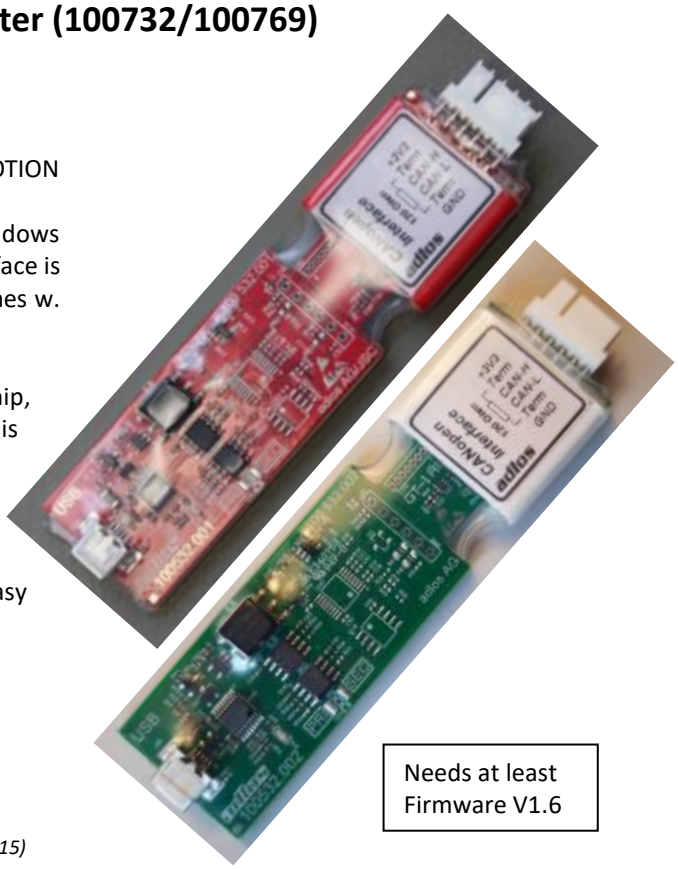
AN300044: Using USB-CANopen Converter (100732/100769)

Introduction



This document shall enable Users to work with KannMOTION USB-CAN-Interface.
100732 CAN Interface is a simple Hardware to Enable Windows Application to talk with CAN/CANopen devices. CAN Interface is electically isolated from PC-USB-GND and Earth, so it comes w. an isolation up to 200V potential difference.

This Interface board is based on the very common FTDI-Chip, combined w. an STM32-CPU. Users' application talks w. this Interface over a virtual COM-Port connection. The used protocol is based on KannMOTION Gen2 serial protocol description.

For this protocol, Adlos offers a USER-API, for quick and easy work with our protocol.

Connector:
- Molex 0022012067 KK254-6Pol
(Mouser 538-22-01-2067)
- Molex 0008510108 KK TERM
(Mouser 538-08-51-0108-LP or 538-79758-0015)

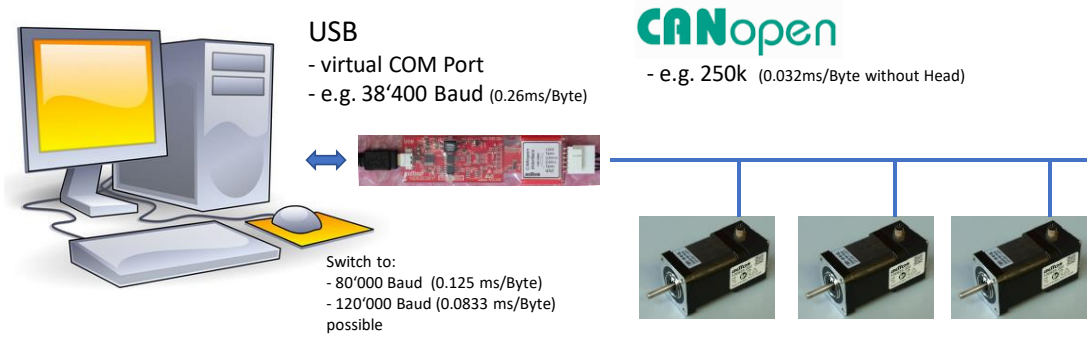
Needs at least Firmware V1.6

Base Documentation of protocol
https://www.kannmotion.com/wp-content/uploads/2022/06/100570_008_KannMotionProtokoll.pdf
https://kannmotion.adlos.com/download/comwatchtool/updatedata/190078_USB_CANOpen.xml

Converter Details

Basic principle /Data rates info

Adlos USC-CAN converter acts as an bridge between an Virtual COMport and CAN-bus, see Illustartor.

