
CANopen®

API for C++/C#

Available as *open source* (GPL v.3.0) or *commercial* license



www.datalink.se

©Ulrik Hagström 2008-2010

Contents

1	Product introduction.....	4
2	Licensing conditions.....	5
2.1	GPL v.3.0 license.....	5
2.1.1	What is 'CAN dispatcher Library'?	6
2.2	Commercial license.....	6
2.2.1	Available value packages.....	7
3	Supported hardware	7
3.1	Available CAN interface plug-ins.	8
3.1.1	Zanthic Technologies CAN4USB FX.....	8
3.1.2	Lawicel CANUSB / CAN232.....	8
3.1.3	KVASER CANLIB (USB, PCI, PC104, PC104+, PCMICA,WLAN)	9
3.1.4	Movimento Castor (USB).....	9
3.1.5	EMS Dr. Thomas Wünsche (USB, Ethernet, PCI, PC104).....	10
3.1.6	PEAK System (PCAN-Light API)	10
3.1.7	Create your own plug-in?.....	11
4	Software layout structure for C++ applications.....	12
5	Software layout structure for C#/.NET applications	13

CANopen[®]

Is a registered Community Trademarks of CAN in Automation.

<http://www.can-cia.org>

This library has not been validated by CAN in Automation but
is widely used and works to our best knowledge!

1 Product introduction

The purpose of this library is that it should be easy to create 3rd party CANopen® compliant applications in a high level language such as C# or C++. The library is well suited for helping develop diagnostic or tools for maintenance or service or tools for node configuration. The driver supports;

- SDO expedited transfer (read and write 1-4 bytes in object dictionary of your CANopen node(s)).
- SDO segmented transfer (read and write typically strings in object dictionary of your CANopen node(s)).
- SDO block transfers (write only, use this for firmware upgrade).
- NMT Master Node control (start, stop, reset...)
- NMT Master error control (node-guard, heart-beat monitor)
- NMT Slave
- CAN raw mode (send and receive any CAN frames).
- EDS/DCF file support.

Download source code for free: http://www.canopen.nu/files/source_code_package.zip

2 Licensing conditions

CANopen API (C++/C#) is made available under the terms of the following dual license (like QT and OpenOffice):

2.1 GPL v.3.0 license

You may use CANopen API (C++/C#) in accordance with the terms of the GNU General Public License, version 3.0.

This license allows *for free use*, but also requires you to distribute your own code using CANopen API (C++/C#) under the terms of the GPL. See "gpl-3.0.txt" for complete information with additional permissions below since source code for "can_dispatcher.lib" is not included:

IMPORTANT: Additional permission under GNU GPL version 3 section 7:

If you modify this Program, or any covered work, by linking or combining it with 'CAN-dispatcher Library' (*can_dispatcher.lib*) (or a modified version of that library), containing parts covered by the terms of "CANopen API (C++/C#) Commercial License", the licensors of this Program grant you additional permission to convey the resulting work. {Corresponding Source for a non-source form of such a combination shall include the source code for the parts of 'CAN-dispatcher Library' used as well as that of the covered work.}

Read more about linking GPL-code to closed source here:

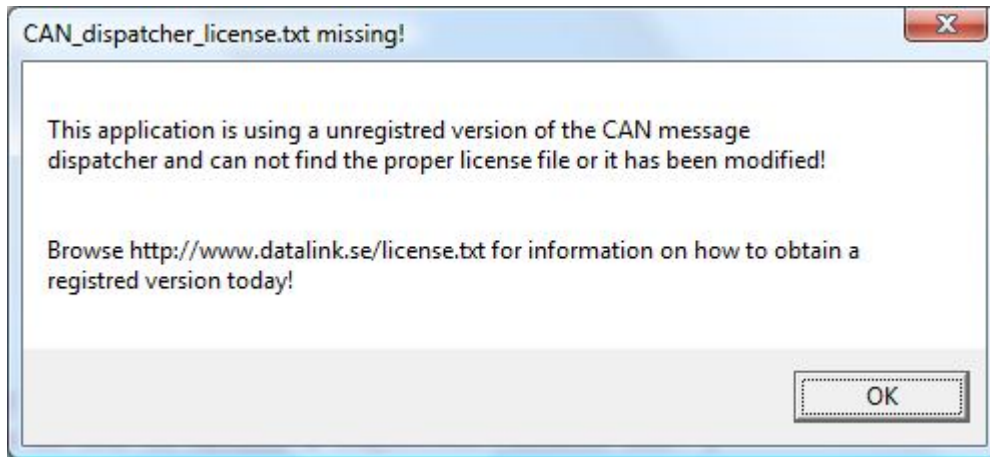
<http://www.gnu.org/licenses/gpl-faq.html#GPLIncompatibleLibs>

