FreeCAN

Beijer Electronics iX FreeCAN

Startup document

English

V18, 2013-09-13, uer, cbr, amr

Foreword

This document is a startup document that describes the Beijer Electronics FreeCAN driver in general, and some additional functionalities.

Please see this document as an addition to the driver manual. In the driver manual you will find more detailed descriptions and some more hints.

The driver manual you will find in the driver settings in iX Developer.

Chapter 1: General descriptions about CAN

Chapter 2: Describes the basic usage of this driver by creating a simple project

Chapter 3: Explains special CAN protocol possibilities such as:
J1939, NMEA, Simple CanOpen Slave

Chapter 4: FAQs and Hints for FreeCAN handling

Order no: SUEN283

Please read the entire installation manual prior to installing and using this equipment.
Only qualified personnel may install, operate or repair this equipment. Beijer Electronics AB, including all its group companies, is not responsible for modified, altered or renovated equipment. Because the equipment has a wide range of applications, users must acquire the appropriate knowledge to use the equipment properly in their specific applications. Persons responsible for the application and the equip-ment must themselves ensure that each application is in compliance with all relevant requirements, standards and legislation in respect to configuration and safety. Only parts and accessories manufactured according to specifications set by Beijer Electronics AB may be used.

BEIJER ELECTRONICS AB, INCLUDING ALL ITS GROUP COMPANIES, SHALL NOT BE LIABLE TO ANYONE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM THE INSTALLATION, USE OR REPAIR OF THIS EQUIPMENT, WHETHER ARISING IN TORT, CONTRACT, OR OTHERWISE. BUYER'S SOLE REMEDY SHALL BE THE REPAIR, REPLACEMENT, OR REFUND OF PURCHASE PRICE, AND THE CHOICE OF THE APPLICABLE REMEDY SHALL BEAT THE SOLE DISCRETION OF BEIJER ELECTRONICS AB.

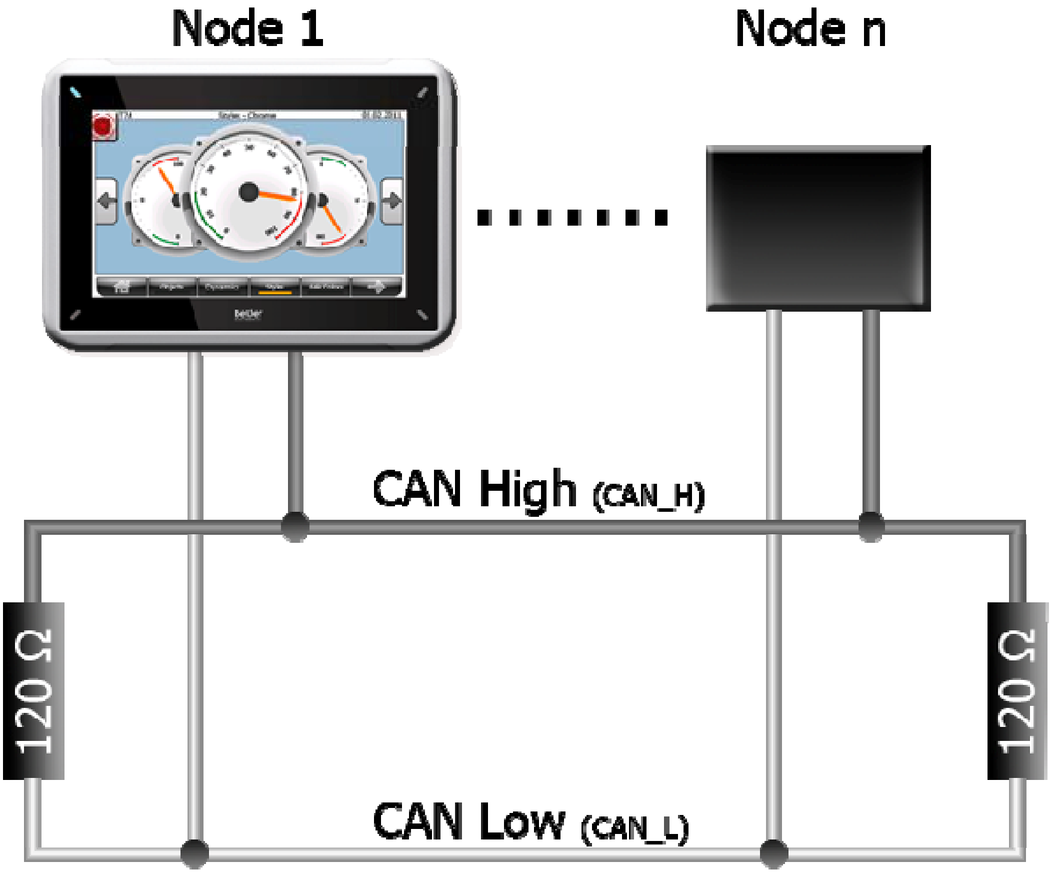
Content

1 General

1.1 What is CanBus?

CAN Bus (**C**ontroller **A**rea **N**etwork) is basically an automotive bus that was developed by company Bosch. It's a dual wire serial protocol that was created to reduce the huge amount of cables in cars. The maximum data transfer rate is 1MBaud. The bus must be terminated on each end with 120 Ohm. The international ISO No is 11898.

Topology:



1.2 What is iX FreeCAN?

IX FreeCAN

=

iX Free configurable ***Can***Bus

With the FreeCAN it’s possible to connect an iX Panel to a CAN Bus system.
Therefore you need an iX Panel and a CiXFreeCAN communication module.

The CAN Bus supports a lot of different protocols.
The iX FreeCAN idea was to have one totally free and independent driver that supports CAN on a base parameter level and not to have dozens of different iX drivers.

The FreeCAN CiX-CAN module can be connected to most CAN2.0 networks.
Its 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.

Base definitions for tags (CAN parameters) are done in an MS Excel sheet,
available at support page of http://www.beijerelectronics.com/ and part of the start up guide as well.
These definitions will be loaded into the CiX-CAN Module.

2 My first iX FreeCAN project

2.1 What is needed for the project?

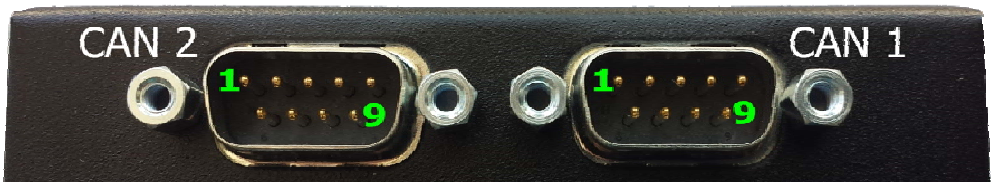
2.1.1 Hardware:

1. **Beijer iX TxA, TxB or TxC**

2. **CiX-CAN module**

CAN module interfaces

The CiX module has two galvanically isolated CAN interfaces, so two CAN busses can be driven fully separated.

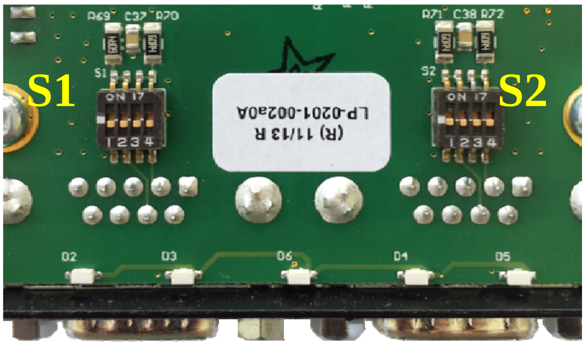


CAN1 / CAN2 : 2 x 9 pole SUB-JD (male)

Interface Pinning	
PIN	**Description**
2	CAN_L
3	CAN_GND
5	CAN_SHLD
7	CAN_H

DIL switches on the base

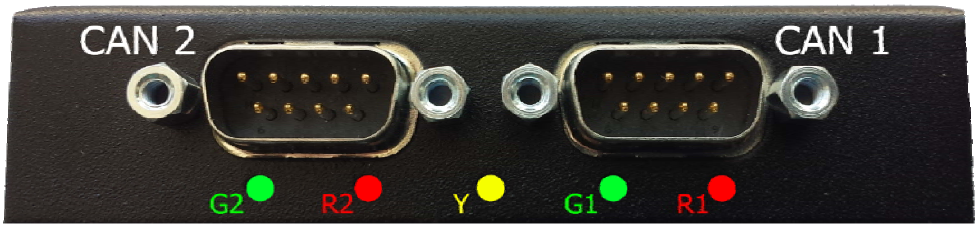
Termination on/off for CAN1/CAN2.
Unscrew the CAN module to see the switches. Termination is off by default at delivery.



If the CiX module is the first / last device that is connected to a CAN cable, add an external R120 resistor between H and L, or switch module termination to on.

Bus Termination for S1 (CAN1) / S2 (CAN2)				
Switch1	**Switch2**	**Switch3**	**Switch4**	**Description**
off	off	off	off	Termination **off**
on	on	on	on	Termination **on**

LEDs



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

LED CAN Bus State			
G 1/2	**R 1/2**	**Y**	**Description**
-	-	on	Module is **not configured** or firmware is **loading**
on	flashing	flashing	Module is **working**: Yellow flashing: Module communicates to panel Red flashing: Module receives CAN telegrams
-	on	off	**Error** on CAN

- means not relevant

CAN cable

Termination can be switched on the base of the CAN module (see **CiX-CAN module**).

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

CAN Cable length	
Baudrate	**Max length**
500k, 800k, 1M	40 m
100k, 125k, 250k	100 m
50k	500 m
20k, 10k	1000 m

Note: 1. For a basic cable connect all CAN_L (PIN2) to one wire and all CAN_H(PIN7) to another wire.
Connecting CAN_GND (PIN3) is only needed for CAN Open.
2. Don’t connect more than 32 nodes on a CAN cable (without repeater).

3. CAN analyzer (nice to have)

This helps to see what happens on the CAN Bus.
We recommend a USB-to-CAN compact device from IXXAT.

2.1.2 Software:

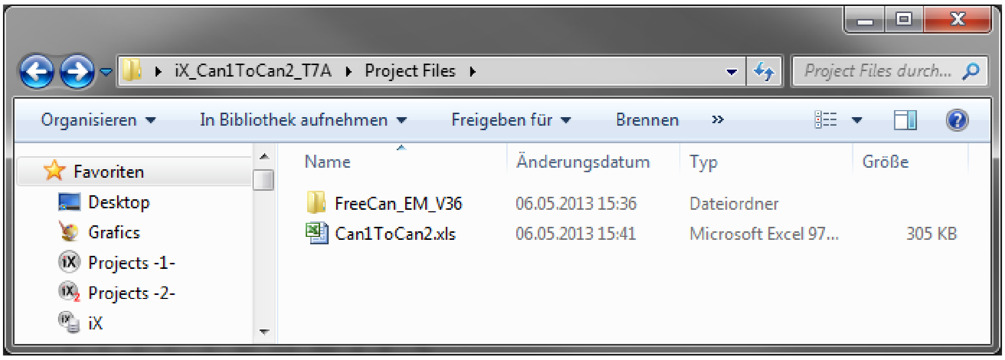
4. iX Developer V2.0 SP1 (version 2.0.463.0 or higher)

5. Installed FreeCAN _EM driver in version 5.00.36 or higher

6. iX_Can1ToCan2_T7A sample project

Download: http://www.beijerelectronics.com/

Content of the sample project:



Name	Änderungsdatum	Typ	Größe
FreeCan_EM_V36	06.05.2013 15:36	Dateiordner	
Can1ToCan2.xls	06.05.2013 15:41	Microsoft Excel 97...	305 KB

- Folder FreeCAN_EM_V36 = FreeCAN driver in version 36(2 files)
- File Can1ToCan2.xls = Excel template (Excel 97-2003 version)
for tag definitions

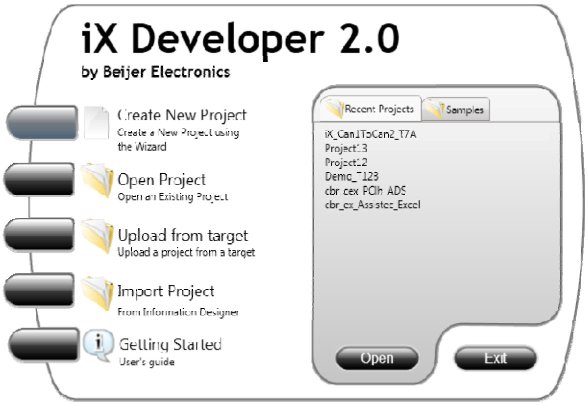
2.2 How should the project basically work?

The project will send CAN telegrams in cycles (1 second) on **CAN1** channel and receives them on **CAN2**. For the cyclic send we use the internal iX Demo driver, as this driver has some cyclic counter tags. CAN1 and CAN2 channels must be directly connected via CAN cable.

Procedure:

Install two drivers in an iX project
a) DEMO driver for cyclic tag writing
b) FreeCAN driver for CAN data exchange

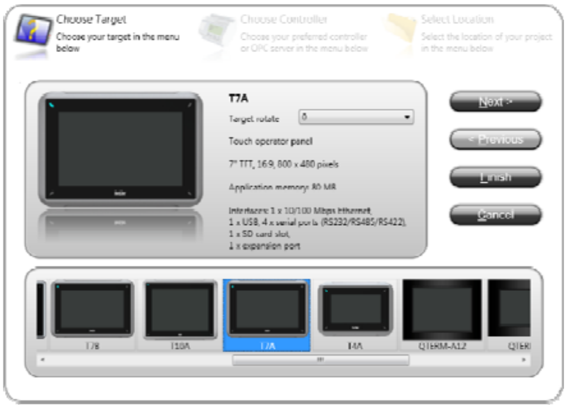
2.3 Create a new T7A project with demo driver



Recent Projects:
- iX_Can1ToCan2_T7A
- Project13
- Project12
- Demo_T123
- cbr_cex_PCIh_ADS
- cbr_cx_Assistec_Excel

Start iX Developer

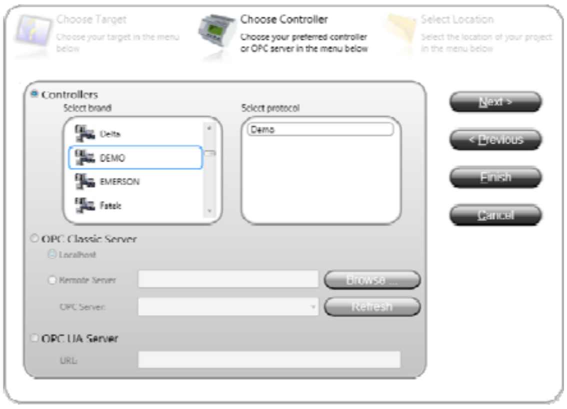
Create a new project


Choose Target
Choose your target in the menu below
Choose Controller
Choose your preferred controller or OPC server in the menu below
Select Location
Select the location of your project in the menu below
Next >
< Previous
Finish
Cancel
T7B
T10A
T7A
T4A
QTERM-A12
QTERI