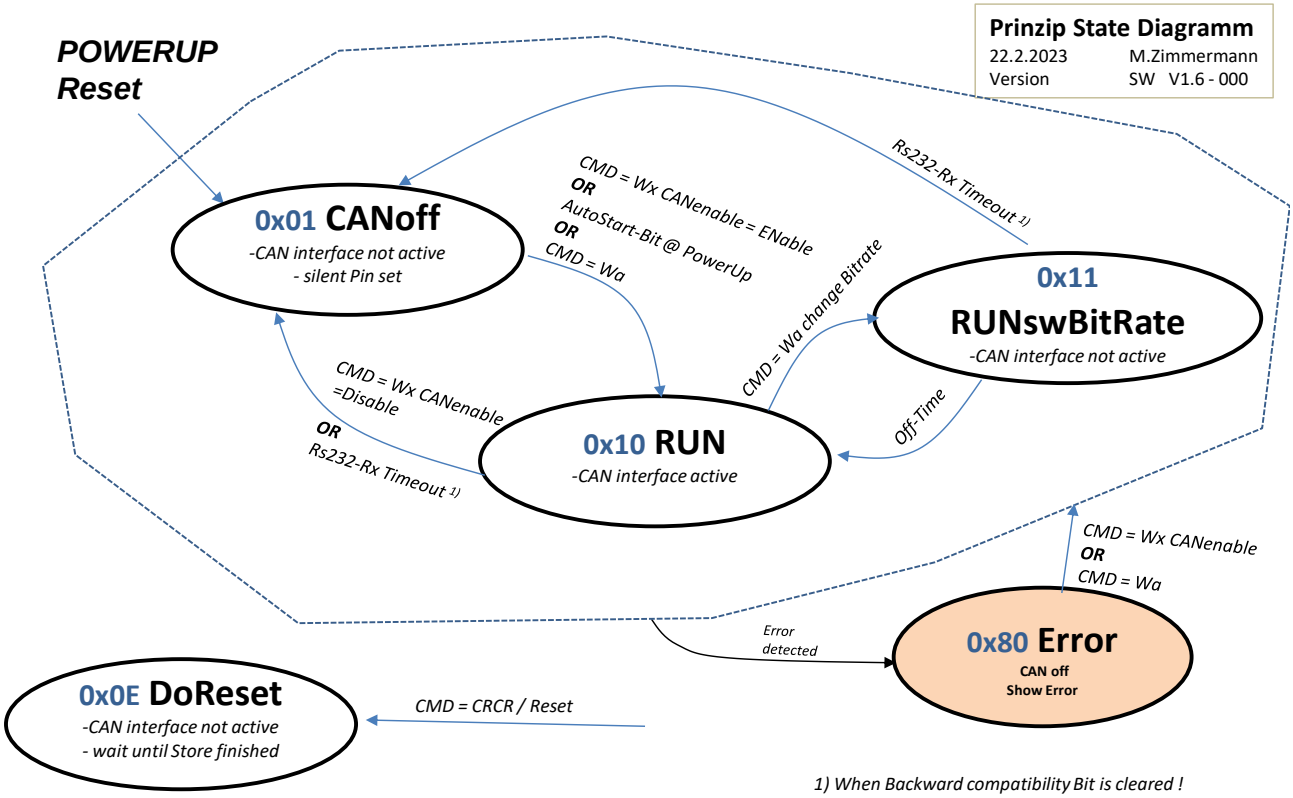


In CAN applications w. a lot of bus traffic, it is a must, to use integrated Filter-Settings to relieve CANopen converter from being overloaded! Depending on CAN- bus rate, and virtual-Comport setting, the converter does not have the possibility to forward all can- bus packets to COMport, due to a bit rate gap, what will result in data loss.

Accessories

Part number	Description	
100 767	CIA Adapter cable Molex to D-Sub-9	
100 742	Converter / motor Demo connection cable Molex to M8-motor	

Main States



LED-Display interpretation

The converter has 3 LEDs, whereby the front green LED is directly controlled by the FTDI, i.e. this LED basically indicates communication via USB.
Interpretation table of the other 2 LEDs:

LED red	LED green	Meaning
	○	Device not running
	✸	Device is running, Baud rate NOT Standard (> 38400)
	✹	Device is running, Baud rate = Standard (38400)
	●	Device not running
○	✸✸	Device is ok, CAN is running-connected
✸		Device is in CAN off State (CAN is not connected, initialized)
●		Device is in Error State

○ LED off
✸ LED flashing 1:1 (100ms ON / 100ms OFF)
✹ LED flashing 1:4 (100ms ON / 500ms OFF)
● LED ON

Command set

Commands overview

For Command/protocol details, please consider our documents 100570 /190078.xml (see link on Base doc).

