

www.vskom.de

VSCAN Manual

Edition: March 2021



Tel: +49 40 528 401 0

Fax: +49 40 528 401 99

Web: www.visionsystems.de

Support: faq.visionsystems.de

The software described in this manual is furnished under a license agreement and may be used only in accordance with the terms of that agreement.

Copyright Notice

Copyright © 2009-2021 Vision Systems. All rights reserved. Reproduction without permission is prohibited.

Trademarks

VScom is a registered trademark of Vision Systems GmbH. All other trademarks and brands are property of their rightful owners.

Disclaimer

Vision Systems reserves the right to make changes and improvements to its product without providing notice.

Vision Systems provides this document “as is”, without warranty of any kind, either expressed or implied, including, but not limited to, its particular purpose. Vision Systems reserves the right to make improvements and/or changes to this manual, or to the products and/or the programs described in this manual, at any time.

Information provided in this manual is intended to be accurate and reliable. However, Vision Systems assumes no responsibility for its use, or for any infringements on the rights of third parties that may result from its use.

This product might include unintentional technical or typographical errors. Changes are periodically made to the information herein to correct such errors, and these changes are incorporated into new editions of the publication.

Contents

1	Installation	7
1.1	USB-CAN Plus Device	7
1.1.1	Windows	7
1.1.2	Linux	8
1.2	Network CAN Device	11
1.2.1	Operational Modes	11
	CAN Server Mode	11
	UDP mode	11
	CAN over IP Router	11
1.2.2	Configuration Overview	14
1.2.3	Webbrowser Server Configuration	15
1.2.4	Webbrowser Channel Configuration	18
1.2.5	Webbrowser Tools	19
1.2.6	SNMP Configuration	22
1.2.7	Factory Settings	22
1.2.8	viaVPN Remote Access System	23
	Activation of option viaVPN	23
	Configuration of viaVPN Parameters	23
	Operation with viaVPN System	25
1.3	Linux Installation (SocketCAN)	26
	Limitations	26
	Usage Examples	27
	Programming Examples	27
1.4	Windows Driver Installation	28
1.5	General Information	29
1.5.1	LED Status	29
1.5.2	Baud-rates and Handshake	29
1.5.3	Pin-out of the 9 Pin D-Sub Connector	29
1.5.4	Pin-out of the 4 Pin Connector (USB-CAN Plus mPCIe)	29
1.5.5	CAN Topology, Wiring and Termination	30
1.5.6	CAN Signal Levels	31
1.5.7	Termination Resistors (USB-CAN Plus Devices)	32
1.5.8	Terminal Block Power	32
	NetCAN Plus Devices	32
	NetCAN Plus Mini Devices	32
1.6	Products	32
1.7	Dimensions	34
1.7.1	NetCAN Plus	34
1.7.2	USB-CAN Plus	39

2	Application Programming Interface	40
2.1	Introduction	40
	Windows	40
	Linux	40
	Other Operating Systems	40
	Python	40
	Programming Language Bindings	40
2.2	Functions	42
2.2.1	VSCAN_Open	42
2.2.2	VSCAN_Close	43
2.2.3	VSCAN_Ioctl	44
	VSCAN_IOCTL_SET_DEBUG	44
	VSCAN_IOCTL_SET_DEBUG_MODE	44
	VSCAN_IOCTL_GET_API_VERSION	45
	VSCAN_IOCTL_GET_HWPARAM	45
	VSCAN_IOCTL_SET_SPEED	45
	VSCAN_IOCTL_SET_BTR	46
	VSCAN_IOCTL_SET_FILTER_MODE	46
	VSCAN_IOCTL_SET_ACC_CODE_MASK	46
	VSCAN_IOCTL_SET_FILTER	47
	VSCAN_IOCTL_GET_FLAGS	47
	VSCAN_IOCTL_SET_TIMESTAMP	48
	VSCAN_IOCTL_SET_BLOCKING_READ	48
	VSCAN_IOCTL_SET_KEEPALIVE	48
2.2.4	VSCAN_Read	49
2.2.5	VSCAN_SetRcvEvent	50
2.2.6	VSCAN_Write	52
2.2.7	VSCAN_Flush	53
2.2.8	VSCAN_GetErrorString	54
2.3	Types and Structures	55
2.3.1	VSCAN_HANDLE	55

2.3.2	VSCAN_STATUS	55
2.3.3	VSCAN_APLVERSION	55
2.3.4	VSCAN_HWPARAM	56
2.3.5	VSCAN_MSG	56
2.3.6	VSCAN_BTR	57
2.3.7	VSCAN_CODE_MASK	57
3	ASCII Command Set	58
3.1	Introduction	58
3.2	Commands	59
3.2.1	Open the CAN Channel	59
3.2.2	Close the CAN Channel	59
3.2.3	Setup the Bus Timing (Standard)	59
3.2.4	Setup the Bus Timing (Advanced)	60
3.2.5	Transmitting a Standard Frame	61
3.2.6	Transmitting a Standard Remote Request Frame	61
3.2.7	Transmitting an Extended Frame	62
3.2.8	Transmitting an Extended Remote Request Frame	62
3.2.9	Set Time-Stamps	63
3.2.10	Set Filter Mode	63
3.2.11	Set Acceptance Code and Mask Register	63
3.2.12	Set Advanced Filter	64
3.2.13	Get Status Flags	65
3.2.14	Get Version Information	65
3.2.15	Get Serial Number	65
3.2.16	Get Extra-Information	66
4	Tools	67
4.1	Firmware-Update	67
4.2	Busmaster	67
4.3	vscandump and vscansend	71
	vscandump.exe	71
	vscansend.exe	71
4.4	vs_can_test_simple.exe	73
4.5	Wireshark	74
4.6	CANopen	75
4.6.1	Introduction	75
4.6.2	Running Example	75
4.6.3	Compilation Instructions	76
4.7	J1939	78
4.7.1	VSCAN J1939 Library	78
4.7.2	SocketCAN Based Implementation	78
4.8	Wrapper DLL System	78
4.9	ZOC	80

4.10	putty	80
4.11	vs_can_search	82
4.12	LabVIEW	83
4.12.1	Open CAN Channel	83
4.12.2	Read CAN Frame	84
4.12.3	Write CAN Frame	85
5	Example Configurations	86
5.1	CAN over IP Router (CAN Bridge Mode)	86
	Server Configuration	86
	Client Configuration	86
6	Frequently Asked Questions	87
6.1	All output from the CAN adapter will be written in one line in HyperTerminal?	87
6.2	I've updated the driver of my USB-CAN Plus, but the alias baudrate 9600 is not functioning anymore?	87
6.3	The Error LED is permanently on	87
6.4	SocketCAN Troubleshooting	88
	FTDI driver	88
	slcan driver	88
	slcan0 interface	88
	CAN Error LED	89
6.5	USB-CAN Plus Troubleshooting in Windows	90
	Powerup LED Blinking Sequence	90
	Checking Driver Installation	90
	Check CAN Wiring	90
	Perform Self-Diagnostic Tests	90
	Check CAN Filter Settings	90
	Perform Basic CAN Transmission Tests	90
7	Licensing Information	91
7.1	GNU GENERAL PUBLIC LICENSE	91
7.2	GNU LESSER GENERAL PUBLIC LICENSE	97

1 Installation

VSCAN devices support both Windows and Linux operating systems. The ASCII protocol is used to exchange data and control information with the devices, and hence a serial interface is required to enable communication.

USB-CAN Plus requires Windows driver installation. This is best done by connecting the device to the PC, and have Windows download the appropriate drivers from Windows Update via the Internet. Usually, this does not require manual interaction. Modern Linux distributions often include those drivers by default, so no need to install the driver. NetCAN Plus has Windows drivers only, but can be used operating system independent without a driver in TCP raw mode. Please read the relevant installation instructions below for your particular device.

For ease of use, ASCII protocol is implemented in `vs_can_api` library with appropriate [API](#). Linux users can alternatively use the SocketCAN driver, which also implements ASCII protocol.

1.1 USB-CAN Plus Device

1.1.1 Windows

On a modern Windows system, the USB driver probably installs automatically being downloaded from Microsoft on the first connection to the PC. If this is not desired, download the driver from our company website¹. Execute the installer, then plug in the adapter in a USB port of your choice. The device will be registered and you can use the COM port for your further work (see Figure 1).

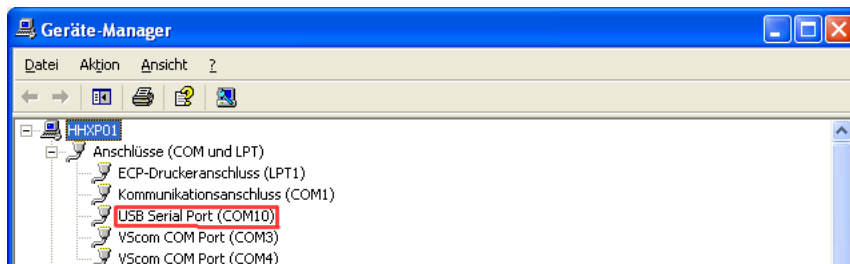


Figure 1: Device Manager

To get better performance and set every standard baudrate as an alias for 3Mbit (**except 115200**), you should use the VS USB-COM Configurator software (Figure 2). This will a) set 'Latency Timer' to 1ms, b) activate 'Event on Surprise Removal', and c) set all baud rates to 3MBit except 115.2k.

¹<http://www.visionsystems.de/produkte/usb-can-plus-usb-can-plus-iso.html#downloads>

On the left list, select the Com Port of your USB-CAN Plus device (Make sure that it is the CAN adapter). Then click 'Optimize for USB-CAN' and finally click 'OK'.

On Windows XP as an administrator, open a command console window (DOS Box). Navigate to the folder where the script „*regmodify.vbs*” is located and execute it with the desired COM port like this:

```
cscript regmodify.vbs COM10
```

Please disconnect the device from the system for about 5 seconds after this step and connect again.

Then you can open the COM port of the USB-CAN Plus with any standard baudrate² (see Figure 3).

1.1.2 Linux

As for the Linux driver, it is already available in modern kernels. The device name will be `/dev/ttyUSBx`. Use port configuration as 3M,8N1 with RTS/CTS flow control. See [SocketCAN Troubleshooting](#) for details.

²300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 230400, 460800, 921600

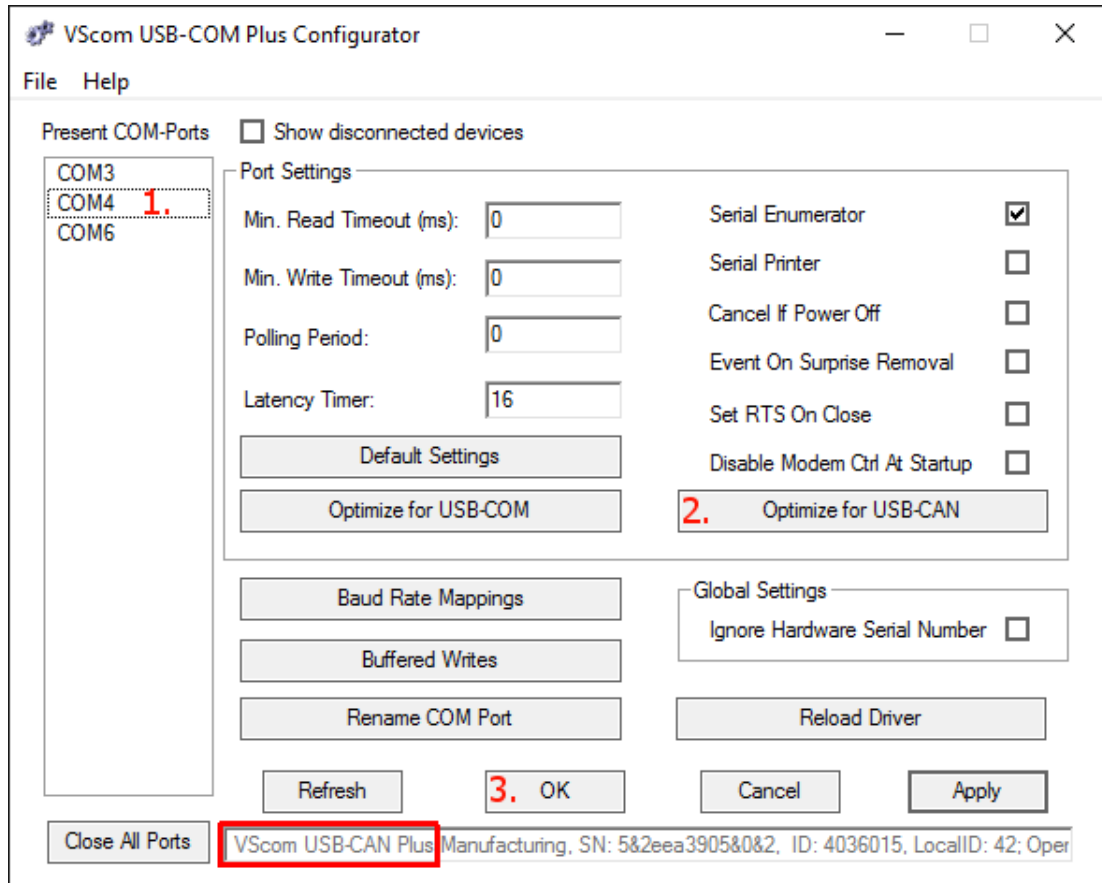


Figure 2: USB-COM Configurator

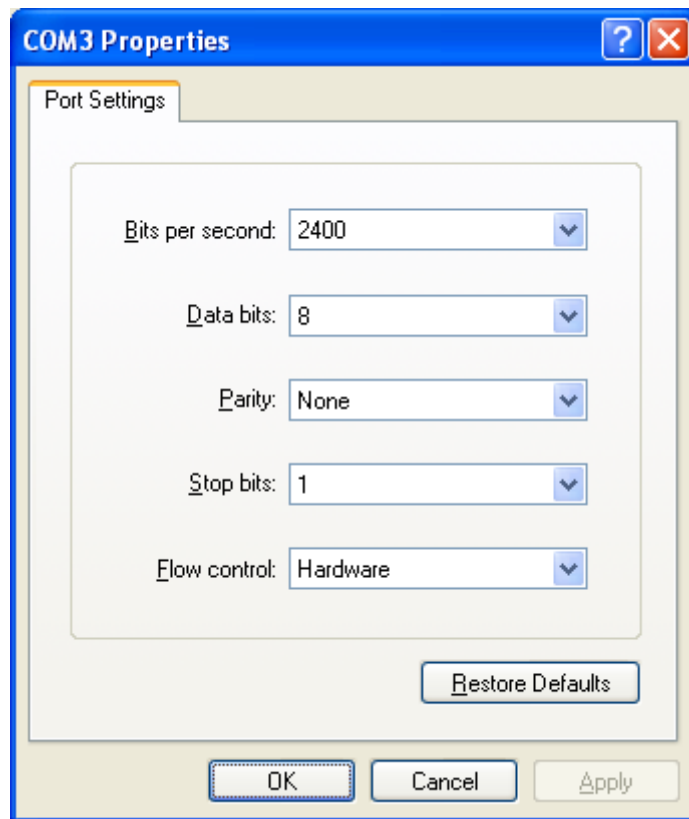


Figure 3: COM Port Properties

1.2 Network CAN Device

NetCAN Plus gateways provide CAN communication over network in versatile ways. For brevity the name NetCAN Plus is sometimes shortened to NetCAN⁺.

1.2.1 Operational Modes

NetCAN Plus can act either as CAN Server (CAN frames will be exchanged between NetCAN⁺ and an application connected via COM port or TCP socket) or as CAN over IP Router (two or more NetCAN⁺ exchange CAN frames between various CAN networks).

CAN Server Mode For CAN Server the following approaches are provided:

Driver mode: in this mode the network is transparent for the application. To use this mode the installation of the Windows driver is required (please refer to section [1.4 on page 28](#) for detailed installation instructions). After driver installation the new virtual COM port will be available to the system, so NetCAN Plus can be used in the same way as USB-CAN Plus. Due to the virtual COM port protocol overhead the performance is lower than by the TCP raw mode. **You don't have to set the baudrate and hardware handshake explicitly - it's fixed to 3Mbit and RTS/CTS.**

TCP raw mode: the communication will be handled directly via IP address and port number. In this mode no driver installation is required.

NetCAN⁺ devices can be operated with either [ASCII](#) or [VSCAN API](#). For the API, there is no difference whether Driver mode or TCP raw mode is used, but due to performance issues, TCP raw mode is recommended.

UDP mode This mode is similar to TCP raw mode, but the communication will be handled via UDP protocol. Two UDP sockets are required to be able to send and receive CAN frames. The user's application connects to NetCom's *UDP Port(Local)* and configures the CAN channel using the ASCII protocol commands like 'S', 's', 'Z', etc. At the end of the configuration, the channel will be opened with 'O', 'L', or 'Y' command. Now, CAN frames can be sent using 'T' and 't' commands. To receive CAN frames user's application must listen to its own UDP port that will be configured via *Destination* and *UDP Port(Dest)* in the NetCAN Plus configuration. The following Python program³ shows how UDP communication with NetCAN Plus works.