T7A
Target rotate: 0
Touch operator panel
7" TFT, 16:9, 800 x 480 pixels
Application memory: 80 MB
Interfaces: 1 x 10/100 Mbps Ethernet,
1 x USB, 4 x serial ports (RS232/RS485/RS422),
1 x SD card slot,
1 x expansion port

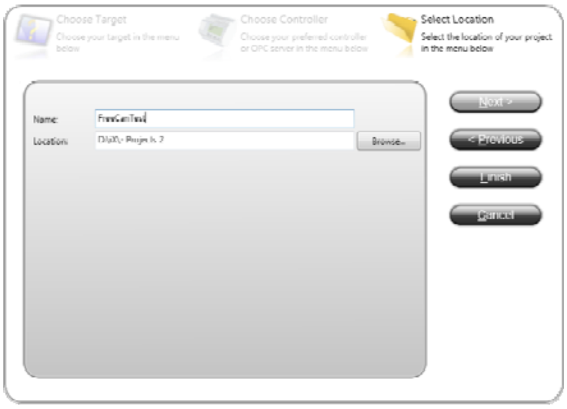
Select T7A

Press ***Next***



Select Controller brand: ***DEMO***

Press ***Next***


Choose Target
Choose your target in the menu below
Choose Controller
Choose your preferred controller or OPC server in the menu below
Select Location
Select the location of your project in the menu below
Browse...
Next >
< Previous
Finish
Cancel

Name: FreeCanTest
Location: D:\iX-Projects 2

Enter a project ***name*** e.g. ***FreeCANTest***

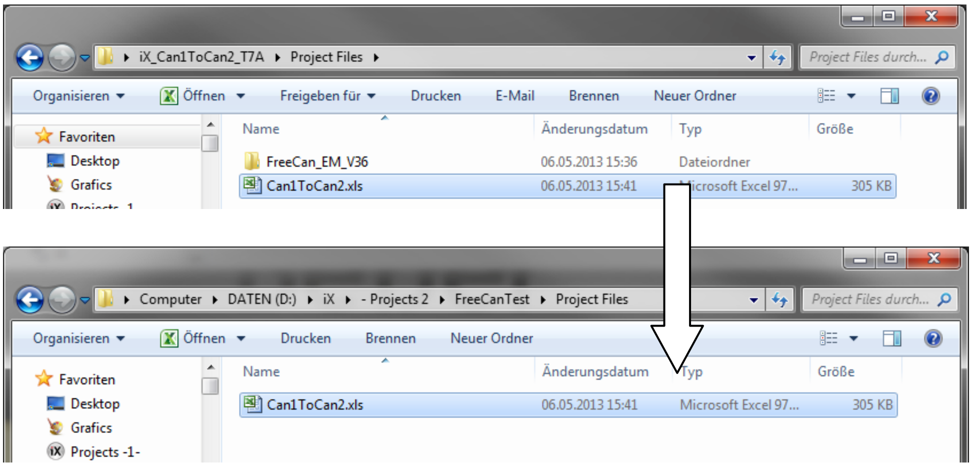
Select the ***project path***

Press ***Finish***

2.4 Parameter Definition in *.xls File

In the sample project **iX_Can1ToCan2_T7A** you will find a folder named ***Project Files*** with a file ***Can1ToCan2.xls***.

Copy that file to the new project in its ***Project Files*** folder.

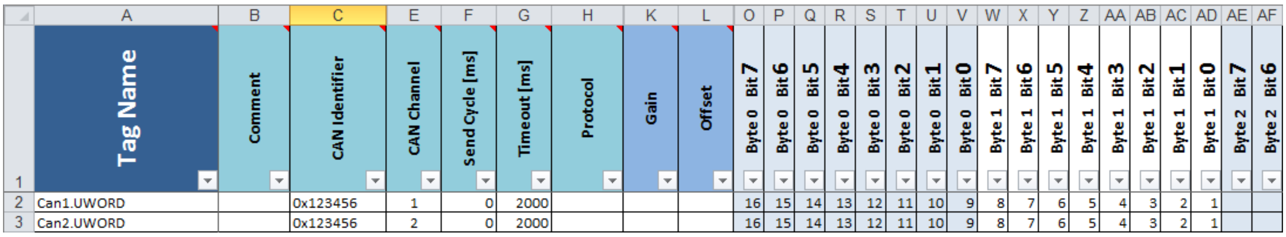


Name	Änderungsdatum	Typ	Größe
FreeCan_EM_V36	06.05.2013 15:36	Dateiordner	
Can1ToCan2.xls	06.05.2013 15:41	Microsoft Excel 97...	305 KB

Name	Änderungsdatum	Typ	Größe
Can1ToCan2.xls	06.05.2013 15:41	Microsoft Excel 97...	305 KB

This Microsoft Excel file is used as a **template definition file**, where you are able to define the access to the CAN telegrams. Basically you enter the tag name, the CAN header address and the value bits in the data field.

See example:**Can1ToCan2.xls** file where already 2 Tags / CAN parameters are entered:



	A	B	C	E	F	G	H	K	L	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF
1	Tag Name	Comment	CAN Identifier	CAN Channel	Send Cycle [ms]	Timeout [ms]	Protocol	Gain	Offset	Byte 0 Bit 7	Byte 0 Bit 6	Byte 0 Bit 5	Byte 0 Bit 4	Byte 0 Bit 3	Byte 0 Bit 2	Byte 0 Bit 1	Byte 0 Bit 0	Byte 1 Bit 7	Byte 1 Bit 6	Byte 1 Bit 5	Byte 1 Bit 4	Byte 1 Bit 3	Byte 1 Bit 2	Byte 1 Bit 1	Byte 1 Bit 0	Byte 2 Bit 7	Byte 2 Bit 6
2	Can1.UWORD		0x123456	1	0	2000				16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		
3	Can2.UWORD		0x123456	2	0	2000				16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		

- **2 parameters** = iX Tags defined : **Can1.UWORD** and **Can2.UWORD**
  Tag.Datatype for later correct iX Datataype:
  UWORD = Unsigned 16 bit value will be converted to UINT in iX

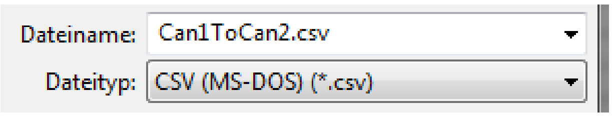
CAN header address is **0x123456** (extended 29 Bit).

- Tag **Can1** works on CAN channel1
- Tag **Can2** works on CAN channel2
- **Timeout** time is **2000 ms** = 2 seconds.
- Data: **First 2 bytes** of CAN data (**byte0 / byte 1**).

The receive list can save up to 3500 CAN telegrams, meaning that 3500 different CAN-ID’s can be saved.

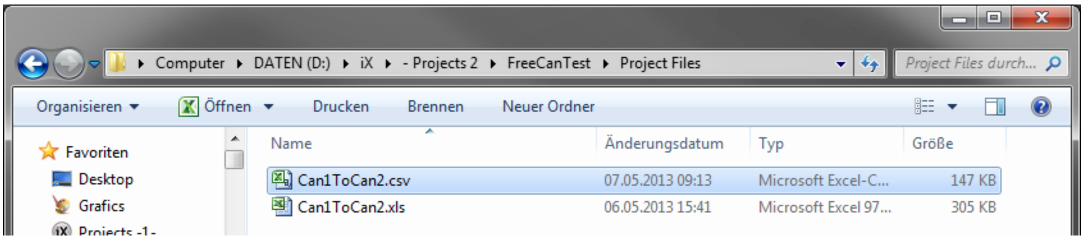
2.5 Create Parameter File in *.csv Format

Now the *.xls parametertemplate file must be saved in the format ***CSV (MS-DOS)***.



Dateiname: Can1ToCan2.csv

Dateityp: CSV (MS-DOS) (*.csv)

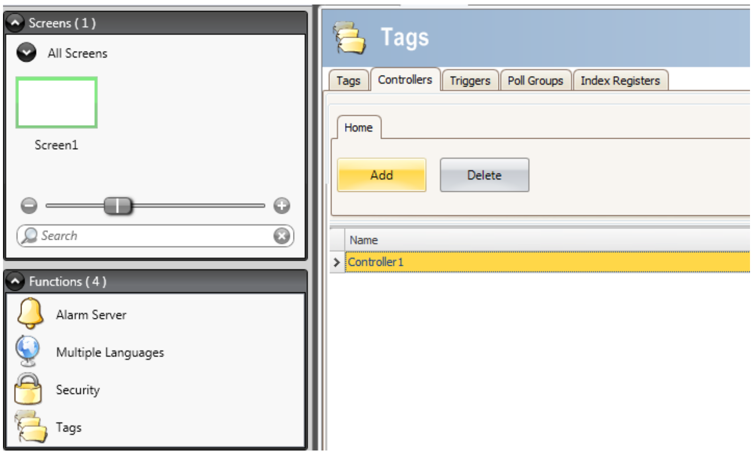


Name	Änderungsdatum	Typ	Größe
Can1ToCan2.csv	07.05.2013 09:13	Microsoft Excel-C...	147 KB
Can1ToCan2.xls	06.05.2013 15:41	Microsoft Excel 97...	305 KB

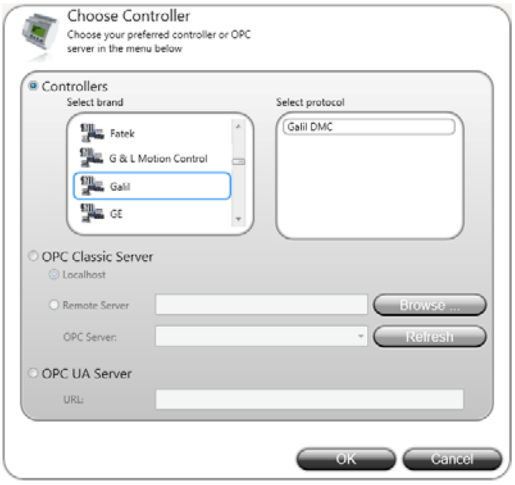
This is necessary for the iX tag import later on (2.8).

2.6 Installation of iXFreeCAN Driver

If you know that the correct driver is already installed, please continue with 2.7
Otherwise check if the driver is installed and which version.
Go to ***Function*** ➔ ***Controllers***. 1 Controller is already entered. (DEMO Driver).Click on ***Add***.



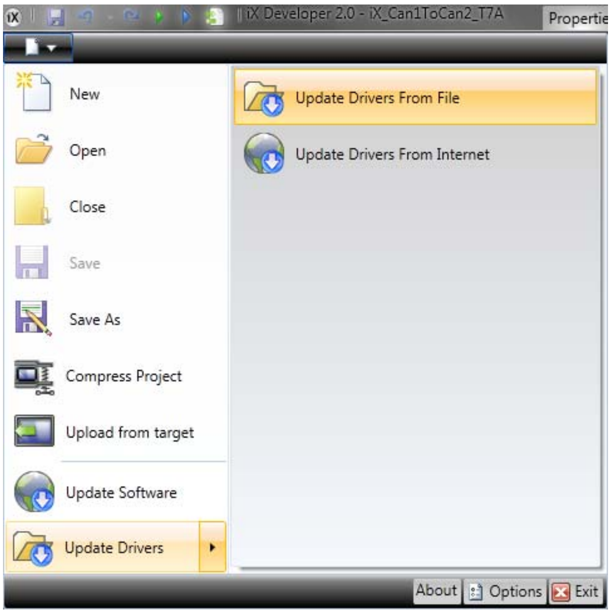
Name
Controller1



See if there is a ***FreeCAN*** Controller installed

If yes go on with 2.7

If no FreeCAN Controller can be selected you have to install the driver first.

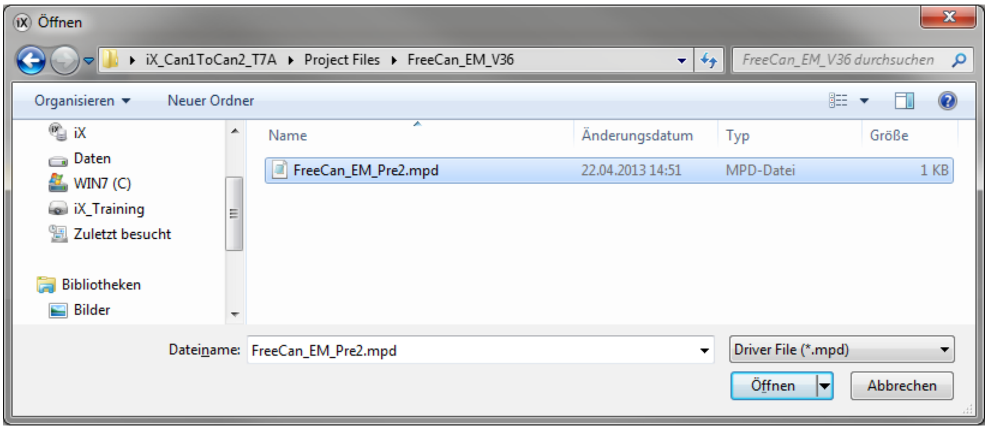


Go to ***Screen ComboBox***

Select ***Update Drivers***

Select ***Update DriversFrom File***

Choose the **FreeCAN_EM_Pre2.mpd** from sample project **iX_Can2Can_T7A\Project Files**

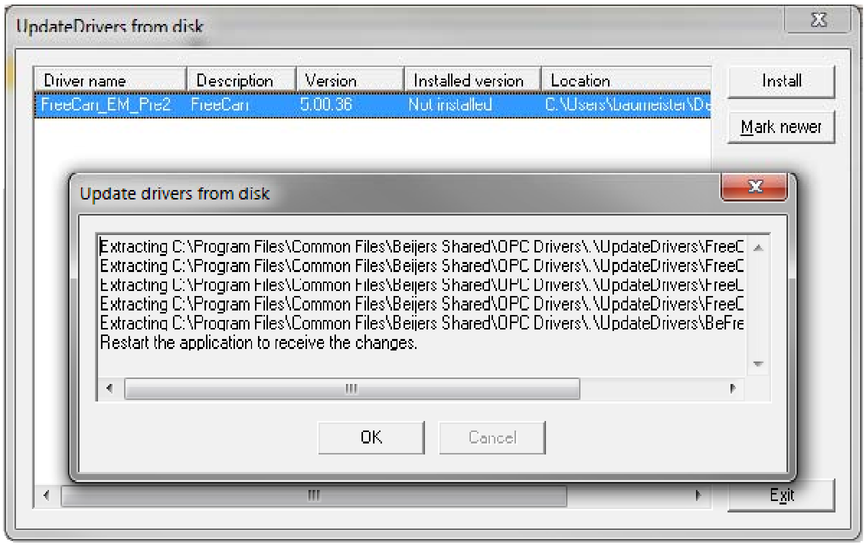


Name	Änderungsdatum	Typ	Größe
FreeCan_EM_Pre2.mpd	22.04.2013 14:51	MPD-Datei	1 KB

Dateiname: FreeCan_EM_Pre2.mpd

Driver File (*.mpd)

Mark the driver and press ***Install***



Driver name	Description	Version	Installed version	Location
FreeCan_EM_Pre2	FreeCan	5.00.36	Not installed	C:\Users\baumeister\De

