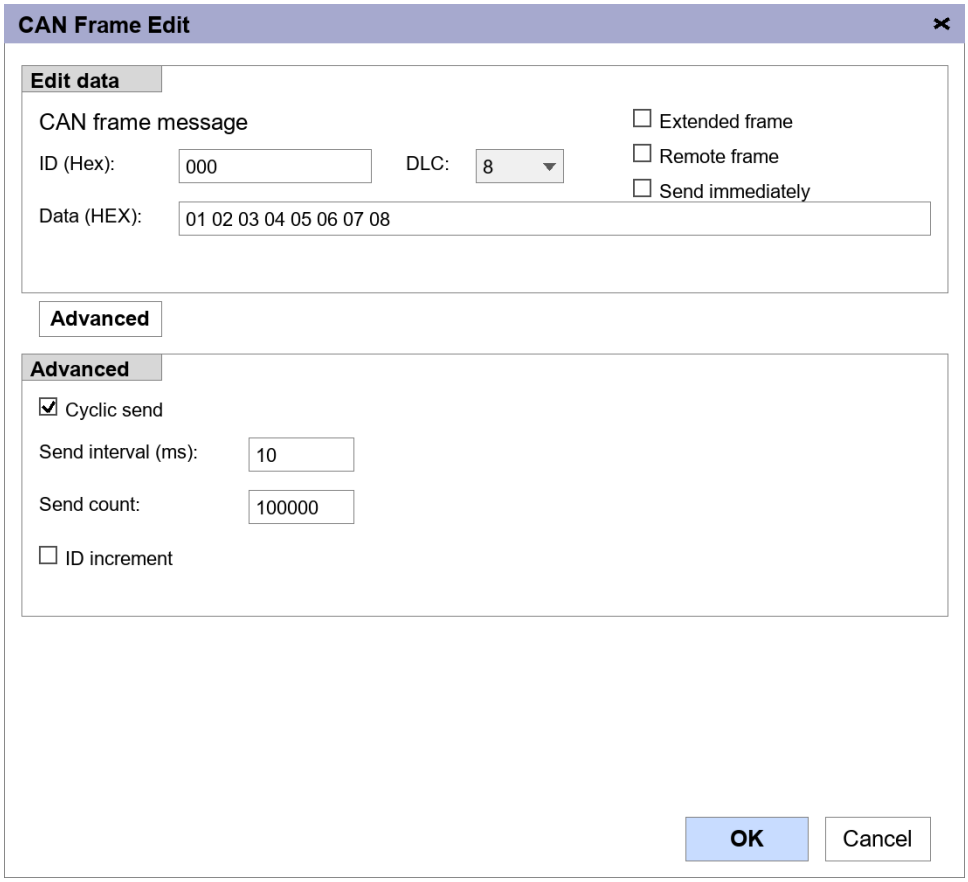

Figure 4.16 Edit-frame dialog

When set, add the CAN message to the send queue. Click the box to the left of "send" to control send/stop. When sending ends the check box automatically resets. For cyclic send packets, the number of successfully sent packets is shown in "Frames sent".

4.4.3 File Send

Select "Send file" to open the send-file dialog shown in Figure 4.17.

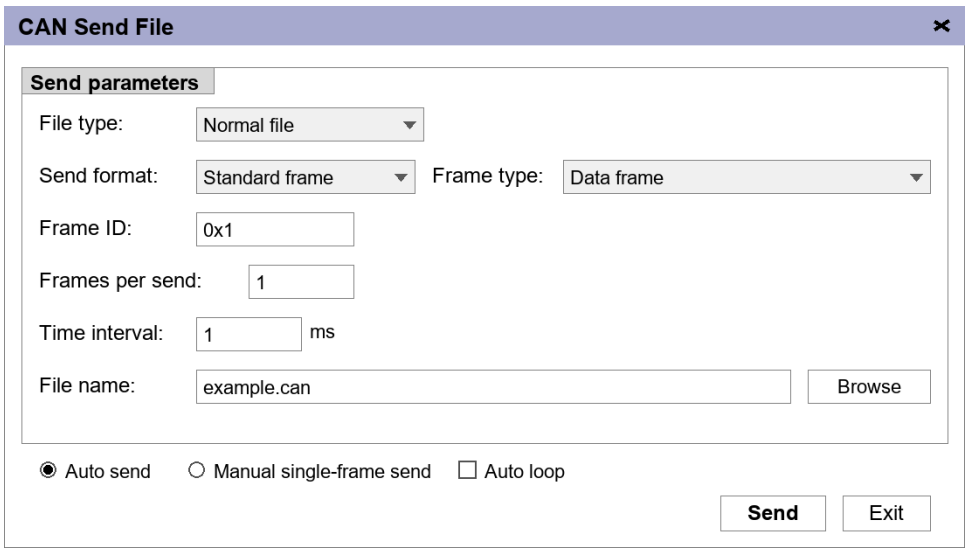

Figure 4.17 Send-file dialog

The file type can be a normal file or a batch file (.CAN). A normal file is used to program CAN-bus devices and requires you to develop the programming software yourself. The batch-file send is introduced below.

GCANTools can save received data as a batch file (.CAN). A batch file can be opened in Notepad to view the saved frame data and format. The "example.can" shown in Figure 4.18 can be found in the GCANTools installation directory.

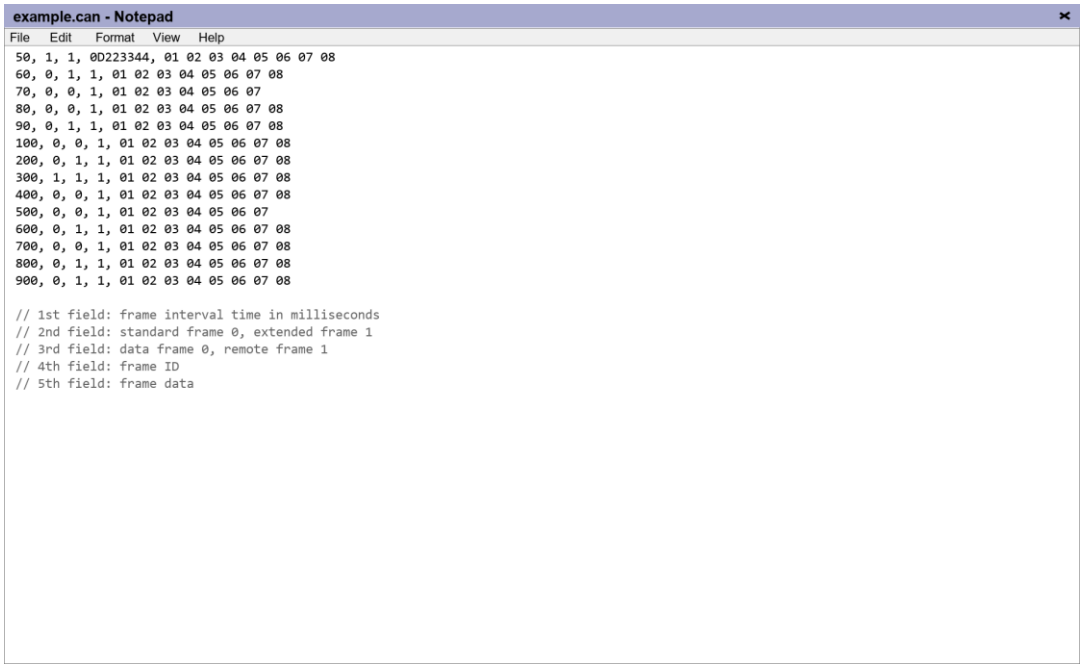

Figure 4.18 example.can batch file

After opening the batch file (.CAN) you can directly view the saved data parameters, which you can modify and add to. The meaning of each field is given in the table below. Note: a batch file must not contain any non-standard data; the comments in the figure are for explanation only and should not be added in actual editing.