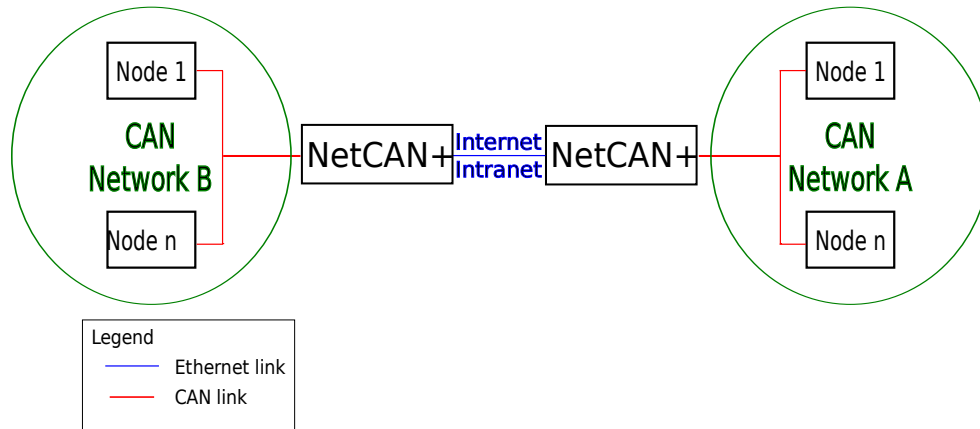
CAN over IP Router In this mode, two or more NetCAN Plus devices are interconnected to enable seamless communication between two or more CAN networks. Figure [4a](#) shows the first case, where two NetCAN⁺ devices act as a tunnel between CAN Network

³https://github.com/visionsystemsgmbh/programming_examples/blob/master/CAN/python/vsudpdump.py

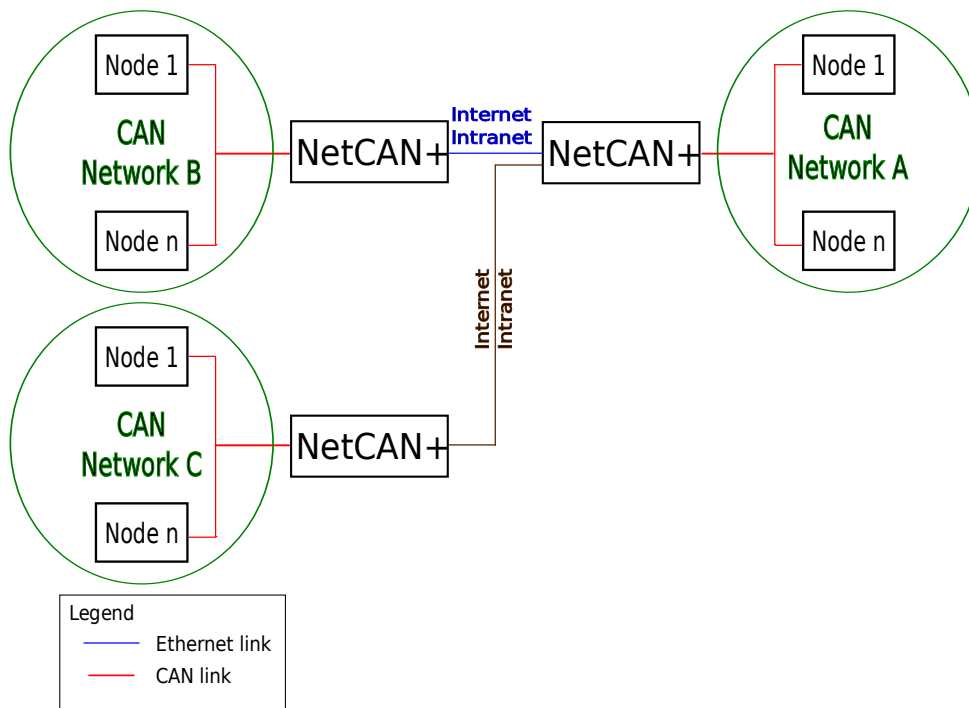
A and B, so all frames sent inside Network A will be transported to Network B and vice versa.

Figure 4b shows the extension of the tunnel. In this case, additional CAN networks can be attached so that CAN frames sent inside Network A will be transported to both B and C, but CAN Network B and C communicate only with Network A, so frames from Network B could not be seen by Network C and vice versa. See Sections 1.2.4 and 5.1 for configuration instructions.

CAN Acceptance Code and Acceptance Mask can be set to filter the CAN frames, so only dedicated frames are passed via TCP link.



(a) Tunnel



(b) 1-to-n

Figure 4: CAN over IP

1.2.2 Configuration Overview

NetCAN Plus device can be configured in the following ways:

- via web interface
- via Telnet
- via NetCom Manager
- SNMP (refer to [SNMP Configuration](#))

Before you can use the above-mentioned methods, you first need to know the IP address of your NetCAN⁺ device. On the first start, the device tries to obtain its IP address using the DHCP⁴ protocol. If there is no DHCP server on the network, a default IP address 192.168.254.254 will be used. In both cases, your PC and NetCAN⁺ must be in the same subnet⁵ to talk to each other.

Once this requirement is satisfied, the NetCAN⁺ device will appear in the Network Places of Windows Explorer (press Windows key + 'e' to open the Windows Explorer and click on the Network symbol). You'll see the following icon as shown in Figure 5 on page 14 providing information about the device's serial number and IP address. Clicking on this icon would open your default Internet browser showing the main page where NetCAN⁺ welcomes you with its "Home" screen (see Figure Figure 6 on page 15).

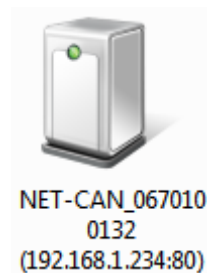


Figure 5: UPnP Device Display

Click on the icon for your desired option. In many menus, you'll see a blue question mark. This is a symbol for help. Click it to get a short explanation, informing about the function of this parameter. Some other settings require a reboot to save and activate them. Whenever this situation occurs, the NetCAN⁺ requests for a Reboot (see Figure 7).

You can instantly reboot or do that later when the configuration is finished.

⁴https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol

⁵<https://support.microsoft.com/en-us/help/164015/understanding-tcp-ip-addressing-and-subnetting-basics>

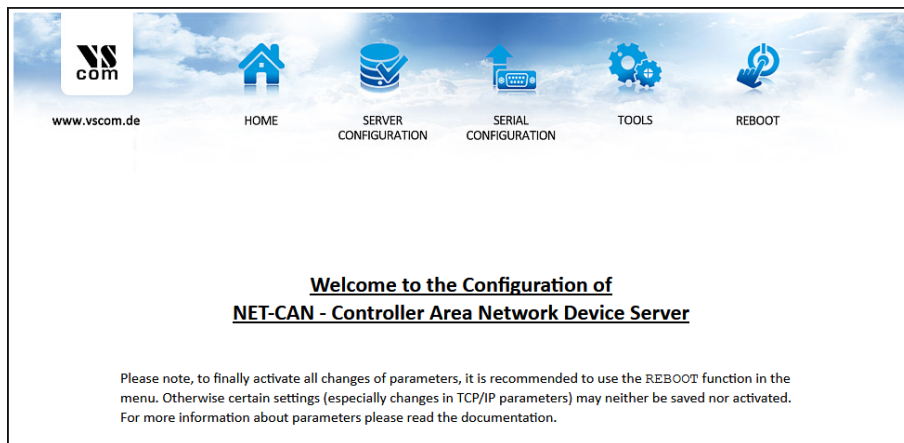


Figure 6: Web Interface for Configuration

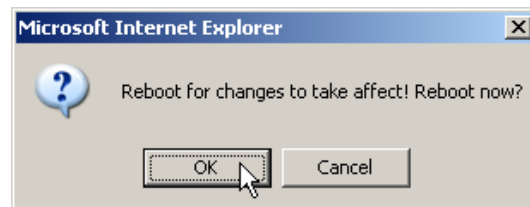


Figure 7: Web Interface Request to Reboot

1.2.3 Webbrowser Server Configuration

The *Server Configuration* is a very long menu (see Figure 8). There is basic server information (see Figure 8a), the server parameters related to the IP-configuration (see Figure 8b), the section for wireless communication (NetCAN Plus 120 WLAN only) (see Figure 8c), the section for encrypted communication (see Figure 8d), Password settings (see Figure 8e), and finally the configuration for date and time (see Figure 8f).

Information about the selected NetCAN⁺ is displayed as *Server Info*. Starting with the *Server Type*, this is the model of the NetCAN⁺, followed by the version of Software and Hardware. This will give a rough overview, which features are implemented, or need an upgrade of the firmware. You can check whether a new firmware version is available via clicking on the „Check for new version” button. The *Serial Nr.* is important to identify the device you are configuring right now. For further information the *UpTime* is listed. *Contact* and *Location* are user-defined information. They may later help to find the device in the installation, and the person responsible for management.

The *Server Parameter* allow configuration of the NetCAN Plus’ name and of course all parameters in IP-settings. Generally it is used as information, e.g. in the NetCom Manager program or in SNMP.

Manual changes of IP parameters are only available with *DHCP* set as Disabled. When

DHCP is not used, enter *IP Address* and *Netmask*, as well as the *Broadcast* address. *Gateway* is required, if there are Routers in the network. DNS is used to access other stations by name. The *ConfigPort* is used to access the NetCAN⁺ for administration via Telnet. It is suggested to use the standard value for Telnet, TCP port number 23. However it may be changed for different purposes. This does not change the function of the Telnet menus.

KeepAlive is an intrinsic function of the TCP/IP protocol. If used, it causes network traffic, but connection problems are detected earlier. In a LAN this is usually not a problem. However, if used via DialUp connections this may cause problems. If this function is used, you must define an interval in seconds. NetCAN⁺ has a better chance to react on network problems, or failed hosts. Even dropping an old connection may be useful in certain environments. You can also use this functionality from the PC side, see [VSCAN_IOCTL_SET_KEEPAIVE](#) for more information.

For detailed information about further Server Configuration options please refer to section „Server Configuration” of the [NetCom Plus User Manual](#).

Server Parameter

Warning: for changes like network settings the server must be rebooted

Server Info

Server Type Plus 110
Software Version 3.4.3 Check for new Version
Hardware Version 4.2
Serial Nr. 0210100826
UpTime 0 day(s) 00:01:32
Contact <unset>
Location <unset>

Server Name NET-CAN_0210100826
MAC Address 74:6A:8F:00:30:A4
Interface Priority Cable, Wireless
DHCP Enabled
IP Address 192.168.1.124
Netmask 255.255.255.0
Broadcast 192.168.1.255
Gateway 192.168.1.1
DNS 213.209.99.202
Domain visionsystems.de
ConfigPort 23
KeepAlive Off
KeepAliveInterval 0
UPnP Broadcast On

(a) Server Info
(b) Server Parameter

Wireless Parameter

Chipset TI WL18XX
SSID NET-CAN_0210100826
OperationMode AP
CountryRegion ETSI (1-13)
Channel 7
Encryption Type Off
Encryption Key

OpenVPN Parameter

OpenVPN Disabled
Logging Off
Config ZIP-file: Browse... No file selected. Upload

(c) Wireless Parameter
(d) OpenVPN Parameter

Date and Time Settings

Date & Time 01-01-2000 00:01:28 UTC+0

Authentication

Security Settings

Password:
Retype Password:

Simple Network Time Protocol

State Off
Mode IP Address
Interval 1800
Server

(e) Authentication
(f) Date and Time Settings

Figure 8: Server Configuration

1.2.4 Webbrowser Channel Configuration

NetCAN Plus can be operated in the following modes (see Figure 9):

Driver Mode - Only very few parameters have a function in *Driver Mode* (see Figure 9a).

NetCAN⁺ is operating as a Server. It accepts two connections for the CAN channel. One connection is used to transmit the serial data, this is the TCP Port(Data). And the other is used to transmit control information, TCP Port(Control). This control connection is mostly used to request the status of the virtual serial port. Software may intend to change serial parameters like baudrate or parity, such requests are honored. However they are ignored, because the serial parameters are fixed in the NetCAN⁺.

The NetCAN⁺ can check if the connected Client is still alive. This may be done, when a second Client wants to establish a connection (On Connect). It may also be done in regular intervals (Polling). The Driver Mode allows for one Client only.

TCP Raw Server - A TCP Raw Server NetCAN⁺ operates very simple (see Figure 9b).

It only waits for incoming data connections in Raw IP mode. As with the Driver Mode only the data connection is defined, there is no connection for control.

You can connect multiple times to the NetCAN⁺ also from different machines.

CAN Bridge Server - Configures server side of the CAN over IP Router functionality.

In this mode NetCAN⁺ waits for incoming connections. Max. Clients value defines how many CAN Bridge Clients can connect to the server. CAN Speed sets the appropriate baudrate of the attached CAN network. Acceptance Code and Mask allow CAN frame filtering so only the dedicated frames pass over the TCP link (See Figure 9d).

CAN Bridge Client - Configures client side of the CAN over IP Router functionality. In

this mode NetCAN⁺ connects to the other NetCAN⁺ in the CAN Bridge Server Mode. This is specified as destination by IP Address and the TCP Port number (e.g. 192.168.1.97:2001). CAN Speed sets the appropriate bitrate of the attached CAN network. Acceptance Code and Mask allow CAN frame filtering so only the dedicated frames pass over the TCP link (See Figure 9e).

UDP Mode - Configures a listen port where the NetCAN Plus receives ASCII com-

mands. Received CAN frames will be sent to a host defined by destination IP address and port number. The other settings should be left by their default values. See Figure 9c.

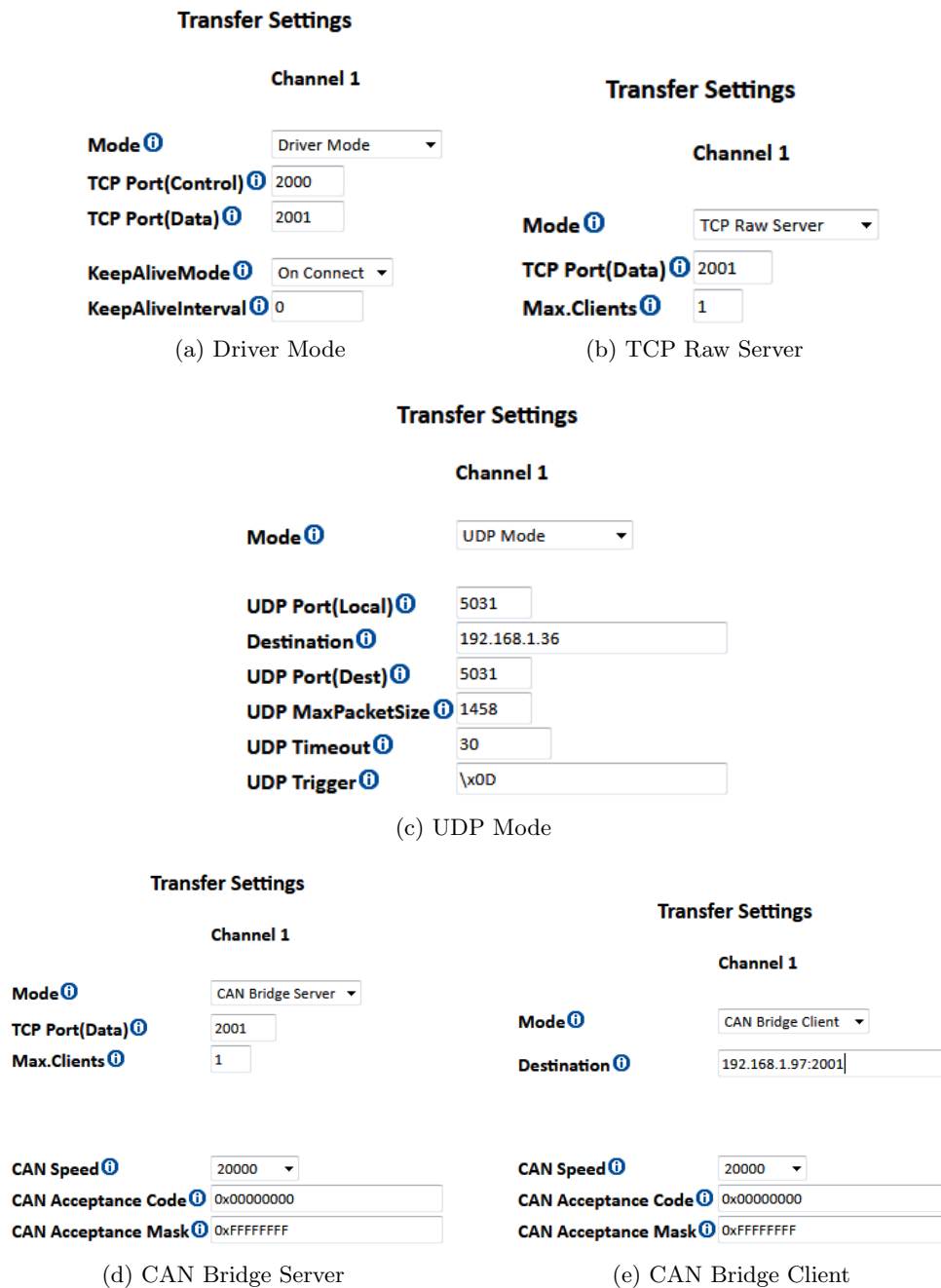


Figure 9: Channel Configuration

1.2.5 Webbrowser Tools

The available tools are (see Figure 10):

- *The Ping* utility will be used to check if a station is available (see Figure 10a).

Enter the IP-Address or the name of a station in the field, and click the *Ping* button. The network connection is checked by sending certain ICMP data packages. If the target responds, the network between the NetCAN⁺ and the target is operational. The time required for an echo depends on the speed of the network. In a typical Ethernet this is only very few Milliseconds, while it can be several seconds throughout the Internet.

- *The Netstat* utility will be used to monitor TCP connections (see Figure 10b). Use Netstat to see the actual status of NetCAN⁺ IP Ports. This is a standard tool for network debugging. A *Foreign Address* of 0.0.0.0 is listed when NetCAN⁺ is waiting for an incoming connection (LISTEN). If the value is not 0.0.0.0, the connection is either active (ESTABLISHED) or closed (TIMEWAIT).
- *The Firmware Update* option is used to update the firmware (see Figure 10c). To upload a new version of the firmware, put the name of the file in the field. Your Webbrowser may allow to search for the file. Click on the “Update” button. While loading the file is checked. If it is valid, it is stored in the Flash Memory. When the upload is finished, NetCAN⁺ will Reboot.
- *The Saving of Configuration to / Loading from a file* option will be used to manage NetCAN⁺ configuration (see Figure 10d). It is possible to save the current configuration to a text file. Of course it is also possible to load the saved configuration into a NetCAN Plus.
- *The Syslog* option will be used to send logging information to the syslog facility (see Figure 10e). Syslogging requires a server the information is sent to. Facility allows to select the data sent to that server.
- *The DebugLog* option will be used to show logging information via TCP connection (see Figure 10e). For this kind of logging the NetCAN⁺ behaves as the server. Open a TCP connection to the configured port, and receive all information generated.