Keywords

CMD	*Description*	*double CMD*	*Additional data*	*Answer*	*time*
CR<CR>	**Reset / Restart**	yes	-		ext
CS<CR>	**Safe Adjustment**	yes	-		
R0<CR>	**Read Registered Nodes**	no	-	Echo + 128Bit +CK	norm
R1<CR>	**Read Online Data**	no	-	see XML	norm
R2<CR>	**Read App State Data**	no	-	see XML	norm
Wa<CR>	**Write Adjustment**	no	<BitRate> <Filter><Soll_Node)	Echo + Ack/Nack +CK	norm
RA<CR>	**Read Adjustment**	no	-	Echo + Configuration +CK	norm
RC<CR>	**Read active CAN settings**	no	-	Echo + ActiveCAN +CK	norm
RN<CR>	**Read NMT state of Node**	no	Node	Echo +NMT +CK	norm
Ws<CR>	**Write SDO-Object / CANopen**	no	see Detail Descr.	Echo + Nack/Data +CK	norm
Wc<CR>	**Write CAN**	no	<mode> <Dir> <timeout>	Echo + Ack/Nack +CK	norm
We<CR>	**Write EEPROM**	no	<adr><Cnt><data32>	Echo + Ack/Nack +CK	norm
RE<CR>	**Read EEPROM**	no	<adr><Cnt>	Echo + data +CK	norm
Wf<CR>	**Write Flash via CAN Bootloader**	no	see Detail Descr.		norm
RF<CR>	**Read Flash via CAN Bootloader**	no	see Detail Descr.		norm
Wx<CR>	**Write Generic Command**	no	<CMD><data32>	see Detail Descr.	norm
RB<CR>	**Read CANopen SDO-Block**	no	see Detail Descr.	see Detail Descr.	ext
RS<CR>	**Read CANopen SDO-norm transfer**	no	see Detail Descr.	see Detail Descr.	ext

CK:byte *0..255* Checksum of full Command or Answer Frame (Sum, followed by complement of 2)
ACK:byte *0x06*
NAK:byte *0x15*

Command details

<R0> CMD, registered Nodes

The R0 CMD provides the node addresses received via heartbeat over time. For details see 190078_USB_CANOpen.xml.

Important: If you want to detect e.g. that devices have been removed again, you have to send the CMD Clear Registered Nodes <Wx>, then wait a few seconds (until heartbeats have been collected again) and read out again with R0.

<Wc> CMD, Write CAN (all type of massages can be send)

Standard command to set all CAN registers and parameters. Good CAN knowledge required.

```
Send:       Wc<CR> <MsgId> <Data> <Size> <CK>
Answer:     Wc<CR> Ack/Nack
<MsgId> : CAN Message ID            32-Bit              Liddle endian, MSB set means ExtID
<Data>  : CAN Data bytes (8-Byte)   8-data byte
<Size>  : Data-Size                 1-Byte  (0..8)      New: high nibble might be coded !
<CK>    : Checksum of full answer   data byte [0..255]  e.g. 0x36
// .. Example: Wc#0D#XL2[0x603]#2B#XL2[0x2011]#04#XL4[1000]#06#CK    // MsgId SDO-Node:3/ Write Req.4Byte /
object 0x2011 / Subindex=4 / Data=  100 rpm = 1000 // 8-Datenbyte

Coding of Size 0xNS (e.g. for not acknowledged messages)
  S:  previous function, 0..8 effective size
  N:  defined request timeout (     0: as before fix = 50ms, for SDO and bootloader.
                              1..15: (N-1)*8ms i.e. 0,8,16,24...120ms )
```

<Ws> CMD, Write CANopen SDO (Service data object)

Command for CANopen SDO communication. Read and write SDO.

Send: **Ws<CR> <Node> <ObjIndex> <SubIdx> <Data> <CNTRL> <Size> <CK>**
Answer: **Ws<CR> CANopen SDO answer <CK>**

<Node>	: CANopen Node Adress	7-Bit	1..127
<ObjIndex>	: CANopen SDO Object-Index	16-Bit Liddle Endian	
<SubIdx>	: CANopen SDO SubIndex	8-Bit	
<Data>	: CANopen SDO Databytes	4-Byte	
<CNTRL>	: ControlByte	1-Byte	0x40:RD / 0x20:WR Request
<Size>	: Data-Size	1-Byte (0..4)	
<CK>	: Checksum of full answer	data byte [0..255]	e.g. 0x36

```
// .. Example: Ws#0D#03#XL2[0x2011]#04#XL4[1000]#20#02#CK          // SDO-Node:3/ object 0x2011 / Subindex=4
/ Data=  100 rpm = 1000 / Write Req / Size 2-Byte
```

<RN> CMD, Read CANopen NMT State of Node N

Command to query NMT state of a specific node. Background: the USB-CAN converter stores internally the last received Heartbeat message including time stamp. This information can be queried with this command.

