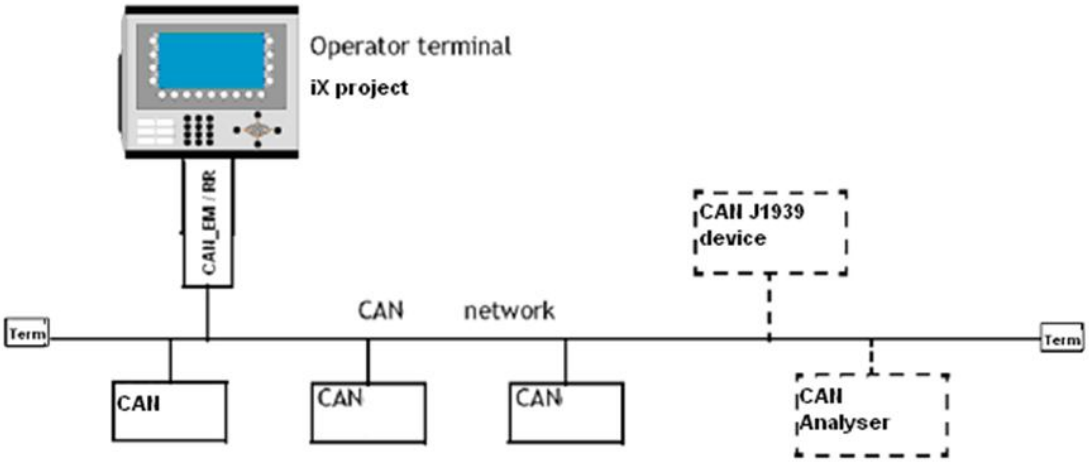
FREECAN configuration

Introduction

This manual describes how to configure and set up FreeCan_EM driver and tag addresses in the panel, connect it to a ciX CAN module, connect the CAN network and also how to configure and set up driver and tag addresses in the panel.

Additional to normal address commands the FreeCan driver can be used with an iX driver import module, which imports tags from a Excel sheet and transfers them to the iX tag list. Using Excel taglist has the advantage of a single overview source and a data bit mask, which allows the definition of every used data bit in the CAN telegrams. By importing this list, a "taglist.lst" file is created, which is loaded into the ciX module on driverstart to make the tag list known to the ciX module.

The FreeCan ciX module can be connected to most CAN2.0 CAN networks. It's purpose is to read and collect the CAN telegrams in a receive list and to write CAN telegrams.
CAN is supported with **11** and **29** bit headers.
Basic knowledge of CAN is recommended.



Release Notes

The HMI panel uses the driver to build commands, which are interpreted in the CAN ciX module hardware (with the firmware). So there are 2 related releases:

Driver Version	**Release**	**Description**
5.00.28	April 2011	Version with taglist and firmware V21 load
Firmware Version	**Release**	**Description**
FreeCanUSB_V21.hex	20.4.2012	FreeCan CAN module program (firmware)

The driver version can be read in the iX Designer in the tags/controller menu:

The CAN module firmware version can be read from a tag: HV returns a double word with the number of the version as a integer, e.g. 2100, which means FreeCan Version 21 subversion 0.

Disclaimer

Please note that changes in the controller protocol or hardware, which may interfere with the functionality of this driver, may have occurred since this documentation was created. Therefore, always test and verify the functionality of the application. To accommodate developments in the controller protocol and hardware, drivers are continuously updated. Accordingly, always ensure that the latest driver is used in the application.

Limitations

- The maximal hold of the receive list is 3500, so 3500 different CAN-ID's can be hold.
- The maximum Excel length is 5000 tags
- With ciX module 2 CAN channels (1,2) can be accessed. They are galvanically isolated from the main board and from each other. So each CAN channel can be connected to a different CAN bus.

Firmware and driver Versions

Up from Driver 5.00.20 and taglist FreecanUSB_V16.hex:
the taglist is loaded into the ciX module by the driver

Up from Driver 5.00.23 and taglist FreecanUSB_V20.hex:
CAN Bus Logging is available

Up from Driver 5.00.28 and taglist FreecanUSB_V21.hex:
Driver loads taglist and firmware. No additional Exe needed any more.

Hints:

If you get communication error after firmware update, please reset your panel by power disconnect.

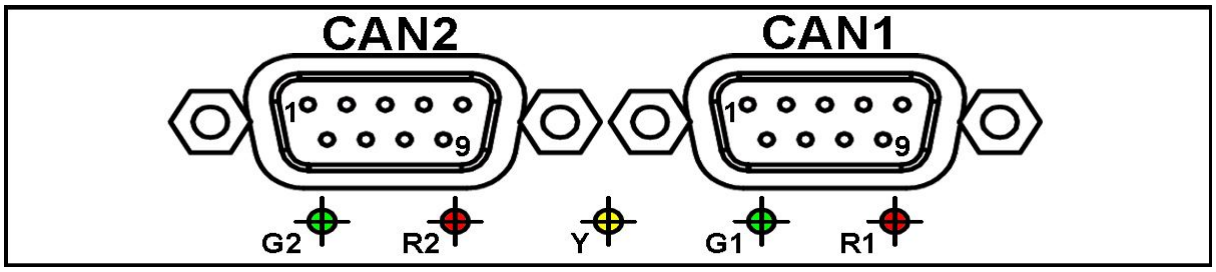
If you update your driver, the driver parameters resets to default values. Please check your driver parameter values, the most important issue is the COM port (default=1).

Connecting the CAN network

As in every CAN bus the connection is of the type Point-to-net (bus). The demands of CAN CAN bus connection must be fulfilled. Terminate the CAN bus on each end with 120 Ohm.

On ciX module the CAN bus is galvanically isolated to the panel and between CAN1 to CAN2. So every CAN bus can be driven seperated.

Here you see the CAN connection on ciX module.



Interface Pinning

CAN1 / CAN2 : 2 x 9 pole SUB-JD (male)

PIN	Description
2	CAN_L
3	CAN_GND
5	CAN_SHLD
7	CAN_H

DIL Switches on Bottom

Termination on/off for CAN1/CAN2.
Unscrew CAN module to see the switches. Termination is **off** by **delivery**.
If the ciX module is the first (or the last) device in the CAN cable, add an external R120 resistor between H and L, or switch module termination to “on”.