Ping

IP Address:

(a) Ping

Netstat

[View connections](#)

(b) Netstat

Firmware Update

Warning: All connections get closed and the server reboots after updating

netcan_Plus_XXX_3.1.1.bin.b64

(c) Firmware Update

Configuration File

Save: *Save the Configuration Parameters in a File*

Load:

(d) Configuration File

Syslogging

Syslog ⓘ

Destination ⓘ

Facility ⓘ

Debuglog ⓘ

Debug Port ⓘ

(e) Syslog

Figure 10: Tools

1.2.6 SNMP Configuration

NetCAN Plus can be also configured using the SNMP protocol. This can be useful if you want to change various settings from your software or script. Starting from version 3.4.8, firmware archives⁶ provide a folder `snmp` holding management information base (MIB) definition files and related documentation:

- `VSCOM-MIB.txt` - general configuration
- `EEE802dot11-MIB.txt` - WLAN settings
- `README.md` - Net-SNMP installation instructions and usage examples

You can also find an example⁷ written in Python in our GitHub repository that configures two NetCAN Plus devices to a CAN Bridge.

1.2.7 Factory Settings

NetCAN Plus provides DIP-switches to set the configuration back to its defaults. Set the DIP-switches to the „Factory Settings” position (refer to the Table 1) and reboot the device (press reset button or perform a power off/on cycle). As soon as the green LED is on again, the DIP-switches must be returned to the „CAN Operation” position.

Operation Mode	S1	S2	S3	S4
CAN Operation	OFF	OFF	OFF	OFF
Factory Settings	OFF	OFF	OFF	ON

Table 1: Switches

⁶<http://www.visionsystems.de/produkte/netcan-plus-110-netcan-plus-120-wlan.html#downloads>

⁷https://github.com/visionsystemsgmbh/programming_examples/blob/master/CAN/python/vssnmpcan.py

1.2.8 viaVPN Remote Access System

The viaVPN system⁸ provides a server available in public Internet, both the NetCAN Plus and the Client PC connect to this. Therefore it is named as “Rendezvous Server”, where both meet. The NetCAN Plus provides special configuration, which is firewall-friendly and enables the connection to the Rendezvous Server. Likewise the PC uses a Client Utility, this allows connection to the Rendezvous Server also in a convenient and firewall-friendly manner. The customer manages all his NetCAN Plus via a web-based account, i.e. via his browser.

The connection from the NetCAN Plus to the Rendezvous Server is in parallel to the normal operation on the LAN. This allows client PCs on the LAN to operate the NetCAN Plus in the same way as without the viaVPN system. And other Client PCs via the Internet can also access the NetCAN Plus without breaking the operation on the LAN. In particular, if the Client PC is a mobile device (typically a Laptop), this PC can operate the NetCAN Plus directly in the LAN, and via the Client Utility while the PC is far away (road warrior mode).

In addition to the easy handling of devices, PCs and connections, all connections to the viaVPN system use strong and modern encryption. This prevents all eavesdropping by any party. Even more important it provides strong means of authentication, which ensures only the customer decides on who has access to the NetCAN Plus.

viaVPN activation and configuration parameters can be found on the *Server Configuration* page.

Activation of option viaVPN viaVPN is an add-on option in the NetCAN Plus Servers, hence it has to be activated before the use. This is done by uploading a special ZIP file in the Server configuration.



Figure 11: viaVPN activation

Please refer to the [viaVPN Manual: Administration Web Interface](#) to create and manage your viaVPN account.

Configuration of viaVPN Parameters The first parameters are to configure the access of NetCAN Plus to the Internet, so it can establish the connection to the Rendezvous Server. Many installations do not require configuration at all, then there is no need for

⁸<http://viavpn.com/>

viaVPN Parameter	
viaVPN Function ?	On ▾
Proxy IP Address ?	0.0.0.0
Proxy TCP Port ?	8080
Proxy User ?	
Proxy Password ?	
Retype Password:	
DHCP Server ?	Enabled ▾
IP Address ?	10.0.10.1
Netmask ?	255.255.255.0
Broadcast ?	10.0.10.255
Range Start ?	10.0.10.100
Range End ?	10.0.10.200

Figure 12: viaVPN Network Parameter (web)

special parameters. But in case such is required, the NetCAN Plus uses a web proxy for https access, in the same way as any Internet Browser in the network does this.

The *Proxy IP Address* is either the address or the name (like proxy.network.local). The value of 0.0.0.0 disables this function, which is the default. The *Proxy TCP Port* is used to get access to the proxy function. In rare configurations the operation of the Proxy requires an identification, the *Proxy User Name* and the *Proxy Password* serve for this. Your Network Administrator will provide you with all necessary values.

For easy administration of connections via the PC Client Utility the NetCAN Plus provides a DHCP Server. This will assign a valid configuration to the Client Utility connection to the viaVPN system. As the result the Client PC can connect to the NetCAN Plus, and perform management operations. The following parameters configure the operation of the DHCP Server. Important Note: These parameters here do not influence the configuration in the local network (LAN). They are effective only on the connection established by the Client Utility.

The *DHCP Server* may be Disabled. But Enabled is the default and strongly recommended. *IP Address*, *Netmask* and *Broadcast* are properties of the NetCAN Plus side of the viaVPN connection. They behave in the same way as the parameters in the Server Configuration, see paragraph 1.2.3 on page 15. *Range Start* and *Range End* are the first (lowest) and last (highest) value the DHCP Server will assign to the Client PC side of the viaVPN connection.

Operation with viaVPN System Without the viaVPN function the green LED for Ready status lights when the firmware finished starting, and the NetCAN Plus is ready for operation. This behavior is modified when the NetCAN Plus is enabled for the viaVPN system.

As usual the Power LED (PWR, red) blinks when the firmware is ready. At this time the NetCAN Plus starts the connection to the viaVPN system. During this process the Ready LED (RDY, green) is not just on, instead it blinks in a slow pace (symmetric On and Off). When the connection to the Rendezvous Server is established, the Ready LED is permanent On.

When a Client PC is connected via the Utility, this is also signaled by the Ready LED. This time the LED light is interrupted (short time Off, most time On).

Power LED	Ready LED	Status
Off		Device off, no power
On	Off	Firmware startup
On	Symmetric Blink	Start connection to viaVPN
On	On	Connection to viaVPN established
On	Interrupted	Client PC connected to device

Table 2: Ready LED with viaVPN

1.3 Linux Installation (SocketCAN)

SocketCAN⁹ is a set of open source CAN drivers and a networking stack contributed by Volkswagen Research to the Linux kernel. As of Linux kernel 2.6.38, ASCII protocol (slcan) will be also supported. To use it, you'll first need to install can-utils either from your Linux distribution repository¹⁰ or from the project's git¹¹ repository¹².

For a USB-CAN Plus device that is attached to /dev/ttyUSB0 device, execute the following commands:

```
slcand -c -o -s8 -t hw -S 3000000 /dev/ttyUSB0
ip link set up slcan0
```

Parameter `-s` in `slcand` specifies CAN bitrate as described in section [Setup the Bus Timing \(Standard\)](#). After execution you'll get your USB-CAN Plus operated at 1MBit/s.

You can also enable "Listen only" mode by adding the `-l` Parameter:

```
slcand -c -o -s8 -l -t hw -S 3000000 /dev/ttyUSB0
ip link set up slcan0
```

For NetCAN Plus devices, you'll first need to create a symbolic link to it via `socat`. Assuming your NetCAN+ has the following IP address: 192.168.1.5, you can create the link as follows:

```
socat pty,link=/dev/netcan0,raw tcp:192.168.1.5:2001 &
or
sudo sh -c "socat pty,link=/dev/netcan0,raw tcp:192.168.1.5:2001 &"
```

Then you can apply almost the same commands as for USB-CAN Plus devices:

```
slcand -c -o -s8 /dev/netcan0
ip link set up slcan0
```

Please note, that NetCAN+ must be either in „Driver Mode” or in „TCP Raw Server” mode in order to properly work with slcand.

Limitations Unlike the "real" SocketCAN interface, slcan doesn't support reading the CAN controller error flags. This is due to the serial stream nature of this protocol. Only CAN frames will be transported via the serial link and hence, the error flags have to be checked proactively. This mechanism is not currently provided by SocketCAN framework.

⁹<http://en.wikipedia.org/wiki/Socketcan>

¹⁰can-utils package is available in Debian since vesion 8. Debian derivates like Ubuntu should also include this package.

¹¹[Git](#)

¹²[SocketCAN git repository](#)

Usage Examples With `candump` and `cansend` you can make first send/receive tests.

To receive CAN frames, invoke:

```
candump slcan0
```

To send a CAN frame with standard ID 0x100 and payload 0x0011, invoke:

```
cansend slcan0 100#0011
```

Programming Examples The following repository on GitHub provides programming examples for SocketCAN: [programming_examples](#).

1.4 Windows Driver Installation

The use of the Driver Mode for NetCAN Plus CAN Bus Gateway is not necessary in most installations. A more easy way of operation is the use of the VSCAN API (section 2 on page 40), which supports TCP Raw Mode. This way of installation is recommended, especially because it will support any future device in the same way.

The only reason for using the Virtual Com Port Driver are applications specifically designed to operate via ASCII Protocol (section 3 on page 58) on Windows Com Ports.

Since the Virtual Com Port Driver is rarely used, it is not covered in detail in this manual. Instead please use the [NetCom Plus User Manual](#) to get more information.



Figure 13: NetCom Plus Driver Installation

Start the installation program, select „Complete Installation” and follow the instructions. When finished your Windows system has a new Com Port, e.g. named as COM5.

1.5 General Information

1.5.1 LED Status

The red error LED lights up, when a bus error occurs. And the green data LED is on, when a frame was sent to or received from the bus.

1.5.2 Baud-rates and Handshake

Ensure you've set the handshake to RTS/CTS (hardware) when you open the port!

Device	Baud-rate	Handshake
USB-CAN Plus	3 Mbit	RTS/CTS
NetCAN Plus	3 Mbit	RTS/CTS

1.5.3 Pin-out of the 9 Pin D-Sub Connector

Pin	Signal	Description
1	-	
2	CAN_L	CAN_L bus line (dominant level is low)
3	CAN_GND	CAN ground
4	-	reserved
5	-	reserved
6	-	
7	CAN_H	CAN_H bus line (dominant level is high)
8	-	reserved
9	-	

1.5.4 Pin-out of the 4 Pin Connector (USB-CAN Plus mPCIe)

Pin	Signal	Description
1	CAN_H	CAN_H bus line (dominant level is high)
2	CAN_GND	CAN ground
3	CAN_L	CAN_L bus line (dominant level is low)
4	CAN_GND	case ground (connected to Pin 2)

1.5.5 CAN Topology, Wiring and Termination

CAN bus requires three signals (CAN_H, CAN_L and CAN_GND) to be connected between all CAN bus participants. In addition both ends of the bus are to be terminated. Figure 14 on page 30 shows standard termination scheme and Figure 15 on page 30 shows split termination scheme. For more information about these two approaches please refer to this blog article¹³.

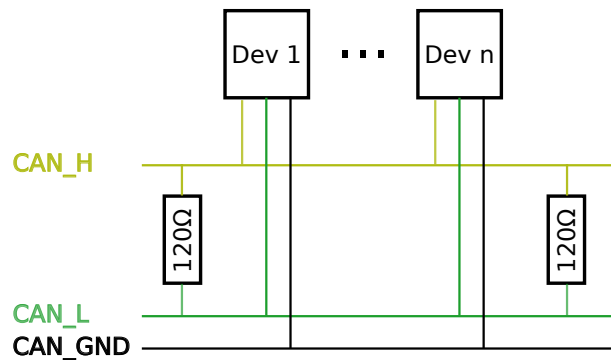


Figure 14: CAN Topology: Standard Termination

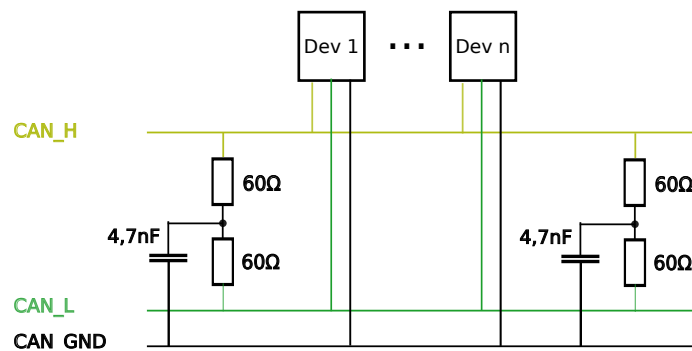


Figure 15: CAN Topology: Split Termination

¹³https://e2e.ti.com/blogs_/b/industrial_strength/archive/2016/07/14/the-importance-of-termination-networks-in-can-transceivers

1.5.6 CAN Signal Levels

VSCAN devices utilize either SN65HVDA540¹⁴ or SN65HVD233¹⁵ transceiver ICs from Texas Instruments Incorporated. Figure 16 on page 31 shows an oscilloscope diagram of a CAN frame sent with the following command:

```
vscansend.exe 192.168.1.157:2001 S1 100#f0
```

Both CAN devices were connected via the cable shown in Figure 14 on page 30 and both CAN_H and CAN_L signal levels correspond to the values defined in the datasheets.

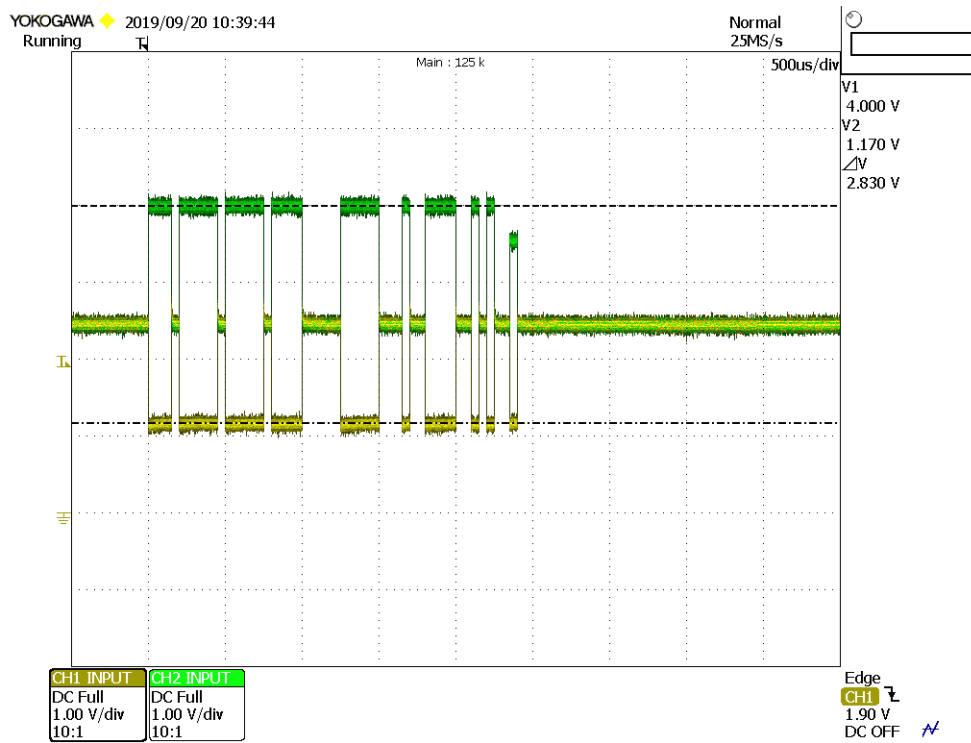


Figure 16: CAN Signals Diagram

¹⁴<http://www.ti.com/lit/ds/symlink/sn65hvda540.pdf>

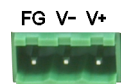
¹⁵<http://www.ti.com/lit/ds/symlink/sn65hvd233-ht.pdf>

1.5.7 Termination Resistors (USB-CAN Plus Devices)

The USB-CAN Plus has an internal termination resistors inside (120Ω), activated by a jumper J1, but external termination shown in Section [CAN Topology, Wiring and Termination](#) is recommended. It's up to you to choose the correct combination and values for your topology.

1.5.8 Terminal Block Power

NetCAN Plus Devices The Terminal Block power connector receives positive voltage on the right (V+) pin. The center (V-) pin connector is negative, which is connected to GND and the case. GND is the same as Field GND (FG), so the standard adapter does not connect to this pin.



NetCAN Plus Mini Devices The Terminal Block power connector for the Mini model receives positive voltage on the left (V+) pin. The right (V-) pin receives ground voltage. Connect the case to Protective Earth Rail.



1.6 Products

Figure 17 shows USB-CAN Plus and NetCAN Plus 120 WLAN devices. Also available are USB-CAN Plus mPCIe and the isolated USB-CAN Plus ISO as also NetCAN Plus 110 without Wireless LAN and NetCAN Plus 110 Mini.



Figure 17: CAN Plus Models

1.7 Dimensions

1.7.1 NetCAN Plus

NetCAN Plus devices use the same case as the NetCom Plus 11x/21x devices.