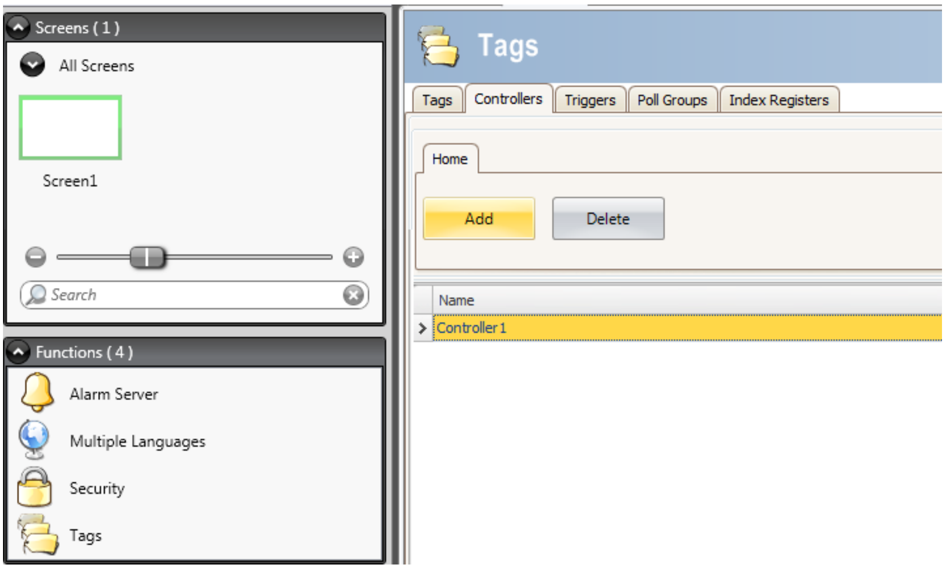
```
Extracting C:\Program Files\Common Files\Beijers Shared\OPC Drivers\.\UpdateDrivers\FreeC
Extracting C:\Program Files\Common Files\Beijers Shared\OPC Drivers\.\UpdateDrivers\FreeC
Extracting C:\Program Files\Common Files\Beijers Shared\OPC Drivers\.\UpdateDrivers\FreeC
Extracting C:\Program Files\Common Files\Beijers Shared\OPC Drivers\.\UpdateDrivers\FreeC
Extracting C:\Program Files\Common Files\Beijers Shared\OPC Drivers\.\UpdateDrivers\BeFre
Restart the application to receive the changes.
```

Confirm with ***OK***, ***Exit,*** and restart iX Developer.

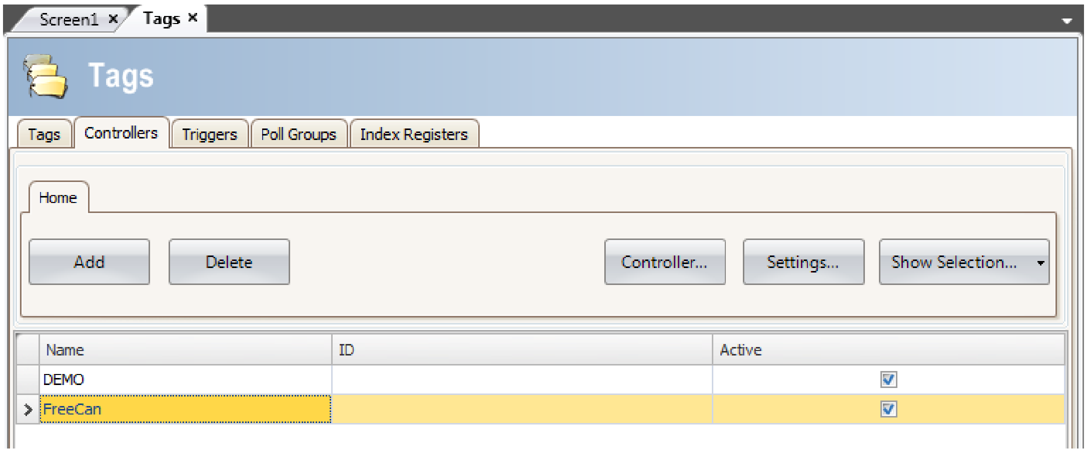
Note: On occurrence of problems in the FreeCAN driver installation, please make sure that you have full administration rights on your system and that iX Developer is started in administration mode.

2.7 IX FreeCAN Driver Configuration

Go to ***Function*** ➔ ***Controllers***. 1 Controller is already entered (DEMO Driver). Click on ***Add***.

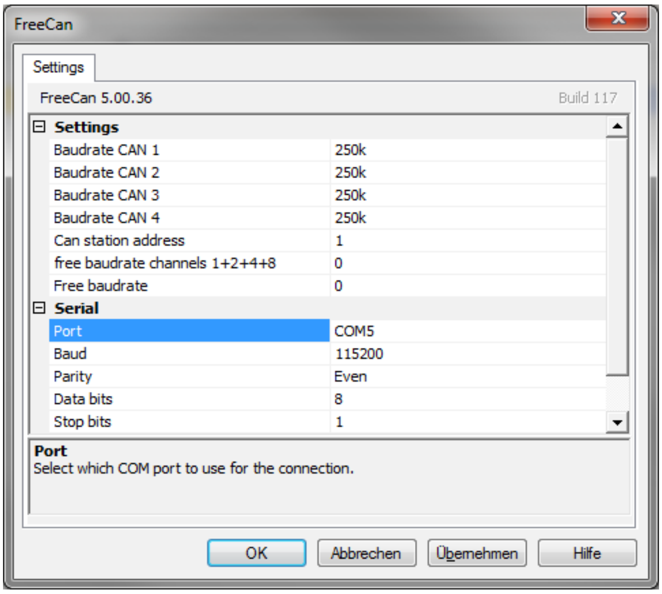


Select the FreeCAN Driver from the list.
Rename Controller1 to ***DEMO*** and the entered Controller2 to ***FreeCAN***.



Name	ID	Active
DEMO		☑
FreeCan		☑

Mark ***FreeCAN*** Controller and press ***Settings***.



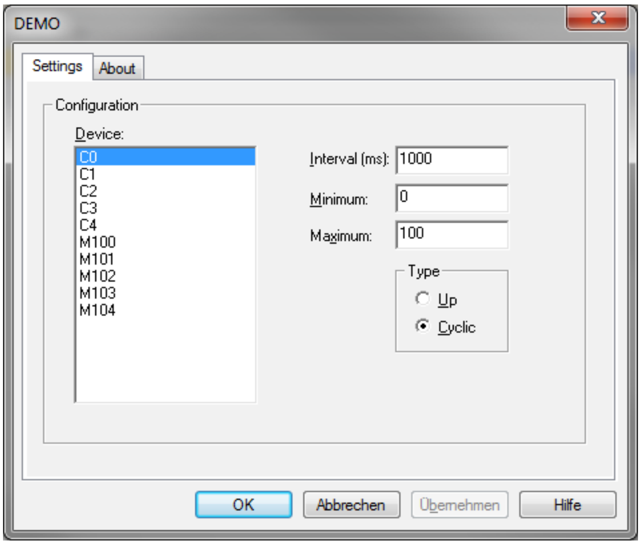
Settings	
Baudrate CAN 1	250k
Baudrate CAN 2	250k
Baudrate CAN 3	250k
Baudrate CAN 4	250k
Can station address	1
free baudrate channels 1+2+4+8	0
Free baudrate	0
Serial	
Port	COM5
Baud	115200
Parity	Even
Data bits	8
Stop bits	1

Port
Select which COM port to use for the connection.

The default setting of the FreeCANdriver defines the used baud rate of 250 kbaud. Please check that the used COM port is set to **COM5** (for ***TxA*** and ***TxB*** panels; for ***TxC*** check the settings in the ***device manager*** to get the correct COM port).

2.8 iX Demo Driver Configuration

Mark ***DEMO*** Controller and press ***Settings***.



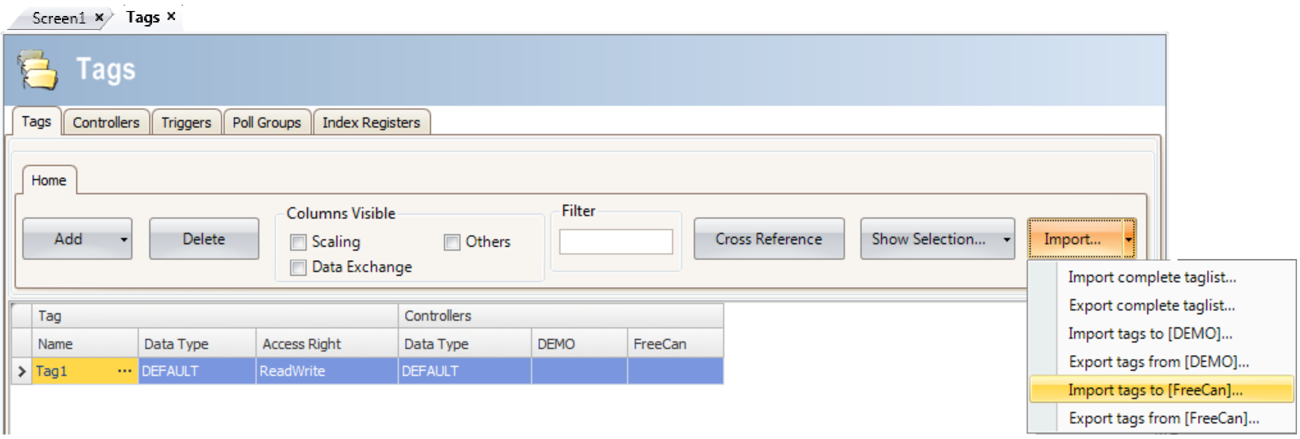
Device:
C0
C1
C2
C3
C4
M100
M101
M102
M103
M104

Interval (ms): 1000
Minimum: 0
Maximum: 100
Type
○ Up
● Cyclic

For the cyclic writing on CAN1 channel the Demo driver can be used. As default the **C0** Counter increments from 0 … 100 and decrements from 100 … 0.

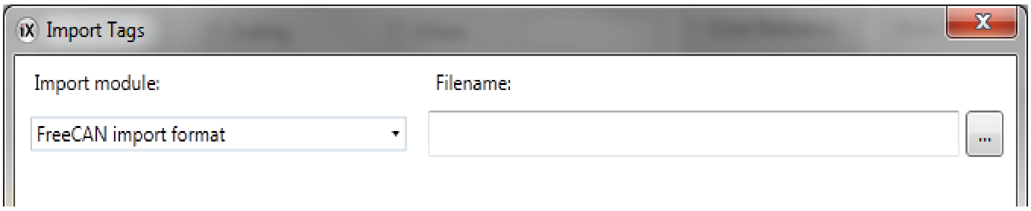
2.9 Import of Tags out of *.csv Parameter File

Go to ***Functions*** ➔ ***Tags*** ➔ ***Import tags to [FreeCAN]***.



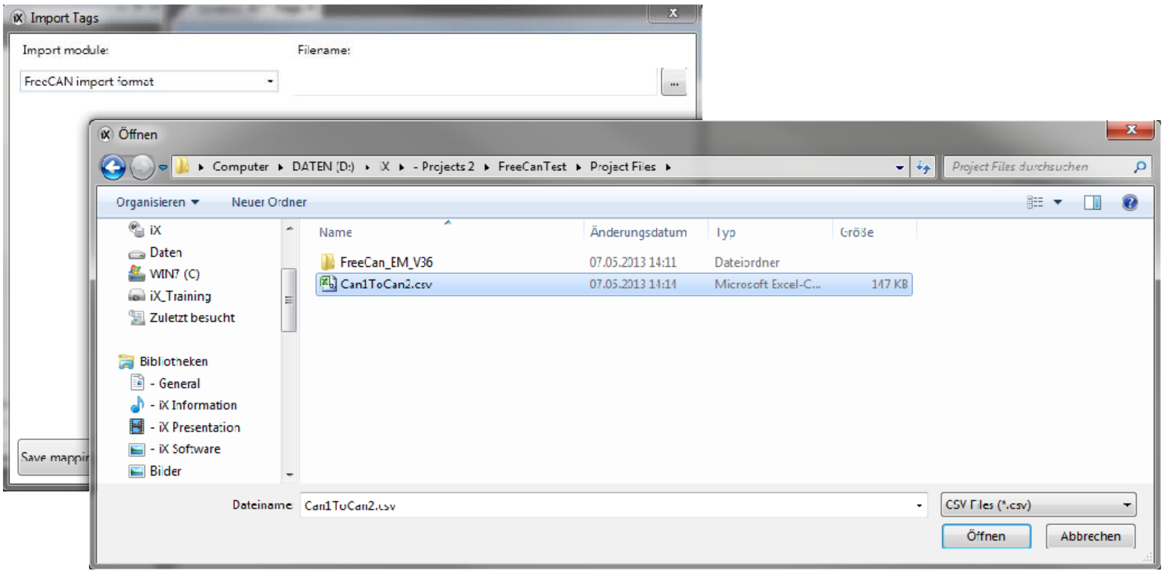
Tag			Controllers		
Name	Data Type	Access Right	Data Type	DEMO	FreeCan
Tag1	DEFAULT	ReadWrite	DEFAULT		

Import to the FreeCAN Controller opens the dialog ***Import Tags*** with the special ***FreeCAN import format***. Pressthe ***Browse Button:***



Import module: FreeCAN import format

Filename:



Import module: FreeCAN import format

Filename:

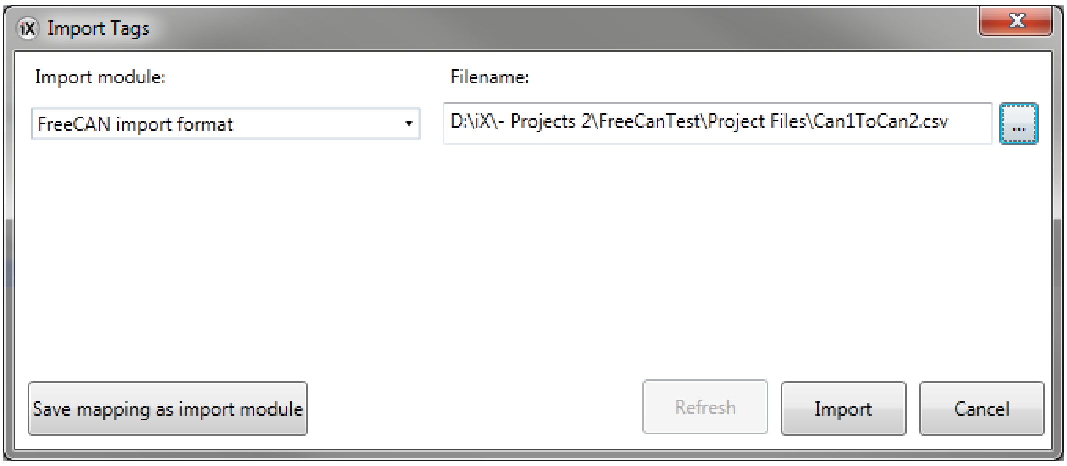
Computer ▸ DATEN (D:) ▸ iX ▸ - Projects 2 ▸ FreeCanTest ▸ Project Files ▸

Name	Änderungsdatum	Typ	Größe
FreeCan_EM_V36	07.05.2013 14:11	Dateiordner	
Can1ToCan2.csv	07.05.2013 14:14	Microsoft Excel-C...	147 KB

Dateiname: Can1ToCan2.csv

CSV Files (*.csv)

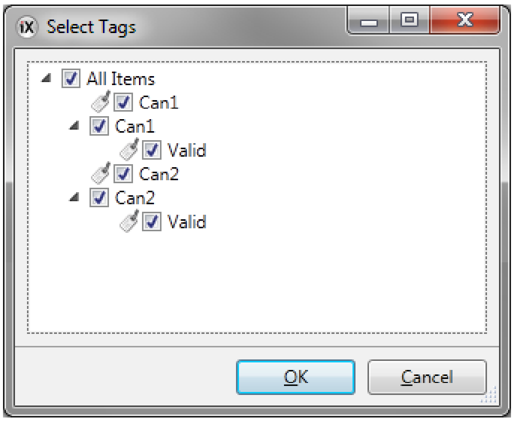
Select your **Can2ToCan2.csv** file (make sure that you have selected the correct folder) and press ***Open***.