Field 1	Field 2	Field 3	Field 4	Field 5
Meaning	Frame interval (ms)	Standard/Extended	Data/Remote, frame ID	Frame data
Example	50	1	1, 0D223344	01 02 03 04 05 06 07 08

Note: you may find a frame interval of 0 in a batch file. In that case GCANTools sends at the minimum send interval for the current baud rate; multiple frames will not be sent simultaneously.

4.5 Bus Diagnosis

With GCANTools you can read whether the bus is normal from the status area in the lower-right corner. Typical examples:



Figure 4.19 Bus diagnosis status

1. Error active or Error passive: indicates the baud rate is set incorrectly, the wiring is wrong, or the bus is empty/unresponsive.

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

4.6 OBD II Vehicle Diagnostics

You can use the GCANTools vehicle signal-analysis function. Connect the device to the vehicle OBD port to parse the actual values of some sensors inside the vehicle. The signal-analysis function supports only the ISO15765 protocol for household gasoline vehicles. Vehicle speed, RPM and coolant temperature can be displayed on the software dashboard shown in Figure 4.20.

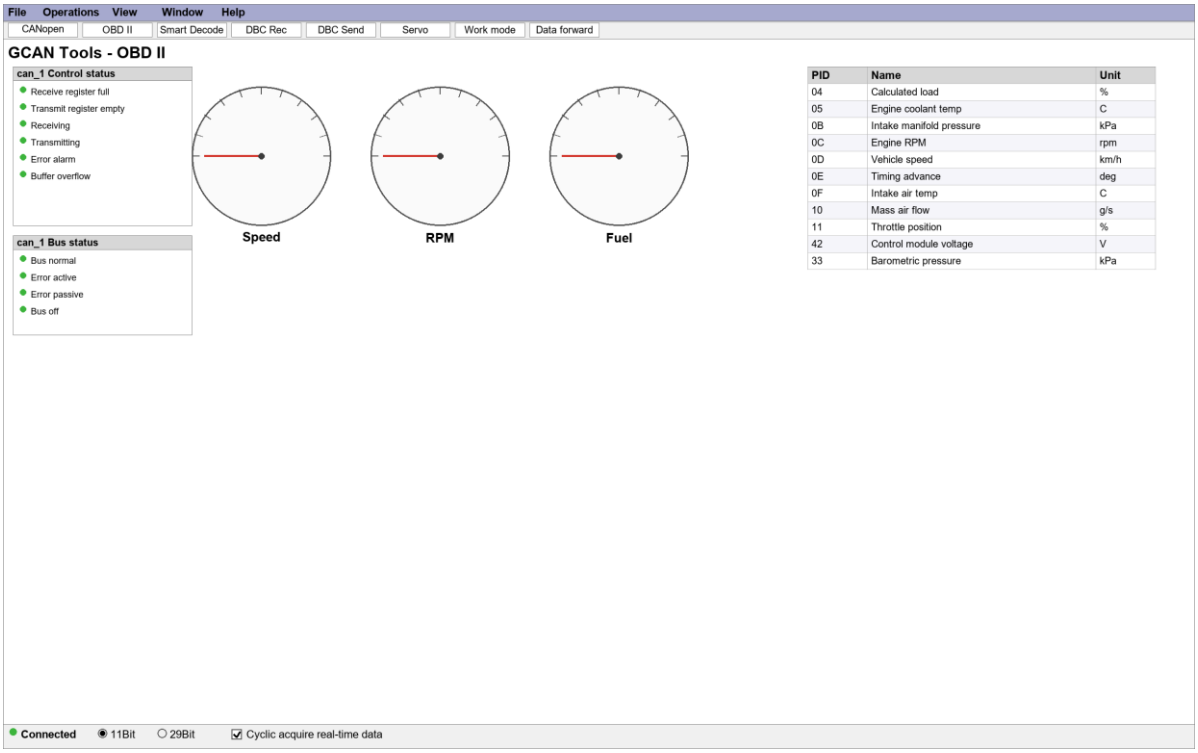

Figure 4.20 OBD II interface

This function currently supports only the CAN1 channel. You can directly view the vehicle's real-time speed, RPM and fuel level to verify the instrument-cluster readings. GCANTools can also read, parse and clear vehicle fault codes, and parse the sensor data specified in ISO15765, including engine RPM, coolant temperature, vehicle speed, voltage, intake

manifold pressure, intake air temperature, air flow rate, throttle position, oxygen sensor voltage, fuel pressure and more. The changes of these data values can be saved to the PC in real time.

To use it, first click the 11Bit button and tick "Cyclic acquire real-time data". If the USBCAN analyzer is connected to the vehicle powertrain bus, it will return parameters such as speed, RPM and coolant temperature. If no data appears, switch to 29Bit and try again. If neither mode returns data, check the baud rate and wiring. Most vehicle powertrain buses use 500 kbps.

Manual PID acquisition: you can manually acquire vehicle parameters by entering a PID value. First select the correct command type (11Bit or 29Bit). The detailed PID values (decimal) are listed in the table below.

PID	Data	Min	Max	Unit
01	Number of current fault codes	0	127	
04	Calculated load value	0	100	%
05	Engine coolant temperature	-40	215	°C
06	Short-term fuel trim (bank 1 & 3)	-100	99.22	%
07	Long-term fuel trim (bank 1 & 3)	-100	99.22	%
08	Short-term fuel trim (bank 2 & 4)	-100	99.22	%
09	Long-term fuel trim (bank 2 & 4)	-100	99.22	%
0A	Fuel pressure	0	765	kPa
0B	Intake manifold absolute pressure	0	255	kPa
0C	Engine RPM	0	9999	rpm
0D	Vehicle speed	0	255	km/h
0E	Ignition timing advance (cyl. 1)	-64	63.5	°
0F	Intake air temperature	-40	215	°C
10	Mass air flow	0	655.35	g/s
11	Absolute throttle position	0	100	%
1F	Time since engine start	0	65535	s
21	Distance with MIL active	0	65535	km
22	Fuel rail pressure (manifold vacuum)	0	5177.265	kPa
23	Fuel rail pressure (atmospheric)	0	655350	kPa
2F	Fuel level input	0	100	%
33	Barometric pressure	0	255	kPa
42	Control module voltage	0	65.535	V
43	Absolute load value	0	25700	%
44	Equivalent ratio command	0	2	
45	Relative throttle position	0	100	%
46	Ambient air temperature	-40	215	°C
47	Absolute throttle position B	0	100	%

5C	Engine oil temperature	-40	210	°C

Table 4.1 OBD II PID values

4.7 CANopen Master Function

Switch to the "CANopen" tab of GCANTools to enter the CANopen master page, shown in Figure 4.21. On the left you can click "Scan slaves" to scan all CANopen slave nodes on the current CAN bus, or manually enter a slave node ID to add a CANopen slave. Scanned or added slave nodes are displayed below; double-click a node ID to activate the slave-communication interface.