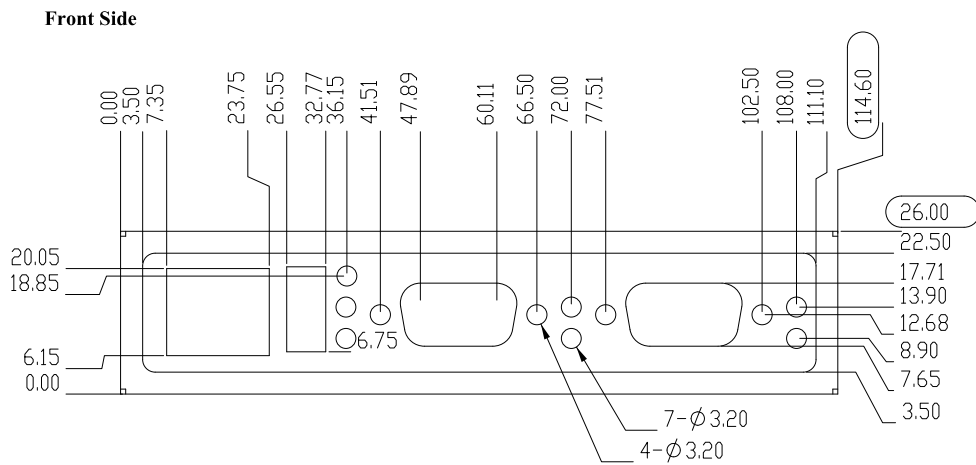


Figure 18: Front Side

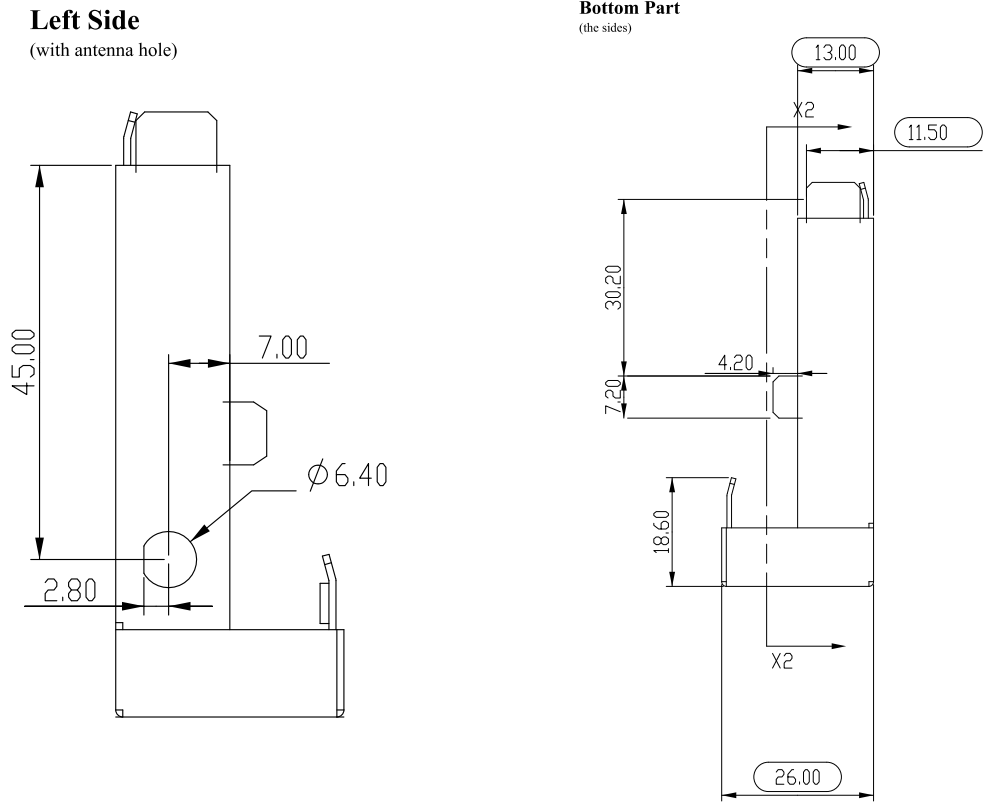


Figure 19: Left and Right Sides

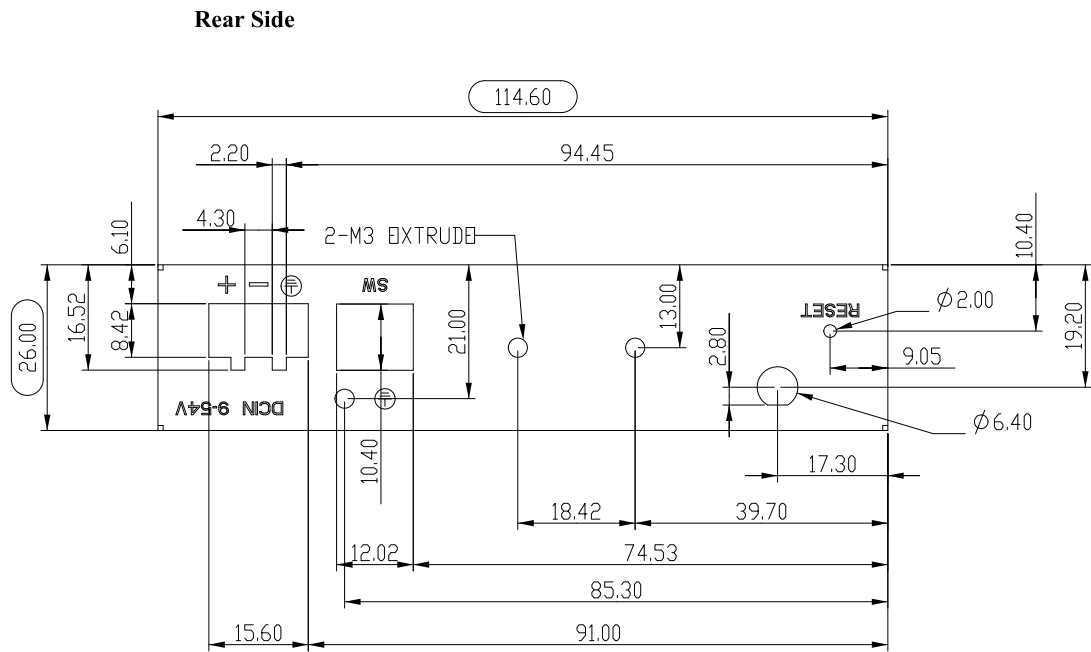


Figure 20: Rear Side

Bottom Side

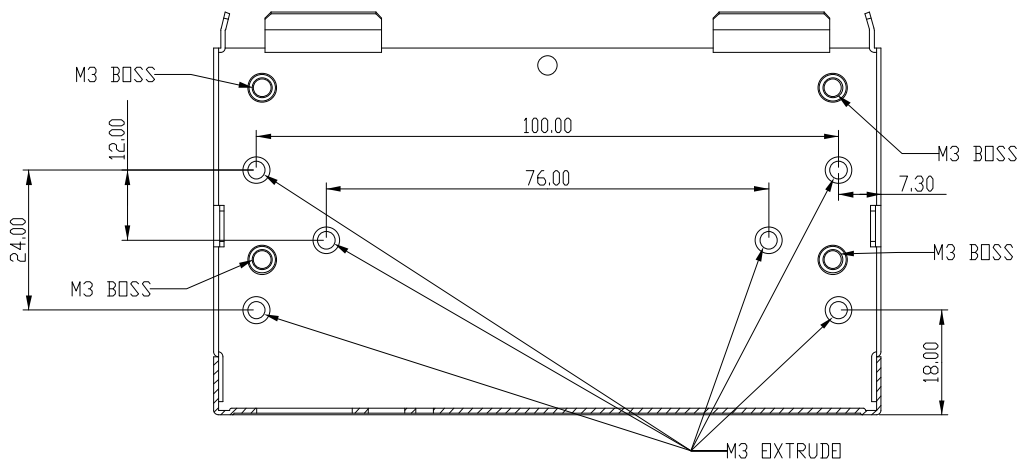


Figure 21: Bottom Side

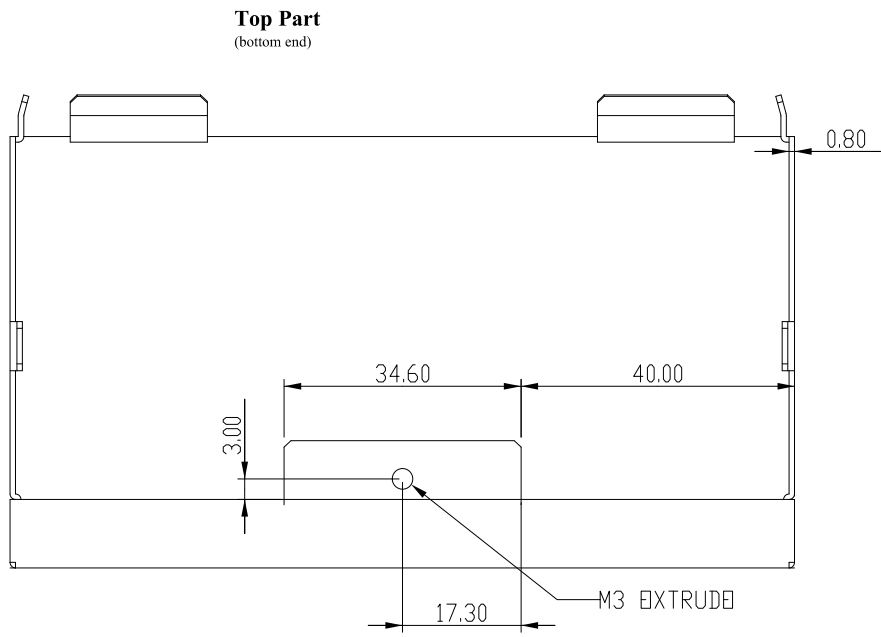


Figure 22: Top Part

1 INSTALLATION

1.7.2 USB-CAN Plus

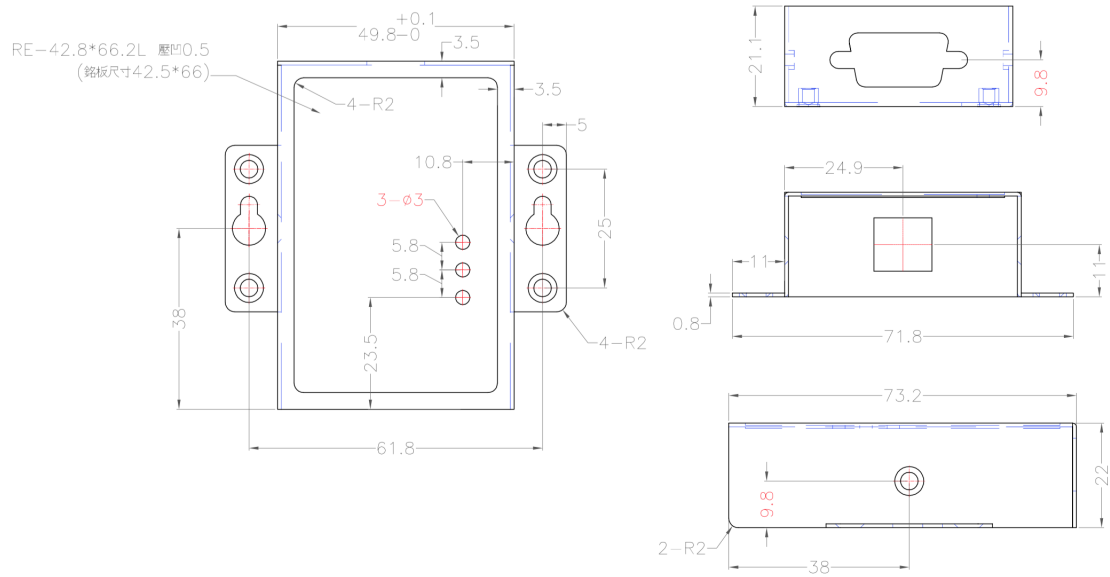


Figure 23: USB-CAN Plus Dimensions

2 Application Programming Interface

2.1 Introduction

The Application Programming Interface (API) gives you the right tools to use all of the functions that the VSCAN devices provide. It will make your life much easier to build your own CAN controlling software with these functions, than to implement your application directly on top of the ASCII protocol. All functions and data structures are explained in the next sub-sections. You'll find programming examples in our repository on GitHub¹⁶.

Windows For Windows, the only thing you must do, is to copy `vs_can_api.dll` (the dynamic link library), `vs_can_api.lib` (the linker input file) and `vs_can_api.h` (the header file) into your project directory. Include the header in your source code and add the `vs_can_api.lib` to your project configuration.

Linux Linux users are advised to use SocketCAN framework described in Section 1.3. If you do want to use the VSCAN API then copy the library (`libvs_can_api.so`) to your global libraries path and add it to your compilation parameters. You must also include the header file (`vs_can_api.h`) in your source file.

Other Operating Systems For other operating systems like MacOS you'll have to implement ASCII protocol yourself. Alternatively you can use Python and `python-can` module. See the next paragraph.

Python `python-can`¹⁷ module provides native support for the ASCII protocol, so VSCAN devices can be used without `vs_can_api` library. You'll find programming examples in our repository on GitHub¹⁸.

Programming Language Bindings You can also use our API library with programming languages other than C/C++. To do so, you'll have to learn how to import the functions from our library to your application. To ease this task, a collection of bindings for the following programming languages will be provided on our GitHub account:

- C#¹⁹
- VisualBasic.NET²⁰

¹⁶https://github.com/visionsystemsgmbh/programming_examples/tree/master/CAN/c/vs_can_api

¹⁷<https://python-can.readthedocs.io/en/3.3.2/>

¹⁸https://github.com/visionsystemsgmbh/programming_examples/tree/master/CAN/python

¹⁹https://github.com/visionsystemsgmbh/programming_examples/tree/master/CAN/c.sharp

²⁰https://github.com/visionsystemsgmbh/programming_examples/tree/master/CAN/VB.NET

- Delphi²¹
- more to come

²¹https://github.com/visionsystemsgmbh/programming_examples/tree/master/CAN/Delphi

2.2 Functions

2.2.1 VSCAN_Open

The VSCAN_Open function opens the CAN channel.

```
VSCAN_HANDLE VSCAN_Open(CHAR *SerialNrORComPortORNet, DWORD Mode);
```

Parameters:

SerialNrORComPortORNet

[in] A char pointer with one of the following values.

- VSCAN_FIRST_FOUND - the first device found will be opened
- Serial number of the specific device
- COM port where the device is located
- IP address and port number of the device

Mode

[in] The mode in which the CAN channel shall be opened.

- VSCAN_MODE_NORMAL - the normal operation mode
- VSCAN_MODE_LISTEN_ONLY - the listen only mode, in which no CAN interaction will be done from the controller
- VSCAN_MODE_SELF_RECEPTION - the self reception mode, in which the device receives also the frames that it sends. *The firmware version must be 1.4 or greater and the DLL version 1.6 or greater.*

Examples:

```
// Windows, Linux
handle = VSCAN_Open(VSCAN_FIRST_FOUND, VSCAN_MODE_NORMAL);

// Windows, Linux
handle = VSCAN_Open("123456", VSCAN_MODE_LISTEN_ONLY);

// Windows
handle = VSCAN_Open("COM3", VSCAN_MODE_NORMAL);

// Linux
handle = VSCAN_Open("/dev/ttyUSB0", VSCAN_MODE_NORMAL);

// Windows, Linux
handle = VSCAN_Open("192.168.254.254:2001", VSCAN_MODE_SELF_RECEPTION);
```

2.2.2 VSCAN_Close

The VSCAN_Close function will close the CAN channel.

```
VSCAN_STATUS VSCAN_Close(VSCAN_HANDLE Handle);
```

Parameters:

Handle

[in] The handle of the CAN device, which shall be closed.

Example:

```
status = VSCAN_Close(handle);
```

2.2.3 VSCAN_Ioctl

You can get and set special information and commands of the CAN device with the `VSCAN_Ioctl` function.

```
VSCAN_STATUS VSCAN_Ioctl(VSCAN_HANDLE Handle, DWORD Ioctl, VOID *Param);
```

Parameters:

Handle

[in] The handle of the CAN device, which should be used.

Ioctl

[in] Tells the function which of the following ioctl should be called.

Param

[in, out] A pointer to the data for the ioctls which are listed below.

VSCAN_IOCTL_SET_DEBUG

You can set the debug verbosity with this ioctl. The higher the debug level the more debug information you get. The `VSCAN_HANDLE` can be `NULL`.

Possible debug levels are:

- `VSCAN_DEBUG_NONE` (*no debug information*)
- `VSCAN_DEBUG_LOW` (*errors from the VSCAN API*)
- `VSCAN_DEBUG_MID` (*information from the VSCAN API*)
- `VSCAN_DEBUG_HIGH` (*errors from system functions*)

Example:

```
status = VSCAN_Ioctl(NULL, VSCAN_IOCTL_SET_DEBUG, VSCAN_DEBUG_HIGH);
```

VSCAN_IOCTL_SET_DEBUG_MODE

You can set the debug mode with this ioctl. It is possible to log the error to the standard error console output (default value) or to save it in a log file. The log file will be saved in the directory from which your application is running and will be named `„vs-can-api.log”`. The `VSCAN_HANDLE` can be `NULL`.

The debug mode defines are:

- `VSCAN_DEBUG_MODE_CONSOLE`
- `VSCAN_DEBUG_MODE_FILE`

Example:

```
status = VSCAN_Ioctl(NULL, VSCAN_IOCTL_SET_DEBUG_MODE, VSCAN_DEBUG_MODE_FILE);
```

VSCAN_IOCTL_GET_API_VERSION

You can request the API version number with this ioctl. Therefore you must commit a pointer of the type VSCAN_API_VERSION to the function. *The DLL version must be 1.6 or greater.*

Example:

```
status = VSCAN_Ioctl(handle, VSCAN_IOCTL_GET_API_VERSION, &version);
```

VSCAN_IOCTL_GET_HWPARAM

This ioctl gives you the possibility to get the hardware parameters (serial number, hardware and software version) of the device. Therefore you must commit a pointer of the type VSCAN_HWPARAM to the function.

Example:

```
status = VSCAN_Ioctl(handle, VSCAN_IOCTL_GET_HWPARAM, &hwparam);
```

VSCAN_IOCTL_SET_SPEED

With this ioctl you can set the speed of your CAN device. The following constant speed values are supported:

- VSCAN_SPEED_1M
- VSCAN_SPEED_800K
- VSCAN_SPEED_500K
- VSCAN_SPEED_250K
- VSCAN_SPEED_125K
- VSCAN_SPEED_100K
- VSCAN_SPEED_50K
- VSCAN_SPEED_20K

With the NetCAN Plus, NetCAN Plus Mini (*firmware version 3.4.8 and greater*) and USB-CAN Plus (*firmware version 2.0 or greater*) you have also the choice to use your desired baudrate directly as the parameter in bit/s.

Example:

```
status = VSCAN_Ioctl(handle, VSCAN_IOCTL_SET_SPEED, VSCAN_SPEED_1M);  
  
// set NET-CAN Plus bitrate to 25kbit/s:  
status = VSCAN_Ioctl(handle, VSCAN_IOCTL_SET_SPEED, (void*)25000);
```

VSCAN_IOCTL_SET_BTR

This ioctl gives you the possibility to configure the speed registers manually (bus timing registers). Therefore you must commit a structure from the type VSCAN_BTR. For more information on this registers, please take a look at the [SJA1000](#) datasheet from NXP Semiconductors or you can use an online bittiming calculator²².