Send: **RN<CR> <Node> <CK>**
Answer: **RN<CR> <TimeSinceLastHeartBeat> <NMT_State> <CK>**

<Node>	: CANopen Node Adress	7-Bit	1..127
<CK>	: Checksum of full answer	data byte [0..255]	e.g. 0x36

```
// Example: RN#0D#XL1[32]#CK   ( query NMT State of Node=32 )
```

```
TX: 0x080A74 10:44:35
  R  N        3
 52 4E 0D 20 33
0x080A74
  R  N     W              ö
 52 4E 0D 57 01 00 00 05 F6
```

<Ev> Event messages

The USB-CAN converter can also send an unsolicited message to the COMWatch (without polling). This happens e.g. if a CANopen node puts an event PDOx on the CANopen bus. If this node is in the filter area, this message is put on the RS232 and sent via <Ev> tag.

Send: **-**
Answer: **Ev<CR> CANopen full Message frame <CK>**

Examples

Kopf(Ev<CR>) MSG-ID Data-8 DLC <CK>

```
0x45;0x76;0x0D;0x07;0x01;0xFC;0x93;0x31;0x39;0x30;0x31;0x30;0x39;0x10;0x02;0x08;0x53;
Bootloader ID  :       0x93FC0107     '190109' V1.0-002
The MSB shows that it is an extended frame
```

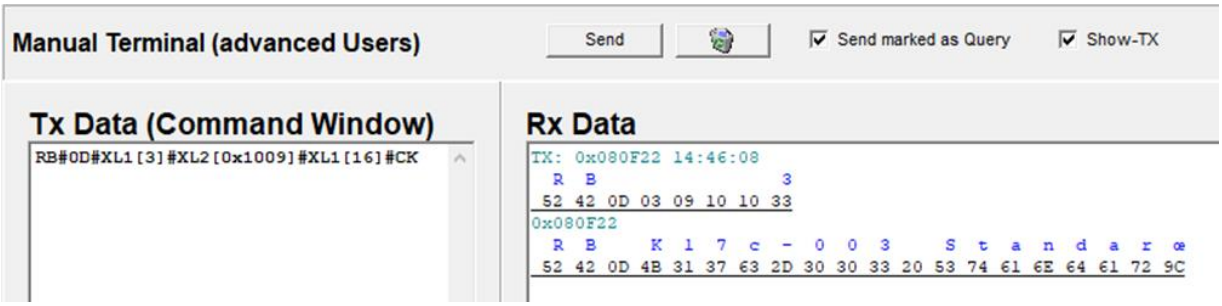
```
0x45;0x76;0x0D;0x03;0x07;0x00;0x00;0x05;0x00;0x00;0x00;0x00;0x00;0x00;0x00;0x01;0x28;
Heartbeat Message …    :       0x703   Data=5
```

<RB> CMD, Read CANopen SDO Block

Command to read SDO which supports block transfer (e.g. Visible string).
This command can return more than 8 data bytes, because our interface compiles the responses and returns them bundled.

```
Send:      RB<CR> <Node> <ObjIndex> <AnswSize> <CK>
Answer:    RB<CR> <Size> <Data> <CK>
<Node>       : CANopen Node Adress        7-Bit                        1..127
<ObjIndex>   : CANopen SDO Object-Index   16-Bit  Liddle Endian
<SubIdx>     : CANopen SDO SubIndex       8-Bit
<AnswSize>   : Expected response size     1-Byte
<CK>  : Checksum of full answer     data byte [0..255]           e.g. 0x36
```

```
// Example: RB#0D#XL1[32]#XL2[0x1008]#XL1[28]#CK   ( Node=32 read Device Name (String) 28-Byte )
```



Communication on CAN-Bus

2635.2486	603h	Rx	Daten	8	40 08 10 00 00 00 00 00
2635.2491	583h	Rx	Daten	8	41 08 10 00 16 00 00 00
2635.2496	603h	Rx	Daten	8	60 00 00 00 00 00 00 00
2635.2501	583h	Rx	Daten	8	00 4B 61 6E 6E 4D 4F 54
2635.2506	603h	Rx	Daten	8	70 00 00 00 00 00 00 00
2635.2511	583h	Rx	Daten	8	10 49 4F 4E 20 4B 31 37
2635.2516	603h	Rx	Daten	8	60 00 00 00 00 00 00 00
2635.2521	583h	Rx	Daten	8	00 63 20 44 72 69 76 65
2635.2526	603h	Rx	Daten	8	70 00 00 00 00 00 00 00
2635.2531	583h	Rx	Daten	8	1D 00 00 00 00 00 00 00
2635.4702	703h	Rx	Daten	1	7F