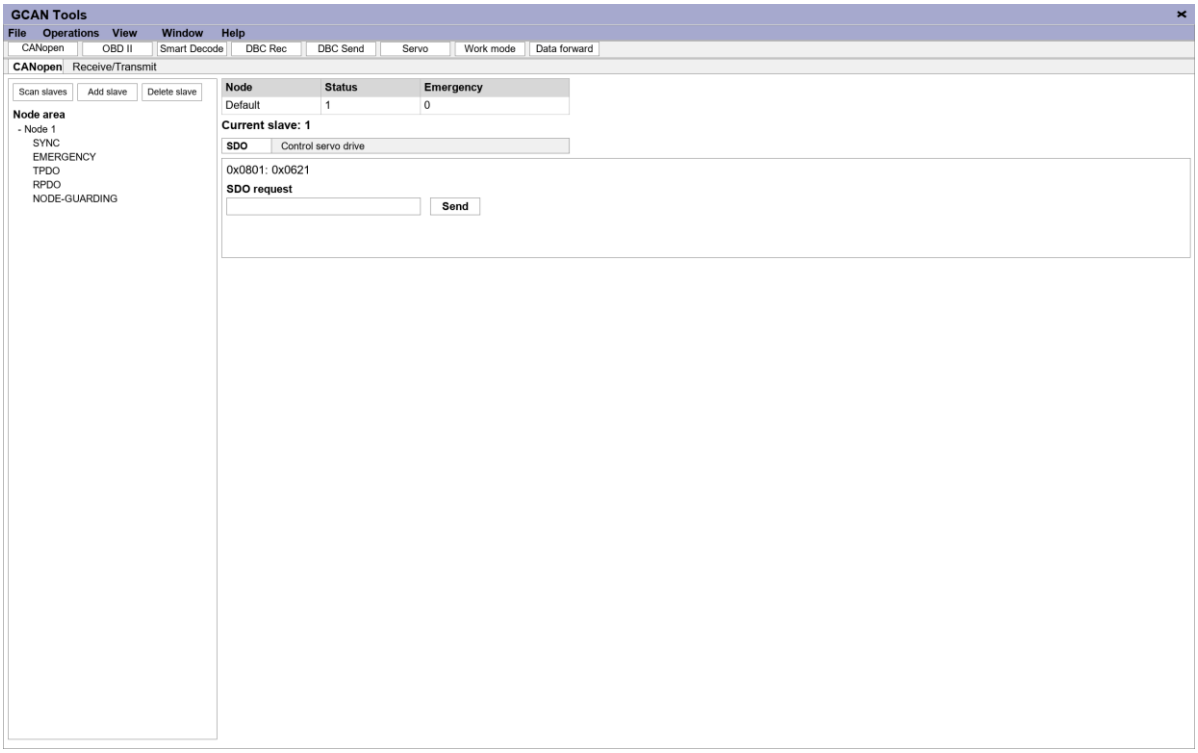

Figure 4.21 CANopen master page

The CANopen master function turns the USBCAN into a CANopen master card that can be inserted into a CANopen network. With a PC and the USBCAN you can control, manage and configure CANopen slaves. GCANTools supports common CANopen master functions such as NMT (Network Management), SYNC, EMERGENCY, PDO (Process Data Object), SDO (Service Data Object) and NODE GUARDING.

The table below lists the function codes and frame-ID ranges corresponding to common CANopen message types.

Function code	Message type	Frame ID range
0	NMT	0x000-0x001
1	SYNC	0x080
2	TIME	0x100
3	EMERGENCY	0x080 + Node ID
4	TPDO	0x180 + Node ID
5	RPDO	0x200 + Node ID

6	SDO (Tx)	0x580 + Node ID
7	SDO (Rx)	0x600 + Node ID

Table 4.2 CANopen function codes

4.7.1 NMT Command



Figure 4.22 NMT command

The NMT command (Network Management) provides network-management services such as initializing, starting and stopping nodes and detecting failed nodes, using a master-slave communication model (so there is only one NMT master). You can use NMT commands to change the operating state of slave nodes:

Start remote node - sets the node to Operational state

Stop remote node - sets the node to Stopped state

Enter pre-operational state - sets the node to Pre-operational state

Reset node - resets the node

Reset communication - resets the node communication

4.7.2 SYNC Message

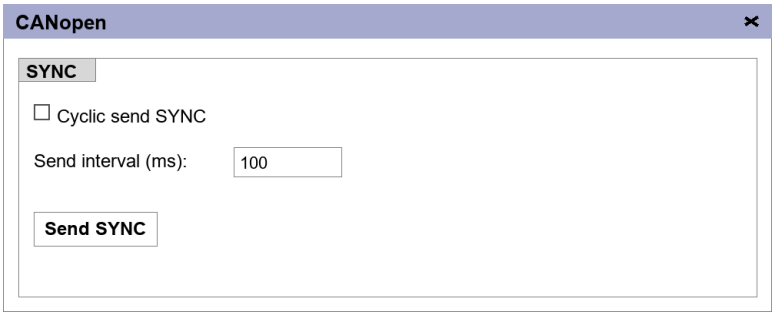

Figure 4.23 SYNC message

SYNC is the synchronization message sent by the master. All slaves with a synchronous PDO communicate with the master according to the SYNC message. You can cyclically send SYNC messages via GCANTools; all synchronous PDOs are sent according to the SYNC message.

4.7.3 EMERGENCY Message

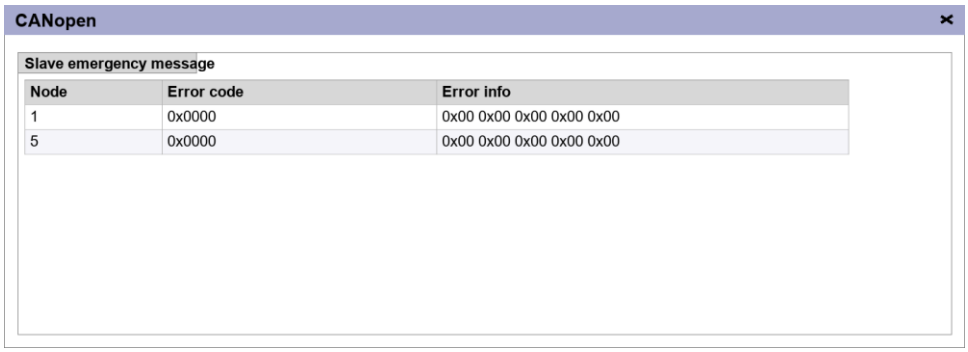

Figure 4.24 EMERGENCY message

The emergency message is triggered by a fatal error inside a device and sent by the relevant device to other devices at the highest priority. It is suitable for interrupt-type error-warning signals. GCANTools can receive EMERGENCY messages sent by slave nodes and the CANopen master function can receive and process them.

