miunske
AUF DEN PUNKT

miunske-toolchain

name:	**miunske-toolchain**
version date:	**06.06.2017**

Table of Contents

1 General

Dear Toolchain user

We are pleased that you have decided to purchase a miunske product including our Toolchain. The Toolchain is a software for parameterisation and programming of the products of miunske GmbH that were developed in-house.

2 Installing the Toolchain

Please follow these instructions to install all required components correctly.

System requirements:

Windows 8 (32bit or 64bit), 2GB RAM
Windows 7 (32bit or 64bit), 2GB RAM
Windows XP is not supported

Screen resolution:

minimum	1280 x 1024 pixel
recommended	1920 x 1200 pixel

- Download the current version of the Toolchain at the following link:

  http://developer.miunske.com/de/downloads

- After a successful download, please launch the downloaded file "Toolchain.exe" as **administrator** to install it.

- After a successful installation, the Toolchain can be started.

Language: You can select your language by choosing it from the menu
“SETTINGS” → “Toolchain” → “General” → “Language”.

3 Connecting devices to the PC

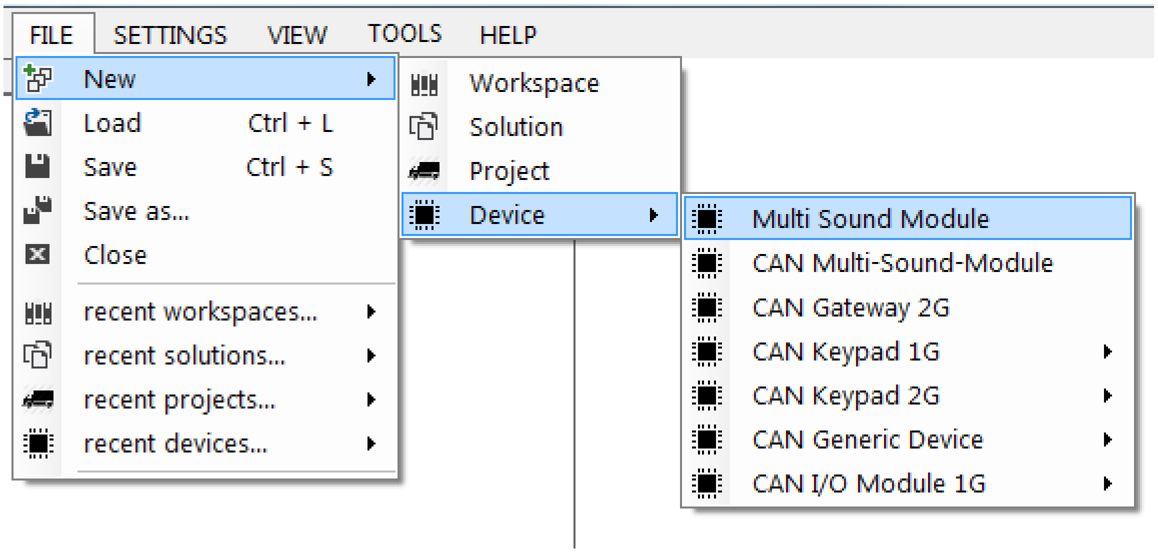
For the reading and writing CAN keyboards, Peak PCAN-USB or Sontheim CANfox interface can be used with the appropriate driver.

4 Toolchain surface

4.1 Creating a project

In the following, an example is shown that illustrates how a new project is created.

“FILE” → “New” → “Device” → „Multi-Sound-Modul“

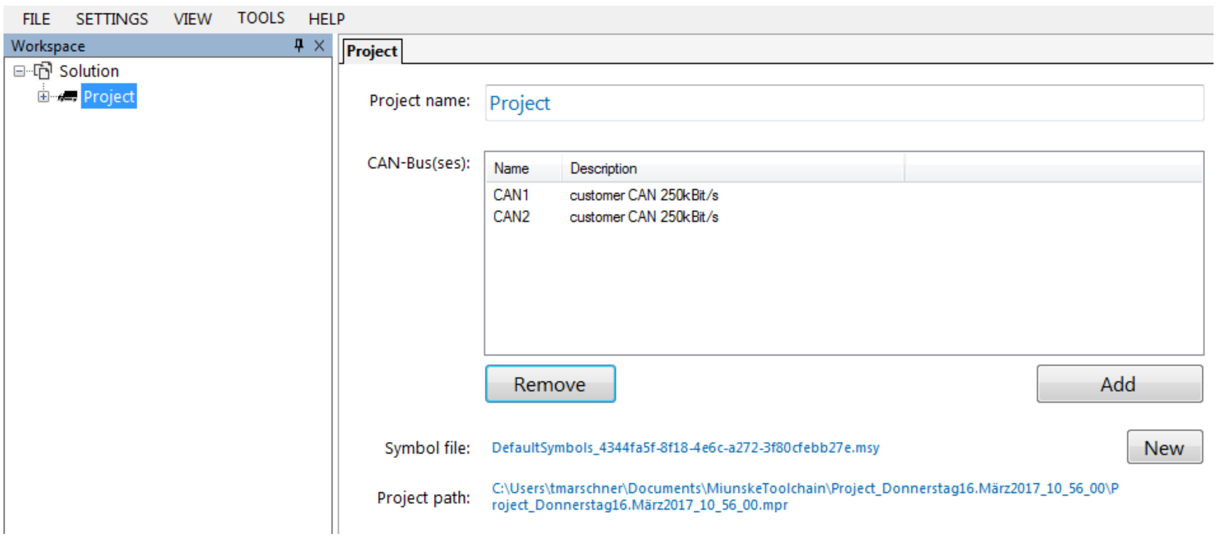


When you choose a Device, a Solution and a Project are also created.

4.2 Create CAN-Busses

The new project appears in the workspace.

The project configurations open, when you click on the name.



Project name: Project

CAN-Bus(ses):

Name	Description
CAN1	customer CAN 250kBit/s
CAN2	customer CAN 250kBit/s

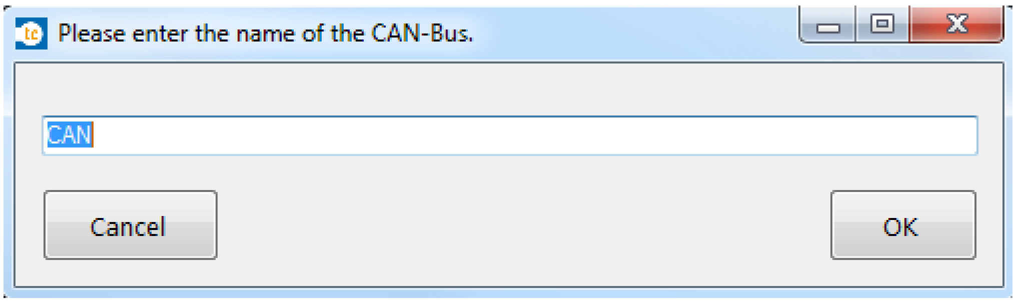
Symbol file: DefaultSymbols_4344fa5f-8f18-4e6c-a272-3f80cfebb27e.msy

Project path: C:\Users\tmarschner\Documents\MiunskeToolchain\Project_Donnerstag16.März2017_10_56_00\Project_Donnerstag16.März2017_10_56_00.mpr

„Projekt name“	The project name is noted here.
„CAN-Bus(ses)“	The CAN busses that should be used can be created here. Two CAN buses already exist with the supplied standard symbol file.
„Symbol file“	With the help of it another configuration can be loaded into the Symbol Manager. With the button “NEW” can be created a new Symbol file.
„Projekt path“	The created project will be exported so that it can be used on another computer again.

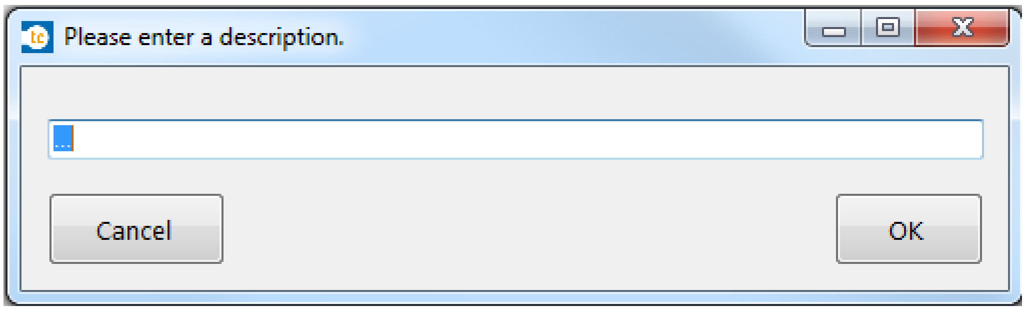
Creating the CAN-Bus is done as follows:

By clicking the button “Add”, a new CAN-Bus is created. Then an input field appears where the name of this CAN-Bus can be entered.


Please enter the name of the CAN-Bus.

CAN

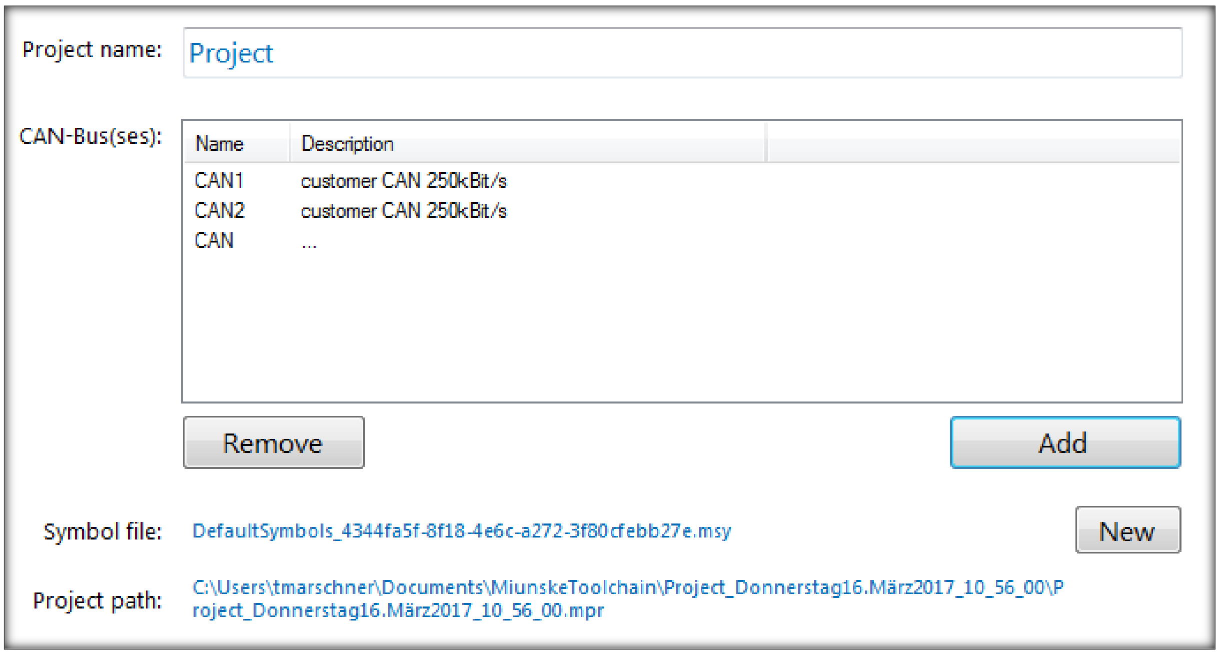
If the name of the CAN-Bus is given, it has to be confirmed by clicking the “OK” button. After confirming, a new input field for the description appears.


Please enter a description.

...

In this field an additional description can be given for the previously defined CAN-Bus. When the description is finished, this can be confirmed by clicking the “OK” button.

Now a new CAN-Bus should be defined in the “CAN-Bus(se)” field.


Project name: Project

CAN-Bus(ses):

Name	Description
CAN1	customer CAN 250kBit/s
CAN2	customer CAN 250kBit/s
CAN	...

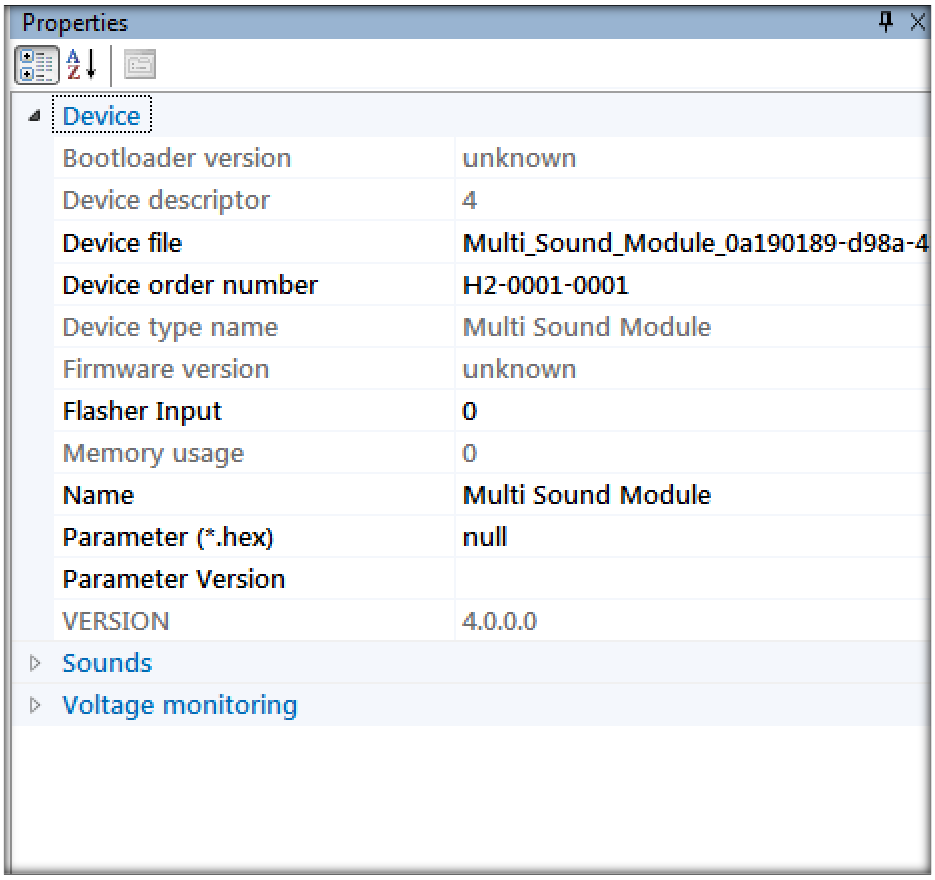
Symbol file: DefaultSymbols_4344fa5f-8f18-4e6c-a272-3f80cfebb27e.msy

Project path: C:\Users\tmarschner\Documents\MiunskeToolchain\Project_Donnerstag16.März2017_10_56_00\Project_Donnerstag16.März2017_10_56_00.mpr

4.3 Properties

You can modify the attributes of the workspace-objects in the in the window „Properties“.

Every object has different attributes, many of them are visualised in the user-interface and do not have to set in the „Properties“.


Properties

Device

Bootloader version	unknown
Device descriptor	4
Device file	Multi_Sound_Module_0a190189-d98a-4
Device order number	H2-0001-0001
Device type name	Multi Sound Module
Firmware version	unknown
Flasher Input	0
Memory usage	0
Name	Multi Sound Module
Parameter (*.hex)	null
Parameter Version	
VERSION	4.0.0.0

Sounds

Voltage monitoring

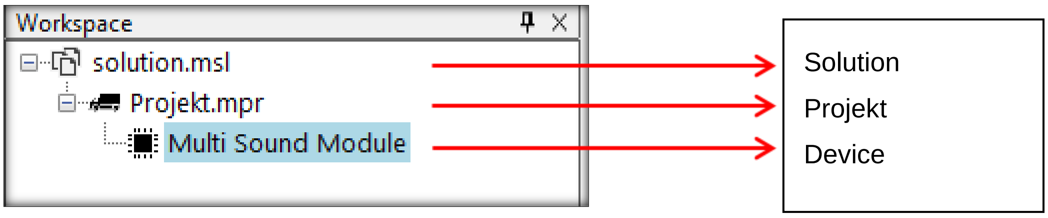
4.4 Workspace

The structure of the Workspace:

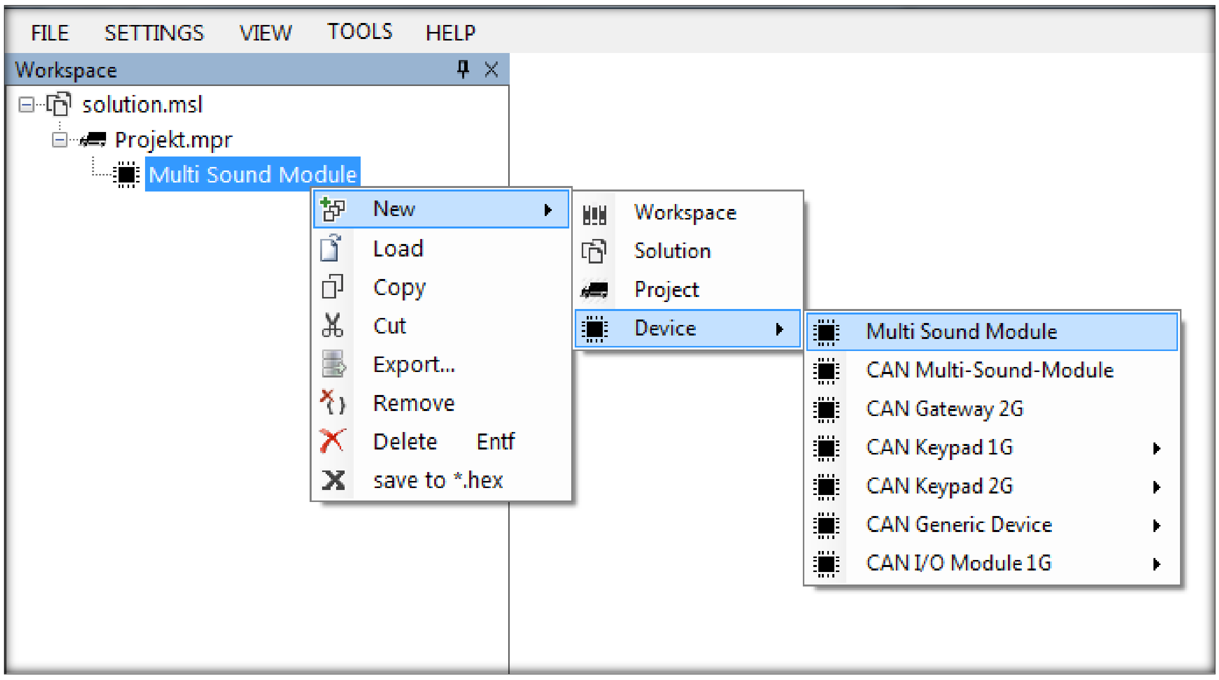
Solution: wraps projects under a name z.B. company name.