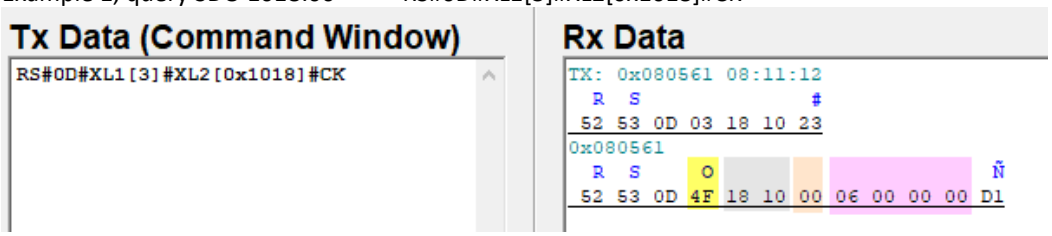
<RS> CMD, Read CANopen SDO Block/ Normal transfer without sizes indication

Command to read SDO which supports block transfer (e.g. Visible string).
Contrary to the <RB> -CMD, the expected response size does not have to be specified here, or does not have to be known, and if the addressed node/SDO does not allow a normal transfer, the communication is terminated directly and the received value is returned, contrary to the RB command.
<RS> is therefore a good choice instead of <Ws> if, for example, the sub-index=0 is to be read for an object.

Send: **RS<CR> <Node> <ObjIndex> <CK>**
Answer: **RS<CR><CMD><ObjIndex><Size/Data32><Sub><CK> Rs<CR><Data><CK>**

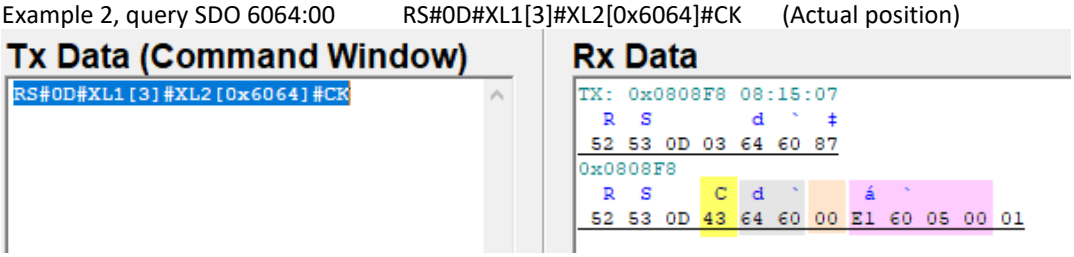
```
<Node>  : CANopen Node Adress                 7-Bit   1..127
<CMD>   : Info whether size or data           8-Bit  [0x41] = standard, 2nd part follows
                                                     [0x43,0x4B..0x80]  =  SDO-CMD
          if CMD <> 0x41 then there will be no 2.part!

<ObjIndex>     : CANopen SDO Object-Index           16-Bit  Liddle Endian
<Size/Data32>  : Size of Data                       32-Bit  Liddle Endian
<Sub >         : Sub-Index (always 0) used to be in Line with <Ws> CMD
<CK>           : Checksum of full answer            data byte [0..255]          e.g. 0x36
<Data>         : 1..Size, data bytes
```

Example 1, query SDO 1018:00 RS#0D#XL1[3]#XL2[0x1018]#CK



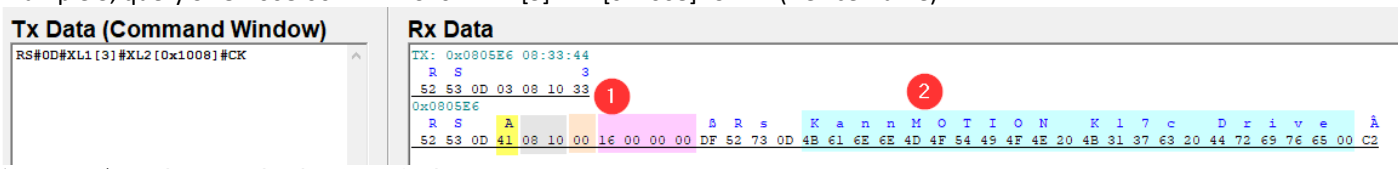

```
RX 1680156672.082 STD          703   1   7F                                .
RX 1680156672.702 STD          603   8   40 18 10 00 00 00 00 00 @.......
RX 1680156672.702 STD          583   8   4F 18 10 00 06 00 00 00 O.......
RX 1680156673.750 STD          702   1   7F                                .
```