4.7.4 PDO

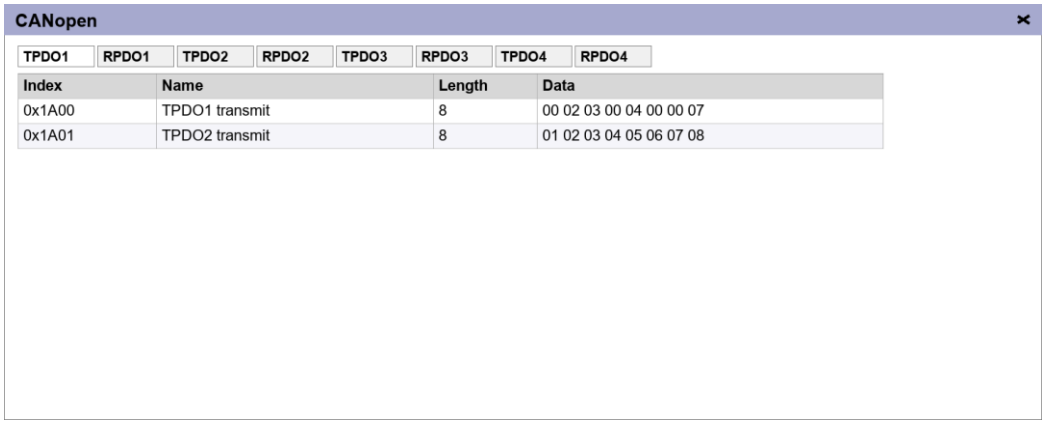

Figure 4.25 PDO

CANopen nodes use PDO to transfer data, mainly for real-time data. A Process Data Object (PDO) is a one-way transfer that does not require the receiving node to return a CAN frame for confirmation; it follows a "producer-consumer" model. GCANTools provides four pairs of send and receive PDOs (TPDO/RPDO) by default.

4.7.5 SDO

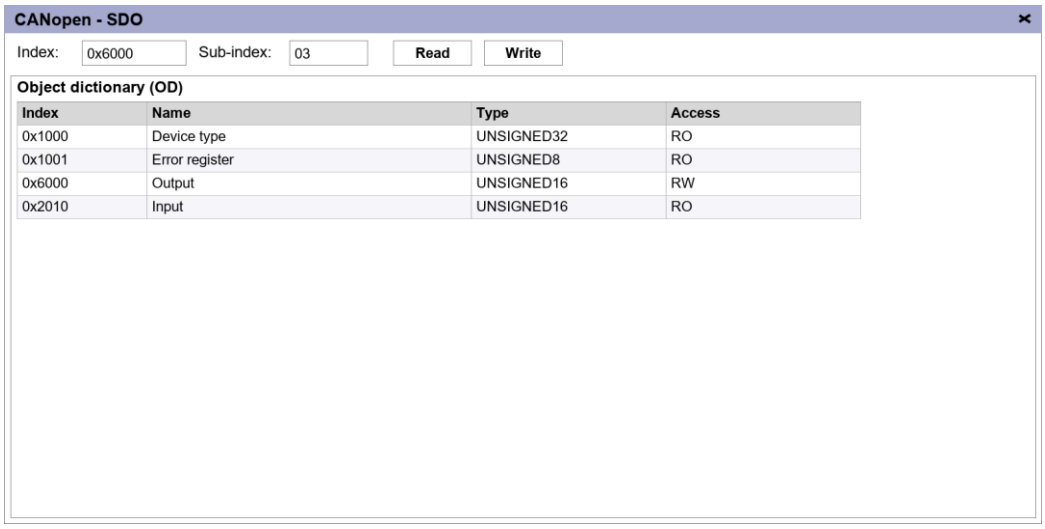

Figure 4.26 SDO and object dictionary

The SDO function of the CANopen master module is divided into send SDO and receive SDO, used to read and write the object dictionary. A Service Data Object (SDO) has the address of the receiving node (Node ID) specified and requires the receiving node to return a CAN frame to confirm reception; if confirmation times out the sending node resends the original frame. This follows the "server/client" (polling) model. In GCANTools you can interact with a slave by editing the index and sub-index, implementing read/write access to entries in the object dictionary.

4.7.6 NODE GUARDING

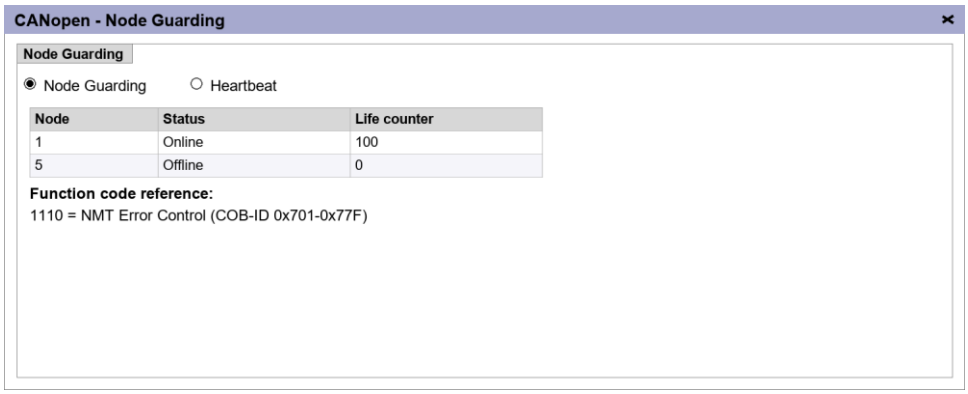

Figure 4.27 Node guarding

The node-guarding function has two implementations: the master actively sends commands to query the slave-node status (Node Guarding), or the slave periodically transmits its status by heartbeat (Heartbeat). GCANTools can also manually simulate CANopen data via a custom message; you can directly send data to a CANopen slave by editing the function code, slave node ID and data.