Project: name of of the project

Device: list of all devices that belong to a project.



Every object has some specific functions in the context menu (right-click).

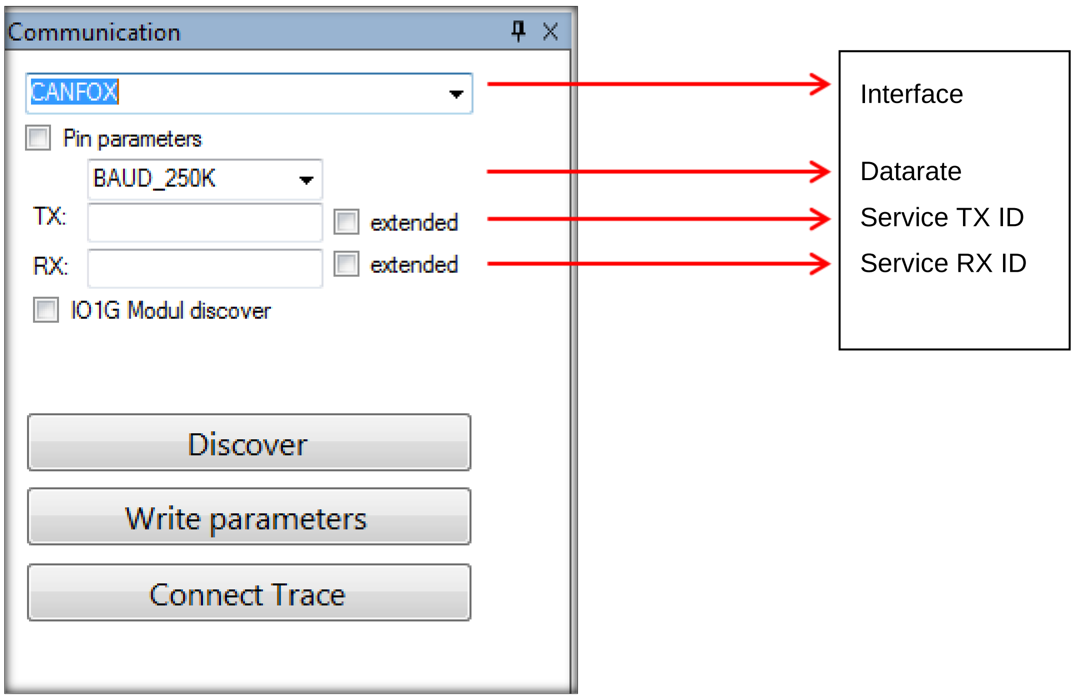


In the following some general functions that are device and project specific.

New	a new device, project or solution can be added
Load	loads a device, project or solution to the selected node
Copy	copies devices or projects
Paste	pastes copied or cut nodes
Cut	devices or projects can be moved to other nodes
Export…	exports node into a file to share it
Remove	removes the node from the Workspace
Löschen	removes the node from the Workspace **and from your harddrive**
save to *.hex	special-function to generate a flashfile

4.5 Communication

Connected devices can be accessed via the „Communication“ window.

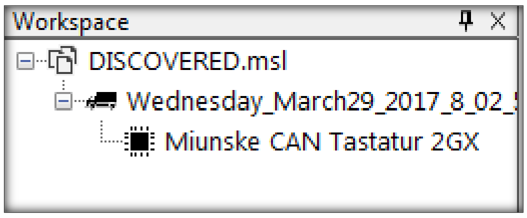


Interface	Choose the Interface that should be used Multi-Sound-Module – „SERIAL“ CAN-Tastatatur – CAN Interface „PEAK“ or „CANFOX“
Pin parameter	Data rate, Service TX ID and Service RX ID can be fixed to transmit them to another device
Datarate	baud rate of the CAN-Bus System
Service TX ID	The Transmit-ID is needed to find the device in the CAN-Bus System
Service RX ID	The Receive-ID is needed to find the device in the CAN-Bus System
extended	The Service ID’s can be in 11Bit or 29Bit (extended) format
IO 1G Module discover	To parameterize the 1G IO Module you have to enable this checkbox

Discover	Discovers all connected devices with different Service ID’s at once explanation: - Only the baud rate matters - recognized devices can be found in the last selected project, if no project is selected, the Toolchain will create one in the solution “DISCOVERED”
Write parameters	Writes parameter to the selected device.
Connect Trace	Interface (CANFOX, PEAK) are prepared. **You can either trace or parameterize.**

miunske standard products are delivered with a 250 kbit/s or 500 kbit/s datarate.

To get the parameters from the device to the Toolchain press Discover.



The recognized devices can be found under their name in a project with timestamp, which is in the solution “DISCOVERED”.

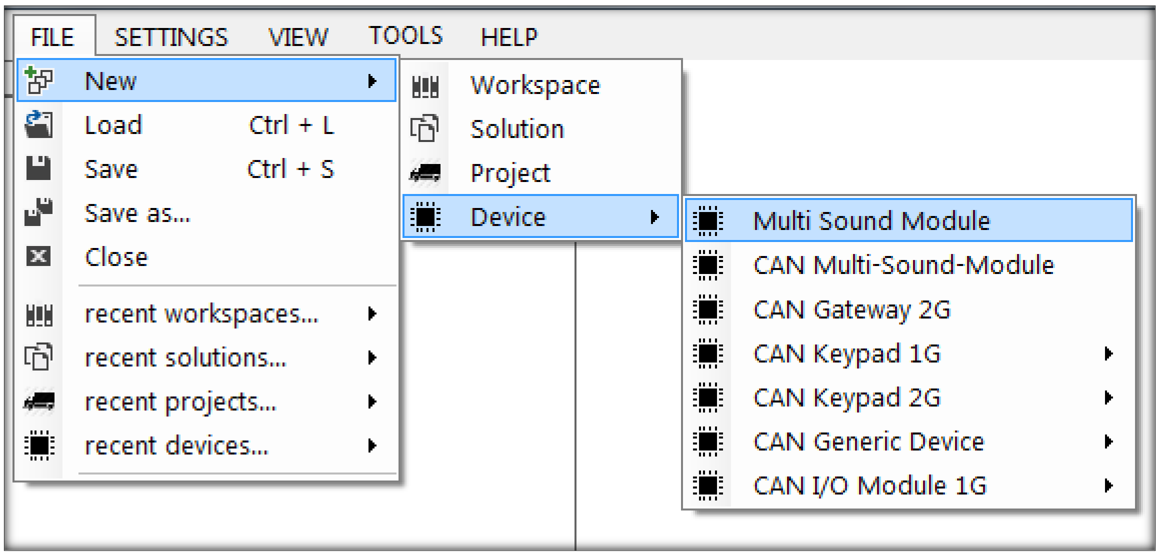
To transmit your configuration, you have to press “Write parameters”.

Make sure the right baud rate and Service-IDs are selected.

4.6 Menu structure

The menu of the Toolchain consists of 5 main categories

4.6.1 FILE



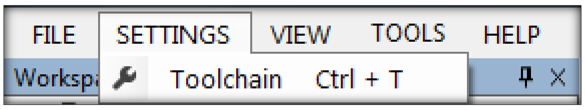
In the File menu are the most important functions for the work with the Toolchain

New	You can add a new device, project, solution or workspace
Load	You can load a new device, project, solution or workspace
Save	Saves the whole workspace
Save as…	Saves the workspace under a given filepath
Close	Closes the Toolchain
Recent workspaces…	Here are the recent workspaces for quick selection. The number of work-spaces can be defined in the settings menu. Every workspace contains a collection of solutions. **You cannot move a workspace to another computer.**
Recent solutions…	Here are the recent solutions for quick selection. The number of solutions can be defined in the settings menu. Every solution contains a collection of projects. The solutions represent for example a customer. **You cannot move a solution to another computer.**
Recent projects…	Here are the recent projects for quick selection. The number of projects can be defined in the settings menu. A project represents a vehicle with devices. You can move projects with the export function.

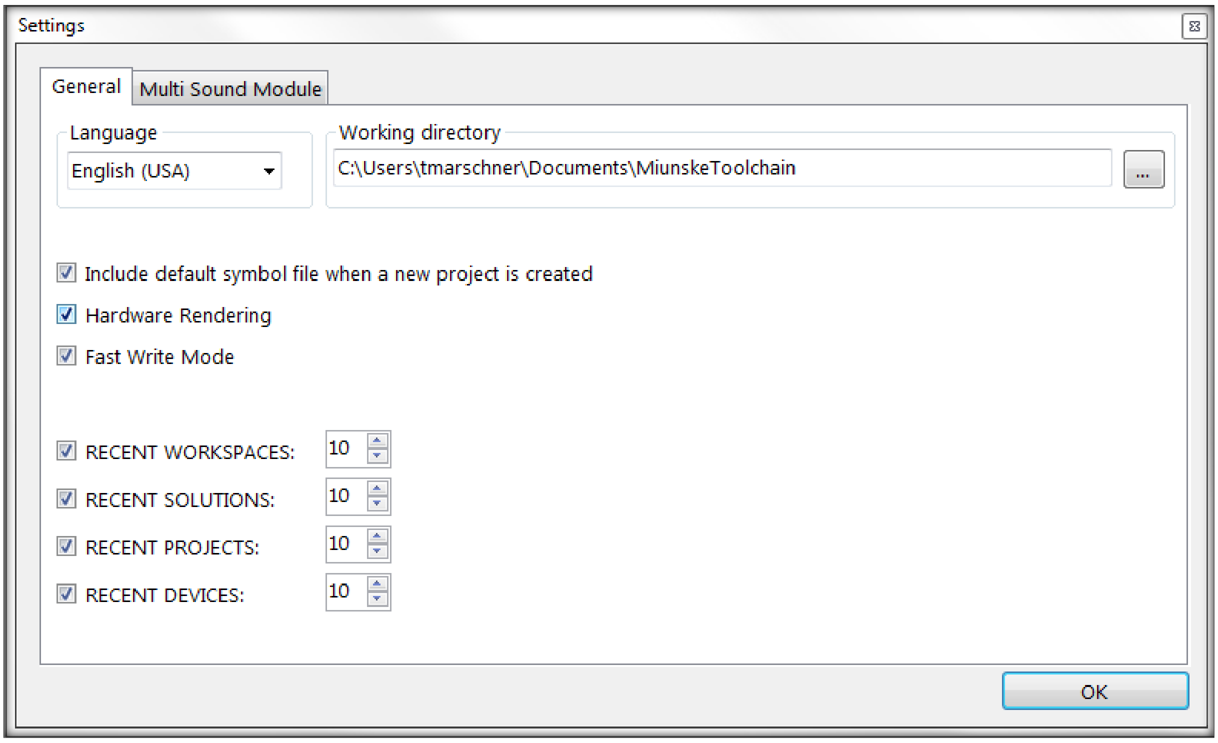
	Please note: the sound library needs to be copied manual
Recent devices…	Here are the recent devices for quick selection. The number of devices can be defined in the settings menu.

4.6.2 SETTINGS

In the menu item settings are global settings for the Toolchain.



You can reach the dialog over “SETTINGS” → “Toolchain” or the shortcut Ctrl+T.



Language: English (USA)

Working directory: C:\Users\tmarschner\Documents\MiunskeToolchain

[✓] Include default symbol file when a new project is created

[✓] Hardware Rendering

[✓] Fast Write Mode

[✓] RECENT WORKSPACES: 10

[✓] RECENT SOLUTIONS: 10

[✓] RECENT PROJECTS: 10

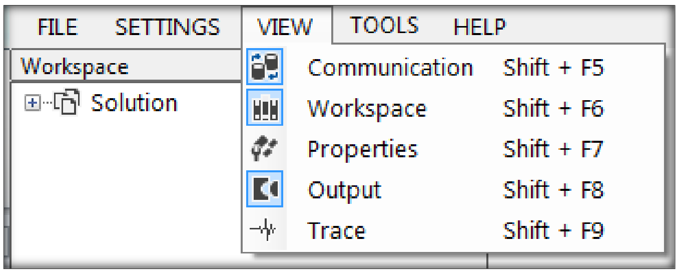
[✓] RECENT DEVICES: 10

In the settings-window are tabs for general and device specific. The device specific settings are explained along with the device.

Language	Switch between German and English
Working directory	Default save location for workspaces, projects and devices and discovered devices • The button **"…“** (open) let you change the working directory
Include default symbol file when a new project is created	When selected the default symbolfile with predefined symbols is included in every new project
Hardware Rendering	User interface is rendered by the GPU
Fast Write Mode	You can disable the CAN-specification to transmit faster than one message per ten milliseconds
RECENT WORKSPACES	Enables the menu item in “FILE” and sets the workspace limit
RECENT SOLUTIONS	Enables the menu item in “FILE” and sets the solution limit
RECENT PROJECTS	Enables the menu item in “FILE” and sets the project limit
RECENT DEVICES	Enables the menu item in “FILE” and sets the device limit

4.6.3 VIEW

Shows and hides important windows.



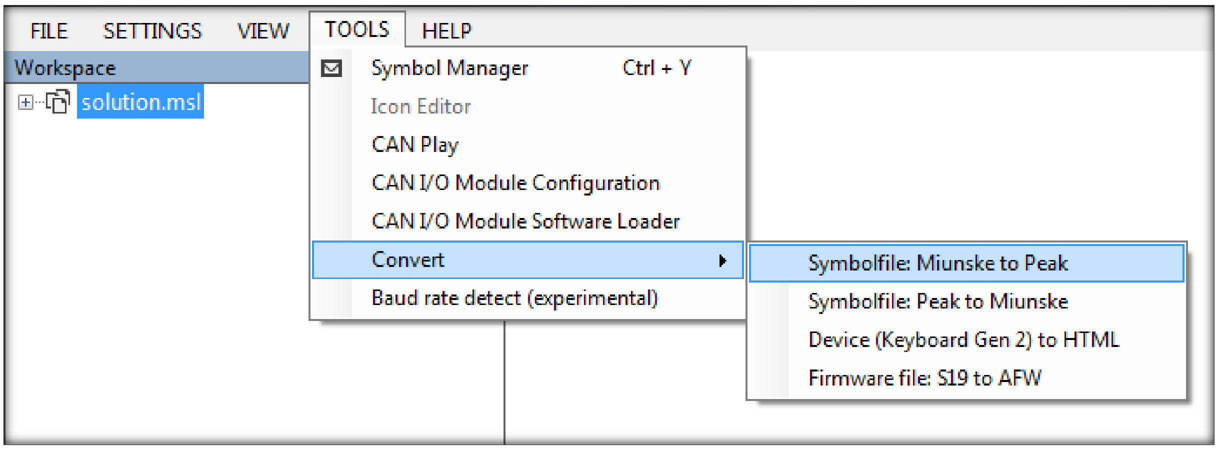
Communication	Interface to read and write parameters from and on devices
Workspace	Overview over all open solutions, projects and devices
Properties	specific properties of the solutions, projects and devices
Output	Logs all relevant events
Trace	Tool to analyse the traffic on the CAN-Bus

Every window can be shown or hidden with the following shortcuts:

Communication	Shift+F5
Workspace	Shift+F6
Properties	Shift+F7
Output	Shift+F8
Trace	Shift+F9

4.6.4 TOOLS

In the menu “Tools” are additional programms and function.