<http://www.gnu.org/licenses/gpl-faq.html#FSWithNFLibs>

<http://www.datalink.se/gpl-3.0.txt>

2.1.1 What is 'CAN dispatcher Library'?

The "CAN Dispatcher Library" (can_dispatcher.lib) is a vital library to make the CANopen driver package operate properly and it also REQUIRE that a valid license code reside in "CAN_dispatcher_license.txt" located in the working directory of the application or else the following pop-up will show:



Download a "CAN_dispatcher_license.txt" for free to be used in open source projects (FREE):

http://www.datalink.se/files/CAN_dispatcher_license.txt

The reason why we have this license file is that we promote open source – and we want to FORCE applications attach information that it is based on open source.

Anyone committed to follow GPL v.3 are welcome to send us information about their project on sales@datalink.se and request the source code for the CAN-dispatcher Library.

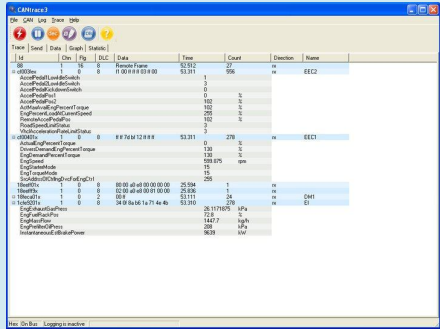
2.2 Commercial license

In exchange for payment you may use CANopen API (C++/C#) in a closed-source application, in accordance with the terms of the CANopen API (C++/C#) Commercial License. CANopen API (C++/C#) Commercial License *also includes priority support.*

Commercial license also includes source code for can_dispatcher.lib.

<http://www.datalink.se/license.txt>

2.2.1 Available value packages

Value package	Pricing and discounts	Price (Ex. VAT and shipping)
Commercial license only	1 pcs. commercial license.	200€
Lawicel commercial	1 pcs. commercial license + 5 pcs. Lawicel CANUSB.	599€
Zanthic commercial	1 pcs. commercial license + 5 pcs. Zanthic CAN-4-USB.	1349€
Kvaser commercial	1 pcs. commercial license + 5 pcs. Kvaser Leaf Light.	2399€
Developer commercial	 <p>1 pcs. commercial license + 2 pcs. Kvaser Leaf Light + 1 license for CAN trace (http://www.tke.fi/pdf/TKE_CANtrace_datasheet.pdf).</p>	2399€

These prices may change without prior notice – please ask for a quotation before ordering on sales@datalink.se.

3 Supported hardware

The structure of the CANopen API (C++/C#) driver is that all CAN-layer dependent calls are made via a DLL called canopenlib_hw.dll (see chapter 4 in this document) – there are no direct dependencies from the CANopen® network to the CAN interface API.

This means that you can choose any of the listed hardware's below and just use the specific DLL for your hardware. You can also develop your own plug-in DLL; source code can be downloaded, free of charge for the listed hardware below as example how to implement your own plug-in for your preferred hardware. It has been kept in mind that this CANopen library should be possible to port to cheap CAN interfaces or even migrate to a embedded environment - without a fancy CAN API – therefore this library CAN-interface class holds a lot of intelligence (CAN message dispatcher, thread safe calls etc).