Import module: FreeCAN import format

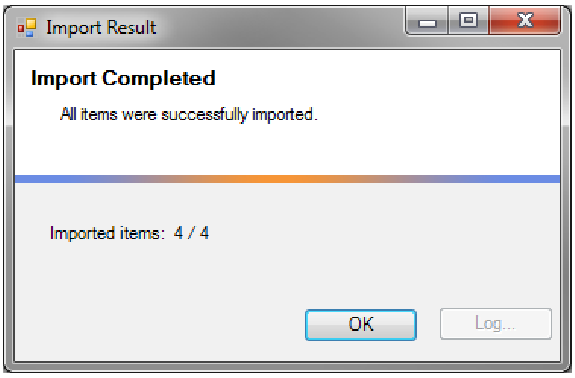
Filename: D:\iX\- Projects 2\FreeCanTest\Project Files\Can1ToCan2.csv

If the *.csv file is a valid import file it’s possible to press ***Import***.



- ☑ All Items
  - ☑ Can1
  - ☑ Can1
    - ☑ Valid
  - ☑ Can2
  - ☑ Can2
    - ☑ Valid

The tag tree view opens. Select ***All Items*** and press ***OK***.



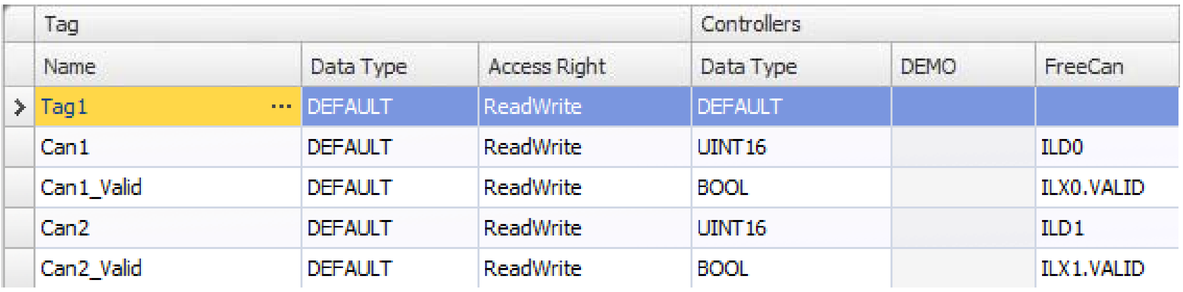
Import Completed

All items were successfully imported.

Imported items: 4 / 4

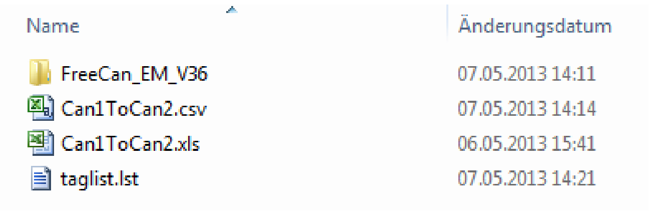
Hint:
Use the option ”*merge*“ if you import the tags for the second time.

Now **4 tags**/items are imported to the iX Tag Editor.



	Tag			Controllers		
	Name	Data Type	Access Right	Data Type	DEMO	FreeCan
>	Tag1	DEFAULT	ReadWrite	DEFAULT		
	Can1	DEFAULT	ReadWrite	UINT16		ILD0
	Can1_Valid	DEFAULT	ReadWrite	BOOL		ILX0.VALID
	Can2	DEFAULT	ReadWrite	UINT16		ILD1
	Can2_Valid	DEFAULT	ReadWrite	BOOL		ILX1.VALID

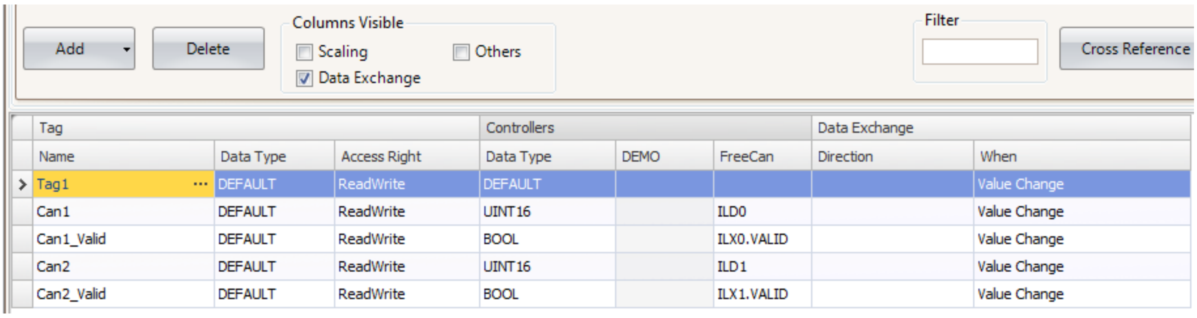
This import produces in parallel a file named ***taglist.lst***, which is a tag reference list.
This reference list will be loaded automatically into the CAN module when the driver loads.



Name	Änderungsdatum
FreeCan_EM_V36	07.05.2013 14:11
Can1ToCan2.csv	07.05.2013 14:14
Can1ToCan2.xls	06.05.2013 15:41
taglist.lst	07.05.2013 14:21

2.10 iX Tag Configuration (Controller Data Exchange)

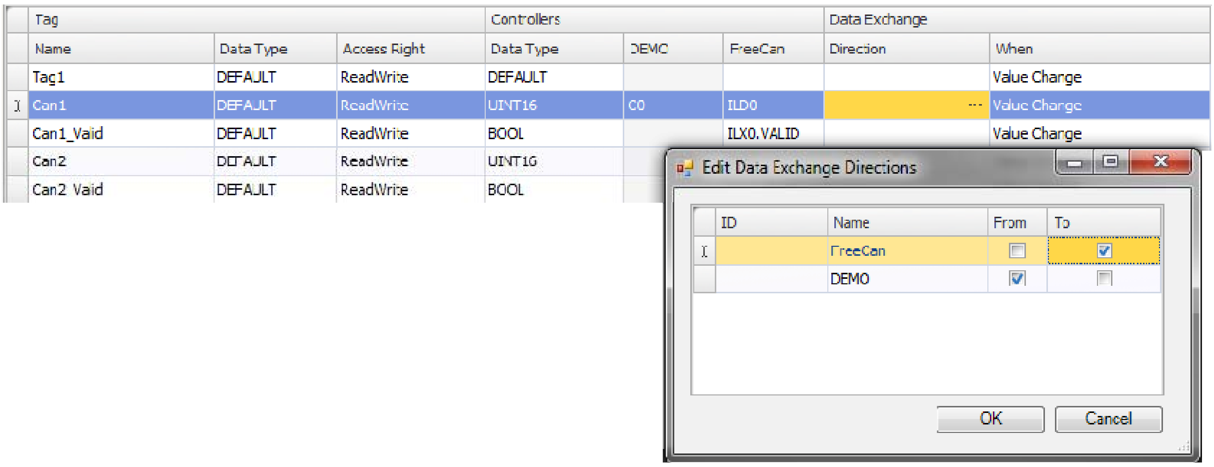
Enable the Data Exchange columns by clicking the checkbox ***Data Exchange***.



Columns Visible: ☐ Scaling ☐ Others ☑ Data Exchange

Tag			Controllers			Data Exchange	
Name	Data Type	Access Right	Data Type	DEMO	FreeCan	Direction	When
Tag1	DEFAULT	ReadWrite	DEFAULT				Value Change
Can1	DEFAULT	ReadWrite	UINT16		ILD0		Value Change
Can1_Valid	DEFAULT	ReadWrite	BOOL		ILX0.VALID		Value Change
Can2	DEFAULT	ReadWrite	UINT16		ILD1		Value Change
Can2_Valid	DEFAULT	ReadWrite	BOOL		ILX1.VALID		Value Change

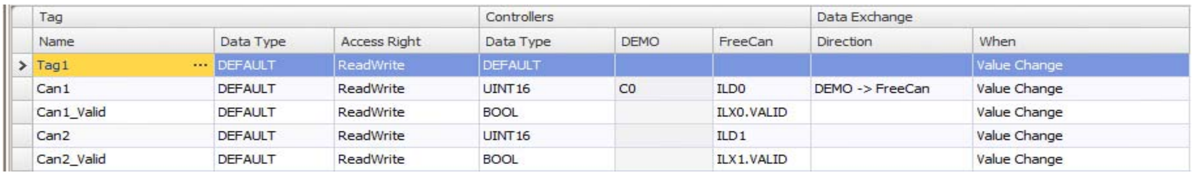
Enter ***C0 (Counter Tag)*** in the DEMO Controller column for the tag ***Can1.***
Then press the ***Direction Browse Button*** for tag ***Can1.***



Tag			Controllers			Data Exchange	
Name	Data Type	Access Right	Data Type	DEMO	FreeCan	Direction	When
Tag1	DEFAULT	ReadWrite	DEFAULT				Value Change
Can1	DEFAULT	ReadWrite	UINT16	C0	ILD0	...	Value Change
Can1_Valid	DEFAULT	ReadWrite	BOOL		ILX0.VALID		Value Change
Can2	DEFAULT	ReadWrite	UINT16				
Can2_Valid	DEFAULT	ReadWrite	BOOL				

ID	Name	From	To
	FreeCan	☐	☑
	DEMO	☑	☐

Set the Checkboxes: ***FromDEMO*** and ***ToFreeCAN***.

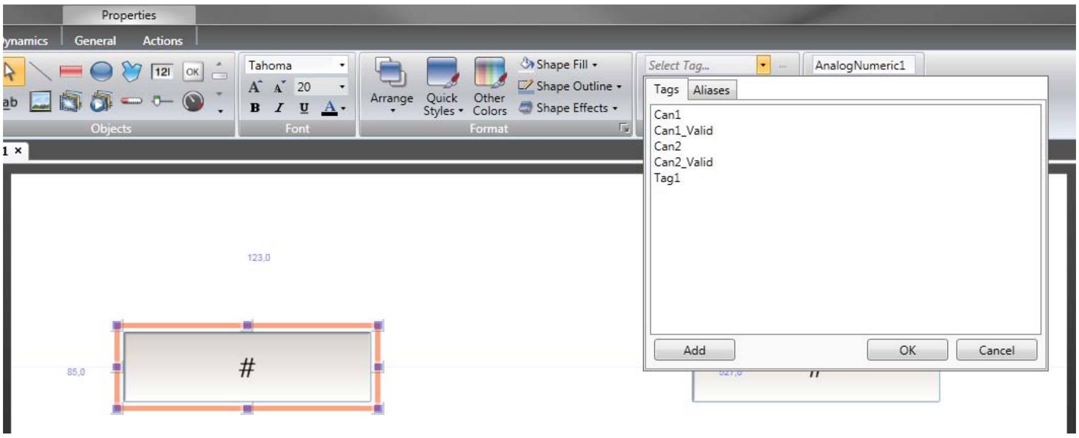


Tag			Controllers			Data Exchange	
Name	Data Type	Access Right	Data Type	DEMO	FreeCan	Direction	When
Tag1	DEFAULT	ReadWrite	DEFAULT				Value Change
Can1	DEFAULT	ReadWrite	UINT16	C0	ILD0	DEMO -> FreeCan	Value Change
Can1_Valid	DEFAULT	ReadWrite	BOOL		ILX0.VALID		Value Change
Can2	DEFAULT	ReadWrite	UINT16		ILD1		Value Change
Can2_Valid	DEFAULT	ReadWrite	BOOL		ILX1.VALID		Value Change

That means that counter Tag ***C0*** increments the value of tag ***Can1*** in a cycle.

2.11 Connecting the Can Tags to Controls

Place two Controls ***Analog Numeric*** on your screen and connect them with tags Can1 and Can2. This can be done in ***Tag/Security.***



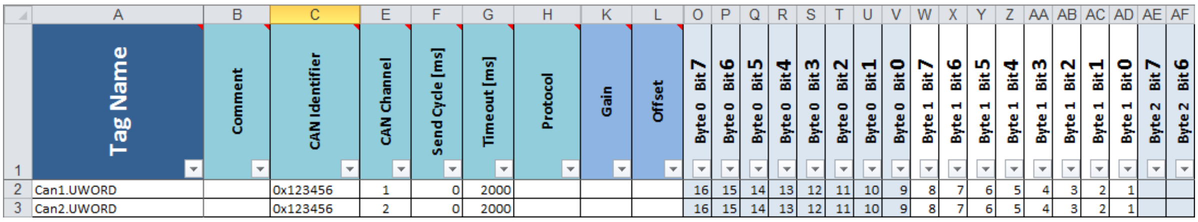
Tags | Aliases

Can1
Can1_Valid
Can2
Can2_Valid
Tag1

Add | OK | Cancel

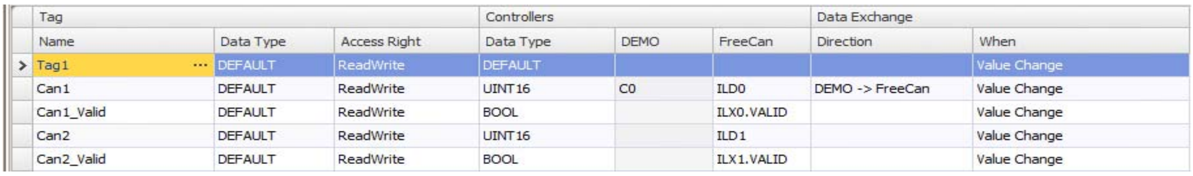
2.12 Showing Can tag timeouts

As we remember, we gave our Can1 and Can2 tags a timeout value of 2000 ms.
Doing that, the tags will be automatically doubled on import:
1.) the value itself
2.) a so called **valid bit**.