FILE
SETTINGS
VIEW
TOOLS
HELP
Workspace
solution.msl
Symbol Manager
Ctrl + Y
Icon Editor
CAN Play
CAN I/O Module Configuration
CAN I/O Module Software Loader
Convert
Baud rate detect (experimental)
Symbolfile: Miunske to Peak
Symbolfile: Peak to Miunske
Device (Keyboard Gen 2) to HTML
Firmware file: S19 to AFW

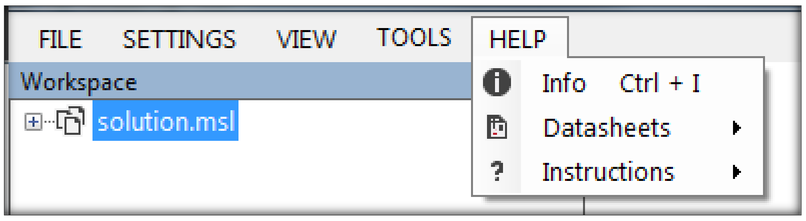
Symbol Manager	Window to configure CAN-messages. More Informations in section Symbol Manager
Icon Editor	Tool to create new icons
CAN Play	Tool to create artificial CAN-Bus traffic
CAN I/O Module Configuration	for the I/O Module exist two programmer adapter. You need this if you have a blue box.
CAN I/O Module Software Loader	for the I/O Module exist two programmer adapter. You need this if you have a blue box.
Convert	Here you can find various programms to convert between fileformats

- Symbolfile: Miunske to Peak
  you can convert symbol files from the Symbol Manager to symbol files for the PCAN-Explorer
- Symbolfile: Peak to Miunske
  you can convert symbol files from the PCAN-Explorer to symbol files for the Symbol Manager
- Device (Keyboard Gen 2) to HTML
  you can visualize Keyboard Gen 2 in HTML to get a clear overview
- Firmwarefile: S19 to AFW
  I/O Module Generation 1 support no S19 files. With this function you can convert from Gen 2 to Gen 1 files.

Baud rate detect (experimental)	This feature can discover the baud rate of the currently connected CAN-Bus system

4.6.5 HELP

In the menu item “Help” are helpful informations for the Toolchain provided.



Info	Information about the Toolchain and contact details. The shortcut is Ctrl+I
Datasheets	technical datasheets for devices
Instructions	User Guides

4.7 Keyboard Shortcuts

4.7.1 Toolchain Interface

Window Communication	Shift+F5
Window Workspace	Shift+F6
Window Settings	Shift+F7
Window Output	Shift+F8
Window Trace	Shift+F9
Toolchain Info	Ctrl+I
Load File	Ctrl+L
Save File	Ctrl+S
Toolchain Settings	Ctrl+T
Window Symbol Manager	Ctrl+Y

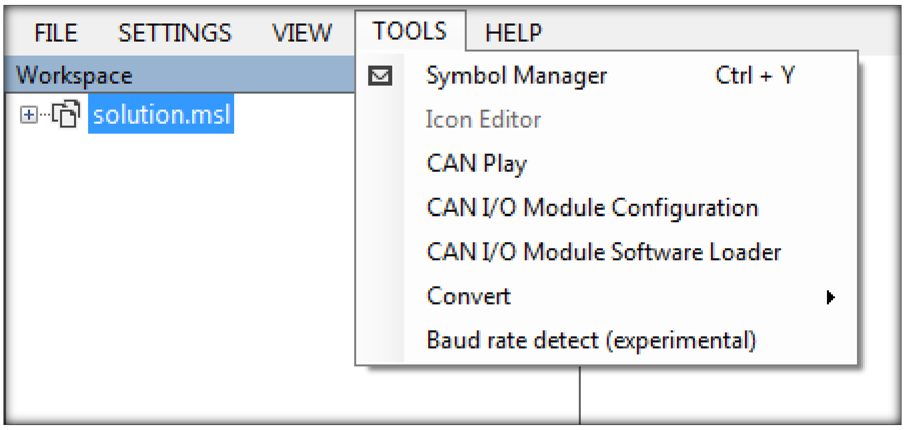
4.7.2 Symbol Manager

New Symbol	Ctrl+ Shift+S
New Variable	Ctrl+ Shift+V
Copy	Ctrl+C
Cut	Ctrl+X
Paste	Ctrl+V
Rename	Ctrl+R
Delete	Ctrl+Del

5 Symbol Manager

The Symbol Manager is used for creating the CAN Messages.

The Symbol Manager can be called in several ways, for example via the menu bar "TOOLS" → "Symbol Manager".

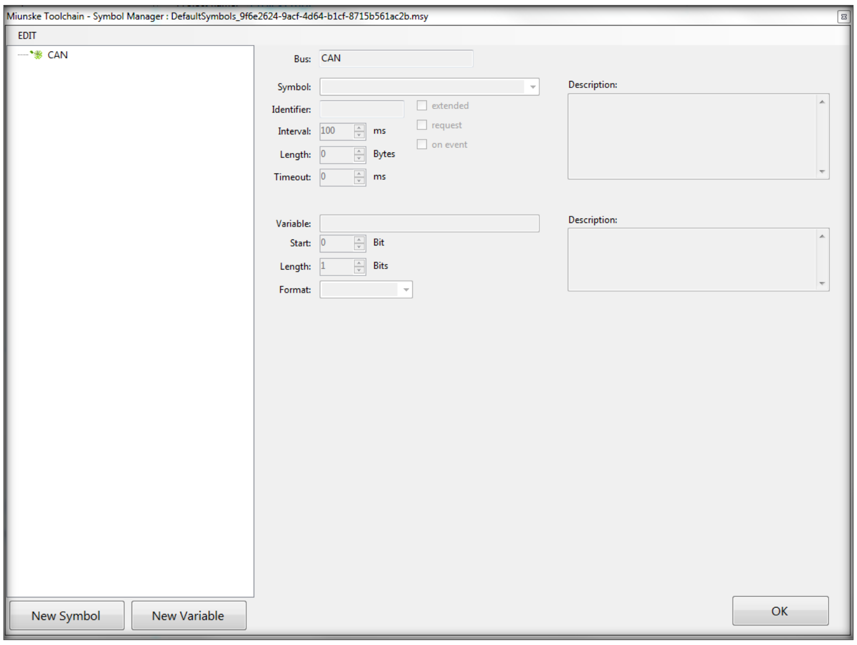


Furthermore, the Symbol Manger can be called with the key combination "Ctrl + Y".

The Symbol Manager can also be accessed within a project, if a CAN-Bus message has to be selected.

5.1 Settings in the Symbol Manager

After the CAN-Bus was created and the Symbol Manager was called, this window appears.


Miunske Toolchain - Symbol Manager : DefaultSymbols_9f6e2624-9acf-4d64-b1cf-8715b561ac2b.msy

Bus: CAN
Symbol:
Identifier:
Interval: 100 ms
Length: 0 Bytes
Timeout: 0 ms
☐ extended
☐ request
☐ on event
Description:

Variable:
Start: 0 Bit
Length: 1 Bits
Format:
Description:

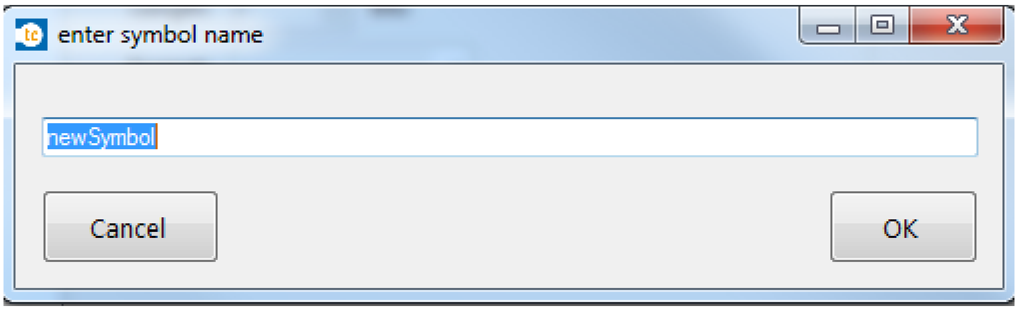
The Symbol Manager distinguishes between “symbol” and “variable”.

„Symbol“	A symbol defines a CAN-Bus ID for receiving or sending a message ID, Standard/Ext, Timeout, Interval
„Variable“	A variable describes the bit position(s) within the CAN message.

5.1.1 Symbol

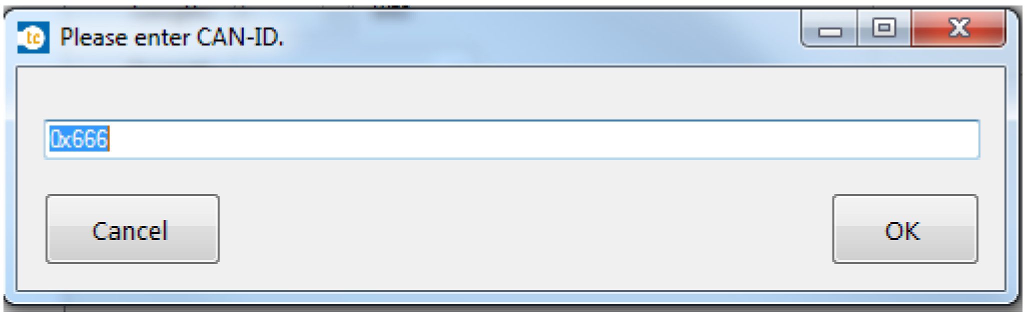
To create a CAN-Bus ID, the button “New Symbol” is used.

By clicking the button a new window appears where the name of the symbol can be defined.



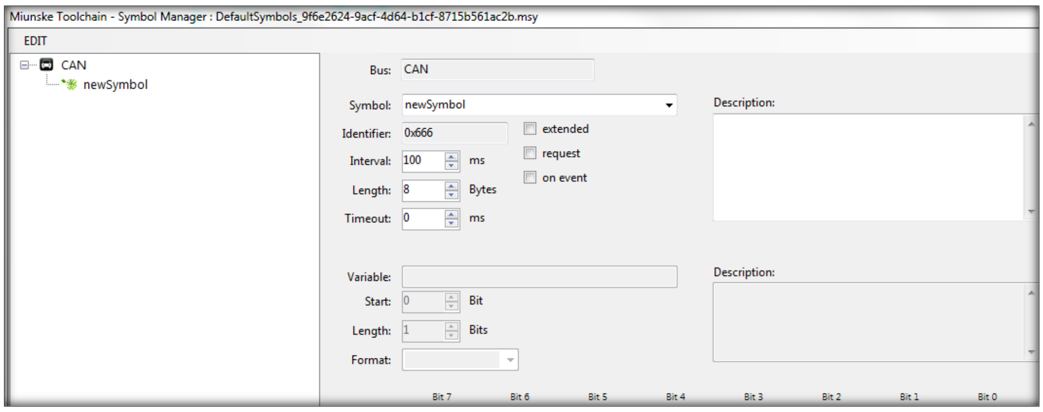
Once the name of the symbol is entered, it can be confirmed with the “OK” button.

After you confirmed the name, you can enter the new ID.



You can enter the ID as hexadecimal with the prefix 0x… or as decimal. After you entered the ID confirm with “OK“.

You can choose the new symbol on the left side. On the right side of the window some fields are highlighted in white. One can now input data into them.


Miunske Toolchain - Symbol Manager : DefaultSymbols_9f6e2624-9acf-4d64-b1cf-8715b561ac2b.msy

EDIT

CAN
newSymbol

Bus: CAN
Symbol: newSymbol
Identifier: 0x666
Interval: 100 ms
Length: 8 Bytes
Timeout: 0 ms
extended
request
on event
Description:

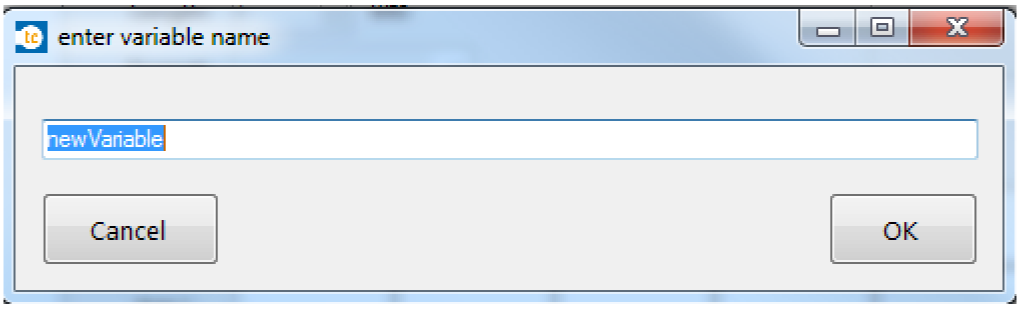
Variable:
Start: 0 Bit
Length: 1 Bits
Format:
Description:

Bus:	Indicates on which CAN-Bus the settings are made.
Symbol:	Indicates which symbol one is editing at the moment.
Identifier:	The CAN-Bus ID, on which the message should be sent or received, is entered here.
Interval:	The interval describes the repetition period until the message is sent again. **The interval should not be less than 10ms.**
Length:	This field defines how long the message is.
Timeout:	This value describes a period of time, after which sending will be cancelled in the event of a fault.
Description:	In this field an exact description of the symbol can be defined.
extended	Defines if the CAN-ID is 11 Bit or 29 Bit long
request	The received or transmitted message is a request frame
on event	Message is transmitted when a state has changed

5.1.2 Variable

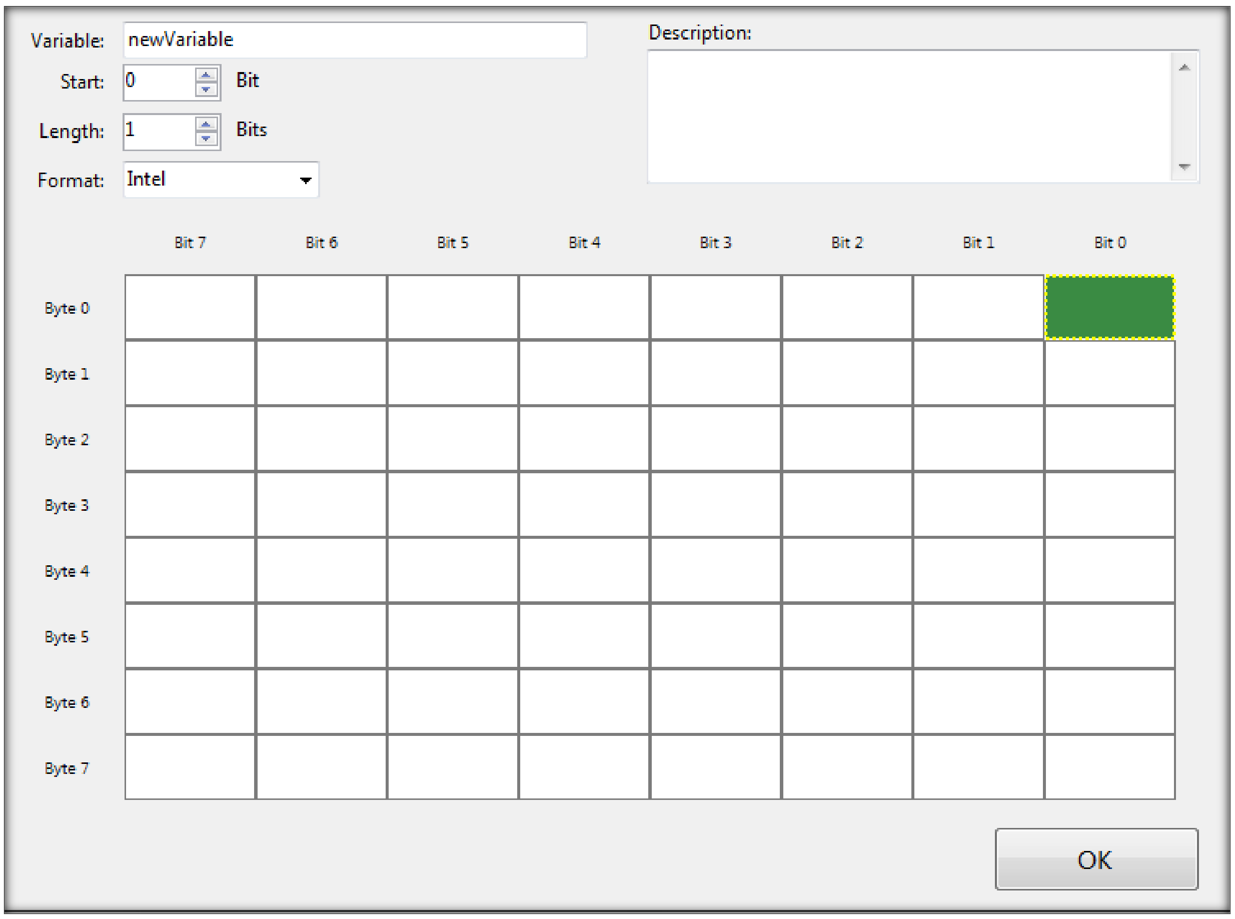
With the help of the button “New Variable” a bit assignment can be set in the symbol created earlier.

By clicking the button a new window appears where the name of the variable can be defined.



Once the name of the variable is entered, it can be confirmed with the “OK” button.

On the right side of the window the rest of the fields are now highlighted in white. One can now input data into them.