S1 (CAN1) / S2 (CAN2)

Switch1	Switch2	Switch3	Switch4	Description
off	off	off	off	Termination off
on	on	on	on	Termination on

LED's

5 LED's showing CAN Bus state for CAN1/CAN2

G2	R2	Y	G1	R1	Description
off	off	on	off	off	Module is not configured or **loading**
on	off	on	on	off	Module is configured and **waiting**
on	flashing	flashing	on	flashing	Module is **working** Yellow flashing: Module communicates to panel Red flashing: Module receives CAN telegrams

CAN cable length

The length of CAN bus cable (without repeater) depends on the baudrate.

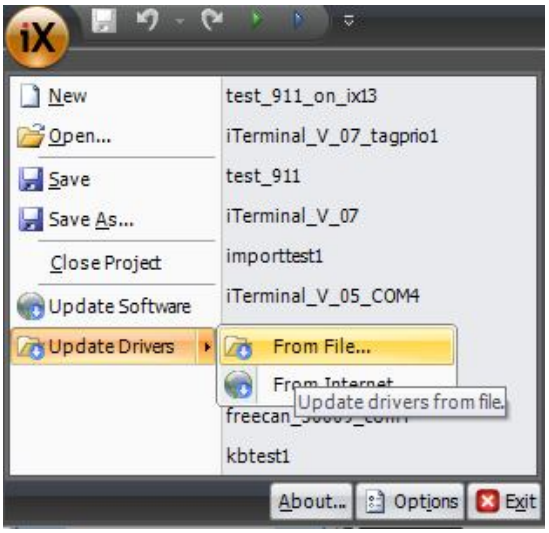
Baud	Max length
500k, 800k, 1M	max 40 m
100k, 125k, 250k	max 100 m
50k	max 500 m
20k, 10k	max 1000 m

Note: Don't connect more than 32 nodes on a CAN cable (without repeater).

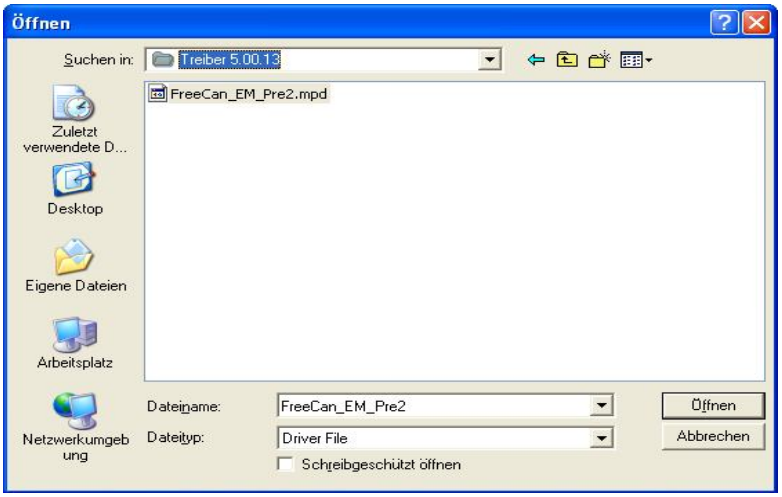
Driver install in iX developer

The actual FreeCan driver must be installed in iX developer

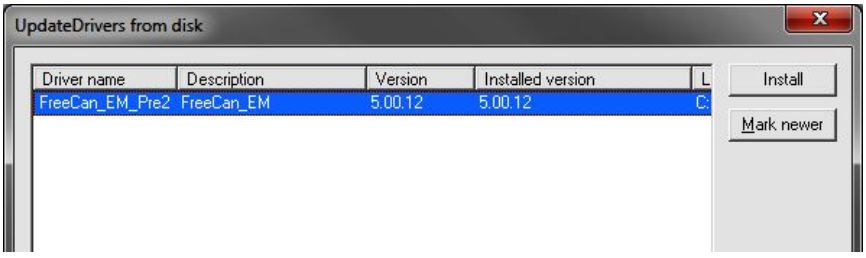
Driver source can be internet or file:



Choose FreeCan driver:



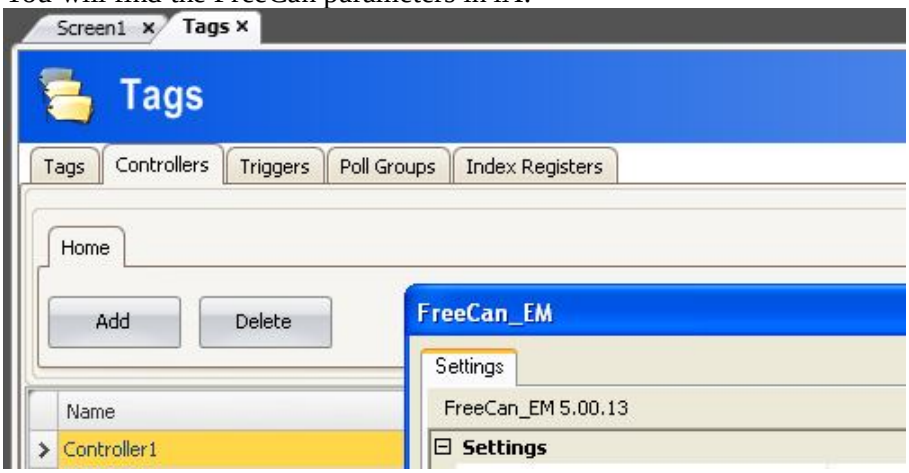
Install newest version:

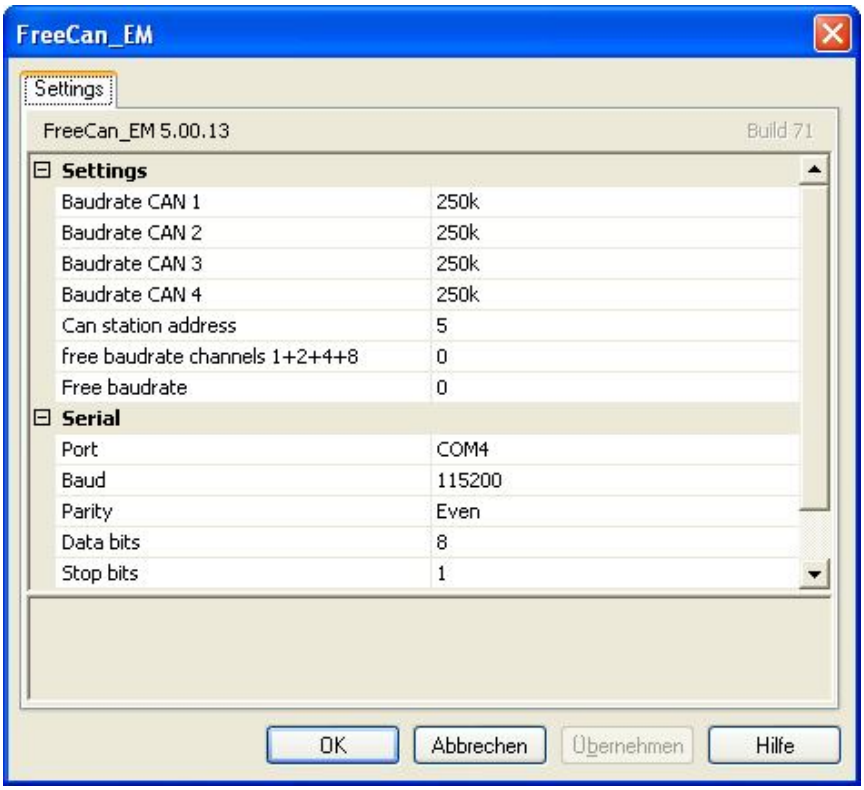


Hint: Restart iX developer after driver update !

Driver Settings

You will find the FreeCan parameters in iX:





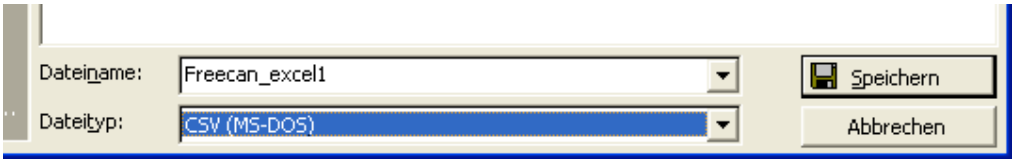
Parameter	Description
Baudrate CAN x	Selects the used baudrate for the CAN channel: 1M, 800k, 500k, 250k, 125k, 100k, 20k, 10k. *CAN3 and CAN4 are not used.* Default: 125k
CAN Station address	The CAN node number of the HMI (1-127). Used for special protocols, e.g. J1939. Default: 1
Free Baudrate Channels 1+2+4+8	Selects the CAN channels for free baudrate setting. 1= CAN1, 2=CAN2, 4=CAN3, 8=CAN4. Add all channels to activate free baudrate sets -> e.g. 9 selects CAN1 and CAN4. Default: 0 (free baudrate is off, pre-set baudrates are used)
Free Baudrate	32 bit value to set any baudrate. The calculation is for a SJA1000 Can with clock 12MHz. The splitting is like this: Bit 0..9 =BTR, 16-19 TSEG1, 20-22=TSEG2, -> e.g. 0x001C0002 sets CAN=250MHz Default: 0
Serial/Port	The used COM Port COM 5 for TxA, TxB panels COM 3 for TA panels, COM 4 for EPC Nautic panels COM 8 for QTERM panels dynamic COM value for PC/TxC (see windows device list) Default: COM 1

All other parameters should not be changed.

Excel list import in iX

Please mind to put your used Excel list into a subdirectory "/Project Files" of your iX project. Only if you do so, the produced taglist.lst (by import) will be transferred to the ciX module on driver start. The subdirectory "/Project Files" is automatically build in a new project.

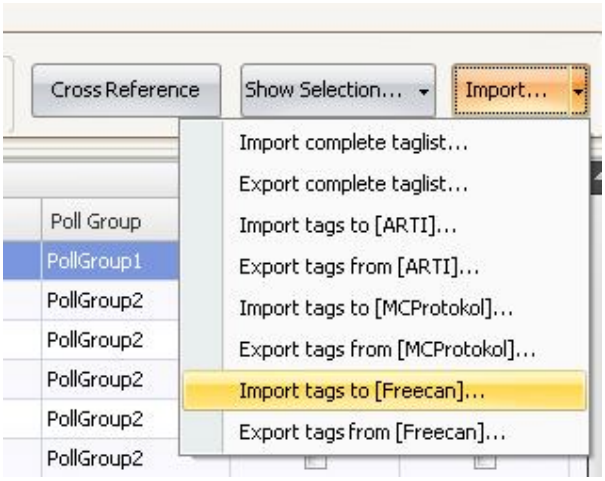
For import of a list to iX, the used Excel list must be saved in CSV Format. Use in Excel menu "save under" and choose CSV (MS-DOS format).



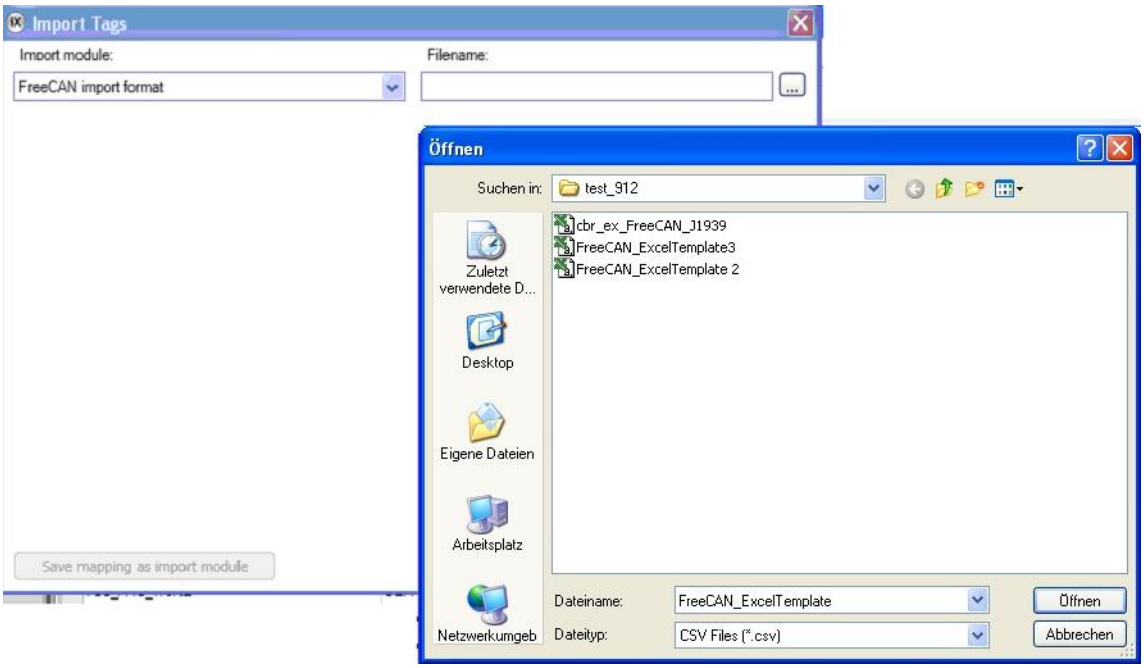
Next you will get a message, that only single sheets can be stored. Confirm all with yes.