	A	B	C	E	F	G	H	K	L	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF
1	Tag Name	Comment	CAN Identifier	CAN Channel	Send Cycle [ms]	Timeout [ms]	Protocol	Gain	Offset	Byte 0 Bit 7	Byte 0 Bit 6	Byte 0 Bit 5	Byte 0 Bit 4	Byte 0 Bit 3	Byte 0 Bit 2	Byte 0 Bit 1	Byte 0 Bit 0	Byte 1 Bit 7	Byte 1 Bit 6	Byte 1 Bit 5	Byte 1 Bit 4	Byte 1 Bit 3	Byte 1 Bit 2	Byte 1 Bit 1	Byte 1 Bit 0	Byte 2 Bit 7	Byte 2 Bit 6
2	Can1.UWORD		0x123456	1	0	2000				16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		
3	Can2.UWORD		0x123456	2	0	2000				16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		

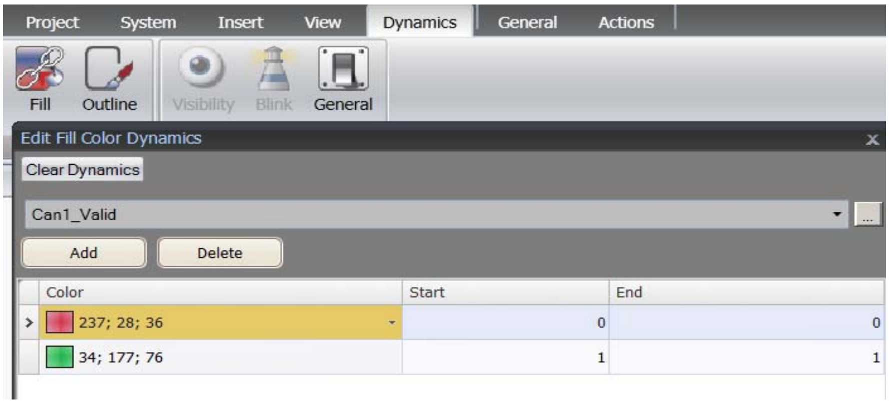
After importing the *.csv parameter file we get the timeout tags: ***[Tag_Valid]***



Tag			Controllers			Data Exchange	
Name	Data Type	Access Right	Data Type	DEMO	FreeCan	Direction	When
Tag1	DEFAULT	ReadWrite	DEFAULT				Value Change
Can1	DEFAULT	ReadWrite	UINT16	C0	ILD0	DEMO -> FreeCan	Value Change
Can1_Valid	DEFAULT	ReadWrite	BOOL		ILX0.VALID		Value Change
Can2	DEFAULT	ReadWrite	UINT16		ILD1		Value Change
Can2_Valid	DEFAULT	ReadWrite	BOOL		ILX1.VALID		Value Change

The valid bit returns ***0*** if the CAN telegram was ***not received*** on ***CAN*** bus ***within timeout*** time.
The valid bit returns ***1*** if the CAN telegram was received on CAN bus ***within timeout*** time.

To visualize the timeout state you could use two controls ***Button*** with an option ***Dynamics* ➔ *Fill:***



Can1_Valid

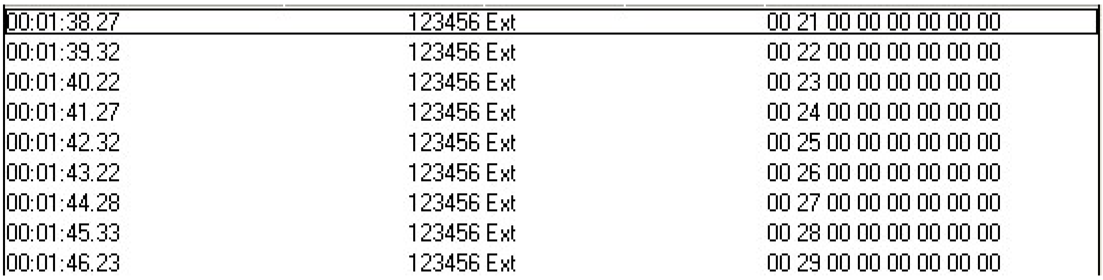
Color	Start	End
🟥 237; 28; 36	0	0
🟩 34; 177; 76	1	1

Enter ***0*** for ***Start*** and ***End*** value and change the color to ***red***.
Enter ***1*** for ***Start*** and ***End*** value and change the color to ***green***.

Now there are two coloured fields that get red, if the Can1 or Can2 tag was not received within 2 seconds.

2.13 iX Runtime

On CAN Bus you can get the following information (Example recorded with a CAN analyzer):

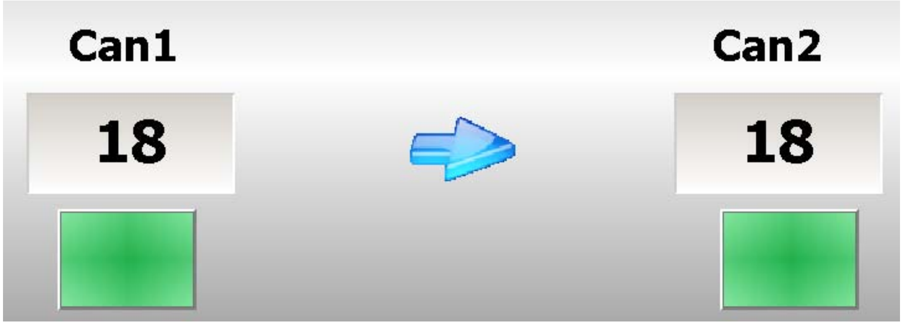


```
00:01:38.27          123456 Ext          00 21 00 00 00 00 00 00
00:01:39.32          123456 Ext          00 22 00 00 00 00 00 00
00:01:40.22          123456 Ext          00 23 00 00 00 00 00 00
00:01:41.27          123456 Ext          00 24 00 00 00 00 00 00
00:01:42.32          123456 Ext          00 25 00 00 00 00 00 00
00:01:43.22          123456 Ext          00 26 00 00 00 00 00 00
00:01:44.28          123456 Ext          00 27 00 00 00 00 00 00
00:01:45.33          123456 Ext          00 28 00 00 00 00 00 00
00:01:46.23          123456 Ext          00 29 00 00 00 00 00 00
```

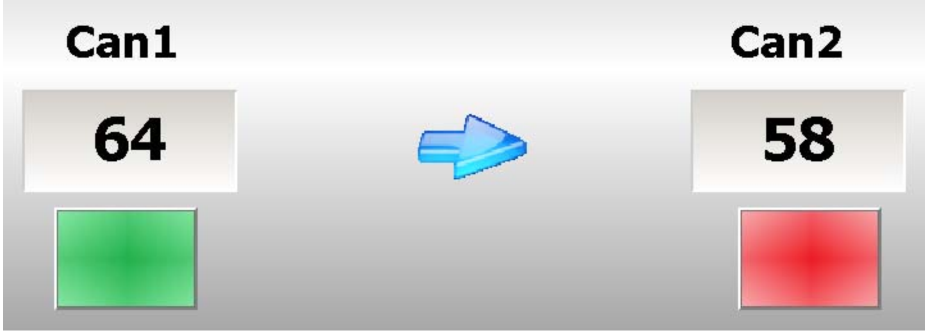
Every second a CAN telegram with header 0x123456 is written, and the value is incremented by 1.

Example of an iX project where we see Can1 and Can2 tags.

CAN1 and CAN2 channels are connected to each other and communication works.
The incremented value of CAN1 will be transferred and will be the same at CAN2.
Also both valid bits are true = green.

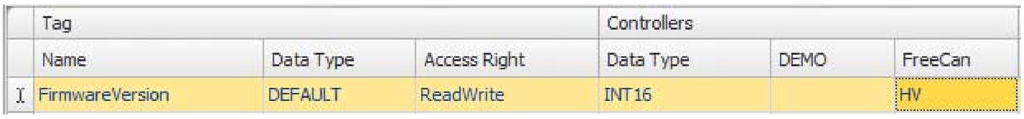


CAN1 and CAN2 channel are ***not*** connected to each other and communication does ***not*** work.
The incremented value of CAN1 will not be transferred, and will not be the same at CAN2.
Can2 valid bit is false = red.



2.14 Showing FreeCAN Firmware Version

Change Tag1 or add a new Tag ***Firmware Version***. Enter ***HV*** in the address column*:*

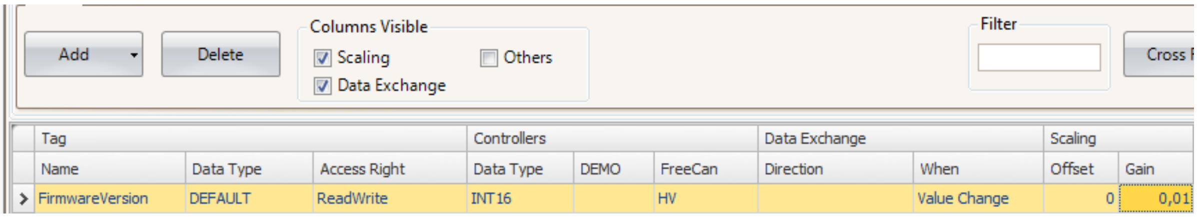


Tag			Controllers		
Name	Data Type	Access Right	Data Type	DEMO	FreeCan
FirmwareVersion	DEFAULT	ReadWrite	INT16		HV

Add an additional Analog Numeric control on the screen and connect it to the ***Firmware Version*** tag.

The driver will deliver a value like 2900, this means Firmware version 29.
If you want to show the basic version number (=29), switch on ***Scaling*** for the tags and insert the ***value 0,01*** for the tag in ***Gain***.

The value will automatically be divided by 100:



Columns Visible
☑ Scaling ☐ Others
☑ Data Exchange

Tag			Controllers			Data Exchange		Scaling	
Name	Data Type	Access Right	Data Type	DEMO	FreeCan	Direction	When	Offset	Gain
FirmwareVersion	DEFAULT	ReadWrite	INT16		HV		Value Change	0	0,01

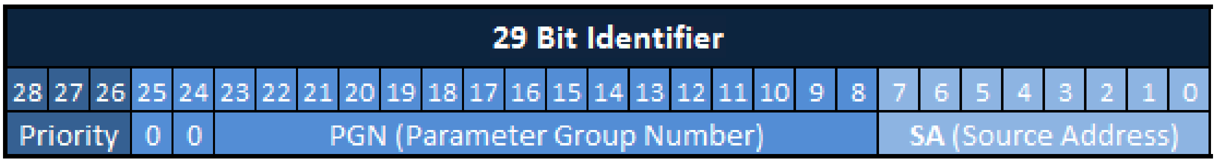
3 Special protocols with FreeCAN

The FreeCAN driver and module supports different Can protocols.
In the following chapter 3 different protocols are described:

Section 3.1: J1939 with FreeCAN
Section 0: NMEA 2000 with FreeCAN
Section 3.3: CanOpen with FreeCAN

3.1 J1939

J1939 is a 29bit (extended header) protocol on 250kbaud.



29 Bit Identifier																												
28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Priority			0	0	PGN (Parameter Group Number)																SA (Source Address)							

(Priority: **3 bits**) + (reserved: **2 bits**) + (PGN: **16 bits**) + (SA: **8 bits**)

Note:

- Bit 24: **Data Page (0** for **J1939)**
- Bit 25: **reserved (= 0)**

J1939 provides many predefined tags in different variations.
There are 2 different priorities: 3 and 6.
For implementation you should have the definition sheet for J1939 parameters.
Priority, PGN and address will be entered in hexadecimal notation.

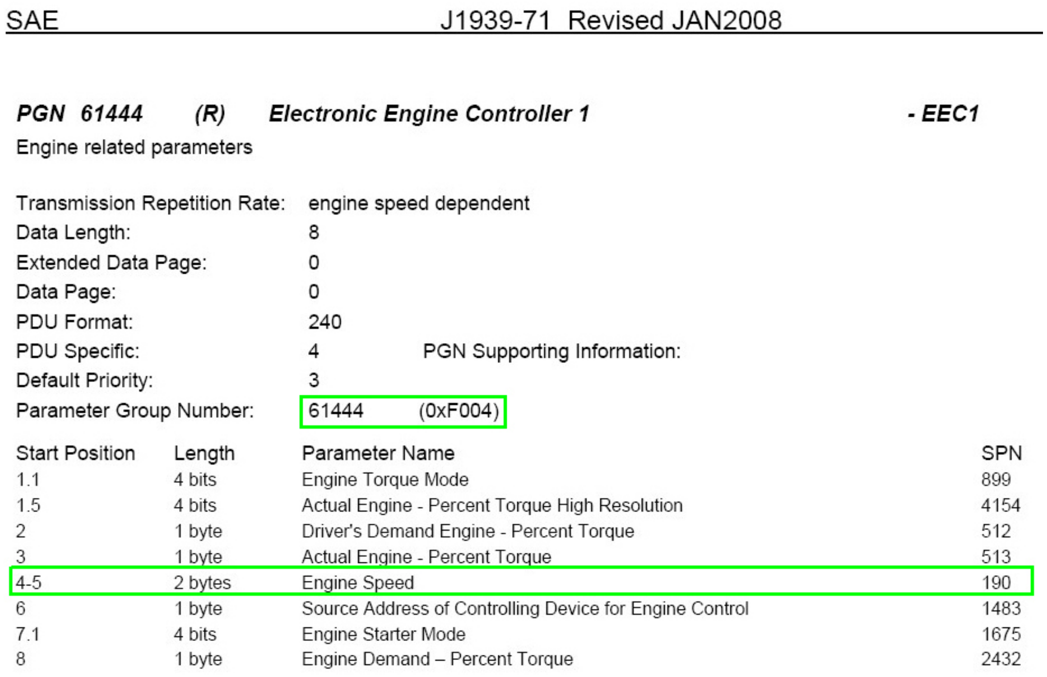
Definition:

- **PGN** (**P**arameter **G**roup **N**umber)
  Defines the header of the CAN J1939 telegram

- **SPN** (**S**uspect **P**arameter **N**umber)
  Defines the different variables in the data of a telegram

- **SA** (**S**ource **A**ddress)
  Specifies the address of the sending CAN device. Sometimes it is called transmitter address.