Example 2, query SDO 6064:00 RS#0D#XL1[3]#XL2[0x6064]#CK (Actual position)




```
RX 1680156906.837 STD          702   1   7F                                .
RX 1680156907.095 STD          603   8   40 64 60 00 00 00 00 00 @d`.....
RX 1680156907.096 STD          583   8   43 64 60 00 E1 60 05 00 Cd`..`..
RX 1680156907.172 STD          703   1   7F                                .
```

Example 3, query SDO 1008:00 RS#0D#XL1[3]#XL2[0x1008]#CK (Device Name)



1) Contains the standard part of the response
2) Contains the effective user data of the normal transfer communication.

```
RX 1680157235.162 STD   702  1  7F                                      .
RX 1680157235.426 STD   603  8  40 08 10 00 00 00 00 00 @.......
RX 1680157235.427 STD   583  8  41 08 10 00 16 00 00 00 A.......
RX 1680157235.427 STD   603  8  60 00 00 00 00 00 00 00 `.......
RX 1680157235.428 STD   583  8  00 4B 61 6E 6E 4D 4F 54 .KannMOT
RX 1680157235.428 STD   603  8  70 00 00 00 00 00 00 00 p.......
RX 1680157235.429 STD   583  8  10 49 4F 4E 20 4B 31 37 .ION K17
RX 1680157235.429 STD   603  8  60 00 00 00 00 00 00 00 `.......
RX 1680157235.430 STD   583  8  00 63 20 44 72 69 76 65 .c.Drive
RX 1680157235.430 STD   603  8  70 00 00 00 00 00 00 00 p.......
RX 1680157235.431 STD   583  8  1D 00 00 00 00 00 00 00 ........
RX 1680157235.830 STD   703  1  7F                                      .
RX 1680157237.669 STD   702  1  7F                                      .
```

<Wx> CMD, Write Generic Command

Execution of various command.

```
Send:      Wx<CR> <CMD><Data32><CK>
Answer:    Wx<CR> Ack/Nack <CK>
<CMD>      : command                  8-Bit
<Data32>   : Daten                    32-Bit
<CK>       : Checksum of full answer  data byte [0..255]          e.g. 0x36
```

CMD	Description	Daten32	
0x00	Heartbeat- and Dropped-Messages	0xFF 0..127 0x100nn 0x20000	- clear all registered Heartbeats - clear Heartbeats of dedicated Node - transfer Heartbeat message of nn **OR** *00=none / FF=all* - clear Dropped Packets counter
0x01	Switch RS232 Baud rate *only accepted in States:* *CANoff **OR** Error!*	38400 80000, 100000, 115200, 120000, 250000, 500000 800000	38400 (Standard) Baud rate
0x02	Activate Filter	0..0x3FFF	Bit0->Filter0 .. Bit13-Filter13
0x03	Deactivate Filter	0..0x3FFF	Bit0->Filter0 .. Bit13-Filter13
0x04	CAN Module EIN/AUS	0,1	0 = off // 1 = on
0x05	CAN Set Sample Point	0 1 2 3 4	Auto (depending on Hardware) 68.75% 75.00% 81.25% 87.5%
0x06	Device Control Set	0..255	Bit0 = Backward compatibility Bit1 = Termination ON (see also cmd 7) Bit2 = TxCMD in Between for Trace *for permanent use, Set CMD save (CSCS)*
0x07	CAN Set Termination	0,1	0 = off // 1 = on *(only at HW 100536.002)*
0x10 0x11	Set Filter 10 - Mask Set Filter 10 - ID	Filter-32Bit ID-32Bit	See filter description! This command can set a full 32-Bit can message filter
0x12 0x13	Set Filter 11 - Mask Set Filter 11 - ID	Filter-32Bit ID-32Bit	See filter description! This command can set a full 32-Bit can message filter

Filter notes CMD 0x10..13:

- Filter 10&11: complete Filter incl. IDE/RTR Bits
- Set filter mask first, then filter ID
- After setting filter ID, filter is automatically activated.

<Wa> Write CAN Adjustment & Filter

The USB-CAN converter can/should be relieved via hardware filter when used in larger CAN networks, because CAN allows much higher transfer rates than our Standard COMWatch interface (38'400).

CAN	z. B.	250 k	-> 25B+8x8B = 89 Bit/Frame	-> 2800 Frames/s
COMWatch		38.4 k	-> Event Frame 12-Byte / 120 Bit	-> 320 Frames/s

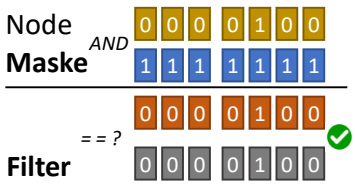
I.e. if a CANopen network is fully used, i.e. large data volume by many participants, our converter is not able to forward all frames from CAN to RS232.!!