Now the Excel sheet is stored in a CSV format with cells divided by ";" and lines by "LF". So avoid ";" and Newlines in the Excel cells – they may lead to wrong imports.

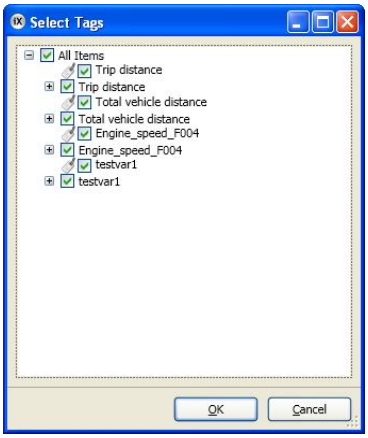
iX Developer, menu Tags, choose Import tags to [FreeCan] :
[FreeCan] = Inserted, renamed Controller



Choose the directory, where the MS Excel list is stored.
It should be the subdirectory "/Project Files" in your project.
Choose the CSV file (*.CSV) and "import" the file.
Note that the produced taglist.lst file (by import) is also stored in this (same) directory !



Now you see a "select Tags window, where you can choose the imported tags.
"All Items" will import all tags. You may avoid problems by only importing new tags.



If you are overwriting an existent list, you will get a window, there this overwrite is managed for every overwritten tag. If you want your choose to be for all items, hook “Apply this action to all”. If you just changed some items, choose “Merge” to update the iX taglist. If you make bigger changes, we recommend to delete all tags before import.

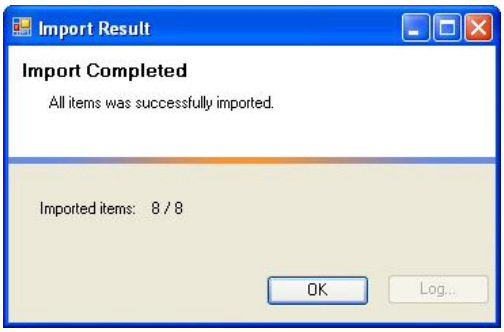
“Change” will keep the old tags and add the new ones. If name is same, it puts the new tag in iX with an addition (e.g. 1), so you may get the same tag in iX tag list twice with different names.

“Overwrite” will overwrite the old tag and load it with the new tag. All tag values in iX will be lost. Also the tags in scripts must be redone.

“Merge” will overwrite only the relevant informations and leave others untouched. Example: you have changed “gain” in a tag in iX, but gain is not used in Excel list. Merge will keep the iX gain value.

“Skip” will not change this item.
“Cancel” will stop the import.

After import you should get this message:



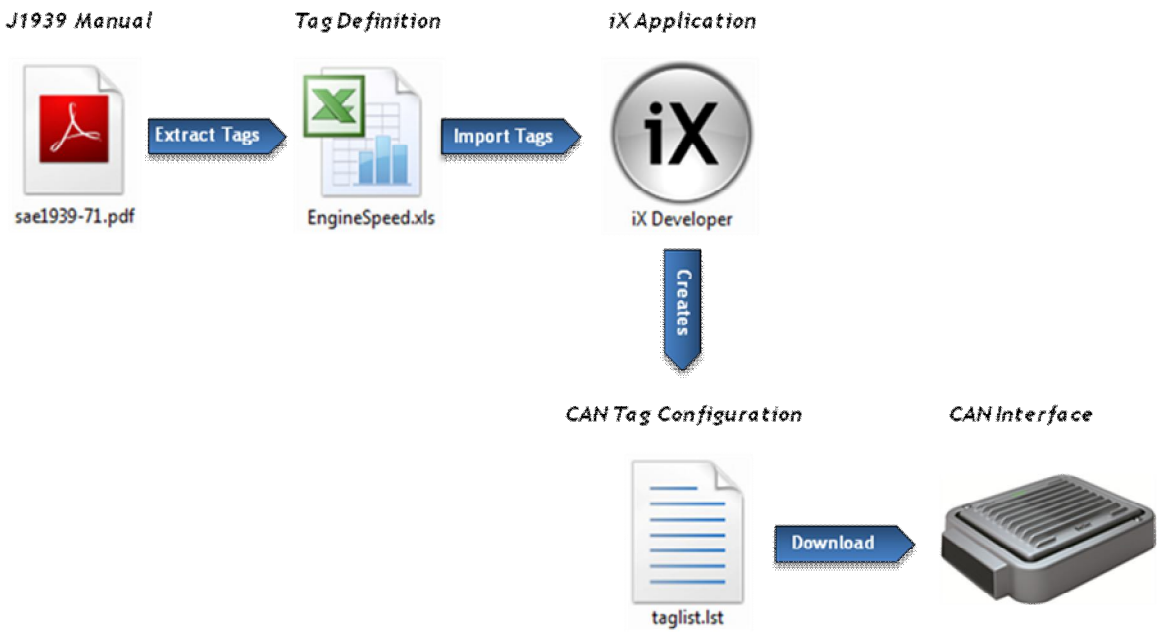
Load taglist into CAN module

The import into iX produces a taglist.lst file, which must be loaded into the CAN module. This taglist is a reference file, connecting your driver tag names to the tag definition data. This gives the Can Module the information, what command “ILn” is.

If you are **not** using the Excel list (only non-list commands), you do not have to mind this.

By FreeCan_EM 5.00.20 version (and up) the taglist is loaded into the ciX module automatically by the driver, so you do not have to mind it. Only make sure, that your Excel/CSV file is in the subdirectory “/Project Files” of your project and you validated your project before transfer.

Schemexample for project build



1. Make up you mind what CAN telegrams to build, e.g. for a J1939 device
2. Define the tags in the Excel sheet, place this Excel file in your project subdirectory “/Project Files”
3. Save Excel sheet as *.csv (MS-DOS)
4. Open iX developer project and choose tags / import to driver
5. Choose *.csv file
6. Import dedicated tags and connect them to a panel control, validate project
7. Transfer project to panel.
8. Start iX project on panel. At driver start the taglist.lst file is loaded into the ciX module.

Addressing

There are 3 types of addresses:

1) Basic signals
2) Non List addresses
3) MS Excel list addresses

Basic signal

These are all addresses, which concern the basic CAN functions, as CAN state.