3.1 Available CAN interface plug-ins.

Today we can provide you with, free of charge, plug-in DLLs for the listed CAN interfaces below.

3.1.1 Zanthic Technologies CAN4USB FX

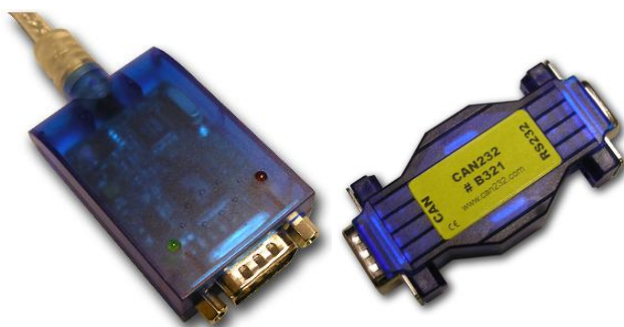
A plug-in for the *high performance* CAN4USBFX (<http://www.zanthic.com/can4usbfx.htm>) from Zanthic Technologies Inc is available which means that it will work with Zanthic Technologies USB adapters.



3.1.2 Lawicel CANUSB / CAN232

A plug-in for the Lawicel ASCII-API is available which means that it will work with both *low price* CANUSB and CAN232.

- CANUSB – http://www.canusb.com/documents/canusb_manual.pdf
- CAN232 – http://www.can232.com/can232_v3.pdf



3.1.3 KVASER CANLIB (USB, PCI, PC104, PC104+, PCMICA,WLAN)

All KVASER's hardware (www.kvaser.com) is using the same interface API – this means that you can choose any of the KVASER CAN-products as they are all supporting the CANLIB API.

Recommended Kvaser hardware is (http://www.tke.fi/pdf/leaf_light.pdf).



3.1.4 Movimento Castor (USB)

Movimento Castor USB is low cost USB to CAN adapter. Contact distributors@datalink.se to reach all preferred retailing contacts and best price.

<http://www.movimentogroup.com/pdf/Movimento%20Castor%20CAN.pdf>



3.1.5 EMS Dr. Thomas Wünsche (USB, Ethernet, PCI, PC104)

All EMS hardware (<http://www.ems-wuensche.com>) is using the same interface API – this means that you can choose any of the EMS CAN-products to use with the EMS plug-in DLL.



Important notice: EMS windows drivers are not delivered with a free of charge API (SDK) – this means that you will have to purchase a proper developer license for your EMS device/devices to make the `canopen_hw.dll` working properly. **Contact EMS for further information about developer licenses.**

3.1.6 PEAK System (PCAN-Light API)

A plug-in for the PEAK System (www.peak-system.com) is available which means that it will work with all PEAK System hardware that is supporting the PCAN-Light API.