Variable: newVariable
Start: 0 Bit
Length: 1 Bits
Format: Intel
Description:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0								🟩
Byte 1								
Byte 2								
Byte 3								
Byte 4								
Byte 5								
Byte 6								
Byte 7								

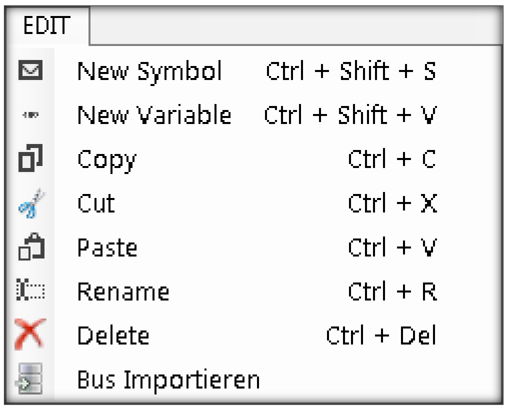
Variable:	In this field the name of the variable can be defined.
Start:	Defines the starting point within the message, as seen in the preceding illustration, highlighted with blue color in byte 1 at bit position 0
Länge:	This field defines the length of the message in bits.
Format:	Defines the byte-order, options are Intel (Little-Endian) or Motorola (Big-Endian)
Description:	In this field an exact description of the variable can be defined.

If all needed IDs for the project are defined, one can close the Symbol Manager by clicking the “OK” button. Now all IDs can be used in the individual devices.

5.1.3 Symbol Manger special functions

In the Symbol Manger are many special functions. The are explained in the next paragraph.

By clicking on the menu item “EDIT” you get the following menu.



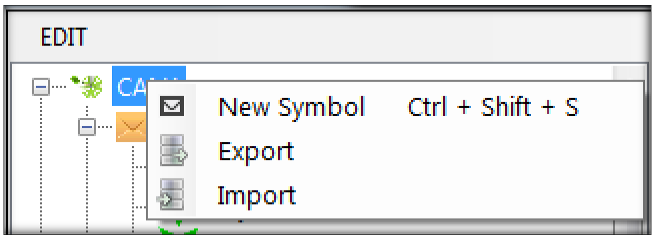
New Symbol	Add a new symbol
New Variable	Define a new variable
Copy	Copy a symbol or variable
Cut	Move symbols to another CAN-Bus or a variable to another symbol
Paste	Past the copied object in the selected node
Rename	Rename symbols or variables
Delete	Deletes a node and all of his sub nodes
Import Bus	Import a CSV (EXCEL) File into the symbol file. Requires the exact order of the values.

For a better workflow many functions have a shortcut:

New Symbol	Ctrl+Shift+S
New Variable	Ctrl+Shift+V
Copy	Ctrl+C
Cut	Ctrl+X
Paste	Ctrl+V
Rename	Ctrl+R
Delete	Ctrl+Del

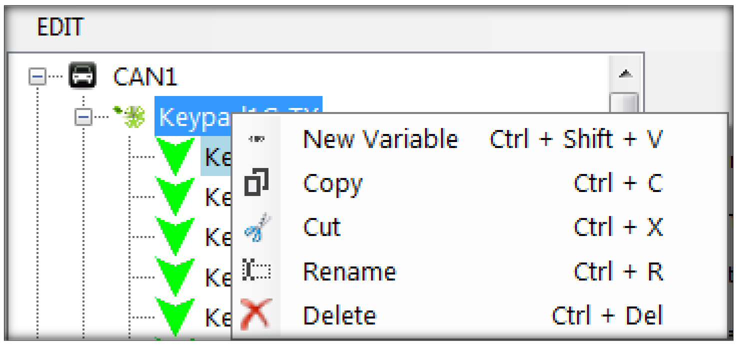
In the context menu (right mouse-click) are additional functions:

CAN-Bus



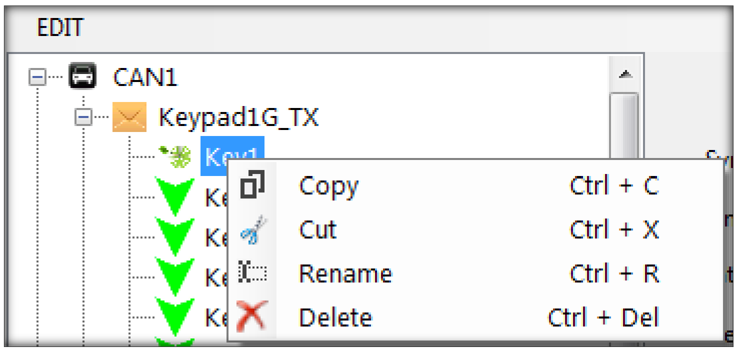
New Symbol	Add a new symbol
Export	Convert CAN symbol file in a CSV (EXCEL) file
Import	Import a CSV (EXCEL) File into the symbol file. Requires the exact order of the values.

Symbol



New Variable	Define a new variable
Copy	Copy the symbol and all contained variables
Cut	Move symbol to another CAN-Bus
Rename	Rename the symbol
Delete	Delete symbol and all contained variables

Variable



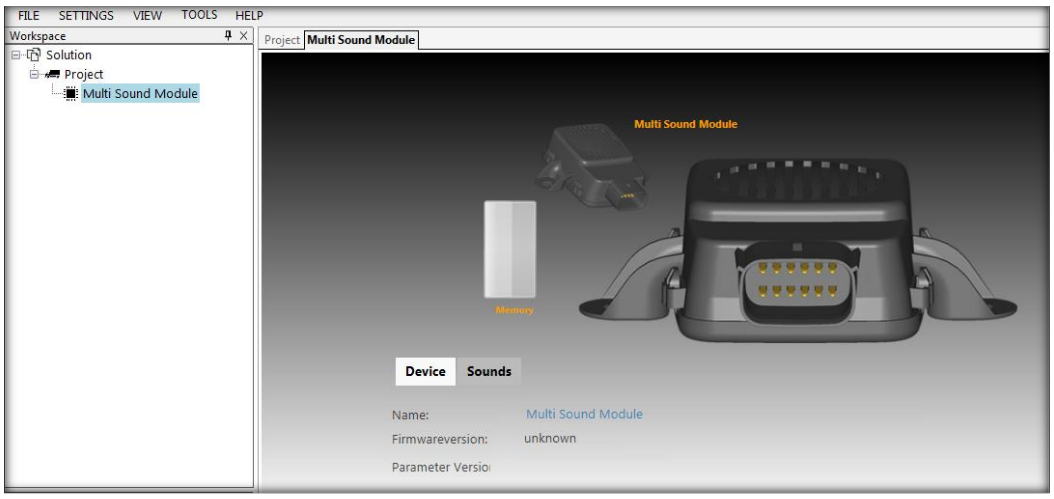
Copy	Copy the variable
Cut	Move variable to another symbol
Rename	Rename the variable
Delete	Delete the variable

6 Operation of Multi-Sound-Module

6.1 Parameterization of Multi-Sound-Module

You can choose the Multi-Sound-Module in the workspace.

After clicking on the tab, it opens itself in the middle of the Toolchain



Name: Multi Sound Module
Firmwareversion: unknown
Parameter Versio

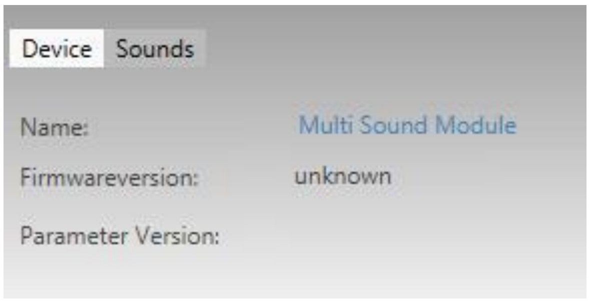
„Device“

Here, general settings are made which affect the entire device.

„Sounds“

You can assign sound files to inputs in this tab.

6.1.1 Tab: „Device“

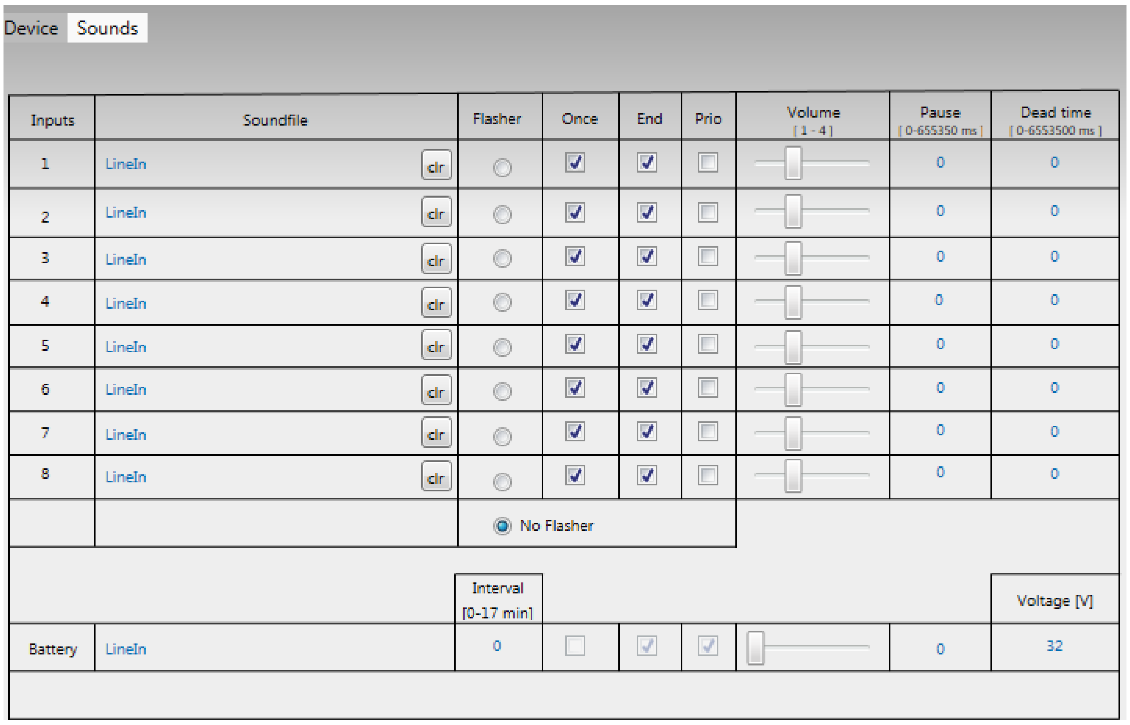


Name: Multi Sound Module
Firmwareversion: unknown
Parameter Version:

Name	The name of the product.
Firmware	is automatically changed to be corresponding firmware version, as soon as a device has been read
Parameter Version	Free versioning as text or number for the data record.

6.1.2 Tab: „Sounds“

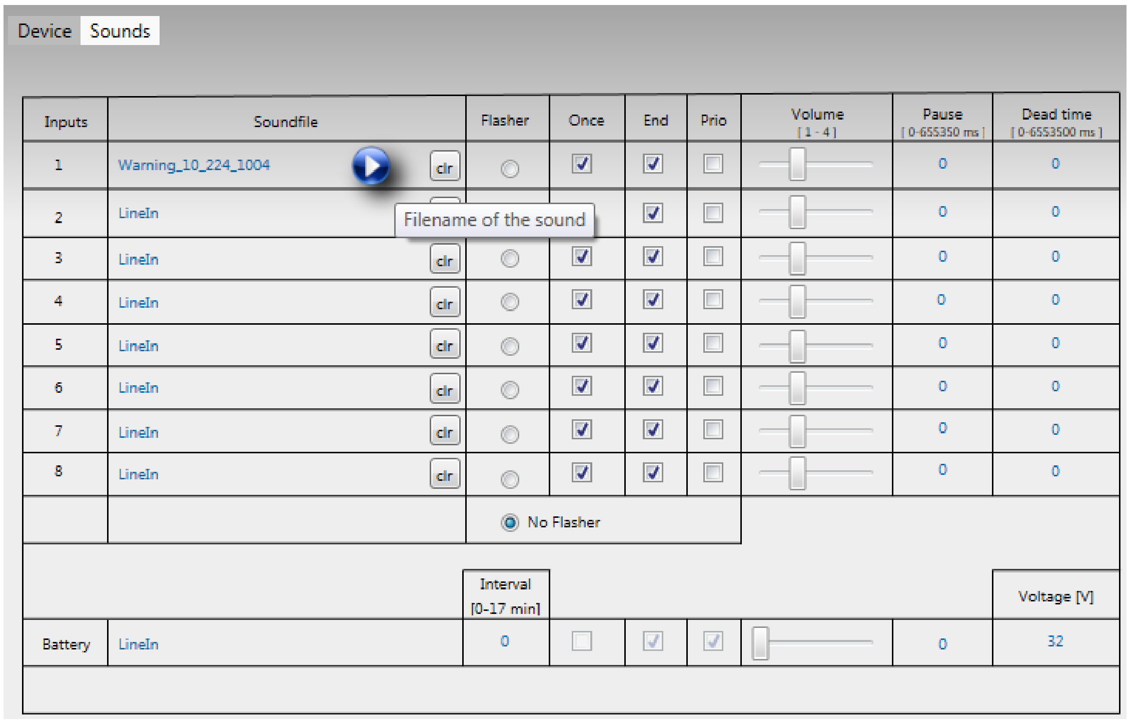
Here are all settings for the sound files of the module.



Device | Sounds

Inputs	Soundfile		Flasher	Once	End	Prio	Volume [1 - 4]	Pause [0-655350 ms]	Dead time [0-6553500 ms]
1	LineIn	clr	○	✓	✓	☐		0	0
2	LineIn	clr	○	✓	✓	☐		0	0
3	LineIn	clr	○	✓	✓	☐		0	0
4	LineIn	clr	○	✓	✓	☐		0	0
5	LineIn	clr	○	✓	✓	☐		0	0
6	LineIn	clr	○	✓	✓	☐		0	0
7	LineIn	clr	○	✓	✓	☐		0	0
8	LineIn	clr	○	✓	✓	☐		0	0
			● No Flasher						
			Interval [0-17 min]						Voltage [V]
Battery	LineIn		0	☐	✓	✓		0	32

If one moves the mouse pointer over the individual elements, information on the function is displayed.



Device | Sounds

Inputs	Soundfile		Flasher	Once	End	Prio	Volume [1 - 4]	Pause [0-655350 ms]	Dead time [0-6553500 ms]
1	Warning_10_224_1004	clr	○	✓	✓	☐		0	0
2	LineIn	Filename of the sound			✓	☐		0	0
3	LineIn	clr	○	✓	✓	☐		0	0
4	LineIn	clr	○	✓	✓	☐		0	0
5	LineIn	clr	○	✓	✓	☐		0	0
6	LineIn	clr	○	✓	✓	☐		0	0
7	LineIn	clr	○	✓	✓	☐		0	0
8	LineIn	clr	○	✓	✓	☐		0	0
			● No Flasher						
			Interval [0-17 min]						Voltage [V]
Battery	LineIn		0	☐	✓	✓		0	32

The Toolchain already contains a large number of sounds out of the box. By clicking on a field in the column “SOUNDFILE” one can enter the library.

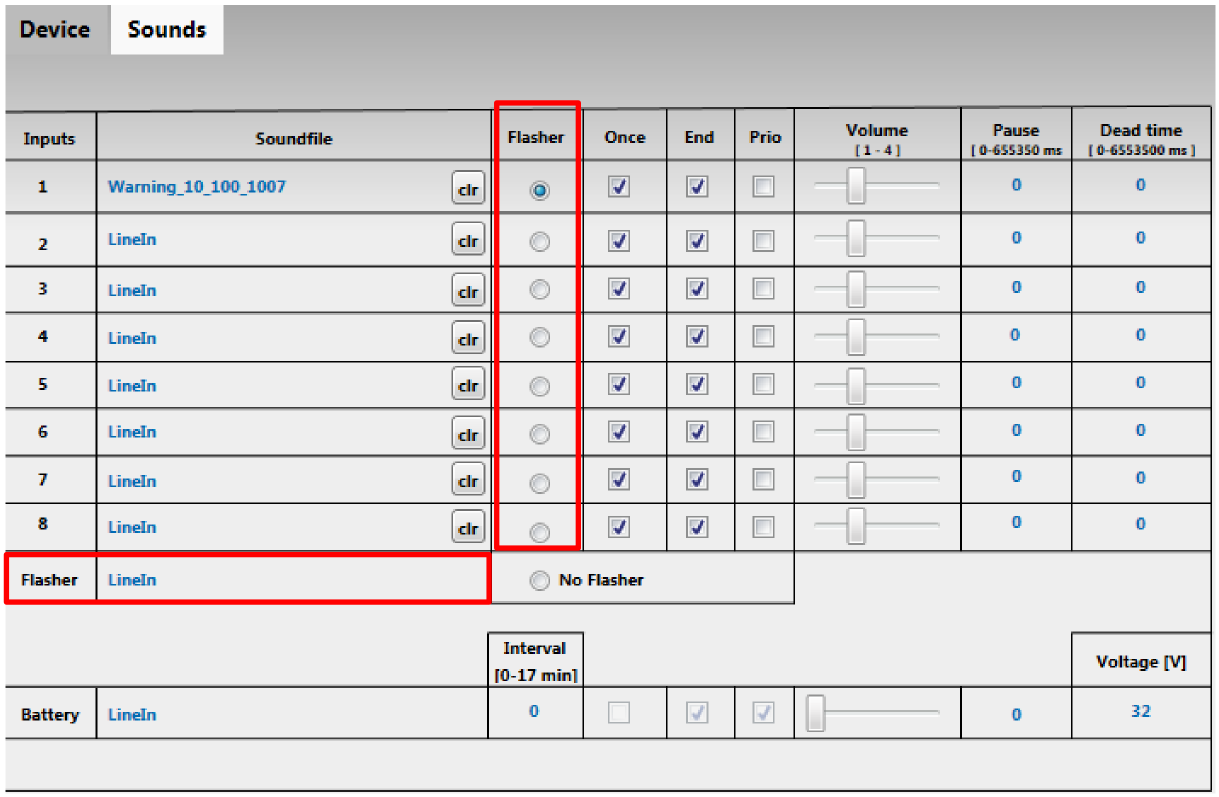
After selecting a sound file, it can be played on the PC by clicking on the blue “play” button.