CAN channel state - Sn

This signal reads the CAN bus activity on 1 second base.
Syntax: Sn where n is the CAN channel 1or2
Example: S1
Return: = 0 if CAN Bus is off; = 1 if CAN Bus is working;

Bus Load – BLn or BLM

The bus load in percent for a CAN channel can be read. It is the real telegram density divided by the max possible telegram density. Do not run a CAN channel over 60%, the collision rate will get too high; a good value is max 30%
BLM: The CAN module internal load. The CAN module collects the CAN telegram and puts them into a receive list. BLM gives a percentage of the input delay. A percentage over 50% will lead to telegram loose.
Syntax: BL1..4 or BLM
Example: BL1
Return: 0..100 (%)

Version - HV

Returns the firmware version as a 32 bit value.
Example: firmware “FreeCan_V16.hex” returns "1600"

Read receive list count - HRL

Returns the filling of the receive list. The list can hold up to 3500 different telegrams. Each different CAN ID holds a receive place. Also a different CAN channel leads to a different receive place. From the same CAN ID only the youngest data are kept. If more than 3500 IDs received, the oldest values will be overwritten.

Clear receive list - HRW

Deletes all data from the receive list. Same happens at a power up or reset.

Bus module status - YS

Gives back a list of important CAN module state informations in a ASCII string. This can be used for debugging. Please do not use it in fast projects, as the read takes time resources. Available since firmware V20.

The read can be shorted by adding 2 numbers:

YSm,n.

m is start number, n is end number.
Example: YS0,5 will read FW to BL4.

The list of informations is:

No	Return name	Full name	Example	Meaning
0	FW	Firmware Version	FW:00120	Version 1.20
1	CAN4-1	CAN state 4-1	CAN4-1:0011	CAN1,2 are active
2	BL1	Busload1	BL1:5	busload 5%
3	BL2	Busload2	BL2:0	busload 0%
4	BL3	Busload3	BL3:15	busload 15%
5	BL4	Busload4	BL4:0	busload 0%
6	BLM	Busload1	BLM:1	internal load 1%
7	LP	Receive Listen pointer	LP:21	21 telegrams in list
8	LR	Receive List checked	LR:21	21 telegrams counted
9	RL	Reference list	RL:45	45 tags for read prepared
10	SL	Sendlist	SL:2	2 telegrams in send list
11	RC	Reset counter	RC:1	Firmware start counter

Remarks:

- LP and LR should be the same number, otherwise your receive list is corrupted or you got CAN telegrams with Header ID”0”
- RC should be 1, otherwise you have software restarts.
- CAN4-1 is built like this CAN4*1000+CAN3*100+CAN2*10+CAN1
  CAN=0: bus inactive, CAN=1: bus runs.

Non List addresses

These accesses can be used without having a taglist loaded to the CAN module.

Read variable - RR

The data content of a CAN telegram can be read / written
So you must pass the CAN ID, channel, datamask
Syntax: RR 12345678 (K0-8C) M0-3F:1-20
Return: value or 0 (if CAN ID is not found in list)

12345678 is the CAN telegram ID (max 29 bits in Hex notation).
By default the telegrams are extended (29Bit) telegrams.
To force a standard (11bit) telegram, the value must be under 0x800 and you must set the highest bit.

Examples:
- 00001234 is Can ID header 0x00001234 extended
- 80000234 is Can ID header 0x00000234 standard

K is channel + intelformat + datacount*16 (in Hex notation)
CAN Channel is 1-4, 0=channel 1.
Intelformat (L,H) is +8, otherwise it is motorola formated data (H,L)
Datacount is 0-8 *16, there 0 = 8 data bytes. This is used for sending telegrams with less than 8 data bytes.

Examples:
K=0 (or nothing) is channel1, Motorola Format and 8 send bytes.
K=3A(b00111010) is CAN channel 2, Intel Format and 3 send bytes.

Mask is the position and length of the data mask (in Hex notation)
Mm:n
m is the position of the mask in the 64 data bits. 0 is the most left bit, 3F is the most right bit.
n is the length in the mask 1-20(hex), so it is 1 to 32 bits (double word).

Examples:
M0:8 is the most left data byte : FF,xx,xx,xx,xx,xx,xx,xx
M8:1 is the second data byte, highest bit : xx,8x,xx,xx,xx,xx,xx,xx
M3F:1 is the most right byte, lowest bit : xx,xx,xx,xx,xx,xx,xx,x1

Write:
Data will be shifted most right in the mask.
As values in iX can only be double words (32 bit), the max data length is 32(0x20) bits.
So writing “1” to M0:20 will set data: 00,00,00,01,xx,xx,xx,xx
All other data in the telegram are left untouched.

Read variable timeout - RT

The age of a CAN telegram can be compared.
So you must pass the CAN ID, channel, timeout-value in sec
Syntax: RT 12345678 (K0-4) T0-65535
Tn is the time in milliseconds(decimal), rounded with 10.

Return: 1 (within timeout) or 0 (timeout)

Example:
RT18FEE000K0T5000 will read telegram ID 0x18FEE000 out of the receive list and compare the receive time within 5 seconds. If this ID was received within 5 sec RT will return a 1, otherwise 0.
If Tn=0, the receive of the telegram will be quoted without time: if this ID was received it returns a 1, otherwise 0.

Read J1939 PGN - RJ

The data content of a J1939 telegram with PGN can be read / written.
The receive list is searched for this PGN, regardless the station or priority. The first found CAN ID is taken.
So you must pass the PGN, channel, data mask
Syntax: RJ 1234 (K0-8C) M0-3F:1-20
Return: value or 0 (if CAN ID is not found in list)

1234 is the PGN 0-FFFF (max 16 bits in Hex notation).
The PGN in J1939 are bits 8-24 in the telegram header.

Examples:
RJF004M18:10 is PGN 0xF004 Engine Speed, SPN 190 (mask xx,xx,xx,LL,HH,xx,xx,xx)

Write Full telegram – HRA, HRB

As the highest tag value in iX is 32 bits, a full telegram must be written in 2 commands.
HRA hold the left 4 data bytes, HRB holds the right 4 data bytes and sends the telegram.