3.1.7 Create your own plug-in?

A number of 'C'-functions needs to be exported from canopenlib_hw.dll in order to make it work with higher layers of the CANopen® layer; they are described in the following list. Support for more than one handle per CAN channel is not required (a dispatcher in canopenlib.dll are dispatching one message to all objects that are listening to this CAN-port).

```

#define CAN_MSG_RTR           0x0001    // Message is a remote request
#define CAN_MSG_EXT         0x0002    // Message has a extended ID

canOpenStatus    __stdcall canPortLibraryInit(void);

canOpenStatus    __stdcall canPortOpen( int port, canPortHandle *handle );
canOpenStatus    __stdcall canPortClose( canPortHandle handle );

canOpenStatus    __stdcall canPortBitrateSet( canPortHandle handle, int bitrate );

canOpenStatus    __stdcall canPortEcho( canPortHandle handle, bool enabled );

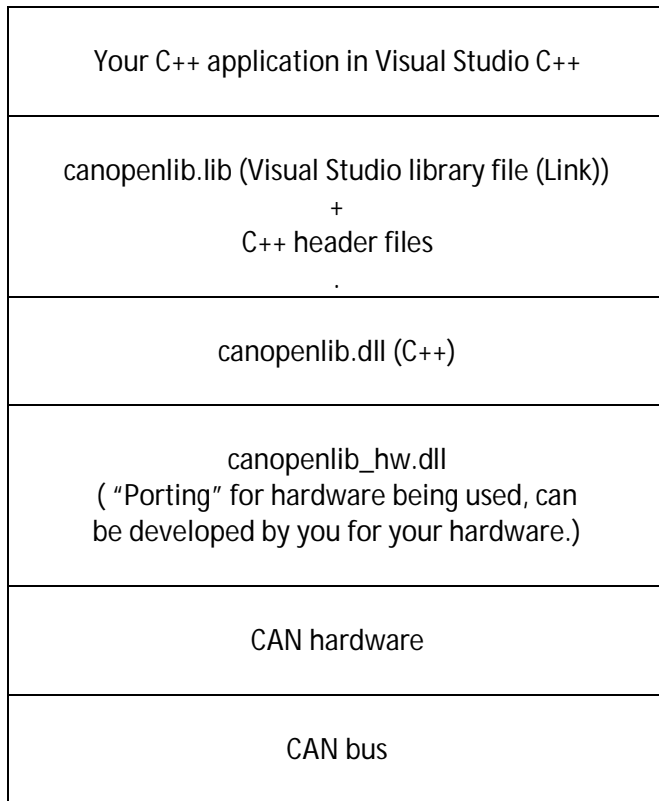
canOpenStatus    __stdcall canPortGoBusOn( canPortHandle handle );
canOpenStatus    __stdcall canPortGoBusOff( canPortHandle handle );

canOpenStatus    __stdcall canPortWrite(canPortHandle handle,
                                        long id,
                                        void *msg,
                                        unsigned int dlc,
                                        unsigned int flags);

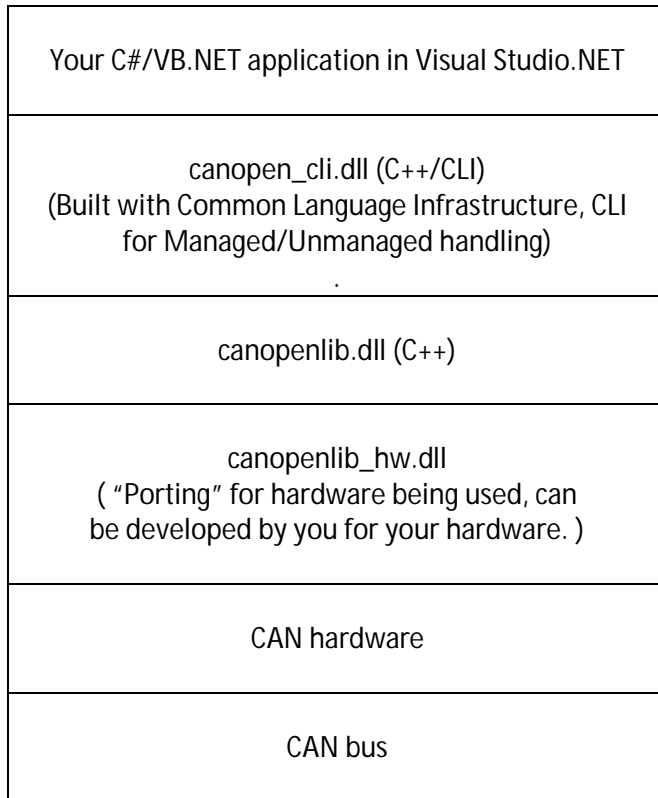
canOpenStatus    __stdcall canPortRead(canPortHandle handle,
                                        long *id,
                                        void *msg,
                                        unsigned int *dlc,
                                        unsigned int *flags);

```

4 Software layout structure for C++ applications



5 Software layout structure for C#/.NET applications



The syntax for the C# API version of this driver is very similar to the C++ and therefore left out in the examples. The appendix of this document includes a C# example program that should hopefully be sufficient.