6.1.3 Flasher function

A flasher has two different levels. In digital measuring technology, these are referred as “falling edge” (turn off), which means the state changes from „1“ to „0“, and „rising edge“ (turn on), which means the state changes from „0“ to „1“.

The Toolchain offers the possibility to choose a sound for each of these levels.

This is done as follows:



Inputs	Soundfile		Flasher	Once	End	Prio	Volume [1 - 4]	Pause [0-655350 ms	Dead time [0-6553500 ms]
1	Warning_10_100_1007	clr	●	✓	✓	☐		0	0
2	LineIn	clr	○	✓	✓	☐		0	0
3	LineIn	clr	○	✓	✓	☐		0	0
4	LineIn	clr	○	✓	✓	☐		0	0
5	LineIn	clr	○	✓	✓	☐		0	0
6	LineIn	clr	○	✓	✓	☐		0	0
7	LineIn	clr	○	✓	✓	☐		0	0
8	LineIn	clr	○	✓	✓	☐		0	0
Flasher	LineIn		○ No Flasher						
			Interval [0-17 min]						Voltage [V]
Battery	LineIn		0	☐	✓	✓		0	32

In the first step, an input of the Multi-Sound-Module must be selected. Then a sound file must be set for it (“rising edge”). Once this is done, the flasher function can be activated in the column “Flasher”.

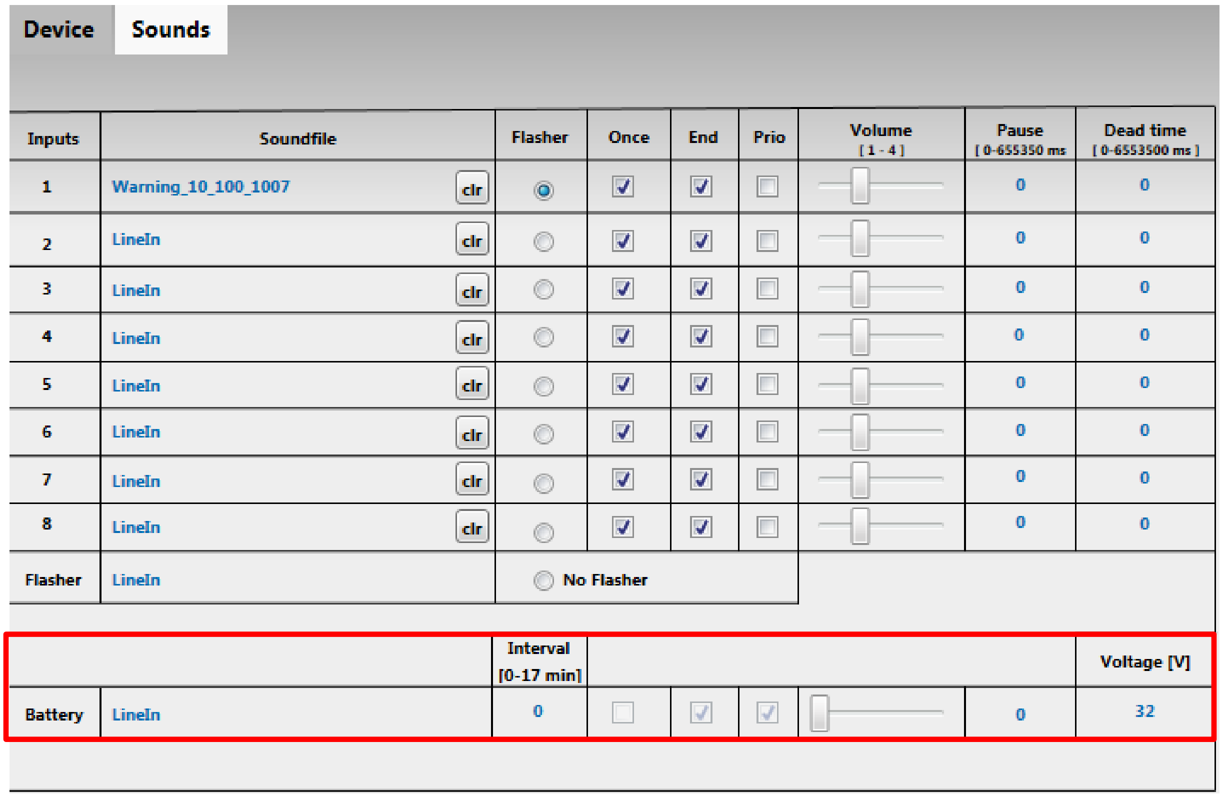
When the flasher is activated, an additional field called “Flasher” appears at the bottom of the table. In this field, another sound file is selected for the “falling edge”.

After selecting a sound file for the flasher, it can be played on the PC by clicking on the blue “play” button. Once a signal is generated, the normal sound file is played, in the end of the signal the sound file for the flasher is played. Only one flasher function can be assigned.

6.1.4 Voltage monitoring

The Toolchain offers the possibility to monitor the operating voltage of 9V - 32V and is able to play a warning signal if it is too low.

Configuration instructions:



Device | Sounds

Inputs	Soundfile		Flasher	Once	End	Prio	Volume [1-4]	Pause [0-655350 ms	Dead time [0-6553500 ms]
1	Warning_10_100_1007	clr	●	✓	✓	☐		0	0
2	LineIn	clr	○	✓	✓	☐		0	0
3	LineIn	clr	○	✓	✓	☐		0	0
4	LineIn	clr	○	✓	✓	☐		0	0
5	LineIn	clr	○	✓	✓	☐		0	0
6	LineIn	clr	○	✓	✓	☐		0	0
7	LineIn	clr	○	✓	✓	☐		0	0
8	LineIn	clr	○	✓	✓	☐		0	0
Flasher	LineIn		○ No Flasher						
			Interval [0-17 min]						Voltage [V]
Battery	LineIn		0	☐	✓	✓		0	32

In the row “BATTERY” a sound file and then the voltage limit is selected at which this sound should be played back.

As voltage value, a value can be defined with two decimal places.

In the column “INTERVAL” a time must be set.

This time determines at which intervals the module should measure the voltage. This can range from 0 to 17 minutes. If the value is set to 0 (default value), no operating voltage is monitored.

Please note that the module requires power during each measurement. This can lead to a dis-charge of the battery over a longer period of time.

6.2 Integrating own sound files

The Toolchain offers the possibility to integrate own sound files.
This is done as follows:

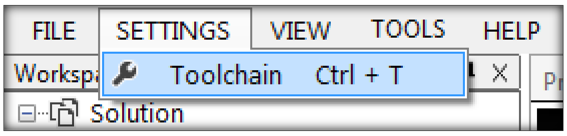
In the first step, the sound file has to be converted to the correct format. This can be done with a free Tool e.g.: Audacity.

The Toolchain accepts sound files in WAV format with the following parameters:

Channels:	Mono
Bitrate:	256 kbit/s
Sample rate:	16000 Hz
Bit depth:	16 bit PCM

If the sound files fulfil these characteristics, they can be integrated into the Toolchain.

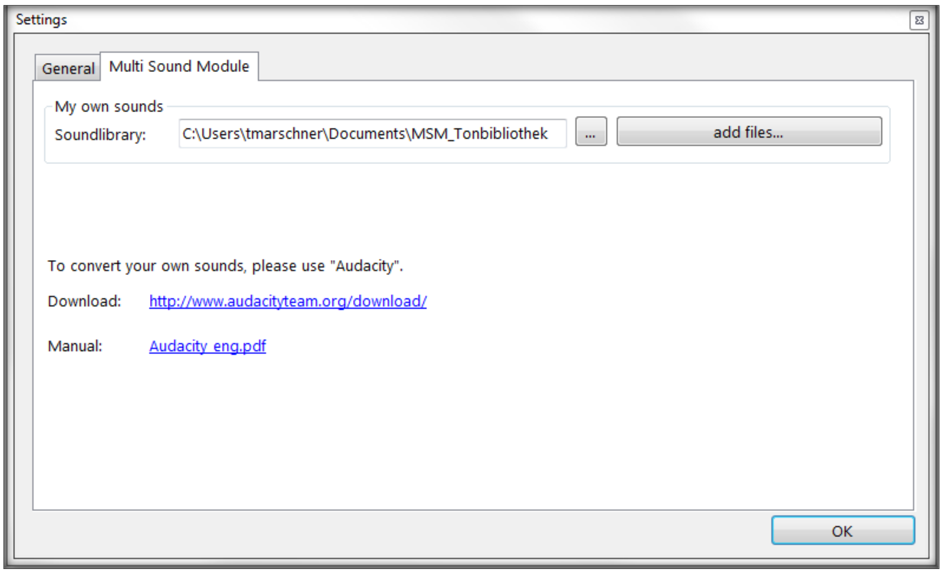
All settings of the Toolchain itself and of the individual devices can be found in the menu "SETTINGS" → "Toolchain".



Each sound file is copied to a user defined folder and the Toolchain attaches a number to the end of the file name. This number is responsible for the assignment of the sound files when a Multi-Sound-Module is read.

Note: This works only if the defined folder is not changed and the sound files were not copied manually to the folder.

The folder path is defined in the tab “Multi-Sound-Module” in the field called “sound library”.



Soundlibrary: C:\Users\tmarschner\Documents\MSM_Tonbibliothek

To convert your own sounds, please use "Audacity".

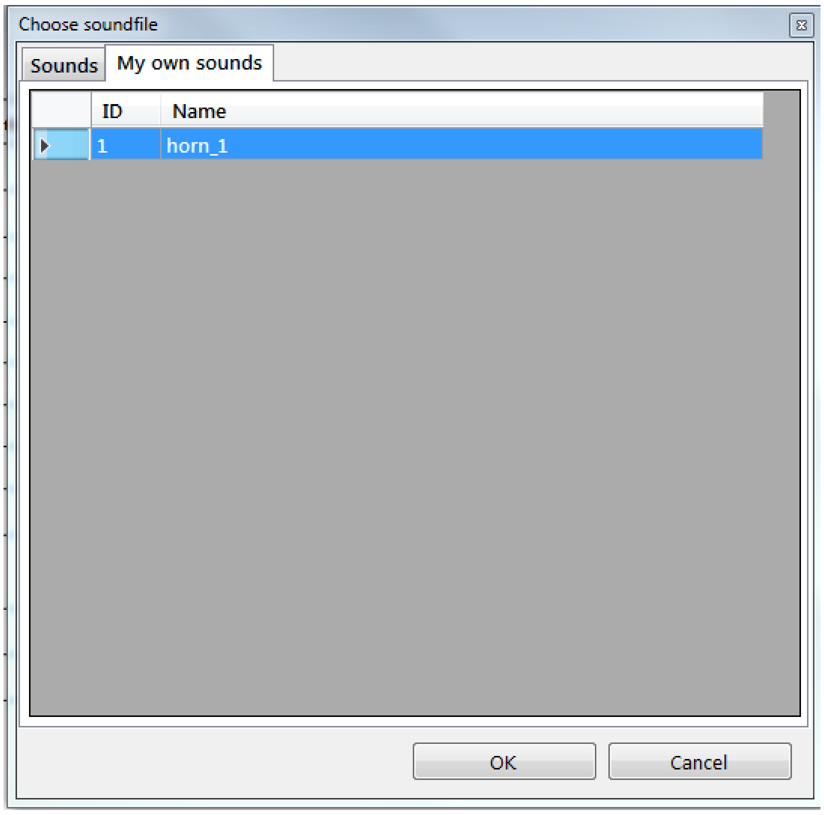
Download: http://www.audacityteam.org/download/

Manual: Audacity eng.pdf

Choose path	With the button **"…“** (open) you can set the path to the sound library
add files …	You can add sound files to the library with the button add files

Once the sound file has been selected, it will be copied to the previously selected folder with a number from 1 to 999 attached to the end of the file name.

If a sound file is selected in the configuration window, the previously added sound files appear in the tab “My own sounds” with a number attached to the end of the name.



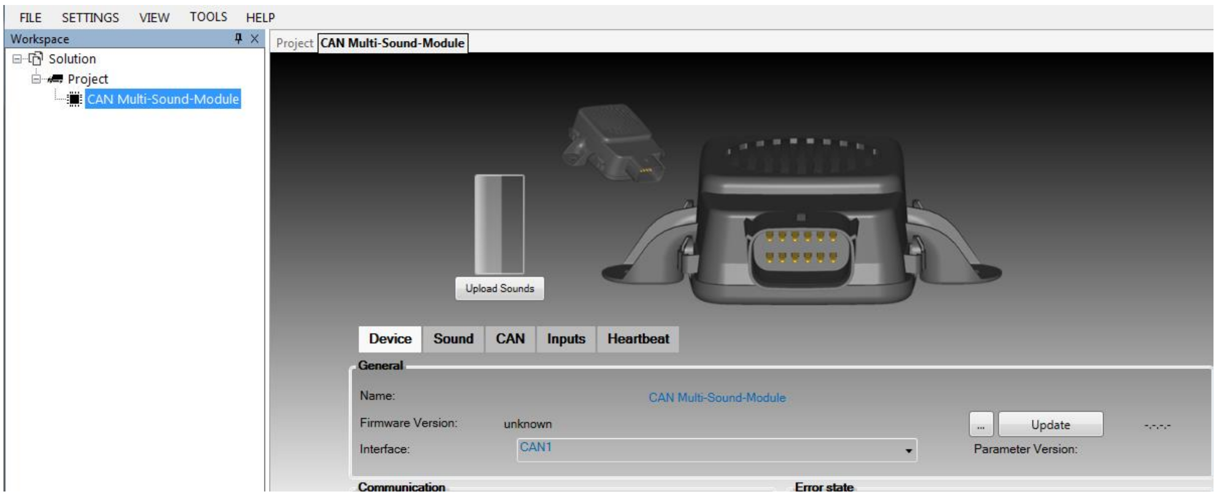
ID	Name
1	horn_1

7 Operation of CAN Multi-Sound-Module

7.1 Parameterization of CAN Multi-Sound-Module

You can choose the CAN Multi-Sound-Module in the workspace.

After clicking on the tab, it opens itself in the middle of the Toolchain



Name: CAN Multi-Sound-Module
Firmware Version: unknown
Interface: CAN1
Parameter Version: -.-.-.-

„Device“

Here, general settings are made which affect the entire device.

„Sound“

Here you can set the sound files that the module should save.

„CAN“

The assignment of the sounds to the CAN messages can be made here.

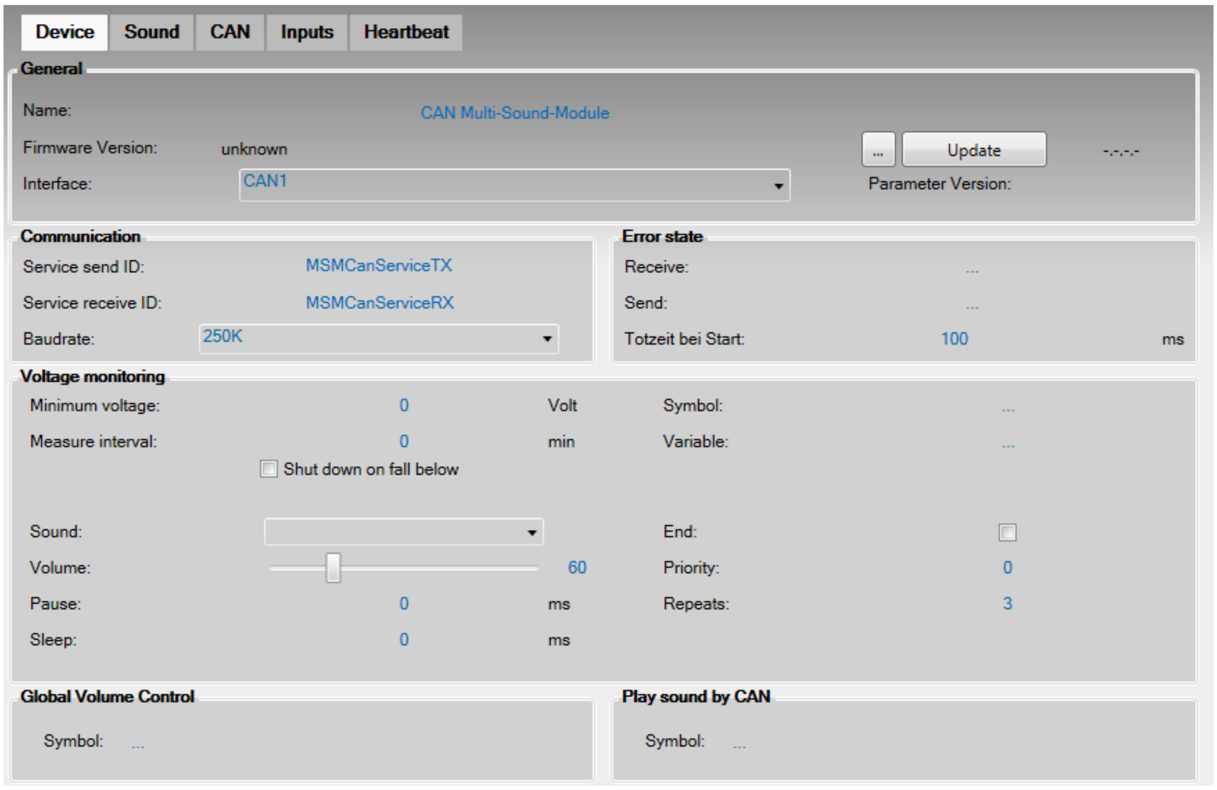
„Inputs“

The assignment of the sounds to the inputs can be made here.