So you must pass the CAN ID and channel
Syntax: HRA (HRB) 12345678 (K0-72)
Returns: send value

Example:
HRA18FEE000K1=0x11223344 and
HRB18FEE000K1=0x55667788
sends Telegram on CAN1, ID 0x18FEE00: 0x1122334455667788

Hint: Force standard header (11bit) by setting highest CAN ID bit.
CAN ID: Highest bit =0 -> extended Header ist send. Eg 0x12345678,eg HRB1
CAN ID: Highest bit =1 -> standard Header ist send. Eg 0x80000723

Hint: Send less than 8 data bytes: adding data cnt to channel high nipple
HRB18FEE000K1 send 8 databytes
HRB18FEE000K41 send 4 databytes on channel 1
HRB18FEE000K72 send 7 databytes on channel 2

Write telegram cyclic – RZ

Sends a telegram cyclic. The telegram must be in receive list (or was send before)
You must pass the CAN ID and channel.
Syntax: RZ 12345678 (K0-8C)
Returns the cyclic value in ms*10.

Example:
HZ18FEE000K1=0x100 sends telegram ID 0x18FEE000 every 2,56 second on CAN1

Detect J1939 station - HLI

Detect a J1939 device (ECU)
Syntax: HLI m , n
m is the station address 0-ff
n is the timeout in sec 0-ff
Return: 1 (within timeout time) or 0 (timeout)

Examples: HLI2,0 checks if J1939 ECU with address 2 exists
HLI4,5 checks if J1939 ECU 4 did send a telegram within the last 5 seconds

In J1939 telegrams the transmitting station is kept in the low 8 bits of the CAN ID header. So HLI searches all received telegrams for this station number and returns if it was received within the last n seconds. If n =0, the total receive from this station is quoted (= Does this device exist?).

Excel list addresses

If you use the FreeCan Import, these tags are automatically build in your project (Please use e your "/Project Files" folder). Using the Excel list gives you a perfect overview on your tags and helps you avoiding transfer error.

ILXn , ILBn, ILWn, ILDn with n=0…65535
The number of the tag in the taglist. So IL0 refers to the first tag in the Excel list.

ILXn returns 1 bit as result on read to taglist variable number n
ILBn returns 8 bit as result on read to taglist variable number n
ILWn returns 16 bit as result on read to taglist variable number n
ILDn returns 32 bit as result on read to taglist variable number n
ILXn.VALID returns true/false as the timeout state of the variable

"IL" reads the data content of one List object. The list objects are defined in a Excel sheet, which can be imported into the FreeCan driver. By importing this list a taglist.lst file is generated, which is loaded into the CAN module by the driver. Please make sure, that driver import and taglist load fit together - otherwise you will get uncertain data.

Can Bus Logging

CAN telegrams can be logged in the ciX module.
All send and received telegrams are logged with a time-info (1ms resolution).

The logging can run parallel to the normal program use, so a logging of the ciX CAN bus actions and its responds is possible. Even though no ciX blockations are known, the debug tool should only be used for debugging cases. The logged datas are hold in the RAM memory and are lost on ciX reset or powerup.

There is a start and a stop and a readout command.

Start command - MS
With the start command, all CAN telegrams are written into a internal buffer of the ciX module. The buffer can hold 19620 telegrams. If the buffer is full, the oldest datas are overwritten (ringbuffer).

Stop command - ME
With the stop command, the logging is stopped.

Readout command - MR
With the read out command, the log is stopped and all logged telegrams are transferred from ciX to a program. The readout can be repeated.
On readout the telegrams are Excel like formatted and a csv file is written.
So the log can be analysed with a basic Excel program.

If the readout runs on a iX program, the readout may take a longer time and block other iX functions. For a readout of a full memory about 30 minutes are necessary, as dataspeed of 10 telegrams per second is typical.

If the readout runs from special log programs, the readout is much faster and a readout of a full memory will take about 3 minutes. This special log programs are available from Beijers support. Also a mix is possible: Start and Stop in iX, end iX and readout with special log program.

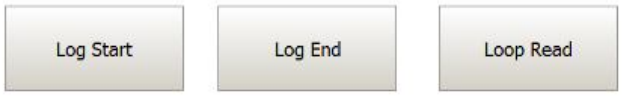
Example: Start, Stop and Readout in iX with scripts

Define in tags:

	Log_Start	DEFAULT	ReadWrite	INT16	MS
	Log_Stop	DEFAULT	ReadWrite	INT16	ME
	Log_Readout	DEFAULT	ReadWrite	STRING[200]	MR

Please mind the String[200] definition for the readout!

Define 3 buttons in a screen:




```
// Script Source Code
//Example for Readout file on a TxA SD card (“\Storage Card”)
//Change curFile=”…” if name or path change is needed
public partial class Screen1
{
	int batchread;
	int i;
	int nr;
	string curFile;
	StreamWriter myfile;
	FileStream fs;

	//”Loop Read”
	void Button3_Click(System.Object sender, System.EventArgs e)
	{
		try {
			//create a new file
			nr=1;
			curFile = @"\Storage Card\logging" +Convert.ToString(nr) +".csv"; //TxA
			//curFile = @"c:\logging" +Convert.ToString(nr) +".csv";	//PC
			while(File.Exists(curFile)){
				nr+=1;
				curFile = @"\Storage Card\logging" +Convert.ToString(nr) +".csv";
				//curFile = @"c:\logging" +Convert.ToString(nr) +".csv"; //PC
			}
			fs = File.Create(curFile);
			fs.Close();
		}
		catch(Exception ex){
			MessageBox.Show("File "+curFile+" could not be created");
			return;
		}
```

```
                //Readout and write to file
                try {
                        myfile = new StreamWriter(curFile,false);                       //overwrite
                        myfile.WriteLine(("CAN Logging Read out at "+DateTime.Now.ToString()
                                +" \n"),true);
                        //read telegrams from ciX
                        batchread=1;
                        i=0;
                        while(batchread==1){
                                i++;
                                //if(i>100) batchread=0; //optional limit
                                if(batchread>0) Globals.Tags.Log_Readout.Read();
                                if(Globals.Tags.Log_Readout.Value == "END") batchread=0; // End
                                if(batchread>0) myfile.Write(Globals.Tags.Log_Readout.Value,true);
                        }
                        myfile.Close();
                }
                catch(Exception ex){
                        MessageBox.Show("Error Logging Readout = "+ ex.ToString() );
                }
        }

        // " Log Start"
        void Button1_Click(System.Object sender, System.EventArgs e)
        {
                Globals.Tags.Log_Start.SetAnalog(0);
                Globals.Tags.loggingIsOn.SetTag();
        }

        // " Log End"
        void Button2_Click(System.Object sender, System.EventArgs e)
        {
                Globals.Tags.Log_Stop.SetAnalog(0);
                Globals.Tags.loggingIsOn.ResetTag();
        }

  }
}
```

In the code above a logfile is created named “logging1.csv”.
This a Excel view of it:

CAN Logging Read out at 29.03.2012 15:56:50

Rec: 308	ID	Ex/Sd	Can	D1	D2	D3	D4	D5	D6	D7	D8	T[ms]	State
1	0x7FF	S	1	0x00	0x12	0x01	0x80	0x00	0x00	0x00	0x00	10	T
2	0x7FF	S	2	0x00	0x12	0x01	0x80	0x00	0x00	0x00	0x00	11	
3	0x7A1	S	1	0x00	0x12	0x01	0x80	0x00	0x10			13	
4	0x7A1	S	2	0x00	0x12	0x01	0x80	0x00	0x10			13	
5	0x200	S	1	0x00	0x00	0x00	0x00	0x00	0x64			19	
6	0x200	S	2	0x00	0x00	0x00	0x00	0x00	0x64			19	
7	0x20A	S	1	0x00	0x00	0x00	0x92	0x02	0x00	0x00	0x01	50	
8	0x20A	S	2	0x00	0x00	0x00	0x92	0x02	0x00	0x00	0x01	50	

There were 308 Telegrams recorded.
They are sorted chronological, as you can see in Row T[ms].
The CAN ID is in HEX notation.
Ex/Sd means if the Can Header is **S**tandard (11bit) or **E**xtended(29bits)
Can is the Channel 1 or 2
D1-D8 are the 8 Data bytes (hex notation)
T[ms] is the recording time in ms from logstart.
State is the Telegram State:

- “ “ is a received telegram without Errors
- “T” is a Transmit telegram, sended by ciX modul
- “TE” means, that the ciX Modul could not send the telegram
- “E” is a transmit or receive Error
- “B” is a CAN Busoff error (Can bus is off)
- “O” is a overrun error, a receive telegram was lost.

FreeCan Import Informations in a Excel sheet

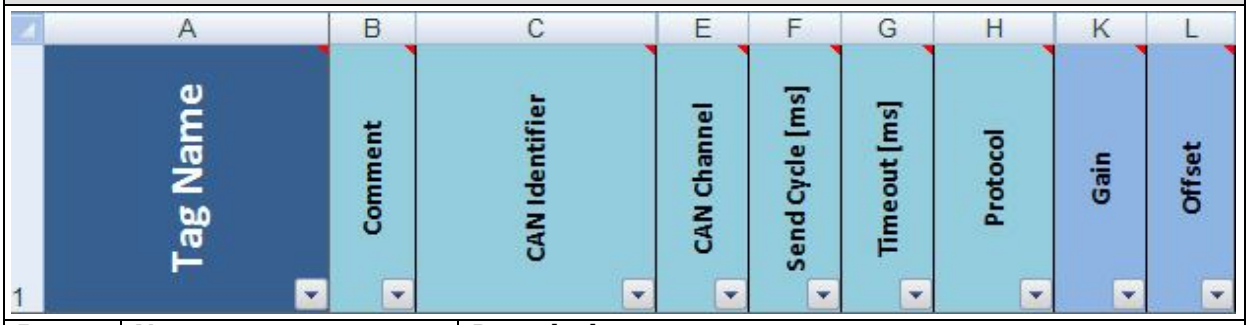
The informations in a Excel sheet are :
Each line is a variable
Each row may be a information in a variable.

For all fields in the sheet you should mind :
- No CR/LF (Enter) should be used !
- Only English characters should be used.

The Excel sheet must be exported as CSV(MS-DOS) for import in iX !

For a start you can use our “FreeCan_ExcelTemplate.xls” form.

Excel sheet row definition

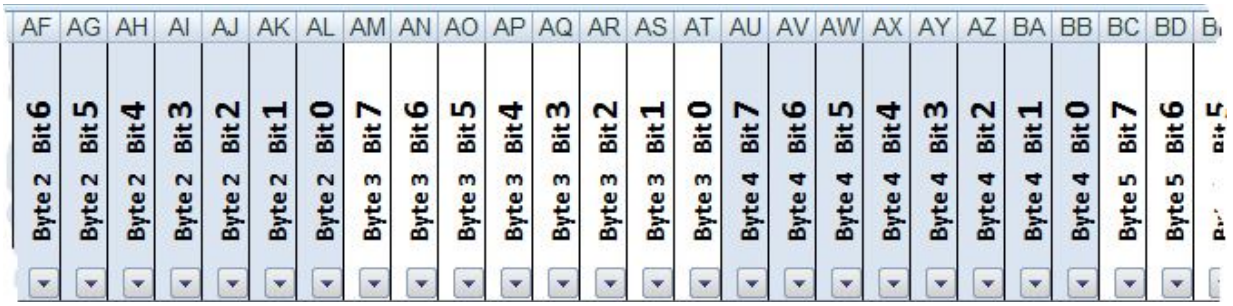


Row	Name	Description
A	**Name**	Symbol tagname, is copied to iX taglist. Must be less than 80 characters, no Space “ “ or special characters may be used.
B	**Comment**	Add here additional comments
C	**CAN Identifier / Variable number**	Can telegram Header ID (0…0x1FFFFFFF). The CAN telegram header, right aligned May be Extended(29bits) or Standard(11bits). Values > 0x7FF will cause a extended header telegram. Setting the format with 8 hex characters will force a extended telegram: 0x00000001 will send ID 1 as a extended telegram.
D		Reserved for protocol data
E	**CAN Channel**	CAN channel (0…4) Nothing = 0 =1 will use CAN1. Value 2=CAN2. Value 3=CAN3. Value 4=CAN4. For EM_CAN/RR use only CAN channel 1 or 2 and activate CAN2 on use with DIL (SB3/SB4)
F	**Send Cycle [ms]**	Nothing = 0 = no cyclic telegram send >0 = send telegram cyclic with value *10. Cycle send will start on first write from panel. Data value will be the last written value.
G	**Timeout [ms]**	Timeout 0…655350 ms, rounded by 10. Nothing = 0 = no timeout detection of this tag