Example:

```
VSCAN_BTR btr = { .Btr0 = 0x00, .Btr1 = 0x14 }; // for 1Mb/s
status = VSCAN_Ioctl(handle, VSCAN_IOCTL_SET_BTR, &btr);
```

VSCAN_IOCTL_SET_FILTER_MODE

This ioctl let you set the desired filter mode for the acceptance code and mask. You can switch between single and dual filter mode. For more informations, please take a look at the [SJA1000](#) datasheet from NXP Semiconductors. *The firmware version must be 1.4 or greater and the DLL version 1.6 or greater.*

Example:

```
status = VSCAN_Ioctl(handle, VSCAN_IOCTL_SET_FILTER_MODE, VSCAN_FILTER_MODE_DUAL);
```

VSCAN_IOCTL_SET_ACC_CODE_MASK

You can set the acceptance code and acceptance mask register with this ioctl. This gives you the possibility to filter for special frame types you want to receive. Therefore you must commit a structure from the type VSCAN_CODE_MASK. For more information on this specific registers, please take a look at the [SJA1000](#) datasheet from NXP Semiconductors.

Example:

```
VSCAN_CODE_MASK codeMask;

status = VSCAN_Ioctl(handle, VSCAN_IOCTL_SET_FILTER_MODE, VSCAN_FILTER_MODE_DUAL);

// will receive the ids between 0x300 and 0x3ff (dual filter mode set filter 2)
codeMask.Code = 0x6000;
codeMask.Mask = 0x1ff0;

status = VSCAN_Ioctl(handle, VSCAN_IOCTL_SET_ACC_CODE_MASK, &codeMask);

// receive all frames on the CAN bus (default)
codeMask.Code = VSCAN_IOCTL_ACC_CODE_ALL;
codeMask.Mask = VSCAN_IOCTL_ACC_MASK_ALL;

status = VSCAN_Ioctl(handle, VSCAN_IOCTL_SET_ACC_CODE_MASK, &codeMask);
```

²²On <http://www.bittiming.can-wiki.info/> select „NXP SJA1000(Philips) or Intel” from the drop-down list and specify the desired bitrate, click on „Request Table” button and you’ll get calculated BTR settings.

VSCAN_IOCTL_SET_FILTER

This ioctl let set you up to 16 advanced filters for the NetCAN Plus, NetCAN Plus Mini (*firmware version 3.4.8 and greater*) and USB-CAN Plus (*firmware version 2.0 or greater*). The logic is `<received_can_id> & Mask == Id & Mask`. *The DLL version must be 1.8 or greater.*

Example:

```
// set two filters
VSCAN_FILTER filter[2];
filter[0].Size = 2;
filter[0].Id = 0x543;
filter[0].Mask = 0xfff;
filter[0].Extended = 1;
filter[1].Id = 0x231;
filter[1].Mask = 0x0ff;
filter[1].Extended = 0;

status = VSCAN_Ioctl(handle, VSCAN_IOCTL_SET_FILTER, filter);

// clear all filters
VSCAN_FILTER filter;
filter.Size = 0;

status = VSCAN_Ioctl(handle, VSCAN_IOCTL_SET_FILTER, &filter);
```

VSCAN_IOCTL_GET_FLAGS

To get extended status and error flags use this ioctl. Commit a DWORD(32bit) pointer as the Param argument. The bit flags and their equivalent macro names are:

- Bit 0: VSCAN_IOCTL_FLAG_RX_FIFO_FULL
- Bit 1: VSCAN_IOCTL_FLAG_TX_FIFO_FULL
- Bit 2: VSCAN_IOCTL_FLAG_ERR_WARNING
- Bit 3: VSCAN_IOCTL_FLAG_DATA_OVERRUN
- Bit 4: VSCAN_IOCTL_FLAG_UNUSED
- Bit 5: VSCAN_IOCTL_FLAG_ERR_PASSIVE
- Bit 6: VSCAN_IOCTL_FLAG_ARBIT_LOST
- Bit 7: VSCAN_IOCTL_FLAG_BUS_ERROR

Take a look at the [SJA1000](#) datasheet from NXP Semiconductors, if you want more information on what's behind bit 2 to 7.

Example:

```
DWORD flags;

status = VSCAN_Ioctl(handle, VSCAN_IOCTL_GET_FLAGS, &flags);
```

VSCAN_IOCTL_SET_TIMESTAMP

You can set on and off the time-stamp functionality with this ioctl. If you switch it on, every received frame will have a valid time-stamp value in the VSCAN_MSG structure. The time base is in milliseconds and will be overrun after 60 seconds (timestamps between 0-60000ms).

Example:

```
status = VSCAN_Ioctl(handle, VSCAN_IOCTL_SET_TIMESTAMP, VSCAN_TIMESTAMP_ON);
```

VSCAN_IOCTL_SET_BLOCKING_READ

This ioctl will set theVSCAN_Read function to blocking mode (default is unblock).

Example:

```
status = VSCAN_Ioctl(handle, VSCAN_IOCTL_SET_BLOCKING_READ, VSCAN_IOCTL_ON);
```

VSCAN_IOCTL_SET_KEEPAIVE

This ioctl will set up TCP keepalive²³ feature for the NetCAN Plus and NetCAN Plus Mini devices (*requires DLL version 1.10.0 or later*). You can define a timeout between entering an idle state, and sending the first keepalive probe, timeout between the probes and number of probes. Though the number of probes can be configured only in Linux, in Windows, this parameter will be ignored.

Example:

```
VSCAN_KEEP_ALIVE KeepAliveConf;
KeepAliveConf.OnOff = 1;           // enable keepalive feature
KeepAliveConf.KeepAliveTime = 30; // send keepalive after 30 seconds of idle
                                   // condition
KeepAliveConf.KeepAliveInterval = 1; // send the next keepalive probe after 1 second
KeepAliveConf.KeepAliveCnt = 5;     // send 5 keepalive probes before reporting the
                                   // socket error

status = VSCAN_Ioctl(handle, VSCAN_IOCTL_SET_KEEPAIVE, (void*)&KeepAliveConf);
```

²³https://en.wikipedia.org/wiki/Keepalive#TCP_keepalive

2.2.4 VSCAN_Read

To read one or more CAN frames from the CAN bus, you must use the `VSCAN_Read` function. The read mode of this function is set to non-blocking mode per default. This means that `VSCAN_Read` will return immediately - even when there are no frames at the moment. Use the ioctl `VSCAN_IOCTL_SET_BLOCKING_READ` to make the `VSCAN_Read` blocking - then it will return only when frames were received.

```
VSCAN_STATUS VSCAN_Read(VSCAN_HANDLE Handle, VSCAN_MSG *Buf, DWORD Size, DWORD *Read);
```

Parameters:

Handle

[in] The handle of the CAN device, which should be used.

Buf

[in] A pointer to one element or an array of the structure `VSCAN_MSG`.

Size

[in] The number of the array elements in `Buf`.

*Read

[out] A pointer to a `DWORD` that will receive the real number of the frames read.

Example:

```
VSCAN_MSG msgs[10];
DWORD read;

status = VSCAN_Read(handle, msgs, 10, &read);
```

2.2.5 VSCAN_SetRcvEvent

With the `VSCAN_SetRcvEvent` function you can set an event which will be set when a frame arrives. There are different versions for Windows and Linux. *The DLL version must be 1.6 or greater.*

```
// Windows Prototype:
VSCAN_STATUS VSCAN_SetRcvEvent (VSCAN_HANDLE Handle, HANDLE Event);
// Linux Prototype:
VSCAN_STATUS VSCAN_SetRcvEvent (VSCAN_HANDLE Handle, sem_t *Event);
```

Parameters:

Handle

[in] The handle of the CAN device, which should be used.

Event

[in] In Windows an event handle of the type `HANDLE` and in Linux a pointer to the `sem_t` union.

Windows Example:

```
// don't forget your own error handling for the API and system functions
// for further informations on these functions take a look at the MSDN

HANDLE hEvent;
DWORD dwRetCode;

hEvent = CreateEvent (NULL, FALSE, FALSE, NULL);

VSCAN_SetRcvEvent (handle, hEvent);

dwRetCode = WaitForSingleObject (hEvent, INFINITE);

switch (dwRetCode)
{
    case WAIT_OBJECT_0 :
        // a CAN frame arrived
        break;
    default:
        // probe for error
}

CloseHandle (hEvent);
```

Linux Example:

```
// don't forget your own error handling for the API and system functions
// take also a look at sem_trywait, sem_timedwait and the rest of the sem_* functions

sem_t sem;
int retCode;

retCode = sem_init(&sem, 0, 0);
VSCAN_SetRcvEvent(handle, &sem);

retCode = sem_wait(&sem);

// a CAN frame arrived

retCode = sem_destroy(&sem);
```

2.2.6 VSCAN_Write

With the `VSCAN_Write` function you can write one or more frames to the CAN bus. The frames will be buffered and send out after some time - this time can grow up to one time slice of the scheduler (Windows = ~16ms and Linux = ~10ms). If you want to send the frames immediately, you must call the function `VSCAN_Flush`.

```
VSCAN_STATUS VSCAN_Write(VSCAN_HANDLE Handle,
                          VSCAN_MSG *Buf, DWORD Size, DWORD *Written);
```

Parameters:

Handle

[in] The handle of the CAN device, which should be used.

Buf

[in] A pointer to one element or an array of the structure `VSCAN_MSG`.

Size

[in] The number of the array elements in `Buf`.

*Written

[out] A pointer to a `DWORD` that will receive the number of frames written.

Example:

```
VSCAN_MSG msgs[10];
DWORD written;

msgs[0].Flags = VSCAN_FLAGS_EXTENDED;
msgs[0].Id = 100;
msgs[0].Size = 1;
msgs[0].Data[0] = 0x1B;

// we will send ten frames with the same data
// to the ids 100-109
for (i = 1; i < 10; i++)
{
    memcpy(msgs + i, &msgs[0], sizeof(msgs[0]));
    msgs[i].Id++;
}

status = VSCAN_Write(handle, msgs, 10, &written);
```

2.2.7 VSCAN_Flush

The VSCAN_Flush function will send all frames immediately out to the CAN bus.

```
VSCAN_STATUS VSCAN_Flush(VSCAN_HANDLE Handle);
```

Parameters:

Handle

[in] The handle of the CAN device, whose data should be flushed.

Example:

```
status = VSCAN_Flush(handle);
```

2.2.8 VSCAN_GetErrorString

The `VSCAN_GetErrorString` function retrieves the associated human readable error string.

```
VOID VSCAN_GetErrorString(VSCAN_STATUS Status, CHAR *String, DWORD MaxLen);
```

Parameters:

Status

[in] The status for which the error string should be retrieved.

String

[out] A pointer of a string array which will receive the error string.

MaxLen

[in] The maximum possible length of the error string (without the terminating zero).

Example:

```
VSCAN_STATUS status = VSCAN_ERR_NO_DEVICE_FOUND;  
char string[33];  
  
VSCAN_GetErrorString(status, string, 32);  
  
printf(string);
```

2.3 Types and Structures

2.3.1 VSCAN_HANDLE

```
typedef int VSCAN_HANDLE;
```

This type definition holds the handle of an opened CAN channel. In this case the value is greater than zero. Otherwise the value is equal to one of the type definition VSCAN_STATUS.

2.3.2 VSCAN_STATUS

```
typedef int VSCAN_STATUS;
```

The type definition VSCAN_STATUS can have one of the following status value.

- **VSCAN_ERR_OK** - indicates that everything is okay
- **VSCAN_ERR_ERR** - indicates a general error
- **VSCAN_ERR_NO_DEVICE_FOUND** - indicates that no CAN device was found with the specific functions
- **VSCAN_ERR_SUBAPI** - indicates that an error occurred in a subordinated library
- **VSCAN_ERR_NOT_ENOUGH_MEMORY** - indicates that there is not enough memory to complete the function
- **VSCAN_ERR_NO_ELEMENT_FOUND** - indicates that there is no requested element available (e.g. from an input buffer)
- **VSCAN_ERR_INVALID_HANDLE** - indicates that the handle which is used is not valid (e.g. CAN channel closed)
- **VSCAN_ERR_IOCTL** - indicates that an ioctl request failed; ensure that you've used the right parameter values
- **VSCAN_ERR_MUTEX** - indicates that there was a problem with a used mutex in the VSCAN API (e.g. timeout)
- **VSCAN_ERR_CMD** - indicates that there was a problem to complete a given command on the CAN device

2.3.3 VSCAN_API_VERSION

This structure holds the version information of the API.

```
typedef struct
{
    UINT8  Major;
    UINT8  Minor;
    UINT8  SubMinor;
} VSCAN_API_VERSION;
```

2.3.4 VSCAN_HWPARAM

This structure holds the values of the hardware parameters.

```
typedef struct
{
    UINT32 SerialNr;
    UINT8  HwVersion;
    UINT8  SwVersion;
    UINT8  HwType;
} VSCAN_HWPARAM;
```

The `SerialNr` element comprised the unique serial number reserved for this device. The `HwVersion` holds the revision of the CAN hardware and in the opposite `SwVersion` the actual software version of the firmware. The upper four bits of these variables hold the major and the lower four the minor number. And `HwType` retrieves the type of CAN hardware (e.g. Serial, USB, Net).

2.3.5 VSCAN_MSG

The structure is used for the information of each CAN frame which will be received or transmitted.

```
typedef struct
{
    UINT32 Id;
    UINT8  Size;
    UINT8  Data[8];
    UINT8  Flags;
    UINT16 Timestamp;
} VSCAN_MSG;
```

The element `Id` holds the identifier of the standard or extended CAN frame. The width of the data bytes is saved in the `Size` element and the maximum eight data bytes itself in `Data`. The member `Flags` is a bit-mask to retrieve or set some of these special flags: `VSCAN_FLAGS_STANDARD` - is set when this message is a standard frame, `VSCAN_FLAGS_EXTENDED` - this bit is set in the case of an extended frame and the `VSCAN_FLAGS_REMOTE` bit could be set, when it was or should be a remote request frame. The `Timestamp` element holds the time-stamp of the received frame, when this special function is activated over the ioctl [VSCAN_IOCTL_SET_TIMESTAMP](#). If a frame was received with a time-stamp, also the flag `VSCAN_FLAGS_TIMESTAMP` is set in the member `Flags`.

2.3.6 VSCAN_BTR

This structure is used for the setting of the bus timing register.

```
typedef struct
{
    UINT8 Btr0;
    UINT8 Btr1;
} VSCAN_BTR;
```

The elements `Btr0` and `Btr1` implements the values for the bus timing register one and two. For more information read the chapter [2.2.3](#) or take a look at the [SJA1000](#) datasheet from NXP Semiconductors.

2.3.7 VSCAN_CODE_MASK

The structure stores the acceptance filter code and filter mask.

```
typedef struct
{
    UINT32 Code;
    UINT32 Mask;
} VSCAN_CODE_MASK;
```

The structure member `Code` stores the acceptance code and `Mask` the acceptance mask. For more information see chapter [2.2.3](#) or take a look at the [SJA1000](#) datasheet from NXP Semiconductors.

3 ASCII Command Set

3.1 Introduction

The ASCII command set gives you the possibility to use the VSCAN device even with a simple terminal program. This makes it very easy for you, to send some frames by hand or to sniff the frames on the CAN bus in a simple human readable view. It will also be possible to use such a simple semantic in a scripting system (e.g. Linux bash-script).

Every binary data will be sent and received by their ASCII hexadecimal equivalents. The return values of all functions will be CR (ASCII 13) if the function succeeds or BELL (ASCII 7) if the function fails. Some functions have extended return values, but this will be described per function in the command description.

The received frames will be send directly to your ASCII communication channel - e.g. serial port or network connection.

3.2 Commands

3.2.1 Open the CAN Channel

The CAN channel will be opened with the command `O[CR]`, `L[CR]` or `Y[CR]`. The difference between these three types is, that the second command will open the channel in a listen only mode, in which no bus interaction will be done from the controller. The last command will open the channel in a self reception mode, in which the device will also receive the frames that it sends (*only available in firmware version 1.4 or greater*). Before you will use one of the commands, you should setup a bus timing with the command `S` or `s`. *Anyway, the last configured bit rate is stored in the device and used as the standard bus timing at power up.*

Examples:

Open the channel in normal operation mode.

```
O[CR]
```

Open the channel in the listen only mode.

```
L[CR]
```

Open the channel in the self reception mode.

```
Y[CR]
```

3.2.2 Close the CAN Channel

The CAN channel will be closed with the command `C[CR]`. The command is only active if the CAN channel is open.

Example:

```
C[CR]
```

3.2.3 Setup the Bus Timing (Standard)

The bus timing will be setup-ed with the command `Sn[CR]`. With the NetCAN Plus, NetCAN Plus Mini (*firmware version 3.4.8 and greater*) and USB-CAN Plus (*firmware version 2.0 or greater*) it is also possible to define the bitrate directly (e.g. `S125000[CR]`). You can only use this command if the CAN channel is closed.

Parameters:

`n`

Could be one of the following values:

- 1 - 20 KBit
- 2 - 50 KBit
- 3 - 100 KBit
- 4 - 125 KBit
- 5 - 250 KBit
- 6 - 500 KBit
- 7 - 800 KBit
- 8 - 1 MBit

Example:

Configure a bus timing of 1 MBit.

```
S8[CR]
or directly with
S1000000[CR]
```

3.2.4 Setup the Bus Timing (Advanced)

A more user defined bus timing could be configured with the command `sxxyy[CR]`. This command is not available for the „NetCAN Plus” series. As with the standard bus timing command above, you can only use it when the CAN channel is closed.

Parameters:

xx

This is the hex value of the bit timing register 0.

For more information please take a look at the [SJA1000](#) datasheet from NXP Semiconductors.

yy

This is the hex value of the bit timing register 1.

Example:

Configure a bus timing of 100 KBit.

```
s041C[CR]
```

3.2.5 Transmitting a Standard Frame

Transmitting a standard frame (11bit) over the CAN bus will be done with command `tiiiidd[0-8]`. The return value will be `z[CR]` or the normal error byte (BELL). As you can imagine, this command is only available when the CAN channel is open.

Parameters:`iii`

Standard frame (11bit) identifier.

`l`

Data length (0-8)

`dd[0-8]`

Data bytes in hex. The number of the bytes must be equal with the data length field.