4.7.7 Servo Drive Control

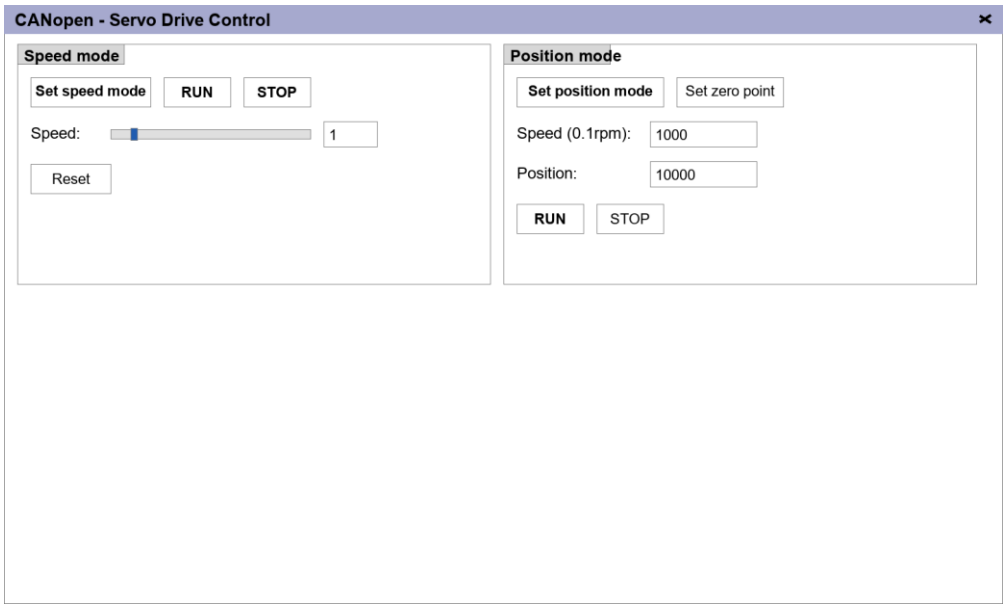

Figure 4.28 Servo drive control

This function supports only servo-motor devices compliant with the CANopen 402 standard; confirm this before use. Currently only "Set speed mode" is available for simple control of servo-motor devices. After connecting the servo motor's CAN bus, click "Scan slaves" - "Start" - "Set speed mode" - "RUN" in turn to drive the servo motor.

4.8 Curve Function

Data value changes can also be displayed in real time as a curve showing the change trajectory, as shown in Figure 4.29. You can select up to four variables to display simultaneously on the same interface.

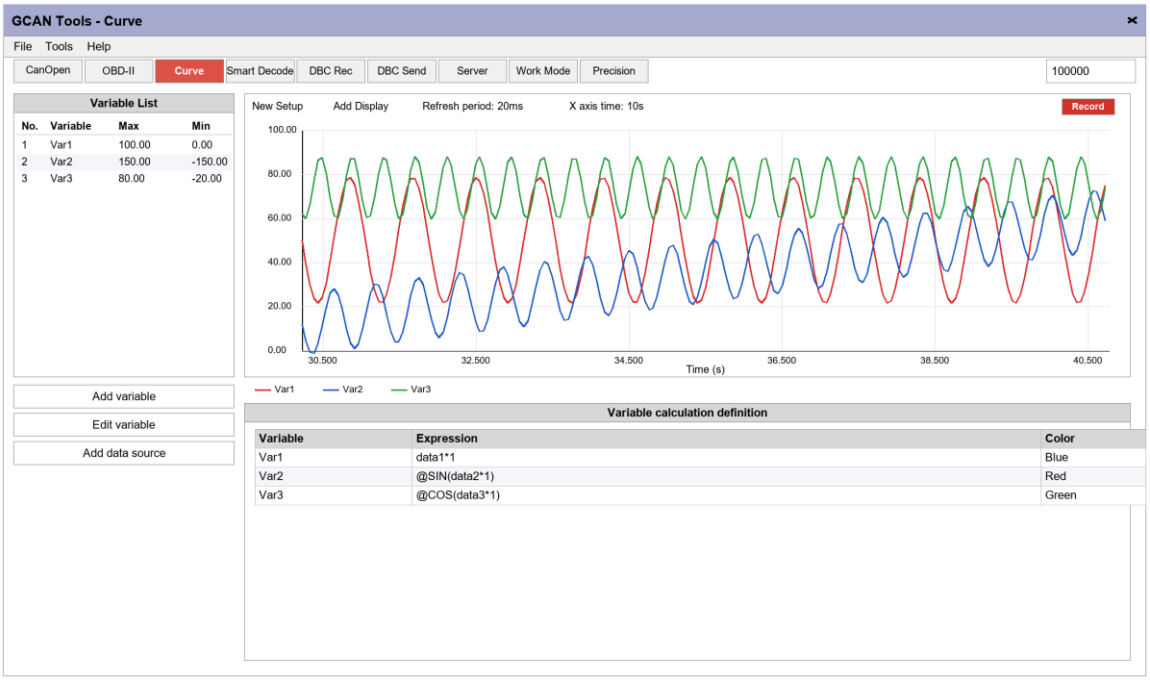

Figure 4.29 Curve display

Through the curve function you can select up to four variables and observe each data change in real time, and intuitively compare the relative changes between multiple values. It can be used to adjust engine idle speed, torque and so on. Note: set up the variables before receiving data; variables created during data reception cannot be displayed on the curve.

Click "Add variable" or "Edit variable" to add and edit variables. These variables can perform simple mathematical operations on the received raw CAN-bus data and display the results as curves. You can add multiple variables; variable names can contain letters, numbers, spaces and underscores for easy identification. The special operators available in the curve function are listed in the table below; write them strictly in the specified form.

Operator	Function
sin / cos / tan	Sine / Cosine / Tangent
sqr	Square root of a non-negative real number
log	Natural logarithm (ln x)
abs	Absolute value
exp	Exponential function base e (e^x)
Max	Current maximum of the variable
Min	Current minimum of the variable

Table 4.3 Curve-function operators

"Add data source" adds the data of a certain byte of a certain frame from the received raw CAN-bus data as a data source to the variable-calculation definition. Click "Add data source", fill in the CAN ID, Standard frame (Extended frame), Data and so on, and click OK; a corresponding statement is generated in the "Variable calculation definition". You can also write it yourself, following that format.

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

4.9 Smart Decode

The smart-decode function currently supports displaying the change frequency of frame data for a specific frame ID on the CAN1 channel. The color changes with the frequency: the faster the change, the deeper the color. The display can be set to HEX, DEC or BIN. For data filtering, click "Filter" to set filter parameters. To use it, tick "Enable receive data CAN1".