Example ***Engine Speed*** (like is defined by SAE1939-71 document):



SAE J1939-71 Revised JAN2008

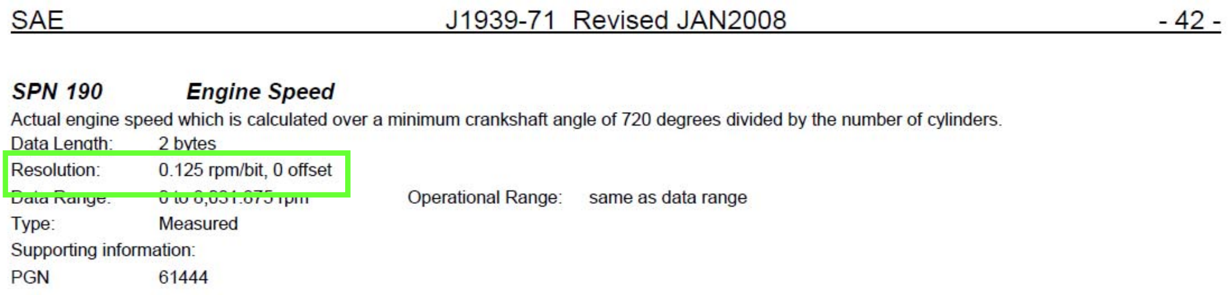
PGN 61444 (R) Electronic Engine Controller 1 - EEC1

Engine related parameters

Transmission Repetition Rate:	engine speed dependent	
Data Length:	8	
Extended Data Page:	0	
Data Page:	0	
PDU Format:	240	
PDU Specific:	4	PGN Supporting Information:
Default Priority:	3	
Parameter Group Number:	61444 (0xF004)	

Start Position	Length	Parameter Name	SPN
1.1	4 bits	Engine Torque Mode	899
1.5	4 bits	Actual Engine - Percent Torque High Resolution	4154
2	1 byte	Driver's Demand Engine - Percent Torque	512
3	1 byte	Actual Engine - Percent Torque	513
4-5	2 bytes	Engine Speed	190
6	1 byte	Source Address of Controlling Device for Engine Control	1483
7.1	4 bits	Engine Starter Mode	1675
8	1 byte	Engine Demand – Percent Torque	2432

Engine Speed:	PGN 61444 (decimal) = **0xF004** (hexadecimal)
Used Bytes:	Byte **4** + Byte **5**
SPN(Suspect Parameter Number):	**190**



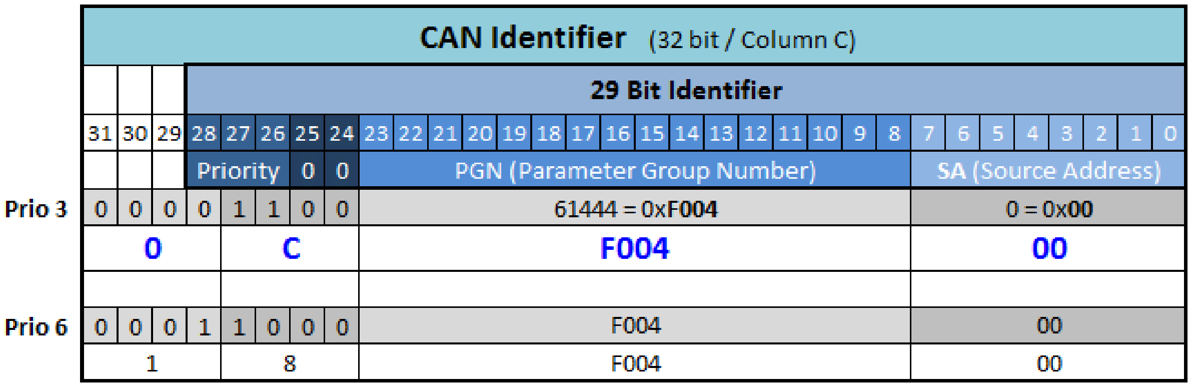
SAE J1939-71 Revised JAN2008 - 42 -

SPN 190 Engine Speed

Actual engine speed which is calculated over a minimum crankshaft angle of 720 degrees divided by the number of cylinders.

Data Length:	2 bytes		
Resolution:	0.125 rpm/bit, 0 offset		
Data Range:	0 to 8,031.875 rpm	Operational Range:	same as data range
Type:	Measured		
Supporting information:			
PGN	61444		

Assumed, the transmitter **address**= **0** and the **priority** =**3**, the header is:



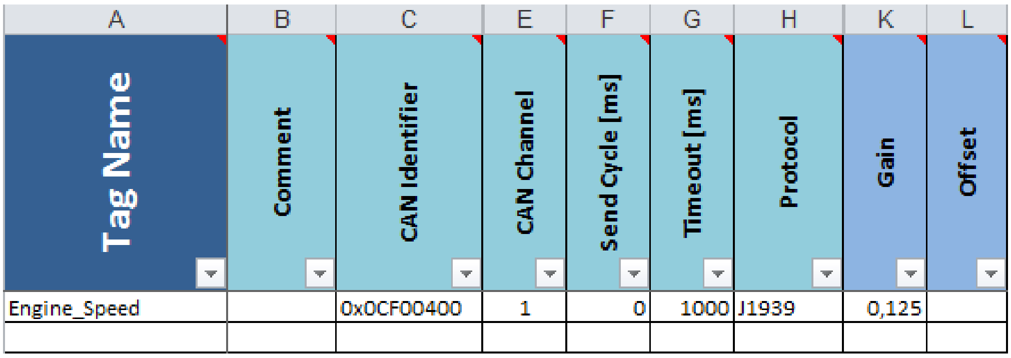
CAN Identifier (32 bit / Column C)

				29 Bit Identifier																												
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Priority			0	0	PGN (Parameter Group Number)																SA (Source Address)							
Prio 3	0	0	0	0	1	1	0	0	61444 = 0xF004																0 = 0x00							
	0				C				F004																00							
Prio 6	0	0	0	1	1	0	0	0	F004																00							
	1				8				F004																00							

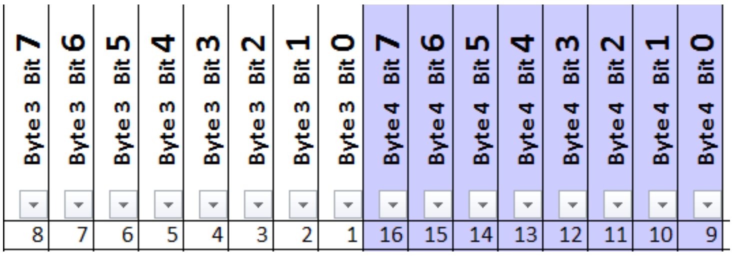
Finally we define the tag ***Engine_Speed*** in the Excel sheet.

Hints:

1. SAE counts up from 1, we have to use byte 3 and byte 4 in the Excel sheet.
2. Resolution 0.125 rpm/bit with 0 offset.



A	B	C	E	F	G	H	K	L
Tag Name	Comment	CAN Identifier	CAN Channel	Send Cycle [ms]	Timeout [ms]	Protocol	Gain	Offset
Engine_Speed		0x0CF00400	1	0	1000	J1939	0,125	



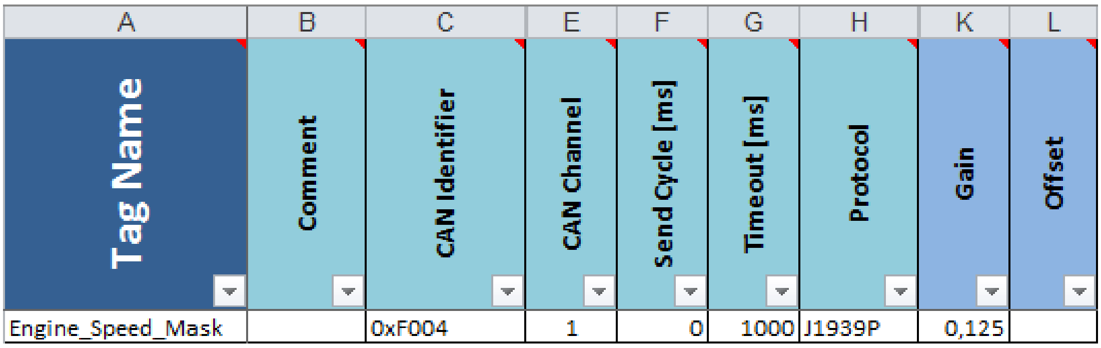
Byte 3 Bit 7	Byte 3 Bit 6	Byte 3 Bit 5	Byte 3 Bit 4	Byte 3 Bit 3	Byte 3 Bit 2	Byte 3 Bit 1	Byte 3 Bit 0	Byte 4 Bit 7	Byte 4 Bit 6	Byte 4 Bit 5	Byte 4 Bit 4	Byte 4 Bit 3	Byte 4 Bit 2	Byte 4 Bit 1	Byte 4 Bit 0
8	7	6	5	4	3	2	1	16	15	14	13	12	11	10	9

Summary:

- ***Engine_speed*** will be the imported tag name.
- ***0x0CF00400*** is the CAN header adress.
- Can bus is on ***CAN1*** plug (channel 1).
- Timeout will produce a *valid bit tag* =***Engine_Speed_Valid*** on import.
- "J1939" in Protocol will fill unused data bits with 1 (default).
- ***Gain*** is ***0,125*** for SPN190.
- J1939 is ***Intel*** formatted, with low byte on left side (Byte 3), high on the right (Byte4).

Hint: Unknown transmit address

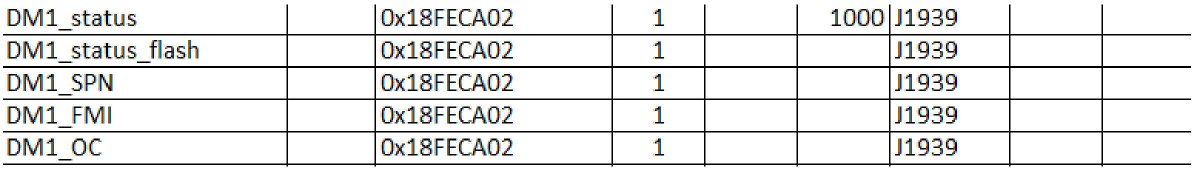
Now you may not know the priority or transmitter address of your J1939 producer. In this case you only define the PGN as header and put "J1939P" in Protocol. Then the priority and transmitter address is ignored, and the first fitting PGN is taken.



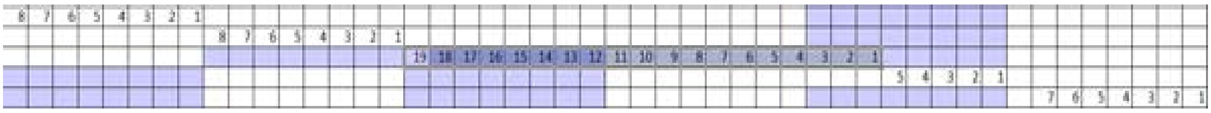
A	B	C	E	F	G	H	K	L
Tag Name	Comment	CAN Identifier	CAN Channel	Send Cycle [ms]	Timeout [ms]	Protocol	Gain	Offset
Engine_Speed_Mask		0xF004	1	0	1000	J1939P	0,125	

Hint: J1939 device alive
If you are not sure if you receive telegrams from a J1939 device, you may use the HLI address in the iX tag definition. HLI04,1 will search for transmitter address 0x04 within 1 second timeout. This results in “1” if transmitter 0x04 sends, but results to “0” if no telegrams within 1 second are received.
HLI22,2 will search for address 0x22 = 34 decimal with timeout 2 seconds.

Hint: DM1 telegrams
DM1 telegrams are error messages by J1939 devices.
DM1 consist of status,SPN,FMI and OC.
For each part we define a tag plus one timeout.
If status gets >0 we have an error message to record.

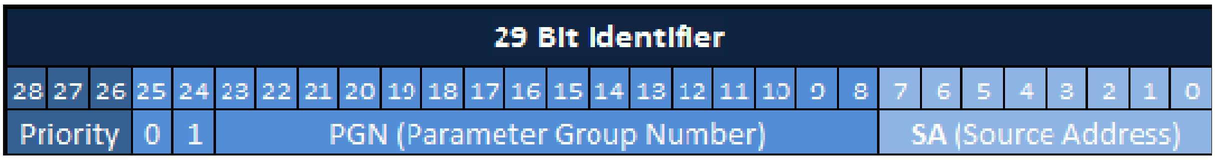


DM1_status		0x18FECA02	1		1000	J1939		
DM1_status_flash		0x18FECA02	1			J1939		
DM1_SPN		0x18FECA02	1			J1939		
DM1_FMI		0x18FECA02	1			J1939		
DM1_OC		0x18FECA02	1			J1939		



3.2 NMEA 2000

NMEA 2000 is basically J1939 using Data Page 1 instead of 0.
However it has variable definitions of its own. For this we extended the protocols.



<table>
<tr><th colspan="29">29 Bit Identifier</th></tr>
<tr><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="3">Priority</td><td>0</td><td>1</td><td colspan="16">PGN (Parameter Group Number)</td><td colspan="8">SA (Source Address)</td></tr>
</table>

(Priority: **3 bits**) + (reserved: **2 bits**) + (PGN: **16 bits**) + (SA: **8 bits**)

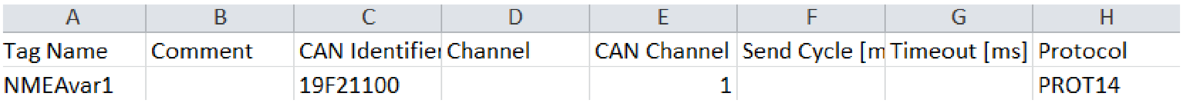
Note:

- Bit 24: **Data Page (1**for**NMEA 2000)**
- Bit 25: **reserved (= 0)**

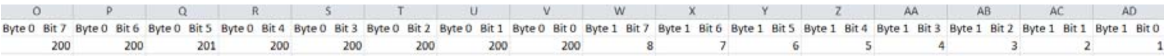
There are 2 different priorities: 3 and 6.
Hence NMEA address header start with 0x19 (high priority) or 0x0D (low priority).These bytes consist of the bits 24 to 31. The bits 29 to 31 are not part of the Bit Identifier but they have to be added to get a full byte. The bits 29 to 31 are always 0.

There are NMEA variables that have a variable inside the data, which extends the header. Normally this would cause a problem to separate the variables. For this we inserted a special protocol "PROT14". Together with the numbers 200 and 201 in a bit column in the Excel sheet it is possible to filter the variable. The number 200 represent a 0 and the number 201 represents a 1.

Example:



A	B	C	D	E	F	G	H
Tag Name	Comment	CAN Identifiei	Channel	CAN Channel	Send Cycle [m	Timeout [ms]	Protocol
NMEAvar1		19F21100		1			PROT14



O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD
Byte 0 Bit 7	Byte 0 Bit 6	Byte 0 Bit 5	Byte 0 Bit 4	Byte 0 Bit 3	Byte 0 Bit 2	Byte 0 Bit 1	Byte 0 Bit 0	Byte 1 Bit 7	Byte 1 Bit 6	Byte 1 Bit 5	Byte 1 Bit 4	Byte 1 Bit 3	Byte 1 Bit 2	Byte 1 Bit 1	Byte 1 Bit 0
200	200	201	200	200	200	200	200	8	7	6	5	4	3	2	1

With this filter applied we get the value of tag NMEAvar1 if telegram 0x19F21100 (PGN 1F211) has the value 0x20 in the first byte.

3.3 Simple CANOpenslave

CANOpen is a dedicated complex bus protocol. With FreeCAN we can ***only*** build a ***simple CanOpen slave***.
The simple CanOpen Slave functionality is supported in FreeCAN driver version **5.00.35**and CiX firmware version **29** or higher.

Simple CanOpen slave with FreeCAN driver
For first tests we build 2 iX projects to run CanOpen communication with 2 TxA panels.

- ***FreeCAN_08_CanOpen***
  A simple CanOpen slave 7, 1 TPDO, 1 RPDO on FreeCAN driver.
  Use FreeCAN_08.EDS in folder /Project Files.

- ***canOpen_master_08***
  A CanOpen master 1 with automatic config, 1 TPDO, 1 RPDO on CanOpen driver (Firmware CanOpenV6 needed).

Start both TxA panels together. At start the master will poll the SDO settings from slave and start a data exchange. As we connected both iX projects to the demo driver counter, you will see the data running.

3.3.1 Simple CanOpen slave project

With FreeCAN firmware V29 (in driver 5.00.36) we implemented a new protocol "**CanOpen**", which allows you to use the FreeCAN driver as a simple CanOpen slave.
We base this document on the FreeCAN manual (in the FreeCAN driver).
So read them first to understand the way we handle things.
Also CanOpen spec "DS401" should be known to you.