Therefore it is strongly recommended to limit the application scope of the converter to the desired node addresses! I.e. communication on the CAN bus outside this scope is ignored and therefore not registered on the USER-PC...
Another possibility is to increase also the Bitrate of Comwatch side, see Wx command Switch RS232 Baud rate!
If a device cannot be addressed or does not respond, the set pre-filtering should be checked.

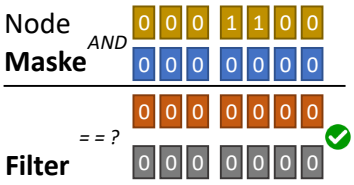
```
Send:      Wa<CR> <Bitrate> <Maske> <Filter> <CK>
Answer:    Wa<CR> Ack/Nack
<Bitrate>    : Bitrate                    8-Bit                        0=125k/       1=250k/
2=500k
<Maske>      : Node-Maske                 8-Bit                        0=Aus
<Filter>     : Node-Filter                8-Bit                        0=Aus
<CK>  : Checksum of full answer     data byte [0..255]          e.g. 0x36
```

Explanation:
If the mask is set <> 0, the additional filtering is activated. The mask marks the bits in the node address, which are to be checked additionally or which must be checked with the filter for equality. The message note is calculated bit by bit with the masked bits by means of a logical AND operator and the result is checked afterwards with the 'Filter' for agreement. If there is a match, the message is processed.

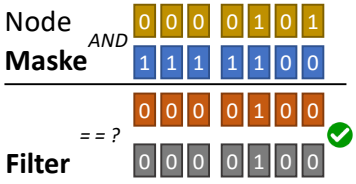
Nur Node = 4 registrieren / zulassen



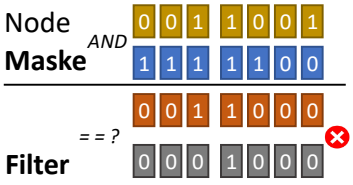
alle zulassen



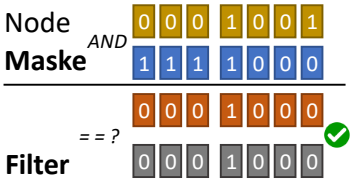
Node: 4-7 registrieren zulassen



Node: 8-11 registrieren zulassen



Node: 8-15 registrieren zulassen



1.1.1.1 Examples

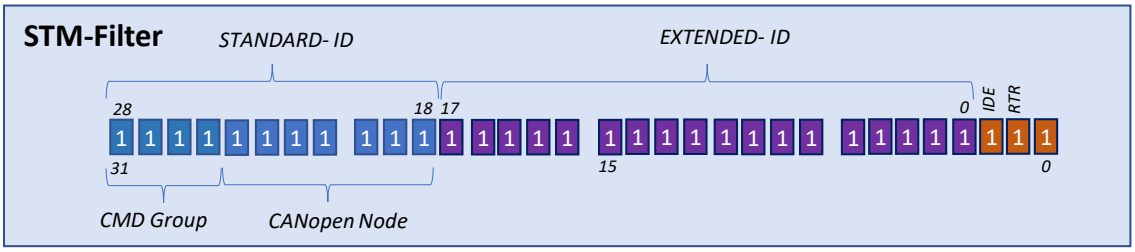
Filter = OFF: Wa<CR> #00#00#CK	All node events and heartbeats are registered
Maske= 0x38 / Filter = 0x10	Nodes: 0x10 .. 0x17 are registered

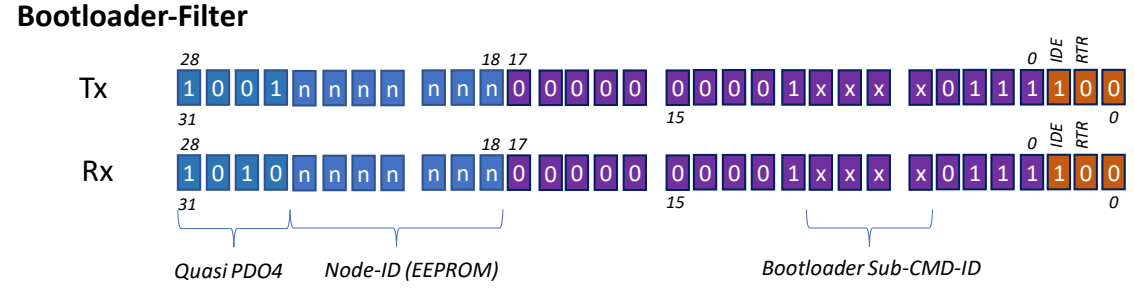
1.1.2 Filter-List

Our controller can accommodate up to 14 different filter criteria, most of the filters are assigned automatically and can be configured e.g. in the node address area with the <Wa> command. In addition, however, individual filters can be subsequently activated/deactivated with the CMD <Wx>.
It is also possible to create your own full filter with the <Wx> command.