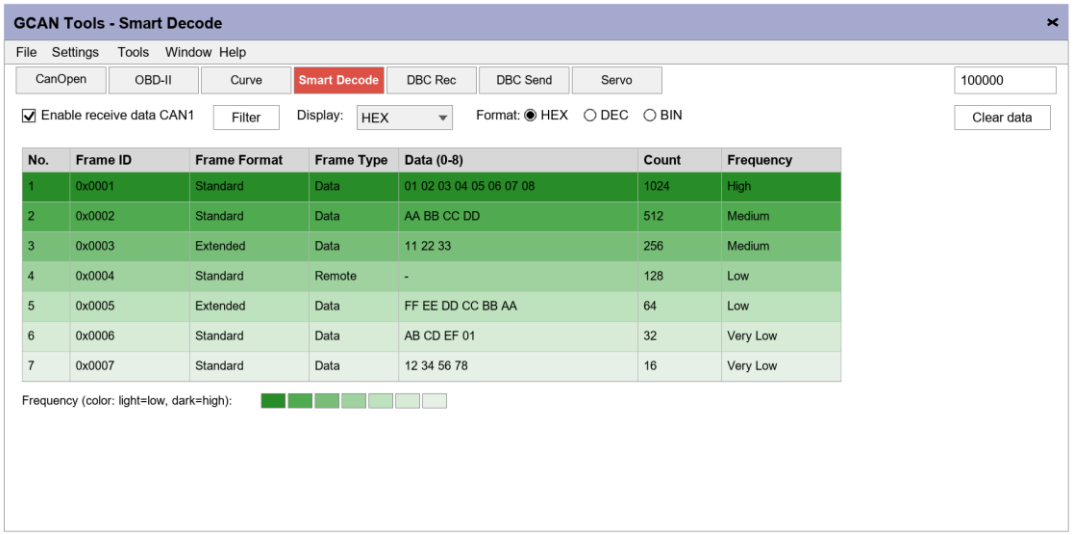

Figure 4.30 Smart decode

4.10 DBC Send

Click DBC Send on the toolbar. This tab is used to send DBC messages; each message can be sent independently without interfering with others. As shown in the figure, the left table is the DBC message selection list, the upper-right table is the to-be-sent message list, and the lower-right table is the signal list of the currently selected message.

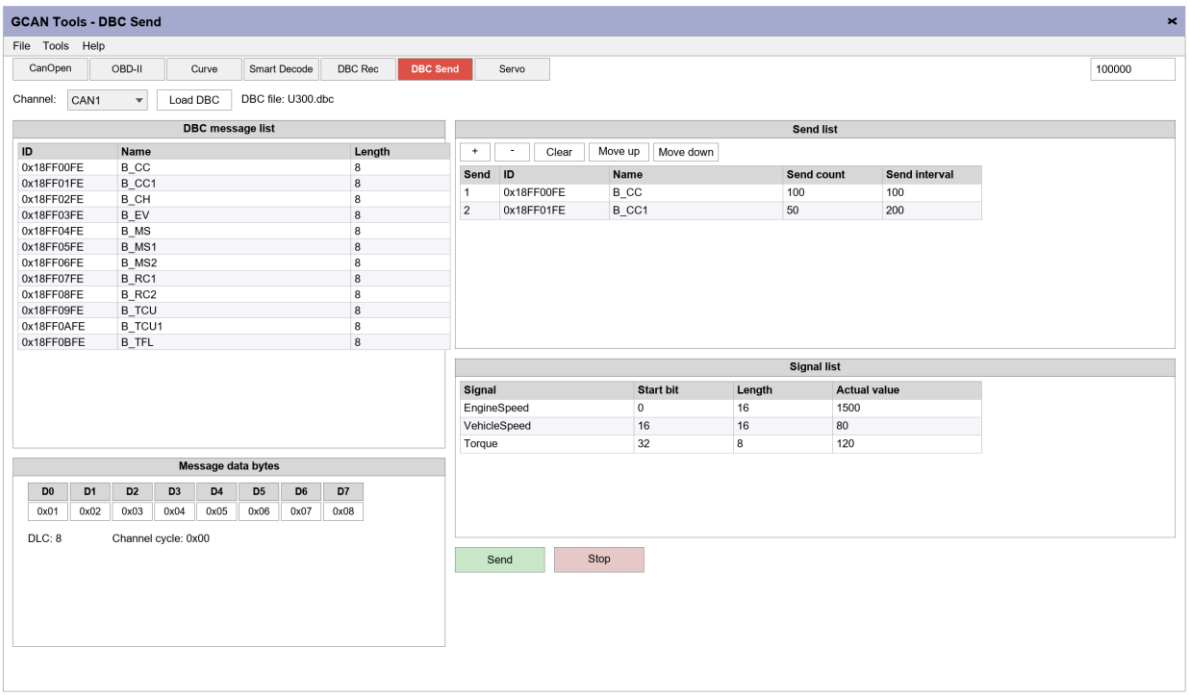

Figure 4.31 DBC Send

Load DBC: first select the DBC protocol, then click "Load DBC" to select the DBC file. The left DBC message list shows all messages in the loaded DBC file; select one and click [+] to add it to the upper-right send list. In the upper-right list you can double-click to edit the send count and send interval. In the lower-right signal list you can double-click to edit the actual signal value; after editing, the message's data field is updated synchronously. [+] adds, [-] removes, Clear removes all, Move up/Move down adjust order. Channel selection: select the

device channel for DBC message sending in the upper-left Channel drop-down. "Send/Stop" starts or stops sending all messages in the list; "Send" starts sending the currently highlighted message.

4.11 DBC Rec

Click DBC Rec on the toolbar. This tab shows the result of parsing the raw messages according to the DBC specification.

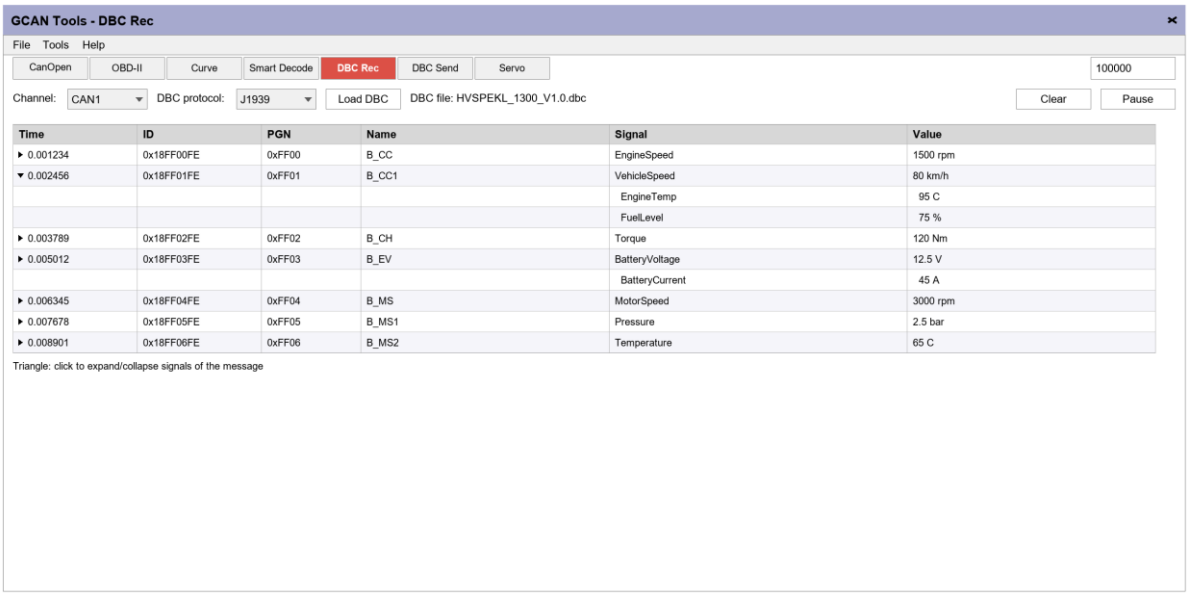

Figure 4.32 DBC Rec

Channel selection: select 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 (see the J1939 specification); a DBC message ID's PGN equal to the message ID's PGN to be parsed is a successful match. Normal: a DBC message ID equal to the message ID to be parsed is a successful match. Load DBC: click to select a DBC file. Triangle: click to view all signals of the current message. Clear: clears the list. Pause: pauses or resumes receiving messages.