Please be aware that you use this driver on your own risk, as we did not certificate it. But it should be good enough to meet the basic needs of a CanOpen slave communication.
If you want to use a certificated CanOpen master or slave with our panel, we offer a special driver and firmware for it. Please contact your local Beijer dealer.

As an example we made a iX project "FreeCAN_08_CanOpen", which allows you to make first steps.
The SDO data for the slave have to be inserted in the Excel sheet "CanOpenSlave"(in folder: /Project Files), so the FreeCAN module is a PREDEFINED CanOpen slave. The SDOs **cannot** be loaded over CAN bus. For inserting the slave in a CanOpen system, use FreeCAN_08.EDS in Folder /Project files.
NMT and heartbeat production/consumption is supported.

The RPDOs (input) are stored in the CAN module and can be used as read tags.
The TPDOs are written on tag change in iX project. The telegrams can be send cyclic.

The example project comes with a useable EDS, which allows you to build a quick CanOpen network. You may change the Excel sheet and the EDS to your demands. The slave number in the example project is set to 7. If you want to change it, you may change controller setting “Can station address” in FreeCAN driver. This changes the reaction on NMT telegram.

Change TPDO and RPDO telegram ID in the Excel sheet.
Change TPDO mapping (SDO1800,1) and RPDO mapping (SDO1A00,1)

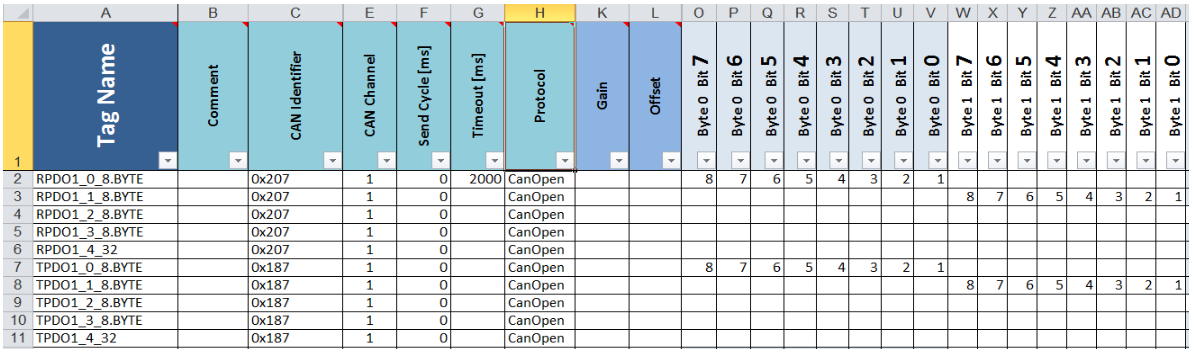
3.3.2 Define the PDOs

The PDOs (process data objects) telegrams are the exchanged data over CanOpen (only in Operational state).

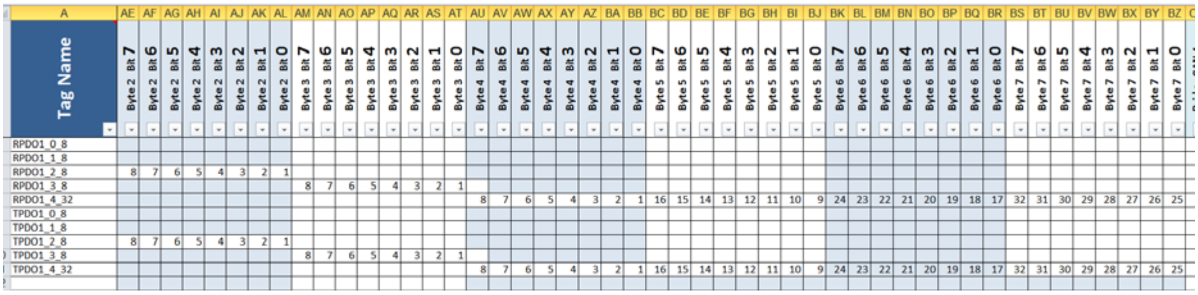
To define the PDOs, the tags for iX program have to be defined. Set “CanOpen” in Protocol row to mark it as CanOpen telegrams. The name extension “.BYTE” (RPDO_0_8.BYTE) will import the tag as a INT16 tag, which is the nearest iX type to Byte.

As CanOpen defaults, the PDO IDs in CanOpen are set like this:
RPDO1 = 0x200 + Slave number
TPDO1 = 0x180 + Slave number
RPDO2 = 0x300 + Slave number
TPDO2 = 0x280 + Slave number
So here for our slave no 7 we get: RPDO1= 0x207, TPDO1=0x187

In this example we defined 1 TPDO1 and 1 RPDO1. The mapping for both is 4 tag bytes and one doubleword (32bit) tag. Below you see the Excel setting for the 5 RPDO and 5 TPDO tags.
Here RPDO1_0_8 has a additional value tag (RPDO1_0_8_Valid) produced by the Timeout setting. This tag can be used to test, if data over RPDO comes in.



	A	B	C	E	F	G	H	K	L	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD
1	Tag Name	Comment	CAN Identifier	CAN Channel	Send Cycle [ms]	Timeout [ms]	Protocol	Gain	Offset	Byte 0 Bit 7	Byte 0 Bit 6	Byte 0 Bit 5	Byte 0 Bit 4	Byte 0 Bit 3	Byte 0 Bit 2	Byte 0 Bit 1	Byte 0 Bit 0	Byte 1 Bit 7	Byte 1 Bit 6	Byte 1 Bit 5	Byte 1 Bit 4	Byte 1 Bit 3	Byte 1 Bit 2	Byte 1 Bit 1	Byte 1 Bit 0
2	RPDO1_0_8.BYTE		0x207	1	0	2000	CanOpen			8	7	6	5	4	3	2	1								
3	RPDO1_1_8.BYTE		0x207	1	0		CanOpen											8	7	6	5	4	3	2	1
4	RPDO1_2_8.BYTE		0x207	1	0		CanOpen																		
5	RPDO1_3_8.BYTE		0x207	1	0		CanOpen																		
6	RPDO1_4_32		0x207	1	0		CanOpen																		
7	TPDO1_0_8.BYTE		0x187	1	0		CanOpen			8	7	6	5	4	3	2	1								
8	TPDO1_1_8.BYTE		0x187	1	0		CanOpen											8	7	6	5	4	3	2	1
9	TPDO1_2_8.BYTE		0x187	1	0		CanOpen																		
10	TPDO1_3_8.BYTE		0x187	1	0		CanOpen																		
11	TPDO1_4_32		0x187	1	0		CanOpen																		



A	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ	BA	BB
Tag Name	Byte 2 Bit 7	Byte 2 Bit 6	Byte 2 Bit 5	Byte 2 Bit 4	Byte 2 Bit 3	Byte 2 Bit 2	Byte 2 Bit 1	Byte 2 Bit 0	Byte 3 Bit 7	Byte 3 Bit 6	Byte 3 Bit 5	Byte 3 Bit 4	Byte 3 Bit 3	Byte 3 Bit 2	Byte 3 Bit 1	Byte 3 Bit 0	Byte 4 Bit 7	Byte 4 Bit 6	Byte 4 Bit 5	Byte 4 Bit 4	Byte 4 Bit 3	Byte 4 Bit 2	Byte 4 Bit 1	Byte 4 Bit 0
RPDO1_0_8																								
RPDO1_1_8																								
RPDO1_2_8	8	7	6	5	4	3	2	1																
RPDO1_3_8									8	7	6	5	4	3	2	1								
RPDO1_4_32																	8	7	6	5	4	3	2	1
TPDO1_0_8																								
TPDO1_1_8																								
TPDO1_2_8	8	7	6	5	4	3	2	1																
TPDO1_3_8									8	7	6	5	4	3	2	1								
TPDO1_4_32																	8	7	6	5	4	3	2	1

A	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN	BO	BP	BQ	BR	BS	BT	BU	BV	BW	BX	BY	BZ
Tag Name	Byte 5 Bit 7	Byte 5 Bit 6	Byte 5 Bit 5	Byte 5 Bit 4	Byte 5 Bit 3	Byte 5 Bit 2	Byte 5 Bit 1	Byte 5 Bit 0	Byte 6 Bit 7	Byte 6 Bit 6	Byte 6 Bit 5	Byte 6 Bit 4	Byte 6 Bit 3	Byte 6 Bit 2	Byte 6 Bit 1	Byte 6 Bit 0	Byte 7 Bit 7	Byte 7 Bit 6	Byte 7 Bit 5	Byte 7 Bit 4	Byte 7 Bit 3	Byte 7 Bit 2	Byte 7 Bit 1	Byte 7 Bit 0
RPDO1_0_8																								
RPDO1_1_8																								
RPDO1_2_8																								
RPDO1_3_8																								
RPDO1_4_32	16	15	14	13	12	11	10	9	24	23	22	21	20	19	18	17	32	31	30	29	28	27	26	25
TPDO1_0_8																								
TPDO1_1_8																								
TPDO1_2_8																								
TPDO1_3_8																								
TPDO1_4_32	16	15	14	13	12	11	10	9	24	23	22	21	20	19	18	17	32	31	30	29	28	27	26	25

3.3.3 Define the SDOs

With the definition of the SDOs the CanOpen configuration of the slave can be set. In the Excel sheet you define the answers of the slave to the requests of the master. Therefore the first data byte 0 holds the respond-code 0x43 = SDO respond with 4 data bytes. Byte 1 to 3 are the SDO index and subindex, inserted by the firmware, so you do not have to define it. Byte 4 to 7 are the SDO data. Read the SDO data from right to left, as in CanOpen the data come Intel-like, which is low, high (byte 4 is the lowest byte, byte 7 the highest).
All “not defined” SDOs are automatically denied on master request, you do not have to do anything.
For “CanOpen” we have a special Excel sheet agreement: “1” is a fixed on bit, “0” is a fixed off bit.
So 10010011 = 0x93 fixed.

It is important that you understand, that this slave is NOT configurable (writeable) over Can Bus. All features are fixed in the Excel list. But the slave responds to SDO uploads. It is to your responsibility to have SDOs and PDOs matching each other.
There are some basic SDOs, that an I/O slave have to bring along (see CanOpen DSP401).
SDO 1000,0 (Slave Type).
Read as SDO 1000, subindex 0. Here we return 0,0,0,0 which means: nothing special.

SDO 1018,0 (Manufacturer).
Here we return 01000145 which is the Beijer manufacturer code.

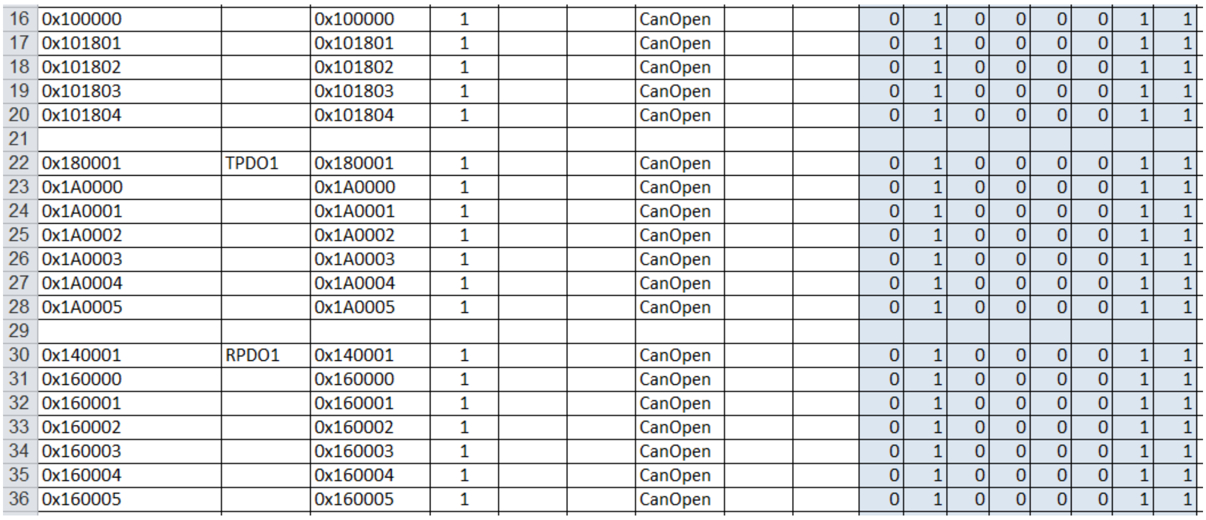
SDO 1800,1 (TPDO1) = 0x40000187 = TPDO1 on Canheader 0x187, no RTR
SDO 1A00,0 (TPDO1 mapping) = 0x40000105 = 5 objects in the mapping
SDO 1A00,1 (TPDO1 mapping obj1) = 0x64000108 = obj 6400 (byte input),1 with 8 bits
SDO 1A00,2 (TPDO1 mapping obj2) = 0x64000208 = obj 6400 (byte input),2 with 8 bits
SDO 1A00,3 (TPDO1 mapping obj3) = 0x64000308 = obj 6400 (byte input),3 with 8 bits
SDO 1A00,4 (TPDO1 mapping obj4) = 0x64000408 = obj 6400 (byte input),4 with 8 bits
SDO 1A00,5 (TPDO1 mapping obj5) = 0x64020520 = obj 6402 (32bit input),5 with 32 bits

SDO 1400,1 (RPDO1) = 0x40000207 = RPDO1 on Canheader 0x207, no RTR
SDO 1600,0 (RPDO1 mapping) = 0x40000105 = 5 objects in the mapping
SDO 1600,1 (RPDO1 mapping obj1) = 0x64100108 = obj 6410,1 with 8 bits
SDO 1600,2 (RPDO1 mapping obj2) = 0x64100208 = obj 6410,2 with 8 bits
SDO 1600,3 (RPDO1 mapping obj3) = 0x64100308 = obj 6410,3 with 8 bits
SDO 1600,4 (RPDO1 mapping obj4) = 0x64100408 = obj 6410,4 with 8 bits
SDO 1600,5 (RPDO1 mapping obj5) = 0x64120520 = obj 6412 (32bit output),5 with 32 bits

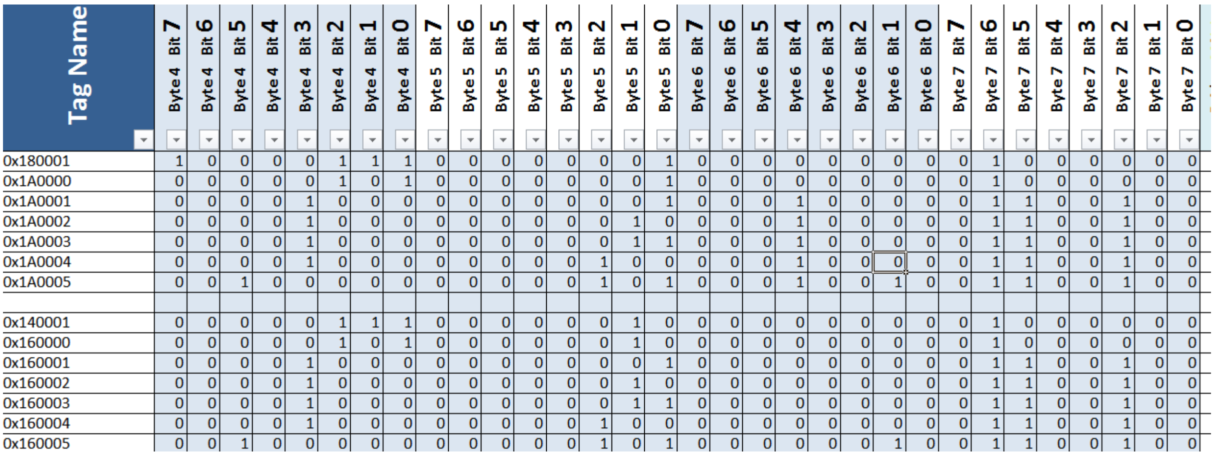
If the master sends SDO 1017,0 (Heartbeat producing time), the slave repeats heartbeat-telegrams on this time rate.
If the master sends SDO 1016,1 (Heartbeat consuming time), the slave timeouts heartbeats and turns to preoperational, if heartbeat from master timeouts.