„Heartbeat“

The CAN Multi-Sound-Module is able to send a periodic heartbeat as CAN message. The heartbeat can signal different states and information of the module.

7.1.1 Tab: „Device“



General

Name: CAN Multi-Sound-Module
Firmware Version: unknown
-.-.-.-
Interface: CAN1
Parameter Version:

Communication

Service send ID: MSMCanServiceTX
Service receive ID: MSMCanServiceRX
Baudrate: 250K

Error state

Receive: ...
Send: ...
Totzeit bei Start: 100 ms

Voltage monitoring

Minimum voltage: 0 Volt
Measure interval: 0 min
☐ Shut down on fall below
Symbol: ...
Variable: ...
Sound:
Volume: 60
Pause: 0 ms
Sleep: 0 ms
End: ☐
Priority: 0
Repeats: 3

Global Volume Control

Symbol: ...

Play sound by CAN

Symbol: ...

General

Name	Name of the product.
Firmware Version	Is automatically changed to the corresponding firmware version, as soon as a device has been read with the buttons "..." (Open) and "Update" the firmware of the “CAN Multi-Sound-Module” can be updated
Interface	Select a CAN bus that the CAN Multi-Sound-Module should use
Parameter Version	Free versioning as text or number for the data record.

Communication

Service send ID	this send ID is responsible for finding the keypad on the CAN-Bus in order to update its parameters
Service receive ID	this receive ID is responsible for finding the keypad on the CAN-Bus in order to update its parameters
Baud rate	setting of the CAN-Bus bit rate used by the keypad • the maximum speed is 1000 kbit/s • the minimum speed is 20 kbit/s

Error state

Request	CAN-ID, which transmits any errors (as a CAN message).
Send	CAN-ID to which the error condition is sent.
Dead time at start	Time that elapses until indicating a present error.

structure of the CAN message for requesting the fault condition:

Name	**Byte 0**	**Byte 1**		**Byte 2**		**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
Bit position	0..7	8- 14	15	16	17..23	24..39		-	-	-

Bit position	**Error**	**Value**
0 – 14 = x	Timeout by ID x	0 -> no Timeout 1 -> Timeout
15	Was not able to send a CAN Message	0 -> no Error 1 -> Error
16	Voltage monitoring	0 -> no undervoltage 1 -> undervoltage
24 - 39	Current voltage when the device is in undervoltage mode.	

Voltage monitoring

Minimum voltage	Defines a minimum value, when the voltage goes under this value the CAN Multi-Sound-Module Stand-by-Modus. You can enter two decimal places.
Measure interval	After the configured time is elapsed, the module measures the voltage. The range of values goes from 0 (which means no voltage monitoring) to 10 minutes. **Please note that every measurement needs power. This may rundown the battery over a longer period.**
Shut down on fall below	The "sleep mode" function is activated when the undervoltage limit is reached. To wake up the keypad from "sleep mode", terminal 30 has to be re-applied. Alternatively, the terminal 15 can be controlled via the hardware input 1. The prerequisite is that hardware input 1 has been parameterized as "sleep mode".
Symbol	ID on which the CAN message is send with the supply voltage value.
Variable	16 Bit Value on which the supply voltage is send in **mV**.
Sound	When the voltage falls under the minimum an alert is played.

Volume	The volume has 255 steps. You can enter a number or use the slider. When the value is **0,** the global volume is set by a CAN message. This message is in the tab „Device“.	
Pause	Time until the sound is played again. The number of repetitions has to be defined in milliseconds.	
Sleep	Time until the Module starts playing the sound in ms.	
End	☐	Interrupts playing when sound with higher priority is played.
	☑	Does **not** interrupt playing when sound with the **same** priority is played.
	■	Does **not** interrupt playing when sound with a **higher** priority is played.
Priority	Set the priority of the sound. The value range goes from 0 to 255 (255 is the highest priority).	
Repetitions	A sound can be repeated 1 to 255 times. Zero means infinite repetitions.	

Global Volume Control

For all sounds, where the volume is 0, the volume can be set with an global CAN message.

Structure of the message for global volume :

Name	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
bit position	0..7	-	-	-	-	-	-	-

bit position	Description
0 – 7	Decimal Volume from 0 – 255

Symbol — ID, on which the Volume is send.

Play sound by CAN

Structure of the message to play a sound:

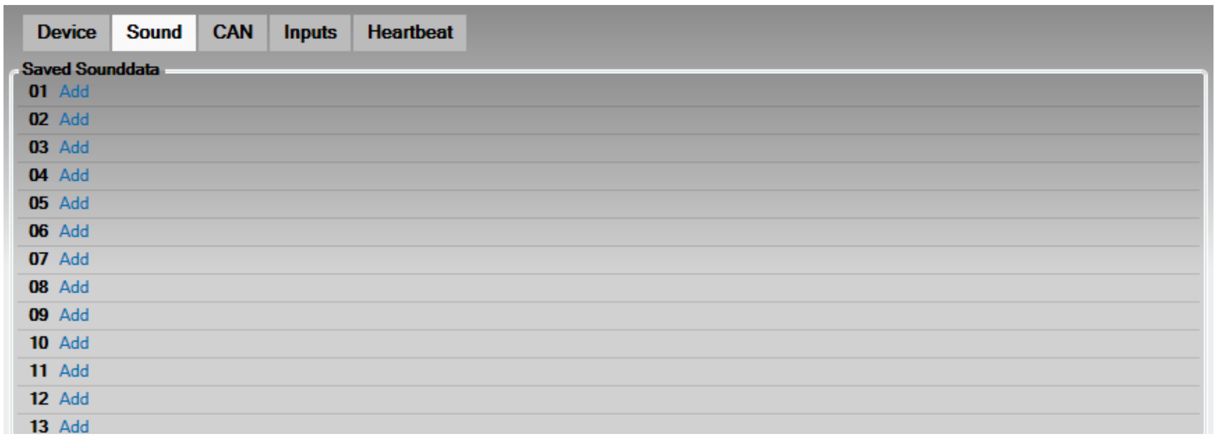
Name	Description
Byte 0	Not used
Byte 1	Line number of the sound 0-49
Byte 2	Volume control 0-255
Byte 3	Pause between repetitions in 100 ms
Byte 4	Number of repetitions
Byte 5	Play sound to end 0x01 Play sound to end with higher priority 0x02
Byte 6	Priority of the sound 0-255
Byte 7	Not used

Symbol

ID, on which the sound is played.

7.1.2 Tab: „Sound“

Here are the sound files for the memory of the CAN Multi-Sound-Modules selected.

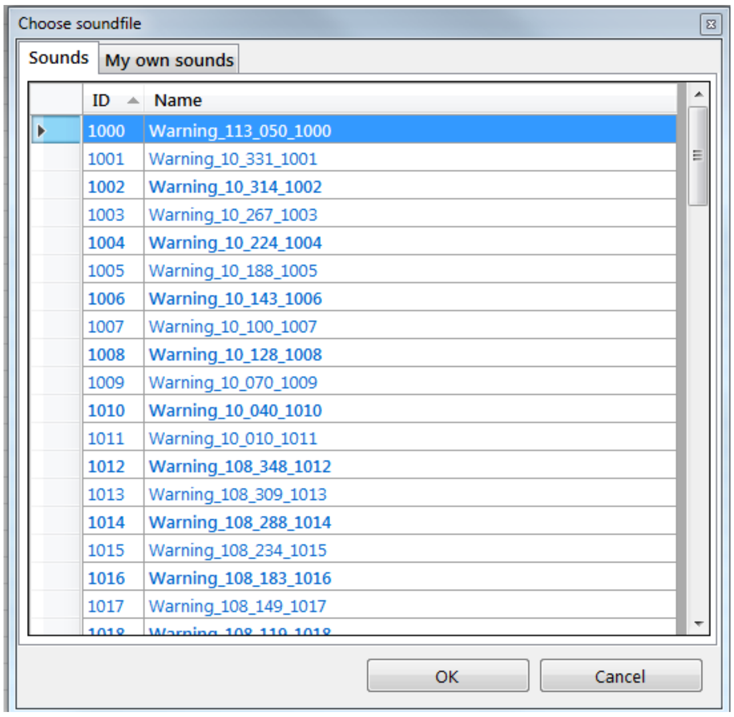


Saved Sounddata

01	Add
02	Add
03	Add
04	Add
05	Add
06	Add
07	Add
08	Add
09	Add
10	Add
11	Add
12	Add
13	Add

You can choose 50 different sounds, which can be parameterized in other tabs.

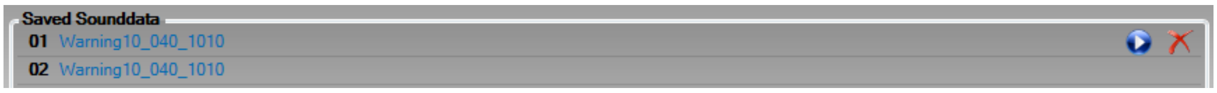
When you click on the caption Add, the sound library will be opened. You can now choose a sound for each line. The Toolchain includes a lot of sounds but when they are not enough you can add your own sounds. You can find the instruction under “integrating own sounds”.



ID	Name
1000	Warning_113_050_1000
1001	Warning_10_331_1001
1002	Warning_10_314_1002
1003	Warning_10_267_1003
1004	Warning_10_224_1004
1005	Warning_10_188_1005
1006	Warning_10_143_1006
1007	Warning_10_100_1007
1008	Warning_10_128_1008
1009	Warning_10_070_1009
1010	Warning_10_040_1010
1011	Warning_10_010_1011
1012	Warning_108_348_1012
1013	Warning_108_309_1013
1014	Warning_108_288_1014
1015	Warning_108_234_1015
1016	Warning_108_183_1016
1017	Warning_108_149_1017
1018	Warning_108_110_1018

When you click on a sound file it is played.

After you selected a sound file, you can play it again on the PC with the blue „Play“-Button.



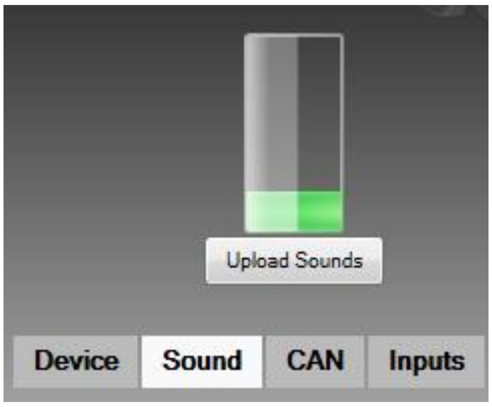
Saved Sounddata

01 Warning10_040_1010

02 Warning10_040_1010

To remove a sound from the list you have to click on the red **X** on the right side.

The more sounds are defined the fuller becomes the memory. You can view the memory in the graph-ic.

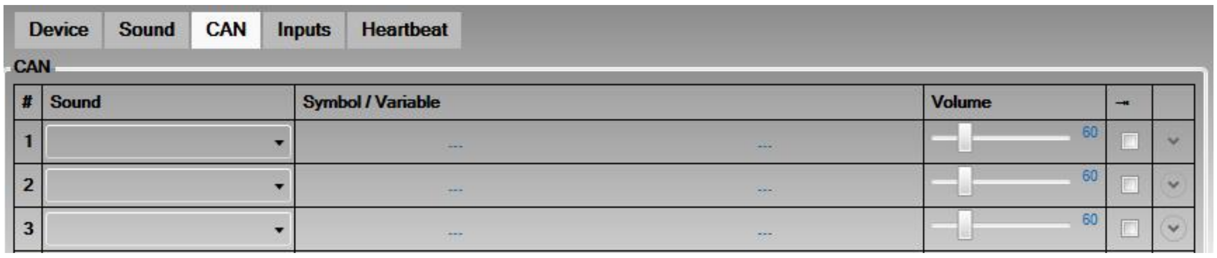


The green bar shows the used memory.

The more memory is used the longer it takes to write the data on the module. The Toolchain uses the CAN bus for the transmission. In the example is the memory usage at 25%, the transmission would take about 10 minutes.

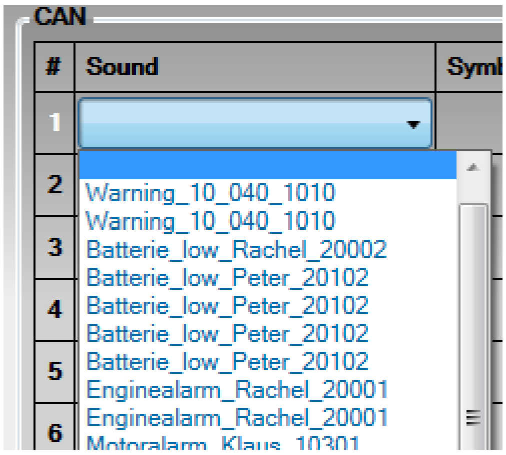
7.1.3 Tab: „CAN“

In the tab CAN you can match sound files to CAN messages. You can match up to 15 sounds here.



#	Sound	Symbol / Variable		Volume	⇥	
1		...	...	60	☐	
2		...	...	60	☐	
3		...	...	60	☐	

Sound Under the item sound you can choose one of the predefined sounds.



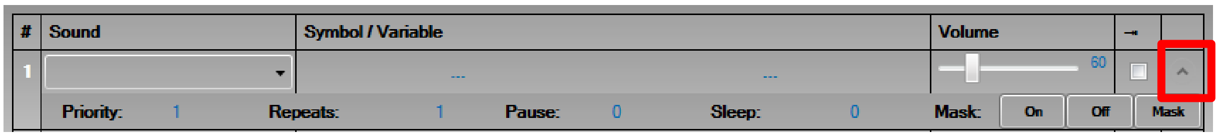
Symbol / Variable When you click on „…“you can choose a symbol and variable in the Symbol Manager for the sound.

Volume The volume has 255 steps. You can enter a number or use the slider. When the value is **0,** the global volume is set by a CAN message. This message is in the tab „Device“.

⇥ Priority

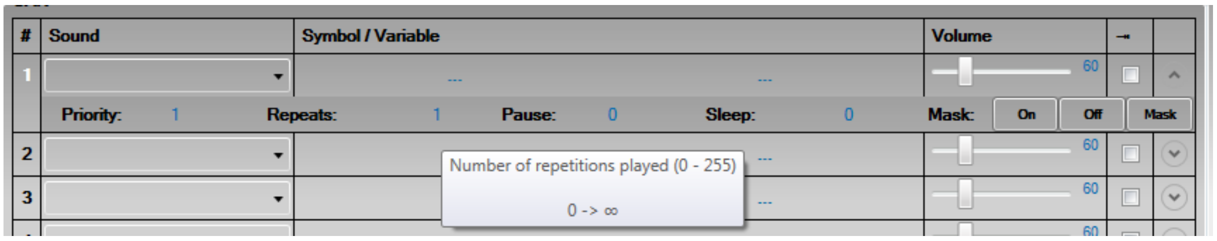
☐	Interrupts playing when sound with higher priority is played.
☑	Does **not** interrupt playing when sound with the **same** priority is played.
■	Does **not** interrupt playing when sound with a **higher** priority is played.

You can expand more parameter with the arrow on the right side.



Priority	Set the priority of the sound. The value range goes from 0 to 255 (255 is the highest priority).
Repetitions	A sound can be repeated 1 to 255 times. Zero means infinite repetitions.
Pause	Time until the sound is played again. The number of repetitions has to be defined in milliseconds.
Sleep	Time until the Module starts playing the sound in ms.
Mask	OR – operation of the mask bits with the bits in the CAN message
On	Bit, to start playing the sound.
Off	Bit, to stop playing the sound

When you place the mouse over the items you get additional information and help.