		Value > 0 = tag will be watched on timeout value. ILXn.VALID is created on import only if value >0 and will return the result of the timeout watch. TRUE = this Can telegram was received within timeout time. FALSE = this Can telegram was NOT received within timeout time.
H	**Protocol**	Nothing or "J1939" or "J1939P" "J1939" will cause a preset data value of all bits =1 "J1939P" will cause a preset data value of all bits =1, and a search for a PGN (row C) without Priority or station in the receive list (this is used for finding J1939 values from unknown stations)
K	**Gain**	Value is copied to iX gain (multiplier)
L	**Offset**	Value is copied to iX offset (addition)
O - BZ	**Data mask**	Data Mask definition (see next chapter)
CJ	**Tag priority**	The sorting of the tags on import can be defined. Give first tags a "1" and number all tags to be sorted, e.g. "2","3",… Unmarked tags are added at end. This function is important to build read tag groups, e.g. all read tags on one iX page. This will fasten up the communication between driver and Canmodule. If it is not used, the tags are copied in same order as in Excel sheet.
CK	**Allways active**	Is copied to iX, "Allways active" gets checked. (Tag is read cyclic, even not used on a screen)
CL	**Poll Group**	Is copied to iX Poll group, if named "PollGroup1" to "PollGroup5"

- Unmentioned rows can be used freely.
- Do not insert rows in the MS Excel Sheet, the import detects the predefined rows.
  If you want to add row, do this from CM.
- Remember not to use special characters or line feed.
- For basic functionality these rows must be filled with values:
  A (name), C (CanID) and O-BZ(data mask).
  All other rows are optional.

Data mask definition



Rows	Name	Description
O - BZ	Byte0 Bit7 – Byte7 Bit0	

Here you define the order of the tag value bits to the CAN telegram data.
The 64 data bits of a CAN telegram can be freely mixed.
The bits in the Excel sheet are defined from left (MSB,Byte0,Bit7) to right (LSB Byte 7, Bit0).
This is gathered in a mask, which is given to the driver as tag value.
Define the new position by numbers 1..64, there 1 is the LSB (most right bit).
Zero (0) or nothing makes the position unused, the data is not copied.
The way is the same for read and write.
On write the unused data bit information is taken from the last received CAN telegram.

Example:

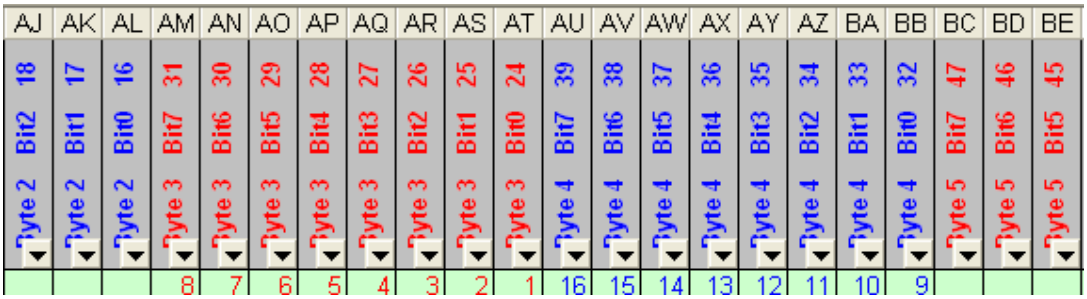
”Tag1”, Data definition: Byte0/Bit 7 =1, rest empty.
CAN telegram Data = 0x 80,00,00,00,00,00,00,00 -> value = 1
CAN telegram Data = 0x 00,00,00,00,00,00,00,00 -> value = 0

”Tag2”, Data definition: Byte0/Bit1 =1, Byte0/Bit0=1, rest empty.
CAN telegram Data = 0x 0F,00,00,00,00,00,00,00 -> Value = 3
CAN telegram Data = 0x,0E,00,00,00,00,00,00,00 -> Value = 2
CAN telegram Data = 0x,F1,00,00,00,00,00,00,00 -> Value = 1
CAN telegram Data = 0x,FC,00,00,00,00,00,00,00 -> Value = 0

Write value =3
Before: CAN telegram Data = 0x,FC,00,00,00,00,00,00,01
After : CAN telegram Data = 0x,FF,00,00,00,00,00,00,01 (this is written on the CAN bus)

Freecan configuration , Preversion 1.14, 9.5.2012 UER

Example for J1939, PGN 0xF004, SPN 190 engine speed:



AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ	BA	BB	BC	BD	BE
18 Bit2 Byte 2	17 Bit1 Byte 2	16 Bit0 Byte 2	31 Bit7 Byte 3	30 Bit6 Byte 3	29 Bit5 Byte 3	28 Bit4 Byte 3	27 Bit3 Byte 3	26 Bit2 Byte 3	25 Bit1 Byte 3	24 Bit0 Byte 3	39 Bit7 Byte 4	38 Bit6 Byte 4	37 Bit5 Byte 4	36 Bit4 Byte 4	35 Bit3 Byte 4	34 Bit2 Byte 4	33 Bit1 Byte 4	32 Bit0 Byte 4	47 Bit7 Byte 5	46 Bit6 Byte 5	45 Bit5 Byte 5
			8	7	6	5	4	3	2	1	16	15	14	13	12	11	10	9			

CAN telegram Data = 0x,FF,FF,FF,23,45,FF,FF,FF -> Value = 0x4523
This value should be multiplied by 0.125 in Row K (gain) to get the correct speed rmp.

Troubleshooting

Error messages

The meaning of error messages from the controller shown by the driver.

Error message	Description
Comm Err stn X	Communication driver to module fails totally. You do have a communication problem from driver to the CAN module. - **After a taglist or firmware Update: Restart panel (power off)** - Check driver and firmware version - **Did you set the correct serial port in the parameter setting?** COM5 is correct for TxA,TxB panels. A driver update defaults this value! - Does your taglist and driver project fit ? Validate project, transfer it and restart panel. - Do you use a USB stick in a bumblebee panel ? It may need too much power. Unplug it and restart.
Bad Reply stn X	Trying to access an illegal or non-existing address / tag. - You may call a unknown tag. Does your taglist and driver project fit ? Validate project, transfer it and restart panel. - Check the length of the read data, especially read strings length must fit to the tag definition. - Check if driver and firmware fit together and are latest version. - Check CAN connection and termination. Check if CAN channel is switched on. - Check the tag addresses used in the project.

Note: Not all access errors are displayed.
If a tag was not received, it returns a 0 as default. -> check tag.VALID to find out if the value is valid.

Errors on import

- Make sure, that your imported CSV file is freshly saved from Excel list
- Check if Excel and CSV files are in project sub folder "/Project Files"

Please check in Excel list:

- did you use any CR (carriage return = new line) in the list ?
  This will destroy the CSV information. Delete all CR/LF.
  Check your CSV file after save, second line must be a legal tag!
- did you use special characters like "äöü" ? Delete them.
- Is your taglist.lst file marked as "read-only" ?
  Remove the read-only bit.
- did you use Space " " in the name (col A)? A space will end the name input, do not use it in a tag name.