Definition of the SDOs in the Excel sheet

1000,0	43,-,-,-,0,0,0,0	
1018,1	43,-,-,-,45,1,0,1	
1018,2	43,-,-,-,0,0,0,0	
1018,3	43,-,-,-,0,0,0,0	
1018,4	43,-,-,-,0,0,0,0	
1800,1	43,-,-,-,87,01,0,40	TPDO1 on 187
1A00,0	43,-,-,-,05, 01,0,40	map 5 following objects
1A00,1	43,-,-,-,08,01 ,10,64	map 8 output bit
1A00,2	43,-,-,-,08,01 ,10,64	map 8 output bit
1A00,3	43,-,-,-,08,01 ,10,64	map 8 output bit
1A00,4	43,-,-,-,08,01 ,10,64	map 8 output bit
1A00,5	43,-,-,-,20,01 ,12,64	map 32 output bit
1400,1	43,-,-,-,07,02,0,40	RPDO1 on 207
1600,0	43,-,-,-,05, 01,0,40	map 5 following objects
1600,1	43,-,-,-,08,01 ,00,64	map 8 input bit
1600,2	43,-,-,-,08,01 ,00,64	map 8 input bit
1600,3	43,-,-,-,08,01 ,00,64	map 8 input bit
1600,4	43,-,-,-,08,01 ,00,64	map 8 input bit
1600,5	43,-,-,-,20,01 ,02,64	map 32 input bit



16	0x100000		0x100000	1			CanOpen			0	1	0	0	0	0	1	1
17	0x101801		0x101801	1			CanOpen			0	1	0	0	0	0	1	1
18	0x101802		0x101802	1			CanOpen			0	1	0	0	0	0	1	1
19	0x101803		0x101803	1			CanOpen			0	1	0	0	0	0	1	1
20	0x101804		0x101804	1			CanOpen			0	1	0	0	0	0	1	1
21																	
22	0x180001	TPDO1	0x180001	1			CanOpen			0	1	0	0	0	0	1	1
23	0x1A0000		0x1A0000	1			CanOpen			0	1	0	0	0	0	1	1
24	0x1A0001		0x1A0001	1			CanOpen			0	1	0	0	0	0	1	1
25	0x1A0002		0x1A0002	1			CanOpen			0	1	0	0	0	0	1	1
26	0x1A0003		0x1A0003	1			CanOpen			0	1	0	0	0	0	1	1
27	0x1A0004		0x1A0004	1			CanOpen			0	1	0	0	0	0	1	1
28	0x1A0005		0x1A0005	1			CanOpen			0	1	0	0	0	0	1	1
29																	
30	0x140001	RPDO1	0x140001	1			CanOpen			0	1	0	0	0	0	1	1
31	0x160000		0x160000	1			CanOpen			0	1	0	0	0	0	1	1
32	0x160001		0x160001	1			CanOpen			0	1	0	0	0	0	1	1
33	0x160002		0x160002	1			CanOpen			0	1	0	0	0	0	1	1
34	0x160003		0x160003	1			CanOpen			0	1	0	0	0	0	1	1
35	0x160004		0x160004	1			CanOpen			0	1	0	0	0	0	1	1
36	0x160005		0x160005	1			CanOpen			0	1	0	0	0	0	1	1



Tag Name	Byte 4 Bit 7	Byte 4 Bit 6	Byte 4 Bit 5	Byte 4 Bit 4	Byte 4 Bit 3	Byte 4 Bit 2	Byte 4 Bit 1	Byte 4 Bit 0	Byte 5 Bit 7	Byte 5 Bit 6	Byte 5 Bit 5	Byte 5 Bit 4	Byte 5 Bit 3	Byte 5 Bit 2	Byte 5 Bit 1	Byte 5 Bit 0	Byte 6 Bit 7	Byte 6 Bit 6	Byte 6 Bit 5	Byte 6 Bit 4	Byte 6 Bit 3	Byte 6 Bit 2	Byte 6 Bit 1	Byte 6 Bit 0	Byte 7 Bit 7	Byte 7 Bit 6	Byte 7 Bit 5	Byte 7 Bit 4	Byte 7 Bit 3	Byte 7 Bit 2	Byte 7 Bit 1	Byte 7 Bit 0
0x180001	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0x1A0000	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0x1A0001	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	1	0	0
0x1A0002	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	1	0	0	1	0	0
0x1A0003	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	1	1	0	0	1	0	0
0x1A0004	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	1	0	0
0x1A0005	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	1	1	0	0	1	0	0
0x140001	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0x160000	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0x160001	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
0x160002	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
0x160003	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
0x160004	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
0x160005	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0

3.3.4 Extending the slave with RPDOs

We suppose that you want to change the slaves setting.
Let’s change it to 3 RPDOs with 8 bytes, 4 words and 2 double words.
This means, that the master sends 3 PDO telegrams to our slave.

If you change the Excel sheet, do not forget to save it as ***.csv (MS-DOS)** and import it to your iX project.

RPDO1 on 0x207, RPDO2 on 0x307, RPDO3 on 0x407
Object 1 byte input = 6400
Object 2 byte input = 6401
Object 4 byte input = 6402

So the definition would be:
SDO 1400,1 (RPDO1) = 0x40000207 = RPDO1 on Canheader 0x207, no RTR
SDO 1600,0 (RPDO1 mapping) = 0x40000108 = 8 objects in the mapping
SDO 1600,1 (RPDO1 mapping obj1) = 0x64000108 = obj 6400,1 with 8 bits
SDO 1600,2 (RPDO1 mapping obj2) = 0x64000208 = obj 6400,2 with 8 bits
SDO 1600,3 (RPDO1 mapping obj3) = 0x64000308 = obj 6400,3 with 8 bits
SDO 1600,4 (RPDO1 mapping obj4) = 0x64000408 = obj 6400,4 with 8 bits
SDO 1600,5 (RPDO1 mapping obj5) = 0x64000508 = obj 6400,5 with 8 bits
SDO 1600,6 (RPDO1 mapping obj6) = 0x64000608 = obj 6400,6 with 8 bits
SDO 1600,7 (RPDO1 mapping obj7) = 0x64000708 = obj 6400,7 with 8 bits
SDO 1600,8 (RPDO1 mapping obj8) = 0x64000808 = obj 6400,8 with 8 bits

SDO 1401,1 (RPDO2) = 0x40000307 = RPDO2 on Canheader 0x307, no RTR
SDO 1601,0 (RPDO2 mapping) = 0x40000104 = 4 objects in the mapping
SDO 1601,1 (RPDO2 mapping obj1) = 0x64010108 = obj 6401,1 with 16 bits
SDO 1601,2 (RPDO2 mapping obj2) = 0x64010208 = obj 6401,2 with 16 bits
SDO 1601,3 (RPDO2 mapping obj3) = 0x64010308 = obj 6401,3 with 16 bits
SDO 1601,4 (RPDO2 mapping obj4) = 0x64010408 = obj 6401,4 with 16 bits

SDO 1402,1 (RPDO3) = 0x40000407 = RPDO3 on Canheader 0x407, no RTR
SDO 1602,0 (RPDO3 mapping) = 0x40000102 = 2 objects in the mapping
SDO 1602,1 (RPDO3 mapping obj1) = 0x64020120 = obj 6402,1 with 32 bits
SDO 1602,2 (RPDO3 mapping obj2) = 0x64020220 = obj 6402,2 with 32 bits

3.3.5 Extending the slave with TPDOs

A extention for TPDOs (telegrams send by slave) would be :
TPDO1 on 0x187, TPDO2 on 0x287, TPDO3 on 0x387
Object 1 byte output = 6410
Object 2 byte output = 6411
Object 4 byte output = 6412

So the definition would be:
SDO 1800,1 (TPDO1) = 0x40000187 = TPDO1 on Canheader 0x187, no RTR
SDO 1A00,0 (TPDO1 mapping) = 0x40000108 = 8 objects in the mapping
SDO 1A00,1 (TPDO1 mapping obj1) = 0x64100108 = obj 6410,1 with 8 bits
SDO 1A00,2 (TPDO1 mapping obj2) = 0x64100208 = obj 6410,2 with 8 bits
SDO 1A00,3 (TPDO1 mapping obj3) = 0x64100308 = obj 6410,3 with 8 bits
SDO 1A600,4 (TPDO1 mapping obj4) = 0x64100408 = obj 6410,4 with 8 bits
SDO 1A600,5 (TPDO1 mapping obj5) = 0x64100508 = obj 6410,5 with 8 bits
SDO 1A600,6 (TPDO1 mapping obj6) = 0x64100608 = obj 6410,6 with 8 bits
SDO 1A00,7 (TPDO1 mapping obj7) = 0x64100708 = obj 6410,7 with 8 bits
SDO 1A00,8 (TPDO1 mapping obj8) = 0x64100808 = obj 6410,8 with 8 bits

SDO 1801,1 (TPDO2) = 0x40000287 = TPDO2 on Canheader 0x287, no RTR
SDO 1A01,0 (TPDO2 mapping) = 0x40000104 = 4 objects in the mapping

SDO 1A01,1 (TPDO2 mapping obj1) = 0x64110108 = obj 6411,1 with 16 bits
SDO 1A01,2 (TPDO2 mapping obj2) = 0x64110208 = obj 6411,2 with 16 bits
SDO 1A01,3 (TPDO2 mapping obj3) = 0x64110308 = obj 6411,3 with 16 bits
SDO 1A01,4 (TPDO2 mapping obj4) = 0x64110408 = obj 6411,4 with 16 bits

SDO 1802,1 (TPDO3) = 0x40000387 = TPDO3 on Canheader 0x387, no RTR
SDO 1A02,0 (TPDO3 mapping) = 0x40000102 = 2 objects in the mapping
SDO 1A602,1 (TPDO3 mapping obj1) = 0x64120120 = obj 6412,1 with 32 bits
SDO 1A02,2 (TPDO3 mapping obj2) = 0x64120220 = obj 6412,2 with 32 bits

Hint:
If you want a TPDO to be send cyclic, set the cyclic time in row “Send Cycle[ms]”.
Example: 1000 would send the TPDO telegram every second AND every time the value changes.
Please mind, that first send of TPDO is done on the first tag change in iX program (or a tag write in the iX init).

4 FAQs and hints

4.1 How a taglist is handled

A taglist is produced, if an Excel file is imported. A taglist is a reference list, that shortens the access to variables. Instead of "Give me address 1234 with mask 5678 on channel 1…" we say "give me listvalue 1". Therefore the taglist must be loaded into the CiX module before communication starts. The driver does this automatically. Another advantage is, that an Excel sheet gives a better overview and a lot of tags can be inserted in a project in one step.

This is why we recommend the use of the taglist.

Another advantage is the possibility to copy the taglist.
This leads to the possibility to have identical iX projects with different taglists (but variable names must be same!)

Example: the use of the same project for CAN device 3 or 4. Make Excel for device 3 and import it into your project. Save taglist.lst as taglist3.lst. Then make Excel for device 4 and import it into your project. Save taglist.lst as taglist4.lst. If you want to load device3, just copy taglist3.lst on taglist.lst and download project. If you want to load device4, just copy taglist4.lst on taglist.lst and download project.

4.2 Avoid firmware update

Normally the firmware is loaded from the driver at project start.
If you want to avoid this, go to the ***/Project Files*** folder of your project and insert a file named ***nofirmwareload.txt*** without any text.
From then the driver won't update the firmware.

Reasons for this could be the use of a special firmware or the freeze of software.

4.3 Different ways to write a CAN telegram

a) **with Excel file**

Every tag in Excel sheet can be written by the iX project. Normally the CAN module takes the last received or send data and overwrites just the value.
By the use of protocol "J1939" the rest of the telegram is filled with "1"(=unused).
If a data outside the value should have other value, it is possible to force the data by setting 100(Bit=0) or 101(Bit=1).

In row F (Send Cycle) a repetition time-value can be set. 1000 will cause a send of the telegram every second. But the first time must be started by writing a value from the iX program (because a value is needed). The value will be send until a new value is written by the iX project.

a) **by direct commands**

Write telegram – HRA, HRB
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.
HRB is the trigger to send the telegram and last content of HRA is taken.

Sends a telegram cyclic by RZ command.

Hint: the cyclic write starts after a value is written to it, then it runs cyclic. For a start in iX write a “script.init” function and write a value to the tag by e.g. “globals.tags.setAnalog(0)”.

4.4 The use of scripts on FreeCAN tags

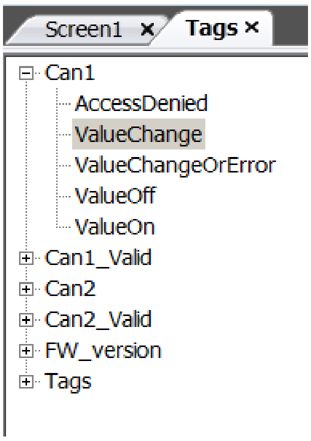
Tags can be read or written in scripts. But be aware, that the data flow is controlled by the driver and there is no absolute way to force a data exchange at a certain moment. Also the driver changes the order of write and read tags. If the order must be kept, it is necessary to collect the script tags commands in a recipe.

Example: tag “mytag”

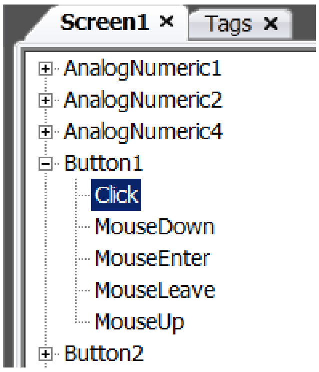
Read:	Globals.Tags.mytag.Read();
Write:	Globals.Tags.mytag.SetAnalog(0x1234);
Copy:	myvar= Globals.Tags.mytag.Value;

Also the place of scripts is evident for its use.

Scripts, which are placed in Tags/scripts (e.g. in ValueChange) cause the driver to read the tag constantly. Please be aware, that this can lead to slow data communication (if too many tags are read).



Scripts, which are placed in buttons are only done on button event:



Please remember: Tags, that are shown in Analog Numeric, are only read on active screen.