5. Linux System Usage

The GCAN USBCAN analyzer series supports various Linux operating systems. We provide 32/64-bit Linux drivers and secondary-development documentation; users can develop and use the devices on Linux themselves.

The general procedure for using our devices on Linux: 1) obtain system administrator privileges; 2) copy the necessary files to the system GCC compile directory; 3) switch to the USBCAN driver folder and compile; 4) run the test program. The detailed steps are as follows:

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



Figure 5.1 Query system version

2. After confirming the system type, copy the corresponding USBCAN driver files to the system (in this example the driver files are placed on the desktop).

3. Obtain administrator privileges for subsequent driver-installation operations. Enter: su (you will be prompted for the administrator password; enter the correct password to obtain administrator privileges).

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

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



Figure 5.2 Link library and compile

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

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

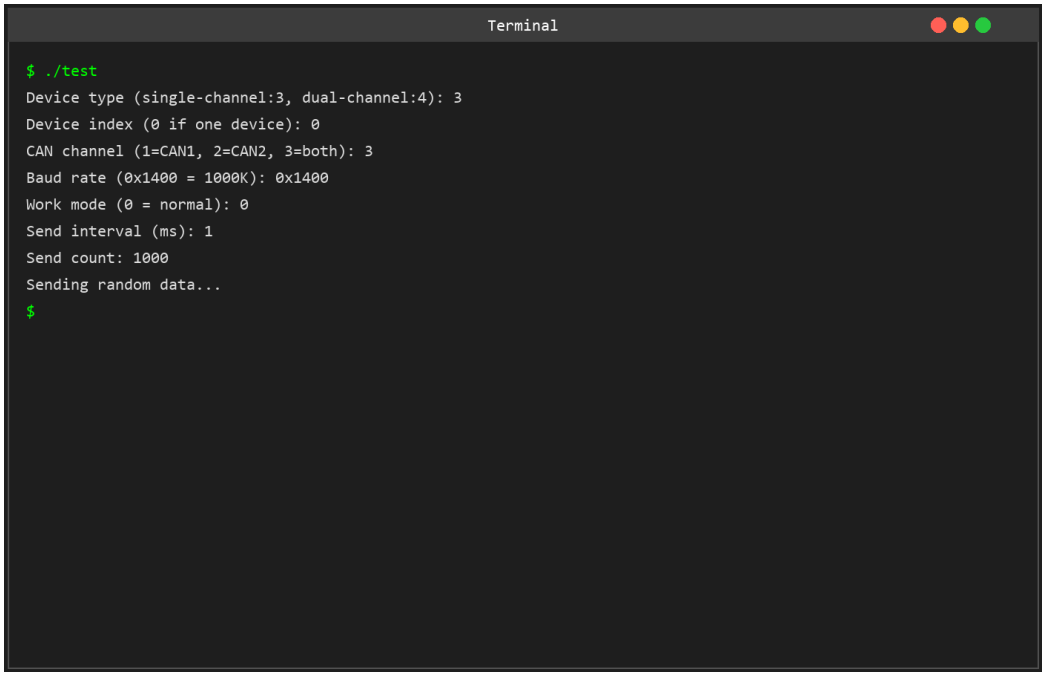

Figure 5.3 Run the test program

After entering ./test, the system shows prompts and examples, where:

1st value (16): device type; single-channel device input 3, dual-channel input 4

2nd value (0): device index; 0 when only one USBCAN is connected

3rd value (3): which CAN channel to open; 1 = CAN1, 2 = CAN2, 3 = both CAN1 and CAN2

4th value (0x1400): set the CAN baud rate; 0x1400 = 1000K; for other baud-rate values see the "EcanVCI Dynamic Library User Manual"

5th value (0): work mode; 0 = Normal mode; for other work modes see the "EcanVCI Dynamic Library User Manual"

6th value (1): send interval in ms

7th value (1000): send count

8. After running the test program, other USBCAN devices can receive the random data it sends. This example only performs the random-data send process and does not include the receive-function part; the user needs to write it themselves (refer to the dynamic-library user manual).

6. Secondary Development

We provide a standard interface function library for secondary-development users, including ECANVCI.h, ECANVCI.lib and ECANVCI.dll. These are standard-format interface functions that can be declared and called in VC, VB, LabVIEW and other programming environments; for details see the "ECAN Dynamic Library User Manual". Figure 6.1 shows the common structure names and the function-library call flow.