Example:

Sending a frame with id 0x111 and three data bytes 0x10, 0x20, 0x30.

```
t1113102030[CR]
```

3.2.6 Transmitting a Standard Remote Request Frame

Transmitting a standard remote frame (11bit) over the CAN bus will be done with `riiil`. The return value will be `z[CR]` or the normal error byte (BELL). This command is only available when the CAN channel is open.

Parameters:`iii`

Standard frame (11bit) identifier.

`l`

Data length (0-8)

Example:

Sending a remote request frame with id 0x111 and request 3 data bytes.

```
r1113[CR]
```

3.2.7 Transmitting an Extended Frame

Transmitting an extended frame (29bit) over the CAN bus will be done with command `Tiiiiiiiidd[0-8]`. The return value will be `Z[CR]` or the normal error byte (BELL). The command is only available when the CAN channel is open.

Parameters:`iiiiiii`

Extended frame (29bit) identifier.

`l`

Data length (0-8)

`dd[0-8]`

Data bytes in hex. The number of the bytes must be equal with the data length field.

Example:

Sending an extended frame with id 0x111 and three data bytes 0x10, 0x20, 0x30.

```
T000001113102030[CR]
```

3.2.8 Transmitting an Extended Remote Request Frame

Transmitting an extended remote request frame (29bit) over the CAN bus will be done with `Riiiiiiiil`. The return value will be `Z[CR]` or the normal error byte (BELL). The command is only available when the CAN channel is open.

Parameters:`iiiiiii`

Extended frame (29bit) identifier.

`l`

Data length (0-8)

Example:

Sending an extended remote request frame with id 0x111 and a request for 3 data bytes.

```
R000001113[CR]
```

3.2.9 Set Time-Stamps

The time-stamp command will set the time-stamp functionality on received frames on or off. This command will only function, when the CAN channel is closed.

Example:

Will set time-stamps on or off.

```
Z1 [CR]
Z0 [CR]
```

3.2.10 Set Filter Mode

The command D1 [CR] switch on the dual filter mode and with D0 [CR] you switch over to the single filter mode. For more information please take a look at chapter 2.2.3. This command is only available if the CAN channel is closed. *The firmware version must be 1.4 or greater.*

Example:

Will set dual or single filter mode.

```
D1 [CR]
D0 [CR]
```

3.2.11 Set Acceptance Code and Mask Register

With the acceptance code command Mxxxxxxxx [CR] and mask register command mxxxxxxxx [CR], you have the choice to filter for specific CAN messages directly on the CAN controller side. For example the acceptance code addresses in SJA1000 format are MC0C1C2C3, same for the mask addresses. If a mask bit is set, then this element from the code bit will be ignored. For more information please take a look at chapter 2.2.3. For the „NetCAN Plus” you could only set the bits for the id part and not from the data. This command is only available if the CAN channel is closed.

Example:

We will filter for all standard frames between 0x300 and 0x3ff (dual filter mode set filter 2).

```
M00006000 [CR]
m00001ff0 [CR]
```

3.2.12 Set Advanced Filter

The advanced filter command is only available for the NetCAN Plus, NetCAN Plus Mini (*firmware version 3.4.8 and greater*) and USB-CAN Plus (*firmware version 2.0 or greater*). Set the filter with `fxxxxxxxx,yyyyyyyy,z[CR]`, where `x` is the id, `y` the mask and `z` for optional extended ('e') or standard frames ('s'). This equals to `<received_can_id> & mask == id & mask`. You can set up to 16 individual filters when calling the command multiple times, or deleting the filter set with id and mask set to zero.

First we set up filter for the id 123 and 234 and then deleting all filters.

```
f123,fff[CR]
f234,ffe,e[CR]
f0,0[CR]
```


3.2.13 Get Status Flags

To get the status bits when an error occurred, you must use the command `F[CR]`. For more information on the bit-mask please take a look at chapter [2.2.3](#). The command is only available if the CAN channel is open.

Example:

Retrieve the status bits as a hexadecimal value. The return value will be formatted like this: `Fxx[CR]`

```
F[CR]
```

3.2.14 Get Version Information

To retrieve the current hard- and software version of the device, you must use the command `V[CR]`. The command is always available and will return the versions formatted like this: `Vxxyy[CR]`. The hardware version is coded in the `xx` (major and minor version) and the software version in the `yy` (also coded as major and minor).

Example:

Retrieving the versions.

```
V[CR]
```

3.2.15 Get Serial Number

With the command `N[CR]` you will retrieve the serial number of the device. This command is always active and will return the decimal serial number like this: `N12345678[CR]`.

Example:

Retrieving the serial number.

```
N[CR]
```

3.2.16 Get Extra-Information

You can retrieve extra information with the command `I[CR]`. The command is always available and will return the values of the bus timing registers, the acceptance code and mask register values, the counter of the arbitration lost interrupt, the arbitration lost capture register, the status register and the value of the error code capture register of the CAN chip. For more information please take a look at the [SJA1000](#) datasheet from NXP Semiconductors.

Example:

Retrieving the extra-information.

```
I[CR]
```

4 Tools

4.1 Firmware-Update

For USB-CAN Plus devices you can use our tool `vs_fw_update.exe`. It will determine the type of device and will use the correct baudrate. If it did not, you can set the baudrate explicitly.

```
vs_fw_update.exe COMx vs_can_1_7.bin 3000000
```

For the NetCAN Plus you should use the base64 web-frontend update file (e.g. `net-can_Plus_XXX_3.1.1.bin.b64`), to update CAN bus and network operation firmware in the device.

4.2 Busmaster

BUSMASTER²⁴ is an Open Source Software utility to test, simulate and analyze data bus systems like CAN. The following usage example was tested on BUSMASTER 3.2.2 (use the download link from our product page).

If you use NetCAN Plus, make sure it is either in the Driver or TCP Raw Server modes.

We will set up a USB-CAN Plus device assigned to COM143 and set CAN bitrate to 50kbit/s.

1. Start BUSMASTER
2. Select VScom CAN API: CAN->Driver Selection->VScom CAN-API
3. In the „Hardware Selection” dialog move „VScom CAN Device” from „Available CAN Hardware” to „Configured CAN Hardware” (see Figure 24)
4. Click „Advanced” button to configure USB-CAN Plus device (see Figure 25). Setup COM143 as Serial Port, set Baudrate to 50kbit and Acceptance Mask to **0xFFFFFFFF**
5. Once in the main window click on the green „Connect/Disconnect” button. If the connection was successfully established, this button would change to red with a caption „Disconnect”
6. If there are CAN frames on your CAN bus, you’ll see them in the „Message Window” (see Figure 26)

To send a CAN message perform the above actions in order to establish a connection and select CAN->Transmit Window (see Figure 27). To create a new CAN frame double-click on „Add Message” in the „Tx Frame List” and enter message ID in hex (in this example it is 0x102) and press „Enter”. A new message is created. Perform single-click

²⁴http://www.vscom.de/download/multiio/winXP/tools/BUSMASTER_Installer_Ver_3.2.2_VScom_1.0.exe

on the „Add Message” and then select the previously created message and the „Send Message” button will be activated. Clicking on the „Send Message” button would send the selected frame.

You can now change the frame payload in the „Data Byte View (HEX)” window below. DLC, Message Type, etc. can be modified in the „Tx Frame List”.

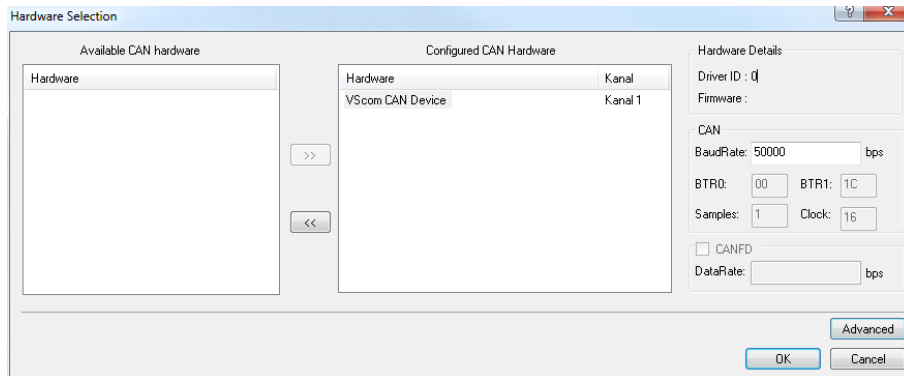


Figure 24: BUSMASTER: Select Hardware Interface

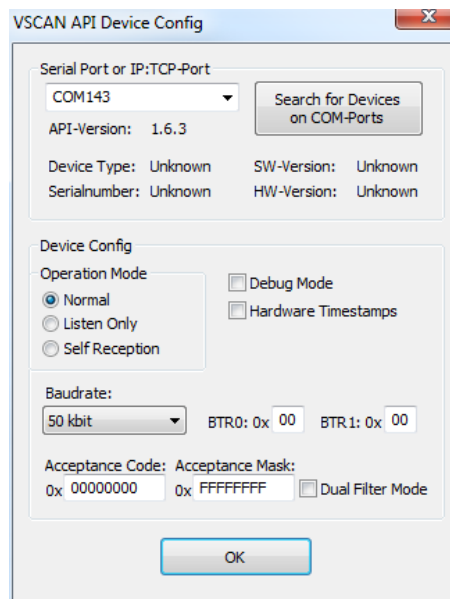


Figure 25: BUSMASTER: Configure Device

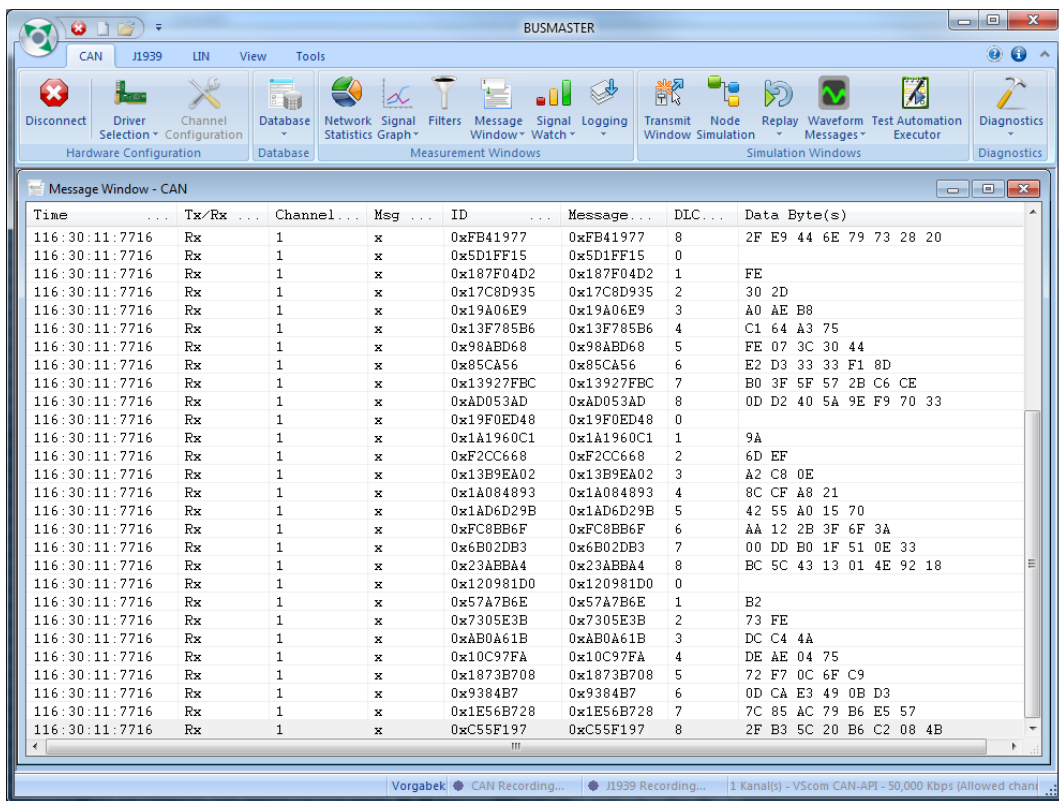


Figure 26: BUSMASTER: Received Frames

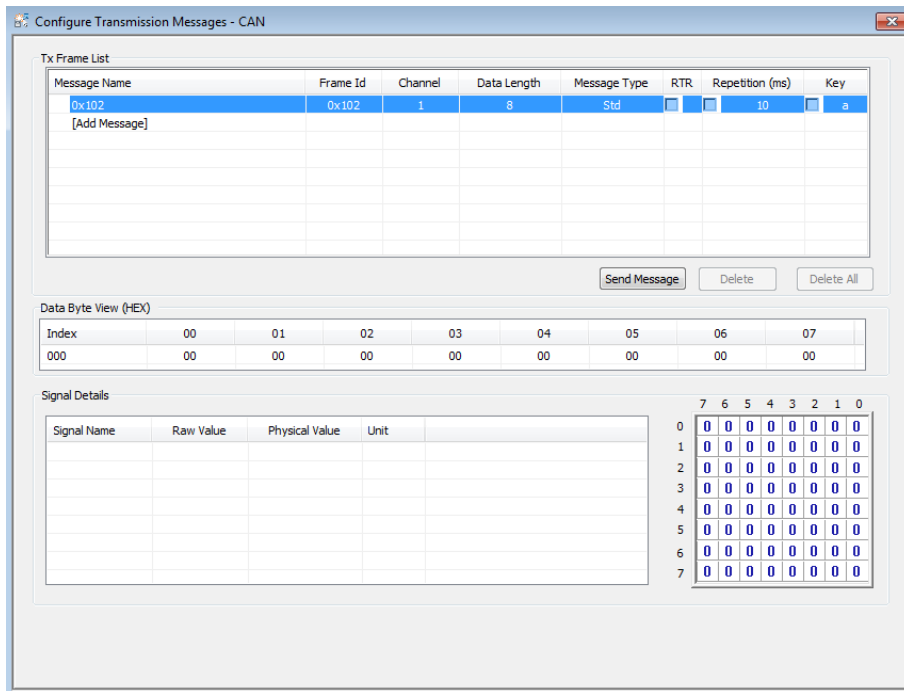


Figure 27: BUSMASTER: Transmit Window

4.3 vscandump and vscansend

`vs_can_tools.zip`²⁵ package provides the following tools for sending/receiving CAN frames on the command prompt:

- `vscandump.exe` - connects to either USB-CAN Plus or NetCAN Plus and shows all received frames
- `vscansend.exe` - connects to either USB-CAN Plus or NetCAN Plus and sends specified CAN frame

Please note that you can use only one program with one CAN adapter at a time. Provided you have two or more CAN adapters, you can then use `vscandump.exe` on one of them and `vscansend.exe` on another.

If you use NetCAN Plus, make sure it is either in the Driver or TCP Raw Server modes.

`vscandump.exe` has the following parameters:

- COM port or IP address and TCP port number
- CAN bitrate in ASCII notation, i.e. S1, S2 etc. (see Section [Setup the Bus Timing \(Standard\)](#))

In this example `vscandump.exe` will connect to a NetCAN Plus with IP address 192.168.254.254 and default TCP port 2001 at 50Kbit/s. You can see two different frames. The first one with ID 0x111 is a standard frame and the extended one with ID 0x00000110. Both have a payload of 4 bytes.

```
c:\>vscandump.exe 192.168.254.254:2001 S2
VSCAN-API Version 1.8.0
111 [4] 00 11 22 33
00000110 [4] 00 11 22 33
```

`vscansend.exe` has the following parameters:

- COM port or IP address and TCP port number
- CAN bitrate in ASCII notation, i.e. S1, S2 etc. (see Section [Setup the Bus Timing \(Standard\)](#))
- CAN frame to send

CAN frame syntax looks like this:

- CAN ID in hex: either 3 digits for the standard ID or 8 digits for the extended ID
- CAN ID delimiter '#'

²⁵http://www.vscom.de/download/multiio/winXP/tools/vs_can_tools.zip

- CAN payload in hex. You can specify between 0 and 8 bytes

The first example sends a standard frame via USB-CAN Plus device, the second example sends an extended frame:

```
c:\>vscansend.exe COM143 S2 100#00aadd
```

```
c:\>vscansend.exe COM143 S2 00000102#00aadd44
```


4.4 vs_can_test_simple.exe

In addition to `vscansend.exe` and `vscandump.exe` `vs_can_tools.zip` also provides a self-diagnostic tool `vs_can_test_simple.exe`. You can use it either with two VScan CAN devices or with only one device. In both cases, you'll need a properly terminated CAN cable.

If you use NetCAN Plus, make sure it is either in the Driver or TCP Raw Server modes.

Once invoked this utility sends a CAN frame on the first given CAN channel and receives it on the other channel. After this, CAN channels will be swapped and a send/receive cycle will be repeated.

`vs_can_test_simple.exe` has the following parameters:

- first CAN channel: COM port or IP address and TCP port number
- second CAN channel: COM port or IP address and TCP port number
- loop flag (-1) is an optional parameter, that repeats the test endlessly

The first example shows a test with two USB-CAN Plus devices:

```
c:\>vs_can_test_simple.exe COM168 COM32
vs_can_test_simple.exe v1.6

DLL version: 1.8.0

SUCCESS: Sending frame from COM168 to COM32 succeeded.
SUCCESS: Sending frame from COM32 to COM168 succeeded.
```

The second example shows a test with only one USB-CAN Plus device. In this case the same COM port is given twice:

```
c:\>vs_can_test_simple.exe COM168 COM168
vs_can_test_simple.exe v1.6

DLL version: 1.8.0

SUCCESS: Sending frame from COM168 to COM168 succeeded.
SUCCESS: Sending frame from COM168 to COM168 succeeded.
```

4.5 Wireshark

Wireshark²⁶ is a well-known network packet sniffer. It is now possible to sniff via Socket-CAN interface (Linux only) and parse CAN frames (Linux/Windows) . Figure 28 shows frames captures via USB-CAN Plus device (/dev/ttyUSB0 attached as slcan0).