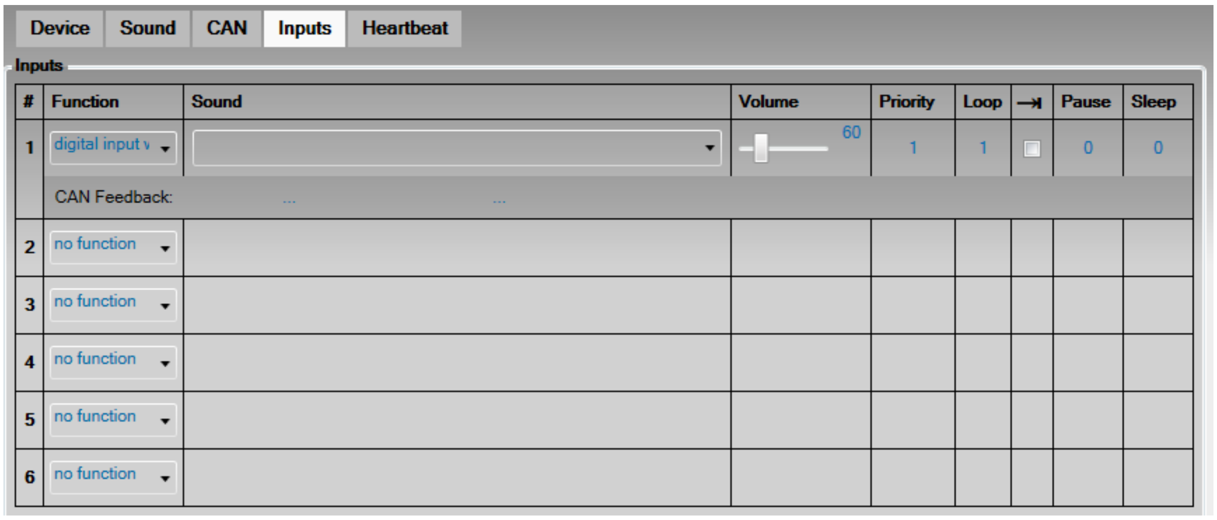


Number of repetitions played (0 - 255)

0 -> ∞

7.1.4 Tab: „Inputs“

In the tab Inputs you can match sound files to hardware inputs. You can match up to 6 sounds here.



#	Function	Sound	Volume	Priority	Loop	⇥	Pause	Sleep
1	digital input v		60	1	1	☐	0	0
	CAN Feedback:							
2	no function							
3	no function							
4	no function							
5	no function							
6	no function							

Function	Every hardware input may have a different parameterization.
no function	The input is not used.

digital input	Input state can be sent on the CAN bus to inform other CAN devices
	High = 1 level
	Low = 0 level
	CAN-message is selected in the line CAN feedback.

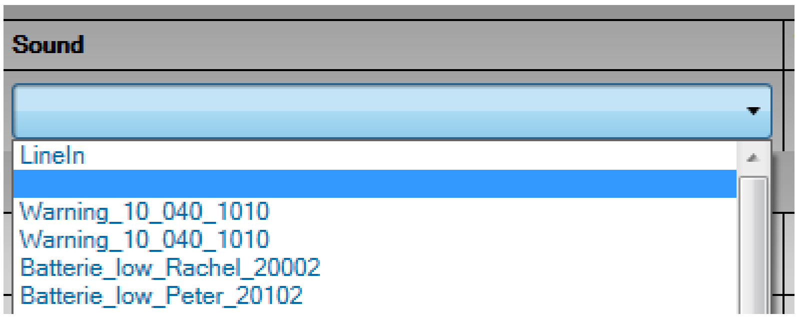
digital input inverted	Input state can be sent on the CAN bus to inform other CAN devices. Inputs are inverted.
	High = 0 level
	Low = 1 level
	CAN-message is selected in the line CAN feedback.

digital input with sound	Input state can be sent on the CAN bus to inform other CAN devices High = 1 level Low = 0 level CAN-message is selected in the line CAN feedback. When the input is on 1 level, the module can play a sound.

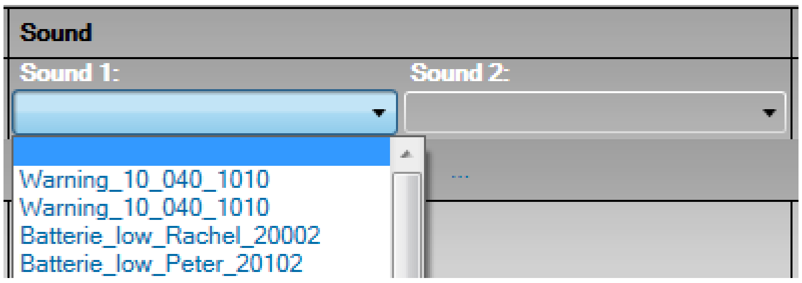
digital input inverted with sound	Input state can be sent on the CAN bus to inform other CAN devices. Inputs are inverted. High = 0 level Low = 1 level CAN-message is selected in the line CAN feedback. When the input is on 1 level, the module can play a sound.
bus quiet on falling edge	The module does not send any messages on the bus when the input level falls from 1 to 0.
bus quiet on raising edge	The module does not send any messages on the bus when the input level rises from 0 to 1.
sleep mode on falling edge	The module goes in sleep mode (low energy usage) when the input level falls from 1 to 0.
sleep mode on raising edge	The module goes in sleep mode (low energy usage) when the input level rises from 0 to 1.
mute on falling edge	Aborts playing when level fall from 1 to 0. Input state can be sent on the CAN bus to inform other CAN devices CAN-message is selected in the line CAN feedback.
mute on raising edge	Aborts playing when level rise from 0 to 1. Input state can be sent on the CAN bus to inform other CAN devices CAN-message is selected in the line CAN feedback.
Indicator	A blinker has two different levels. In the digital measurement technique one speaks of a "falling edge" (switching off), this would be the transition from "1" to "0" and from a "rising edge" (switching on); this would be a level change from "0" to "1 ". raising edge = Sound 1 falling edge = Sound 2 Input state can be sent on the CAN bus to inform other CAN devices CAN-message is selected in the line CAN feedback.

Sound

Under the item sound you can choose one of the predefined sounds.



For the indicator you have to match two sounds.



CAN Feedback

When you click on „**…**“the Symbol Manger opens. There you can choose a message which will be transmitted when the input condition is fulfilled.

Volume

The volume has 255 steps. You can enter a number or use the slider. When the value is **0,** the global volume is set by a CAN message. This message is in the tab „Device“.

Priority

Set the priority of the sound. The value range goes from 0 to 255 (255 is the highest priority).

Repetitions

A sound can be repeated 1 to 255 times. Zero means infinite repetitions.

⇥ Priority

☐	Interrupts playing when sound with higher priority is played.
☑	Does **not** interrupt playing when sound with the **same** priority is played.
■	Does **not** interrupt playing when sound with a **higher** priority is played.

Pause

Time until the sound is played again. The number of repetitions has to be defined in milliseconds.

Sleep

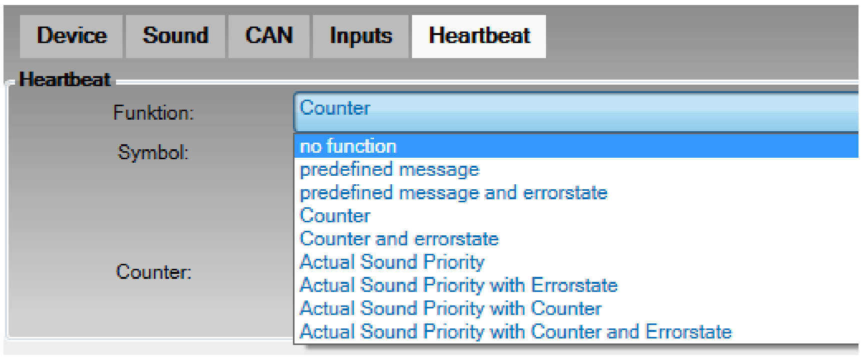
Time until the Module starts playing the sound in ms.

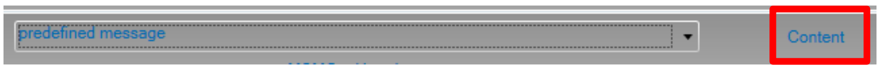
7.1.5 Tab: „Heartbeat“

In this tab is defined whether and how the „Heartbeat“ is send on the CAN-Bus

Heartbeat

Function — The combo box shows all available functions that are provided.


Device
Sound
CAN
Inputs
Heartbeat
Heartbeat
Funktion:
Symbol:
Counter:
Counter
no function
predefined message
predefined message and errorstate
Counter
Counter and errorstate
Actual Sound Priority
Actual Sound Priority with Errorstate
Actual Sound Priority with Counter
Actual Sound Priority with Counter and Errorstate

no function	Heartbeat is not used.
predefined message	Message can be freely defined. 
predefined message and errorstate	Message can be freely defined. You can use an additional byte to transmit the errorstate. You have to request the errorstate. Settings are in the tab “Device”.
Counter	You can define one byte as counter. Every time the module sends the heartbeat the counter is increased by one.
Counter and error-state	You can configure one byte for the errorstate and one for the counter. Every time the module sends the heartbeat the counter is in-creased by one. You have to request the errorstate. Settings are in the tab “De-vice”.
Actual Sound Priority	You can configure one byte to send the actual sound priority, while playing a sound.
Actual Sound Priority with Errorstate	You can configure one byte to send the actual sound priority, while playing a sound and the errorstate. You have to request the errorstate. Settings are in the tab “De-vice”.
Actual Sound Priority with Counter	You can configure one byte to send the actual sound priority, while playing a sound and the counter. Every time the module sends the heartbeat the counter is in-creased by one.
Actual Sound Priority	You can configure one byte to send the actual sound priority,

with Errorstate with Counter	while playing a sound, the errorstate and the counter. You have to request the errorstate. Settings are in the tab “Device”. Every time the module sends the heartbeat the counter is increased by one.
Symbol	ID on which the CAN Multi-Sound-Module sends the heartbeat. (Can be set in the symbol manager)
Variable	Bit which defines the state change. (Can be set in the symbol manager)

7.2 Integrating own sounds

The Toolchain offers the possibility to integrate own sound files.
This is done as follows:

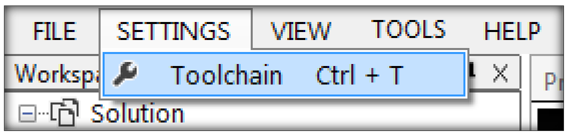
In the first step, the sound file has to be converted to the correct format. You can convert the file with free tools e.g.: Audacity.

The Toolchain accepts sound files in WAV format with the following parameters:

Channels:	Mono
Bitrate:	256 kbit/s
Sample rate:	16000 Hz
Bit depth:	16 bit PCM

If the sound files fulfil these characteristics, they can be integrated into the Toolchain.

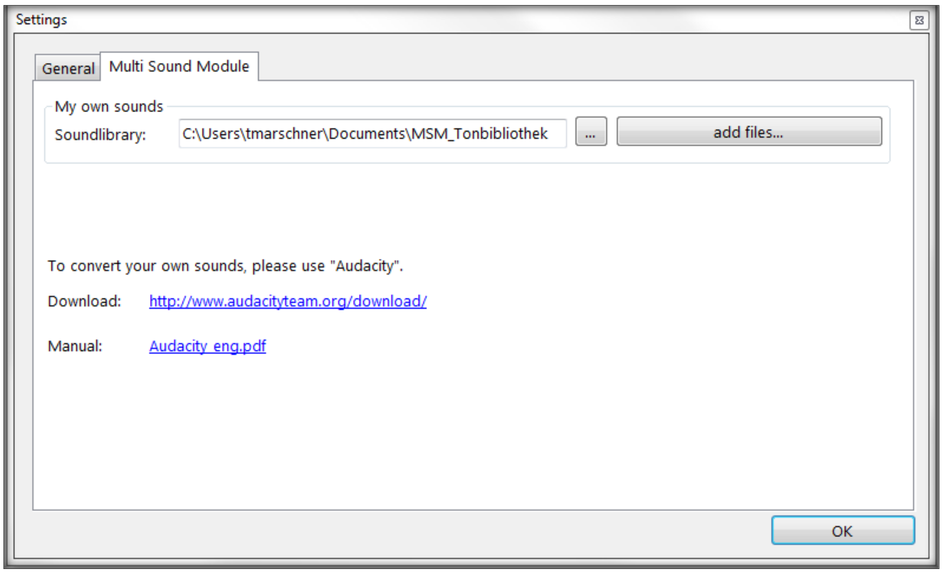
All settings of the Toolchain itself and of the individual devices can be found in the menu "SETTINGS" → "Toolchain".



Each sound file is copied to a user defined folder and the Toolchain attaches a number to the end of the file name. This number is responsible for the assignment of the sound files when a Multi-Sound-Module is read.

Note: This works only if the defined folder is not changed and the sound files were not copied manually to the folder.

The folder path is defined in the tab “Multi-Sound-Module” in the field called “Soundlibrary”.



C:\Users\tmarschner\Documents\MSM_Tonbibliothek

To convert your own sounds, please use "Audacity".

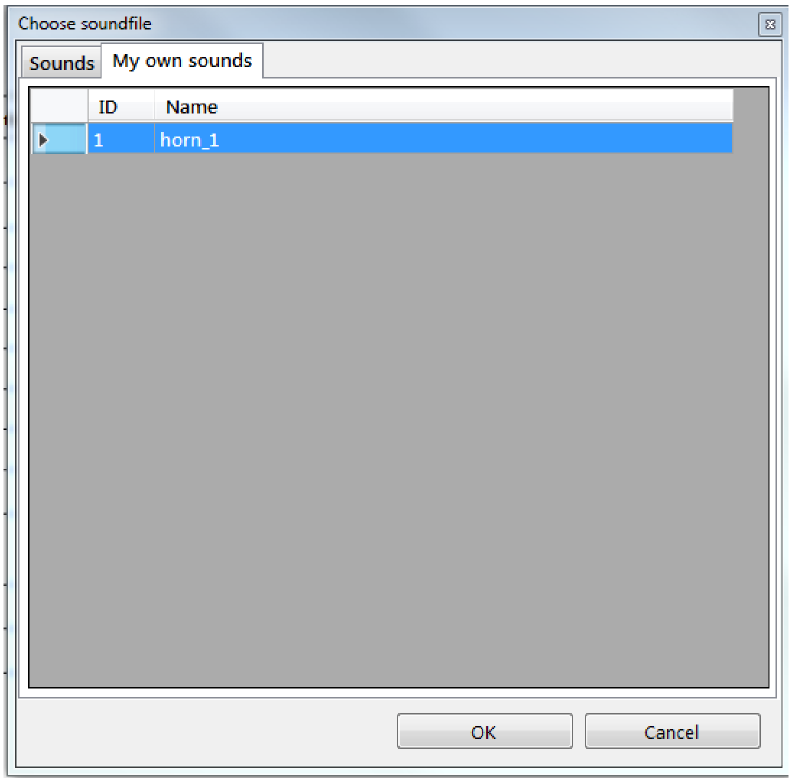
Download: http://www.audacityteam.org/download/

Manual: Audacity eng.pdf

Choose path	With the button **"…“** (open) you can set the path to the sound library
add files …	You can add sound files to the library with the button add files

Once the sound file has been selected, it will be copied to the previously selected folder with a number from 1 to 999 attached to the end of the file name.

If a sound file is selected in the configuration window, the previously added sound files appear in the tab “My own sounds” with a number attached to the end of the name.



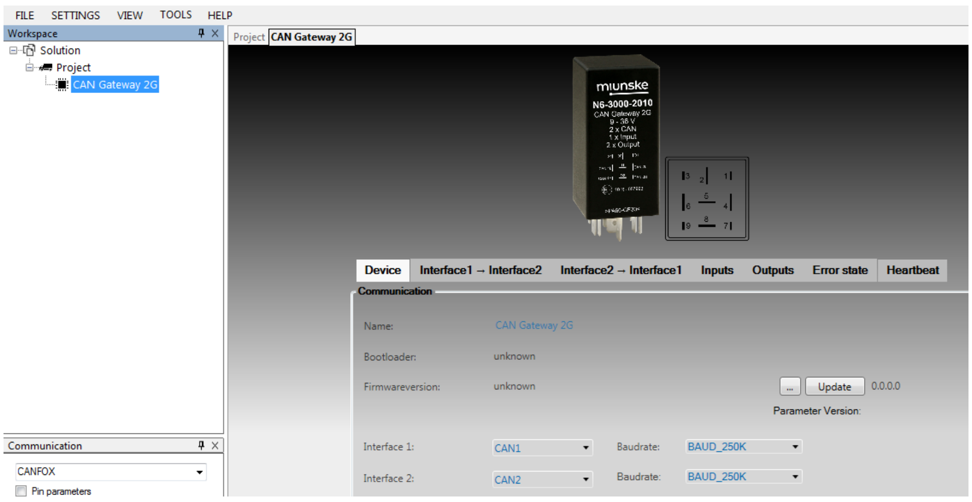
ID	Name
1	horn_1

8 CAN Gateway 2G

8.1 Parameterization of CAN Gateway 2G

The tab “CAN Gateway 2G” appears in the workspace.

After clicking on the tab, it opens itself in the middle of the Toolchain.



Name: CAN Gateway 2G

Bootloader: unknown

Firmwareversion: unknown

0.0.0.0

Parameter Version:

Interface 1: CAN1 Baudrate: BAUD_250K

Interface 2: CAN2 Baudrate: BAUD_250K

CANFOX

Pin parameters

There appear seven tabs:

„Device“

Here, general settings are made which affect the entire device.