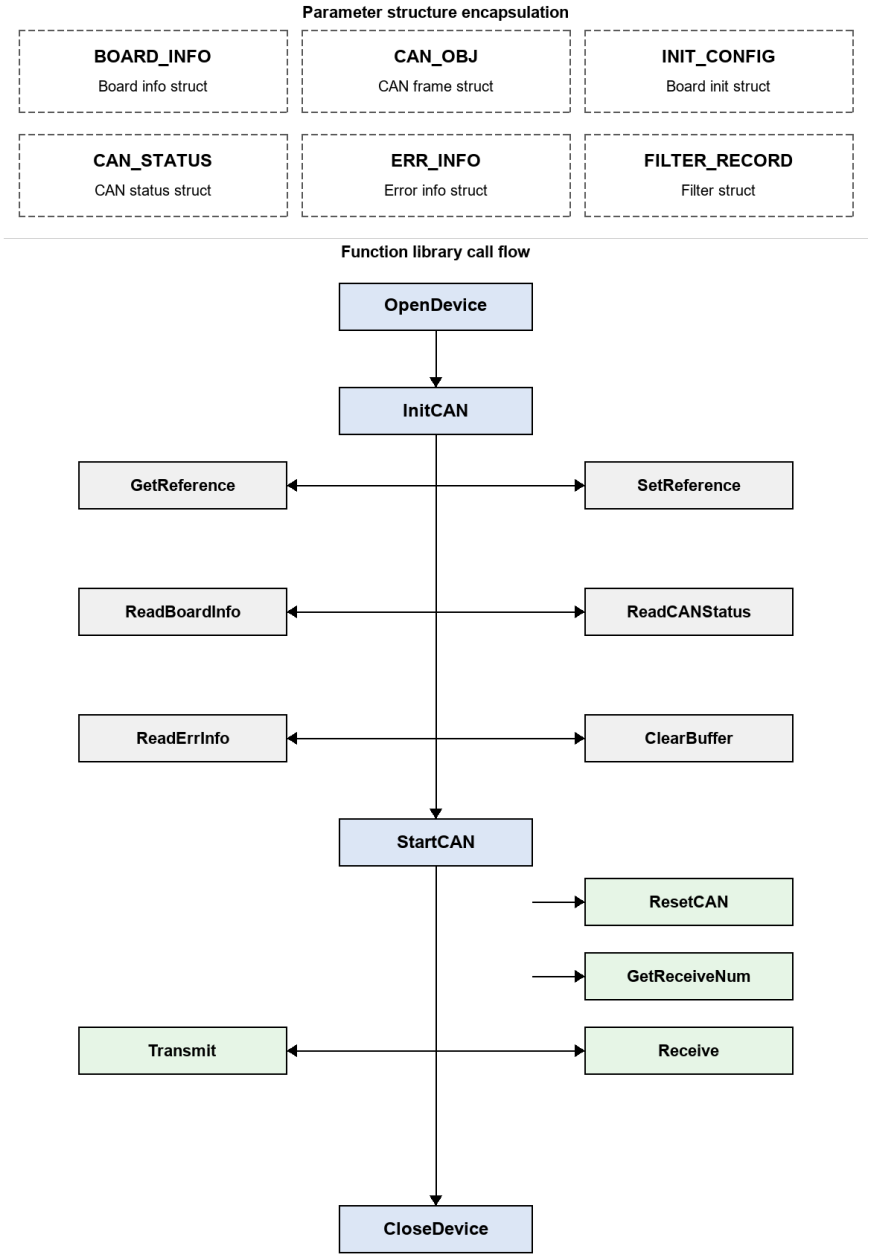

Figure 6.1 Secondary-development flow

7. Technical Specifications

Connection		
	PC side	USB port, type A
	CAN side	OPEN4 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
	CAN baud rate	5 kbit/s - 1 Mbit/s
	Electrical isolation	3000V DC
	CAN terminal resistor	Built-in; selectable via DIP switch
Power supply	**Supply voltage**	**+5V DC (USB port)**
	Supply current	Max 130 mA
Environmental tests	**Operating temperature**	**-40°C to +85°C**
	Operating humidity	15% - 90% RH, non-condensing
	EMC tests	EN 55024:2011-09; EN 55022:2011-12
	Protection rating	IP 20
General	**Outline dimensions**	**96mm x 58mm x 24mm**
	Weight	90 g

Table 7.1 Technical specifications of the USBCAN-I Pro

8. Frequently Asked Questions

1. Why does GCANTools show "Open device error"?

The device type may be selected incorrectly or the USBCAN driver may not be installed correctly. In the PC's Device Manager, check the USBCAN device properties to see whether there is a "!" or "?" in front of the USBCAN device; if so, check the hardware/software conflict and reinstall the USBCAN device driver.

2. Is a 120Ω terminal matching resistor always required?

A 120Ω terminal resistor is recommended to absorb endpoint reflection and provide a stable physical link. For a single-node self-send/self-receive test the 120Ω terminal resistor must be connected to form a loop, otherwise the self-send/self-receive test cannot be performed. A 120Ω terminal resistor is already connected inside the USBCAN high-performance CAN interface card.

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

Older interfaces do not support multi-card simultaneous operation, but current USBCAN interface cards support up to 8 USBCAN interface cards of the same model operating simultaneously.

4. What is the maximum data conversion rate of the USBCAN-I Pro?

A single CAN channel of the USBCAN interface card supports up to 8000 fps CAN-bus data conversion. The frame mentioned here is a standard frame 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 do the CAN status indicators not light up?

All operations of the USBCAN interface card are controlled by the PC. The CAN status indicators are meaningful only after the PC sends a command to start CAN communication.

6. Why does the system perform an illegal operation when calling an interface function?

Read the function description carefully before using interface functions to ensure the input parameters are valid, paying special attention to pointer (address) passing, or refer to the provided example programs. If the problem persists, contact our technical support.

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

The device provides a set of common baud-rate setting values. For other baud rates, contact GCAN customer service to calculate them. Note: the CAN controller of the USBCAN interface card uses a 24 MHz clock; calculate custom baud rates according to this clock frequency.

8. Does system standby or sleep affect reception?

Yes. At that time all processing stops and the hardware receive buffer may overflow. If a program has the device open it will try to prevent the system from entering standby or sleep to ensure normal operation. When using the USBCAN interface card, disable the system standby and sleep functions.

9. How do I handle application errors?

Errors are mainly divided into function-call errors and CAN-bus communication errors. Function-call errors are generally caused by parameter errors, such as device-number out of range or wrong type number; the error number returned by the Win32 GetLastError function is 87. Some are operations on an unopened device (actually an illegal handle operation); specific function-call situations have corresponding standard Win32 error codes for analysis via GetLastError. This debugging is generally done at design time.

CAN-bus communication errors are generally caused by the CAN network or 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, call the ReadCANStatus function.

Note also the data-overflow interrupt error, which may occur in two cases: (1) software receive-buffer overflow - the application cannot process the received data in time; the user should optimize the application or change the communication strategy. (2) Hardware receive-buffer overflow - caused by excessive interrupt latency at the receiving PC; this can only be resolved by improving PC performance or coordinating with other nodes to reduce the send rate.

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

The USBCAN-I Pro provides one CAN port and does not allow opening the device in shared mode; the same device cannot be opened multiple times by different processes via the OpenDevice function. OpenDevice and CloseDevice generally need to be called only once during application initialization and exit. When closing the device, if the current port is no longer in use, call ResetCAN first to disconnect the current 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 can the Transmit function be used better?

The USBCAN driver provides a send FIFO of about 128 frames; each Transmit call can send up to about 128 frames. The send speed depends on the current computer's hardware and software performance; the continuous send speed is generally about 2000 fps (standard data frame, 11 bytes, 1 Mbps). If sending is too fast, the remote receiving device may overflow and stop responding; add an appropriate delay in the application to reduce the send speed.

Each frame has a timeout during sending. The single-frame timeout is about 2 seconds; when sending multiple frames at once, the last frame's timeout is 2 seconds and the others are 1 second. Send timeouts generally occur when the CAN bus is busy and the current node priority is low; this is not a function-call or communication error, and the user can implement a resend (rarely occurs on medium-/low-speed networks). Therefore, ensure during system design that the CAN bus occupancy does not exceed 60-70% of the bus capacity.

12. How can the Receive function be used better?

The device driver provides a software receive buffer of 100,000 frames, giving application programmers sufficient reaction/processing time. When the software receive buffer overflows, the device driver calls ResetCAN to reset the CAN bus and sets the data-overflow interrupt flag bit in CAN status; note that both software-buffer overflow and CAN-controller hardware-buffer overflow use this flag bit.

When Wait is 0, the function returns immediately at call time with the number of frames successfully read; if the receive buffer is empty, it returns 0. When Wait is non-zero, if there is data in the receive buffer at call time it returns immediately with the number of frames successfully read; if the receive buffer is empty, the function waits until a specified timeout expires or data is received, then returns the number of frames successfully received. When Wait is 0xFFFFFFFF it waits indefinitely until data is received; do not set Wait too large, and be especially careful with indefinite waiting.

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 and Wait is 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.