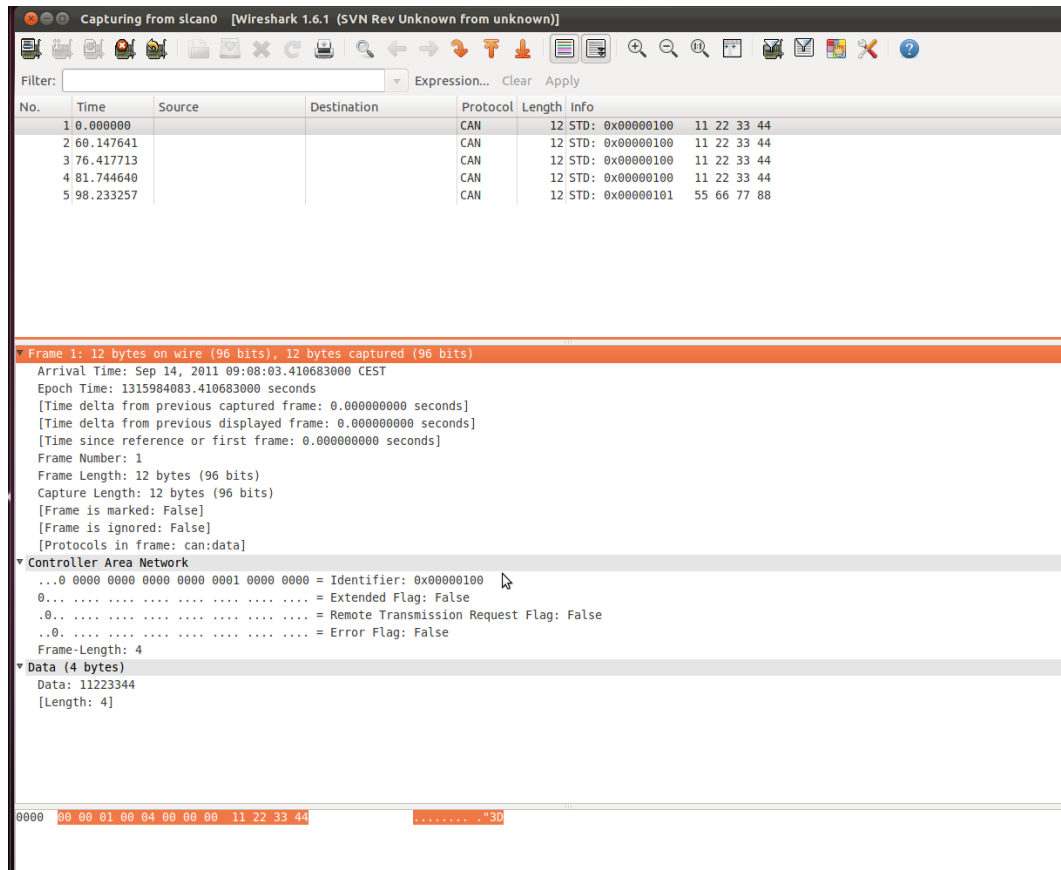


Figure 28: Wireshark

²⁶www.wireshark.org

4.6 CANopen

4.6.1 Introduction

CANopen is a CAN-based higher layer protocol. It was developed as a standardized embedded network with highly flexible configuration capabilities. CANopen was designed for motion-oriented machine control networks, such as handling systems. By now it is used in various application fields, such as medical equipment, off-road vehicles, maritime electronics, railway applications or building automation.

CANopen unburdens the developer from dealing with CAN-specific details such as bit-timing and implementation-specific functions. It provides standardized communication objects for real-time data, configuration data as well as network management data.²⁷

One of the protocol implementations is the CanFestival Project (www.canfestival.org). It is an Open Source (LGPL and GPL) CANopen framework and is part of the Beremiz Project (www.beremiz.org), an Open Source framework for automation. CanFestival focuses on providing an ANSI-C platform independent CANopen stack that can be implemented as master or slave nodes on PCs, Real-time IPCs, and Micro-controllers. VScom devices will be already supported in the latest CanFestival version.

As an alternative to CanFestival, you can also use a Python-based framework `canopen`²⁸. There is also a programming example in our GitHub repository²⁹.

Wireshark can be used to capture and analyze CANopen traffic since version 1.7.1 (see Section 4.5).

Before trying the CanFestival examples please make sure you can send/receive CAN frames using `vscandump` and `vscansend` or `Busmaster`. If you use `NetCAN Plus`, make sure it is either in the `Driver` or `TCP Raw Server` modes.

4.6.2 Running Example

You'll find a small CanFestival example in the CAN examples archive³⁰ showing the communication between master and slave nodes. The following baudrates are supported: 20K, 50K, 100K, 125K, 250K, 500K and 1M. To execute this example decompress one of the following archives:

- `CANopen_example_win32.zip` for Windows
- `CANopen_example_linux.tar.bz2` for Linux

Under Windows connect two VScom CAN devices - for example two USB-CAN Plus devices installed as COM10 and COM11. Open two command windows and change to the directory where the CANopen examples were extracted to and execute

²⁷For more information see the website of CAN in Automation organization www.can-cia.org

²⁸<https://canopen.readthedocs.io/en/latest/>

²⁹https://github.com/visionssystemsgmbh/programming_examples/blob/master/CAN/python/vscanopen.py

³⁰http://www.vscom.de/download/multiio/winXP/tools/CAN_Examples.ZIP.exe

```
TestMasterSlave -s COM10 -S 125K -M none -l libcanfestival_can_vscom.dll
```

in the first window and

```
TestMasterSlave -m COM11 -M 125K -S none -l libcanfestival_can_vscom.dll
```

in the second. Figure 29 shows the output messages of both nodes.

```

(a) Master
Master: 0 0 0 0 0 0 0 0 0 0 0 0
Master : ConfigureSlaveNode 02
TestMaster_operational
TestMaster_post_sync
Master: 0 0 0 0 0 0 0 0 0 0 0 0
TestMaster_post_TPDO MasterSyncCount = 0
Master : Ask RTR PDO (0x1402)
Master : Change slave's transmit type to 0xFF
TestMaster_post_sync
Master: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 1235 0
TestMaster_post_TPDO MasterSyncCount = 1
TestMaster_post_sync
Master: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 1235 0
TestMaster_post_TPDO MasterSyncCount = 2
TestMaster_post_sync
Master: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 1235 0
TestMaster_post_TPDO MasterSyncCount = 3
TestMaster_post_sync
Master: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 1235 0
TestMaster_post_TPDO MasterSyncCount = 4
TestMaster_post_sync
Master: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 1235 0
TestMaster_post_TPDO MasterSyncCount = 5
TestMaster_post_sync
Master: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 1235 0
TestMaster_post_TPDO MasterSyncCount = 6
Master received EMCV message. Node: 02 ErrorCode:
Master received EMCV message. Node: 02 ErrorCode:
TestMaster_post_sync
Master: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 1235 1234
TestMaster_post_TPDO MasterSyncCount = 7

(b) Slave
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 59 1234
TestSlave_operational
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 60 1234
TestSlave_post_TPDO
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 61 1235
TestSlave_post_TPDO
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 62 1236
TestSlave_post_TPDO
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 63 1237
TestSlave_post_TPDO
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 64 1238
TestSlave_post_TPDO
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 65 1239
TestSlave_post_TPDO
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 66 1240
TestSlave_post_TPDO
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 67 1241
TestSlave_post_TPDO
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 68 1242
TestSlave_post_TPDO
TestSlave_post_sync

```

Figure 29: TestMasterSlave under Windows

Under Linux connect two VScom CAN devices - for example two USB-CAN Plus devices installed as `/dev/ttyUSB0` and `/dev/ttyUSB1`. Open two terminal windows and change to the directory where CANOpen examples were extracted to and execute

```
export LD_LIBRARY_PATH=.
./TestMasterSlave -s "/dev/ttyUSB0" -S 125K -M none -l ./libcanfestival_can_vscom.so
```

in the first window and

```
export LD_LIBRARY_PATH=.
./TestMasterSlave -m "/dev/ttyUSB1" -M 125K -S none -l ./libcanfestival_can_vscom.so
```

in the second. Figure 30 shows the output messages of both nodes.

4.6.3 Compilation Instructions

CanFestival is stored in a Mercurial³¹ repository. To get CanFestival source code execute:

```
hg clone http://dev.automforge.net/CanFestival-3/
cd CanFestival-3
```

Configure and compile the library and examples:

³¹Mercurial

```

user@debianvm: ~/projects/src/CanFest
File Edit View Terminal Tabs Help
Master: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
TestMaster_post_sync
Master: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
TestMaster_post_sync
Master: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Master : ConfigureSlaveNode 02
TestMaster_operational
TestMaster_post_sync
Master: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
TestMaster_post_TP00 MasterSyncCount = 0
Master : Ask RTR PDO (0x1402)
Master : Change slave's transmit type to 0xFF
TestMaster_post_sync
Master: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
TestMaster_post_TP00 MasterSyncCount = 1
TestMaster_post_sync
Master: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
TestMaster_post_TP00 MasterSyncCount = 2
TestMaster_post_sync
Master: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
TestMaster_post_TP00 MasterSyncCount = 3
Master: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
TestMaster_post_TP00 MasterSyncCount = 4

```

```

user@debianvm: ~/projects/src/CanFest
File Edit View Terminal Tabs Help
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 40 1234
TestSlave_operational
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 41 1234
TestSlave_post_TP00
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 42 1235
TestSlave_post_TP00
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 43 1236
TestSlave_post_TP00
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 44 1237
TestSlave_post_TP00
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 45 1238
TestSlave_post_TP00
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 46 1239
TestSlave_post_TP00
TestSlave_post_sync
Slave: 1 1 0 0 1 0 1 0 16 ff00ff00 abcd 47 1240
TestSlave_post_TP00
TestSlave_post_sync

```

(a) Master

(b) Slave

Figure 30: TestMasterSlave under Linux

```
./configure --can=vscom
make
```

The TestMasterSlave is located under `examples/TestMasterSlave/` and the vscom library is located under `drivers/can_vscom/`.

For detailed information about these examples and about using CanFestival in your project refer to the `doc/` folder.

4.7 J1939

The SAE J1939 protocol is commonly used in the commercial vehicle area for communication in the commercial vehicle. But there are also other protocols based on J1939 like NMEA2000 for ships, ISOBUS for agricultural vehicles, MilCAN for military use and in the FMS (Fleet Management System) standard.

The Controller Area Bus (CAN) is used as the underlying hardware layer which is robust and has a priority based message protocol. It is also used in retail customer vehicles, but mostly with proprietary higher level protocols.

The CAN standard was upgraded from 11 bit to 29 bit identifiers to fulfill the need to embed all J1939 general message information (Protocol Data Unit) in the CAN identifier field. And a more complex layer was implemented to send more than 8 bytes per J1939 message to its destination.

There are several frameworks that can be used for the J1939 related software development.

4.7.1 VSCAN J1939 Library

We provide a special library called VSCAN_j1939³² for both Windows and Linux operating systems. For more information, please refer to its dedicated manual³³.

Please note that the devices must be ordered with the J1939 feature in order to be able to work with this library. This activation cannot be done after the purchase.

4.7.2 SocketCAN Based Implementation

If you already use SocketCAN, you can utilize the newly introduced J1939 framework. For more information, please refer to our examples repository³⁴.

4.8 Wrapper DLL System

In Linux operating system any application software may use CAN bus adapters from different manufacturers, without modifying the program. An official CAN API named SocketCAN exists for the Linux Kernel.

In Windows systems the situation is different. There is neither an API from Microsoft³⁵, nor a widely accepted de-facto standard used by manufacturers. All products come with a proprietary driver to access the CAN bus adapter. On top is a set of libraries for application programmers, encapsulating the hardware in higher layer function sets.

³²https://www.vscom.de/download/multiio/Windows7/driver/VSCAN_J1939.zip

³³https://www.vscom.de/download/multiio/others/info/VSCAN_J1939_Manual.pdf

³⁴https://github.com/visionsystemsgmbh/programming_examples/tree/master/CAN/c/socketcan

³⁵See serial ports, manufacturers use the Windows API, products are interchangeable.

Application software tailored to use one set of libraries binds the application to certain CAN bus hardware. Those libraries come in the form of DLLs.

VScan provides a system of wrapper DLLs. They provide the possibility to use any VSCAN product as a replacement of products from other manufacturers, when those are given a set of DLLs. The wrapper DLLs use the same name as the original set. They provide the same functions used by the application software, so the existing software shall not notice that exchange.

Installation: Copy the desired wrapper DLL over the original API DLL in your program directory, thus replacing the version from the other manufacturer. It may be useful to create a backup copy before doing so. You must also copy the latest VSCAN-API (vs_can_api.dll) into the same directory.

Configuration: If you want to specify a special configuration for the mapping of a VSCAN product, you can do this over a configuration file (vscan.ini). When there is no configuration file available, the wrapper API uses the first VSCAN product which will be found in the PC. You can also get extra debug information when you configure the debug option for the CAN channel in the configuration file. The debug output will be saved in a file called vs_can_api.log in the program directory.

Configuration File Example:

```
[CAN_1.dll]
Port = "COM5"           ; VSCAN device over a (virtual) COM port
debug = 0

[CAN_2.dll]
Port = "192.168.254.254:2001" ; mapping to NetCAN+ over IP and TCP port (raw mode)
debug = 1               ; debug output is switched on (vs_can_api.log)
```

Note: In this example CAN_1.dll and CAN_2.dll are just sample text. Replace the text in the brackets with the name of the DLL used by your application. So if in reality the names are XCAN_USB.dll and yCAN_PCI.dll, use those names for the section titles. The wrapper DLL finds the desired configuration by searching for a section, which is titled by the DLLs name. The set of wrapper DLLs usually provides a sample configuration file.

4.9 ZOC

ZOC (see Figure 31) is a powerful terminal program which has good logging functionality and will also let you make connections over the network (telnet client).

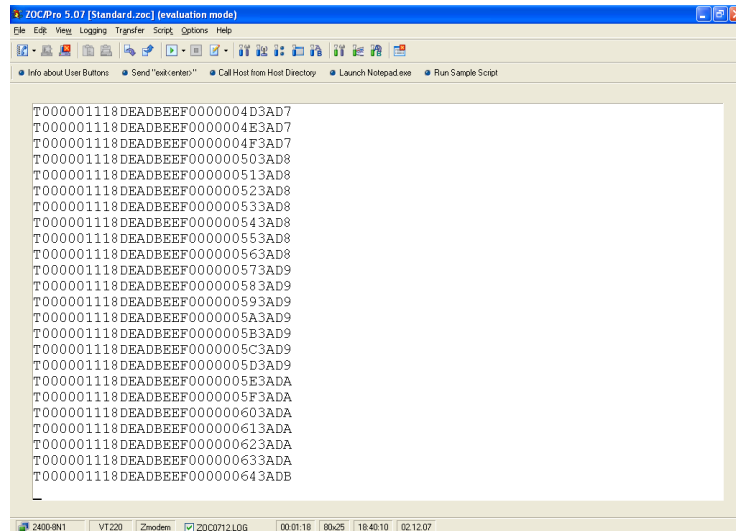


Figure 31: ZOC

4.10 putty

With **putty**³⁶ you can both talk to the network based devices NetCAN Plus (see Figure 32b) and via the serial interface to USB-CAN Plus and NetCAN Plus with the driver for virtual Com Ports (see Figure 32a).

It is important to enable hardware handshake for the USB-CAN Plus devices (see Figure 33a). Also make sure LF is added to every CR and local echo is on for more convenience (see Figure 33b).