Filter	Beschreibung			
0	0x180 - CANopen PDO1	Std -ID	combined with Wa-mask/Node	
1	0x280 - CANopen PDO2	Std -ID	combined with Wa-mask/Node	
2	0x380 - CANopen PDO3	Std -ID	combined with Wa-mask/Node	
3	0x480 - CANopen PDO4	Std -ID	combined with Wa-mask/Node	
4	0x580 - CANopen SDO	Std -ID	combined with Wa-mask/Node	
5	0x080 - CANopen Emergency	Std & Ext-ID	combined with Wa-mask/Node	
6	0x700 - CANopen NMT	Std & Ext-ID	combined with Wa-mask	Heartbeat Collector
7	0x12000107 – Bootloader 0-7	Ext-ID	combined with Node	
8	0x12000107 – Bootloader 8-15	Ext-ID	combined with Node	
9	Reserve			
10	User configurable		freely adjustable with Wx command	
11	User configurable		freely adjustable with Wx command	
12	Reserve			
13	All	Std & Ext-ID		Enabled when **Maske=0 & Filter=0** **!! Attention data traffic !!!**

1.1.3 Filter Representation/View



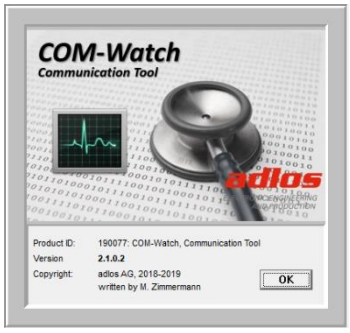


Appendix

Tools

adlos offers for its customers some Helping and Design-In Tools.

ComWatch Communication Tool (190077), for Life values



ComWatch is a helping tool for engineers and technicians to explore device specific parameters, read out tracking data and settings and doing firmware updates.
The software is as it is, and in principle for free for adlos customers, the software is not made for a broad range of standard users, it's made in principle for technical engineers which are used in working w. windows based software and have some understanding of technical things.

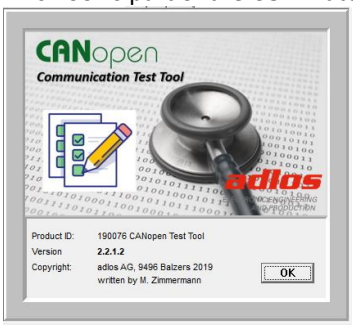
https://kannmotion.adlos.com/download/comwatchtool/ComWatchSetup.zip

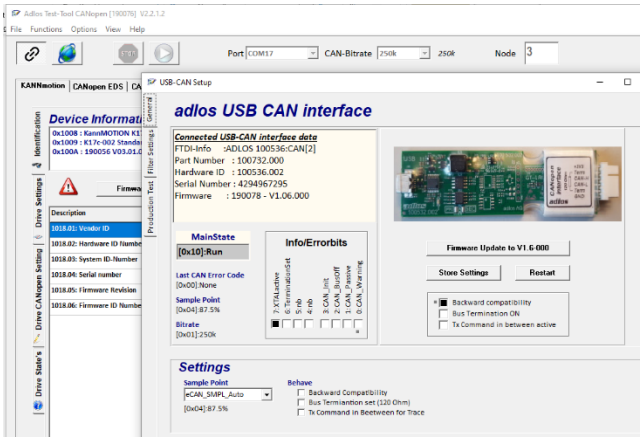
API (190073/190074/190080)

If you want to create your own Windows based application, adlos offers to use its API, means some useful DLLs (DLL: .net Assemblies) to allow quick an save access to our devices.

TestToolCANopen

This Tool is part of the COMwatch Toolsuite.





Contact information

Adlos AG
Föhrenweg 14
FL-9496 Balzers

Thomas Vogt
Thomas.Vogt@adlos.com
Tel: +423 263 63 63

Countries: CH, A, LI, SK, IT
www.adlos.com

KOCO MOTION GmbH
Niedereschacher Straße 54
D-78083 Dauchingen

Olaf Kämmerling
O.Kaemmerling@kocomotion.de
Tel: +49 7720/995858-0

Countries: DE, BE, NL, LU
www.kocomotion.de