³⁶<http://www.putty.org>

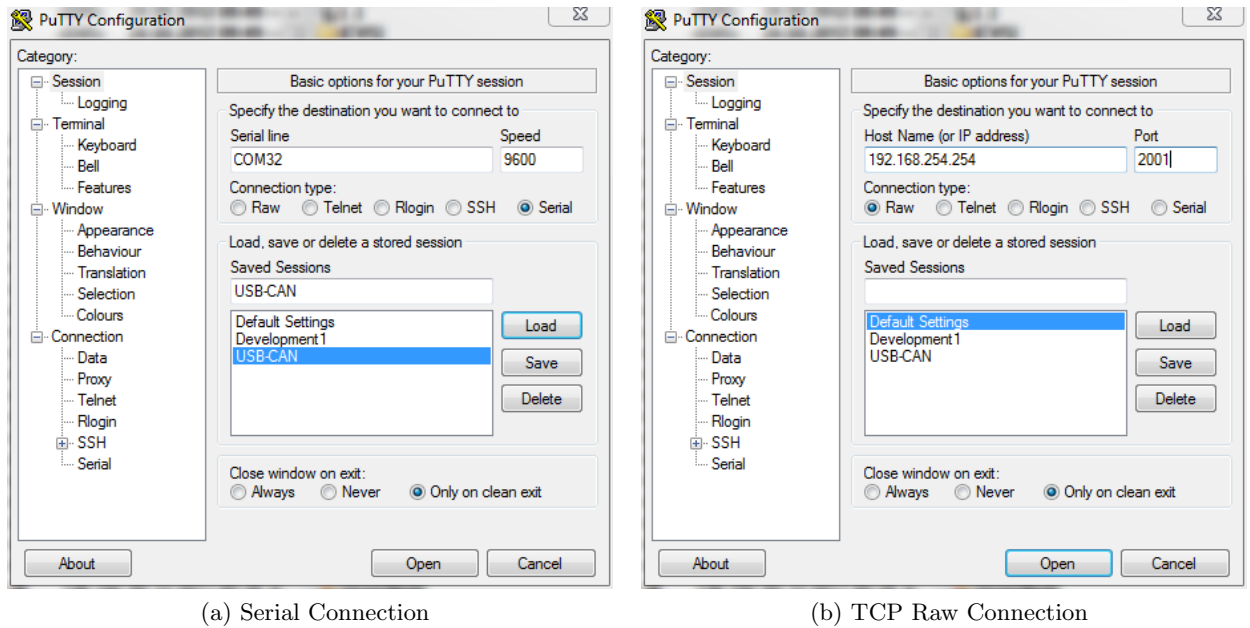


Figure 32: Putty

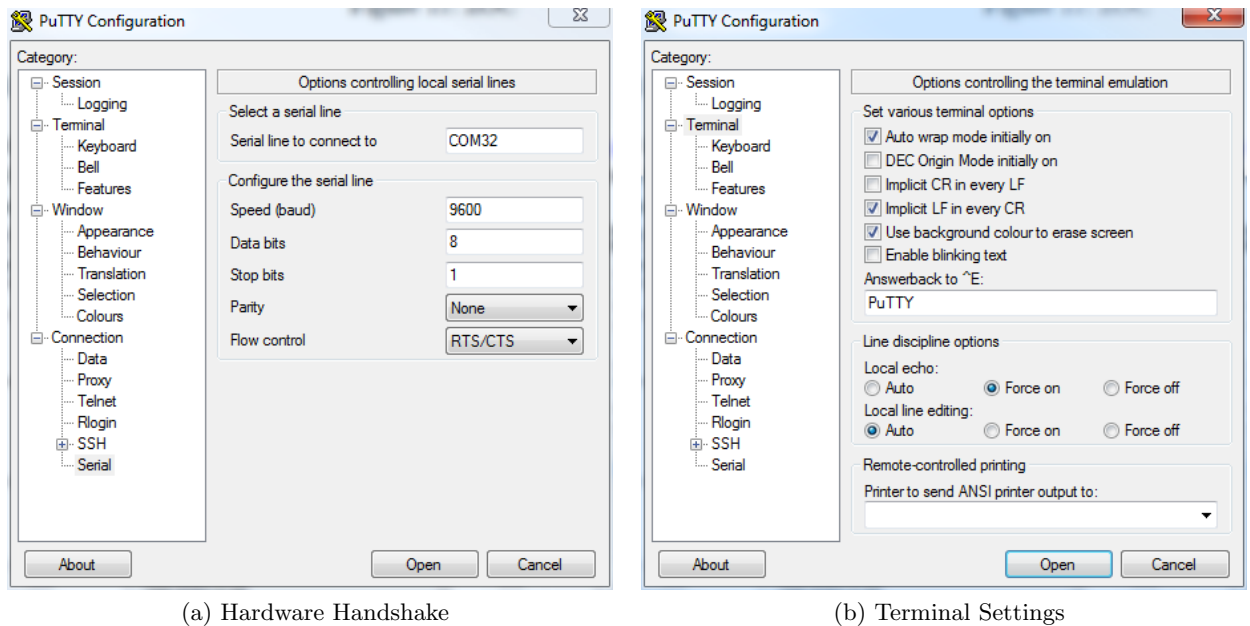
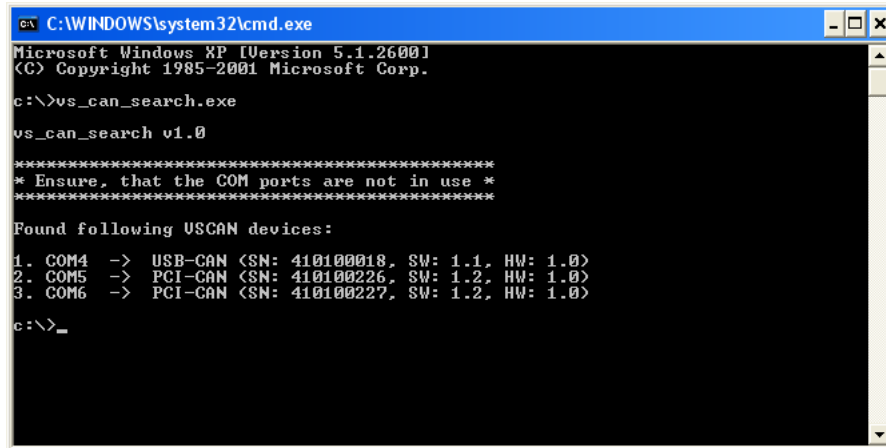


Figure 33: Putty: Additional Connection Settings

4.11 vs_can_search

This tool search for VSCAN devices on every COM port. You can also get extra debug information with the parameter „-d[1-3]” - eg. „vs_can_search.exe -d3”.

If you already know an assigned COM port or IP address, you can get the hardware information directly: „vs_can_search.exe COM4”.



```
ex C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

c:\>vs_can_search.exe
vs_can_search v1.0

*****
* Ensure, that the COM ports are not in use *
*****

Found following USCAN devices:
1. COM4 -> USB-CAN (SN: 410100018, SW: 1.1, HW: 1.0)
2. COM5 -> PCI-CAN (SN: 410100226, SW: 1.2, HW: 1.0)
3. COM6 -> PCI-CAN (SN: 410100227, SW: 1.2, HW: 1.0)

c:\>_
```

4.12 LabVIEW

LabVIEW (short for Laboratory Virtual Instrumentation Engineering Workbench) is a platform and development environment for a visual programming language from National Instruments. This section shows one possible way using `vs_can_api.dll` with LabVIEW. Specially prepared example (see Figure 34) can be found in our GitHub repository³⁷. It lets the user send and read CAN frames after opening the channel with appropriate bitrate. The LabVIEW example was compiled and tested on LabVIEW 8.6.1.

Before trying the LabVIEW example please make sure you can send/receive CAN frames using `vscandump` and `vscansend` or `Busmaster`. If you use `NetCAN Plus`, make sure it is either in the `Driver` or `TCP Raw Server` modes.

You can send CAN frames using the left panel. After configuring your CAN frame just press „Write On/Off” Button and frames will be sent each second. The incoming frames will be shown on the right panel one per second if any available.

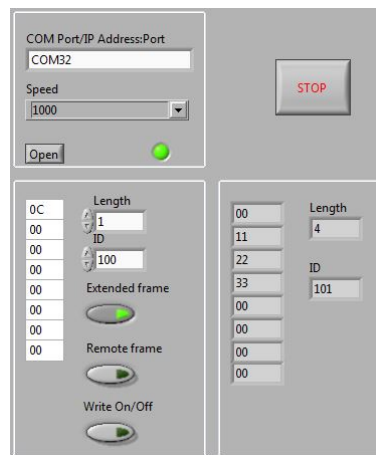


Figure 34: Example Application

4.12.1 Open CAN Channel

The `OpenChannel.vi` has the following input parameters:

- COM Port or IP address and port string, i.e. `COM32` or `192.168.1.100:2001` (as in Section 2.2.1)
- speed as a number between 1 and 8 (as in Section 3.2.3).

Output parameters:

- return error code of `VSCAN_Open()` or `VSCAN_Ioctl()` routines
- CAN channel handle.

³⁷https://github.com/visionsystemsgmbh/programming_examples/tree/master/CAN/LabVIEW

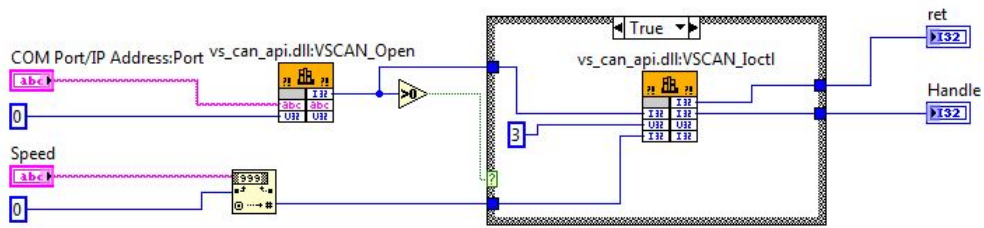


Figure 35: Open CAN Channel

4.12.2 Read CAN Frame

The CanRead.vi has the following input parameter:

- handle

Output parameters:

- return code value
- VSCAN_MSG structure
- number of read bytes

CanRead.vi is designed to read one CAN frame per call. To read more bytes at once, the buffer must be increased.

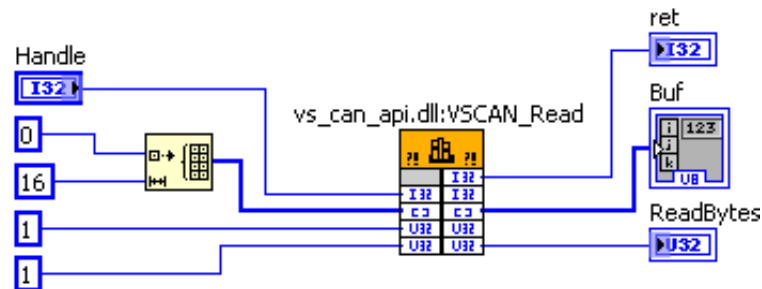


Figure 36: Read CAN Frame

4.12.3 Write CAN Frame

The CanWrite.vi has the following input parameters:

- handle
- parts of the VSCAN_MSG structure

Output parameters:

- return code value
- number of written bytes

The VSCAN_Write call is followed directly by VSCAN_Flush call, so the CAN frame will be sent immediately

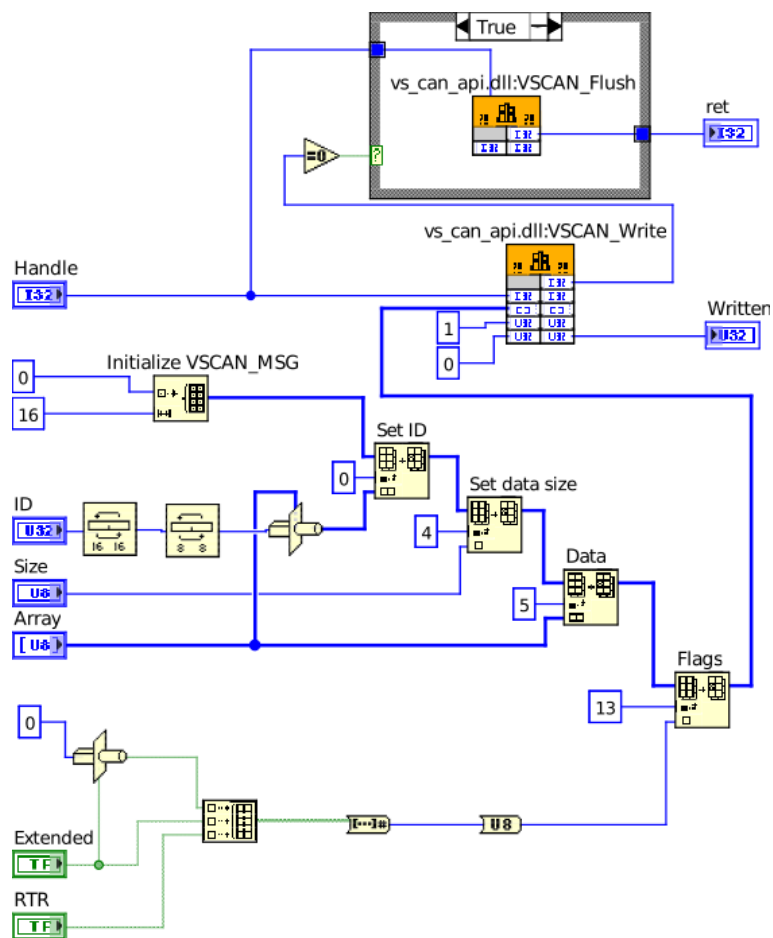


Figure 37: Write CAN Frame

5 Example Configurations

5.1 CAN over IP Router (CAN Bridge Mode)

The CAN Bridge Mode connects two CAN networks over IP and hence requires at least two NetCAN Plus devices: one acting as a server and another acting as a client as described in Paragraph 1.2.1. In the following example, we assume that the server device has IP address 192.168.1.97 and the client device 192.168.1.96 and that both devices work at the CAN bitrate of 20000b/s.

Server Configuration Open the web interface of your server device and configure its *Channel Configuration* to *CAN Bridge Server* as shown in Figure 9d. Change *CAN Speed* value if you do not use 20000b/s.

Client Configuration Open the web interface of your client device and configure its *Channel Configuration* to *CAN Bridge Client* as shown in Figure 9e. As you can see, the *Destination* field is set to your server device IP address and *TCP Port(Data)* values i.e. 192.168.1.97 and 2001. Change *CAN Speed* value if you do not use 20000b/s.

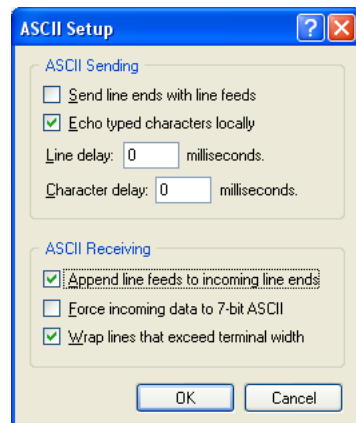
Now restart the server device and some seconds later, the client device to make sure that the server device is up before the client device tries to connect for the first time. If you enable logging (see Section 1.2.5 for details) on the server device, you will see messages stating that the client made a connection.

It is also recommended to set the *Max. Clients* value to at least 2, even if you intend to connect only one client. This way, if a client gets reset without properly closing the connection, it will nevertheless be able to connect again. Also, make sure to enable the *KeepAlive* option in the *Server Parameter* section to detect broken connections (see Section 1.2.3 for details).

6 Frequently Asked Questions

6.1 All output from the CAN adapter will be written in one line in HyperTerminal?

You must configure the correct settings and switch on "Append line feeds to incoming line ends":



6.2 I've updated the driver of my USB-CAN Plus, but the alias baudrate 9600 is not functioning anymore?

There is a VBScript in the driver package, which must be called for each installed virtual USB-CAN Plus COM port:

```
cscript regmodify.vbs COM<x>
```

After executing the script the USB-CAN Plus must be disconnected for about 5 seconds and then connected again in order to use the baudrate aliases. Then you can open the port with any standard baudrate - **except 115kbps!**

6.3 The Error LED is permanently on

This LED state indicates bus error. The most typical causes are:

- wrong wiring and missing termination (refer to Section [CAN Topology, Wiring and Termination](#))
- wrong CAN bitrate (refer to Section [Setup the Bus Timing \(Standard\)](#))

6.4 SocketCAN Troubleshooting

FTDI driver After inserting the USB-CAN Plus device, your `dmesg` should produce the following output:

```
ftdi_sio 1-1.1:1.0: FTDI USB Serial Device converter detected
usb 1-1.1: Detected FT-X
usb 1-1.1: FTDI USB Serial Device converter now attached to ttyUSB0
```

Most desktop Linux distributions provide `ftdi_sio` driver, but the situation can be different on an embedded Linux machine. So if you don't see such a message and no `/dev/ttyUSBx` device appeared, make sure `CONFIG_USB_SERIAL_FTDI_SIO` is activated for your kernel.

Also check, whether it is a USB-CAN Plus device. Grepping the output from `udevadm` must return the following hits:

```
# udevadm info -a -n /dev/ttyUSB0 | grep 'USB-CAN Plus'
  ATTRS{interface}=="VScom_USB-CAN_Plus"
  ATTRS{product}=="VScom_USB-CAN_Plus"
```

Another important check is, whether the port is already opened by another application. Grepping `lsof` output must return no hits:

```
lsof | grep ttyUSB0
```

You can use the `vscantester.py`³⁸ script to perform these checks and more.

slcan driver If you get the following message after starting `slcand`, it means the symbol `CONFIG_CAN_SLCAN` wasn't activated for your kernel and you'll have to recompile the kernel with this symbol enabled.

```
modprobe: FATAL: Module slcan not found in directory /lib/modules/*
```

slcan0 interface As soon as `slcan` driver was loaded, `slcan0` device should appear. So invoking `ip addr show slcan0` should produce the following output:

```
10: slcan0: <NOARP> mtu 16 qdisc noop state DOWN group default qlen 10
    link/can
```

If `slcan0` device is missing, make sure you've selected the proper serial device when invoking `slcand`.

³⁸https://github.com/visionsystemsgmbh/programming_examples/tree/master/CAN/python/dev-tester

CAN Error LED If, after trying to send/receive packets, the CAN Error LED is permanently on, you should first check the wiring as shown in Section [CAN Topology, Wiring and Termination](#) and if the wiring is correct, then check whether `-sx` parameter corresponds to the required CAN bitrate as shown in Section [Setup the Bus Timing \(Standard\)](#).

6.5 USB-CAN Plus Troubleshooting in Windows

If you encounter any issues with USB-CAN Plus like port cannot be opened, no frames sent or received, you can perform the following actions in order to find the issue or at least to gather as much info for the support request as possible. Please always provide output from the tools mentioned below and also include a photo/scheme of your cabling.

Powerup LED Blinking Sequence When you plug the USB-CAN Plus device into USB, both Error and Data LEDs will blink twice and remain off afterward. If you don't see this blinking sequence, either device or firmware is damaged.

Checking Driver Installation First of all make sure you've performed all steps described in Section [USB-CAN Plus Device](#). This is important for a correct function.

Perform CAN device search using `vs_can_search` utility described in Section [vs_can_search](#). You should see your device with the corresponding serial number.

Check CAN Wiring Make sure your CAN wiring corresponds to the wiring described in Section [CAN Topology, Wiring and Termination](#) and Error LED is not permanent on when sending/receiving CAN frames.

Perform Self-Diagnostic Tests Use `vs_can_test_simple.exe` to perform such a test either against another VScom device or with your current device itself as described in [vs_can_test_simple.exe](#).

Check CAN Filter Settings If you can send but not receive CAN frames, it is most likely, that Acceptance Code and Mask registers are set up wrong. These settings are stored in internal non-volatile memory so that they are persistent across power cycles. Try default settings i.e. Acceptance Code `0x00000000` and Acceptance Mask `0xFFFFFFFF`. `vscandump.exe` or `vs_can_test_simple.exe` will make these settings on every invocation.

Perform Basic CAN Transmission Tests If your CAN devices send frames autonomously, you can use `vscandump.exe` to receive the traffic. You can also inject CAN frames using `vscansend.exe`. Both tools are described in Section [vscandump and vscansend](#). Alternatively, you can use BUSMASTER software described in Section [Busmaster](#).

7 Licensing Information

NetCAN Plus devices use Open Source Software to fulfill part of their functions. The main licenses involved are the GNU General Public Licence version 2 (GPLv2) and the GNU Lesser General Public Licence version 2.1 (LGPLv2.1). You'll find the relevant text of those licenses below. In part there are similar public licenses involved.

Please see our Open Source related page for more information³⁹. You'll also find all related license texts and source code archives there.

7.1 GNU GENERAL PUBLIC LICENSE

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

³⁹<http://www.visionsystems.de/opensource>

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all

modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property

right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY

WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and an idea of what it does.>

Copyright (C) <yyyy> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

7.2 GNU LESSER GENERAL PUBLIC LICENSE

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages—typically libraries—of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

- a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
- b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
- c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
- d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
- e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application

of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF

THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and an idea of what it does.>

Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in
the library 'Frob' (a library for tweaking knobs) written
by James Random Hacker.

<signature of Ty Coon>, 1 April 1990

Ty Coon, President of Vice

That's all there is to it!