„Interface1 --> Interface2“

Settings for communication between the interfaces

„Interface2 --> Interface1“

Settings for communication between the interfaces

„Inputs“

The Gateway has a hardware input, which can execute different functions.

„Outputs“

The hardware outputs can be parameterized here.

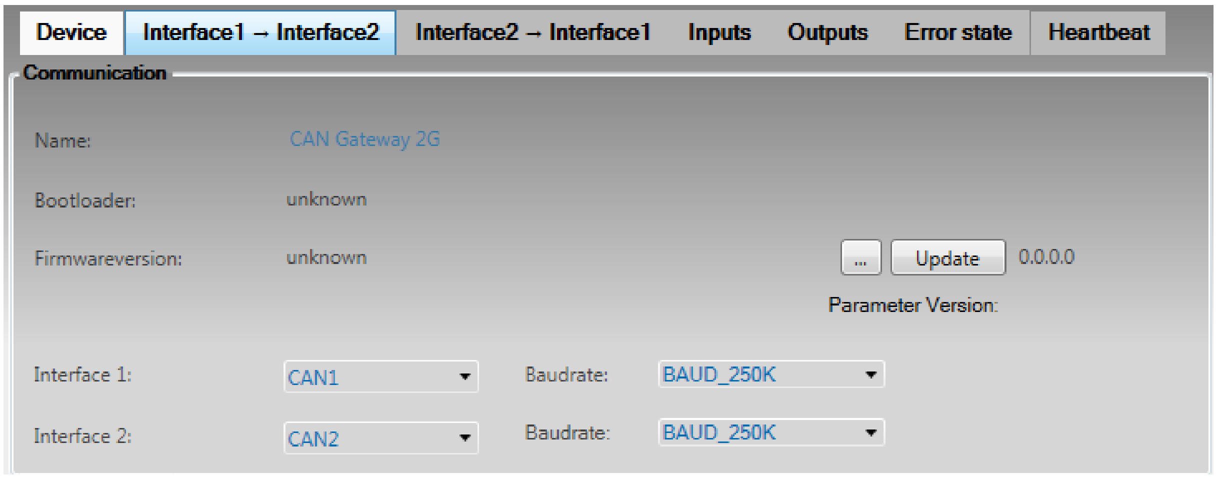
„Error state”

When the Gateway receives a (Request-) CAN message is can send an error state message on the same interface.

„Heartbeat”

The Gateway can send a periodic heartbeat, as a CAN message, on both interfaces.

8.1.1 Tab: „Device“



Name: CAN Gateway 2G

Bootloader: unknown

Firmwareversion: unknown

0.0.0.0

Parameter Version:

Interface 1: CAN1

Baudrate: BAUD_250K

Interface 2: CAN2

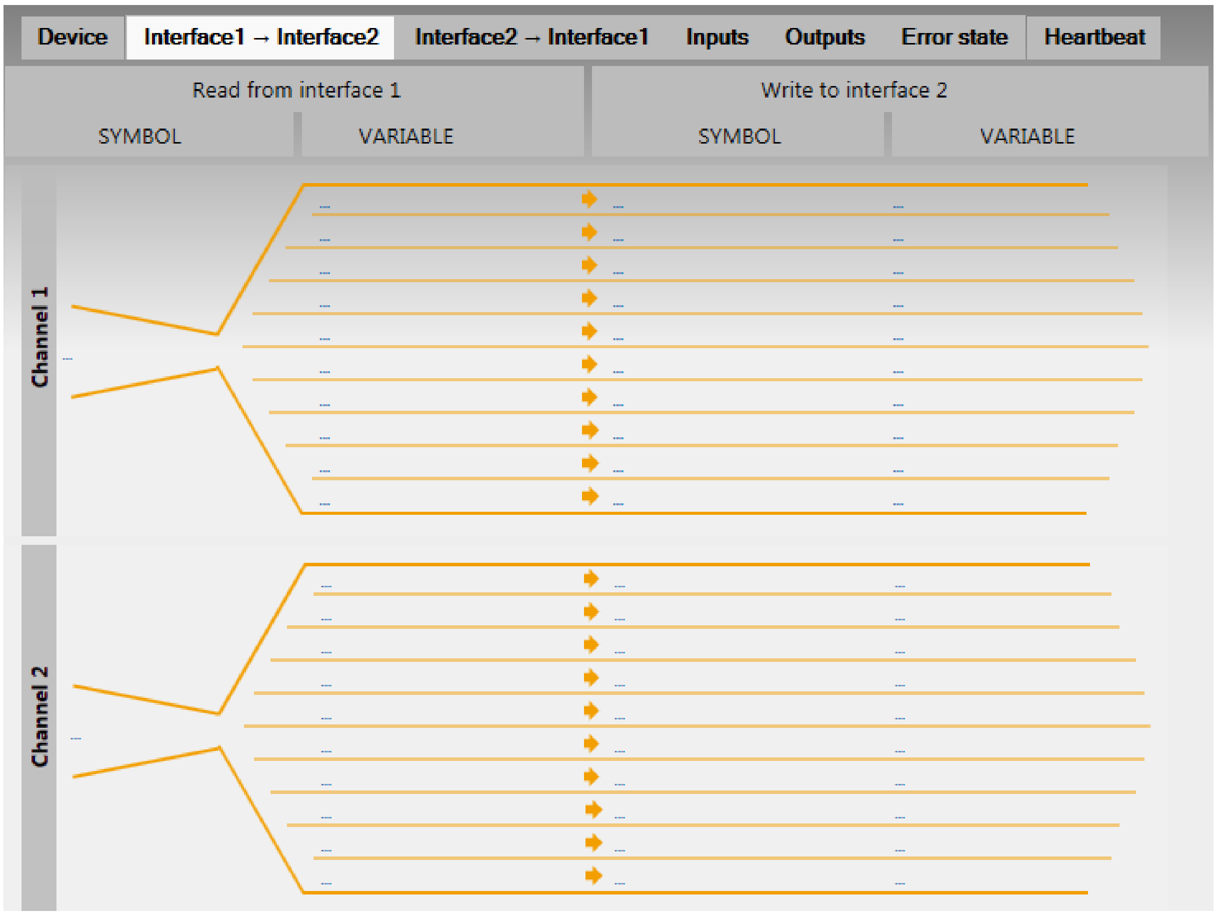
Baudrate: BAUD_250K

Name	the name of the “CAN Gateway 2G” can be changed here
Bootloader	shows the version of the bootloader, after a device has been read
Firmwareversion	is automatically changed to the corresponding firmware version, as soon as a device has been read • With the buttons "..." (Open) and "Update" the firmware of the “CAN Gateway 2G” can be updated
Parameter Version	Free versioning as text or number for the data record.
Interface 1	in the drop-down menu one has to select a CAN-Bus that will be used by interface 1
Baudrate	setting of the CAN-Bus bit rate used by the gateway interface 1
Interface 2	in the drop-down menu one has to select a CAN-Bus that will be used by interface 2
Baudrate	setting of the CAN-Bus bit rate used by the gateway interface 2

8.1.2 Tab „Interface1 --> Interface2“

In this tab all communication settings are made from one CAN-Bus to the other.

For this, there are 10 channels available. In each of these channels 10 different messages can be translated.



Clicking on a variable (“…”) in the left part of the table (**"Read from interface 1"**) opens the Symbol Manager. Then a variable for the previously selected CAN-Bus has to be defined in the Symbol Manager.

In the right part **"Write to interface 2"** a selection of a variable for the second interface has to take place again.

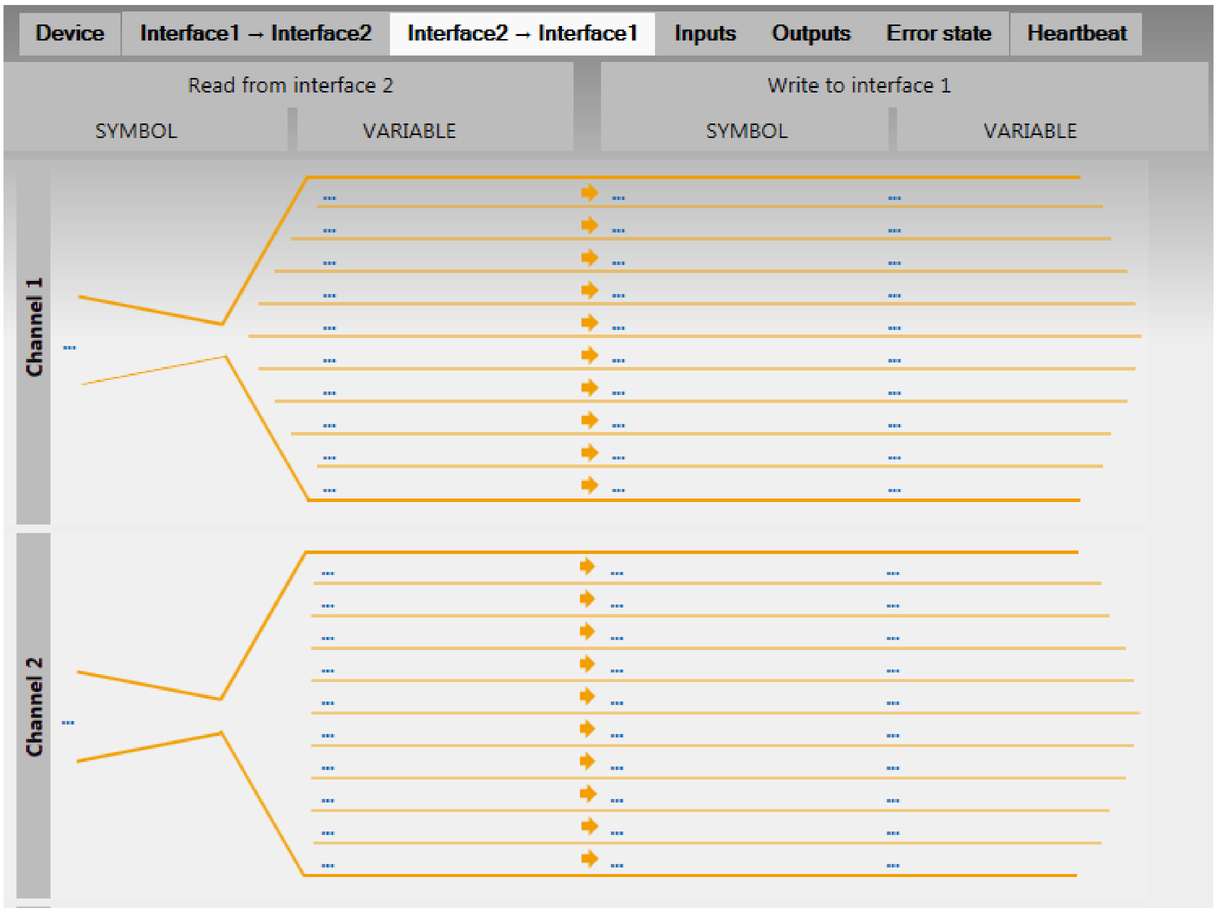
By clicking on the appropriate field of the variable ("..."), the Symbol Manager will be opened, in which one can choose the associated variable.

Writing to interface 2 is an OnChange event, messages are not sent permanently. This is done only once, when a message from interface 1 is received.

If one wants to permanently send the message on interface 2, an interval must be set in the Symbol Manager ("broadcast").

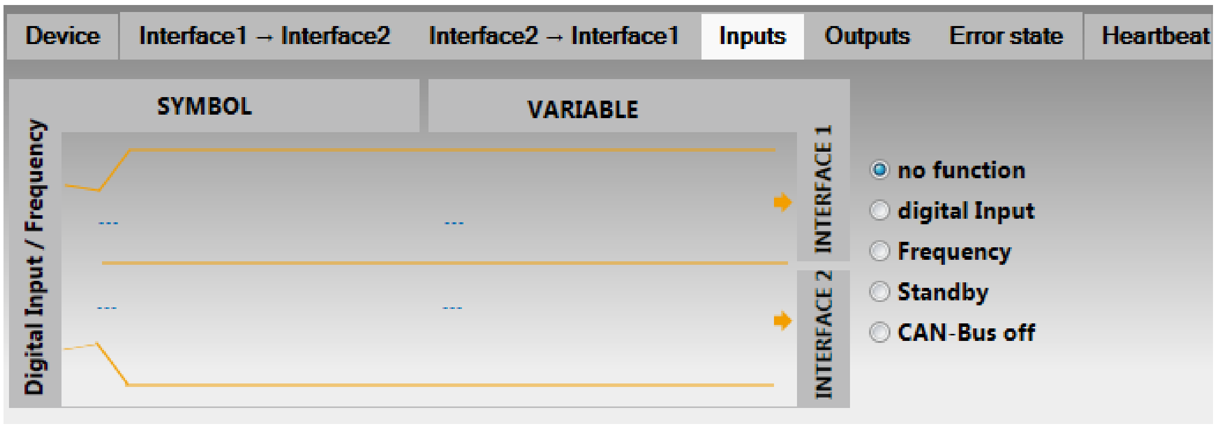
8.1.3 tab „Interface2 --> Interface1“

Basically it is the same mode of operation as in the "Interface1 --> Interface2" tab, with the difference that messages are transmitted from interface 2 to interface 1.



8.1.4 Tab „Inputs“

The “CAN Gateway 2G” has a hardware input that can be used with different functions. The configuration is done as follows.



[✓ no function] digital Input Frequency Standby CAN-Bus off

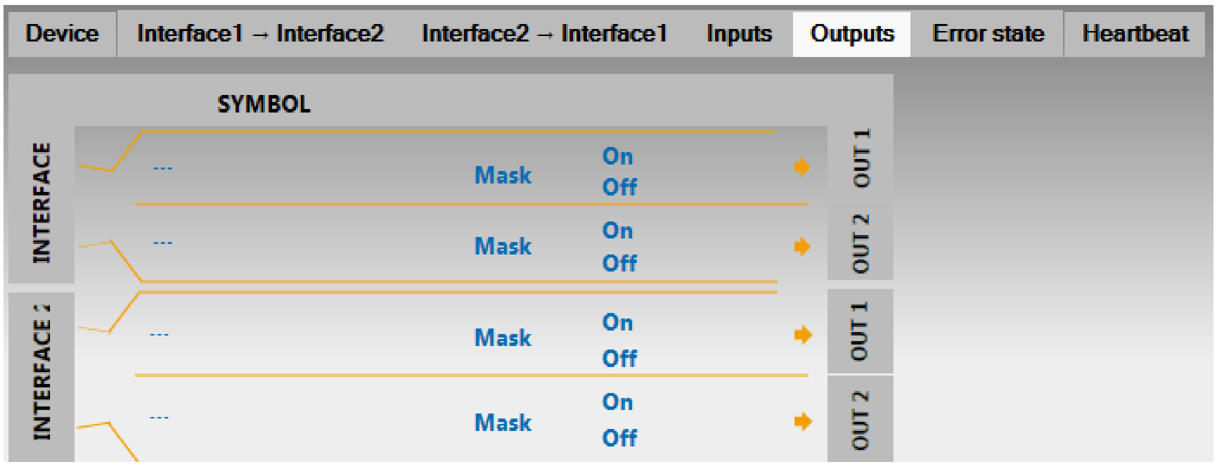
no function	Input is not used

digital Input	Input state can be sent on the CAN bus to inform other CAN devices
	High = 1 level
	Low = 0 level

Frequency	The input can evaluate frequencies between 4 Hz and 1 kHz, to transmit them on the CAN bus.
Standby	CAN Gateway 2G goes to Standby modus where it needs less power, when the input level rises from 0 to 1.
CAN-Bus off	CAN Gateway 2G does not send any messages when the input level rises from 0 to 1.
Symbol	ID on which the CAN message with the input information is send.
Variable	Bit which should represent the input change.

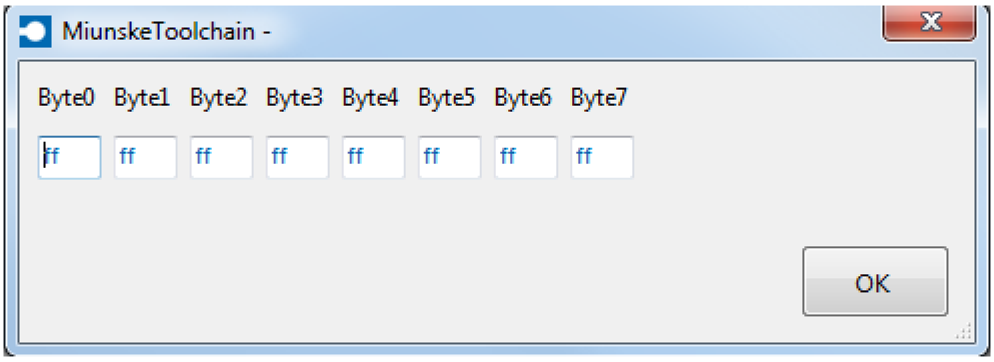
8.1.5 Tab „Outputs“

The two hardware outputs of the “CAN Gateway 2G” can be configured in this tab.



At first choose a symbol from the Symbol Manager with the „…“button.

Afterwards one can set a “Mask” for the bits that should be monitored for the output.



Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
ff	ff	ff	ff	ff	ff	ff	ff

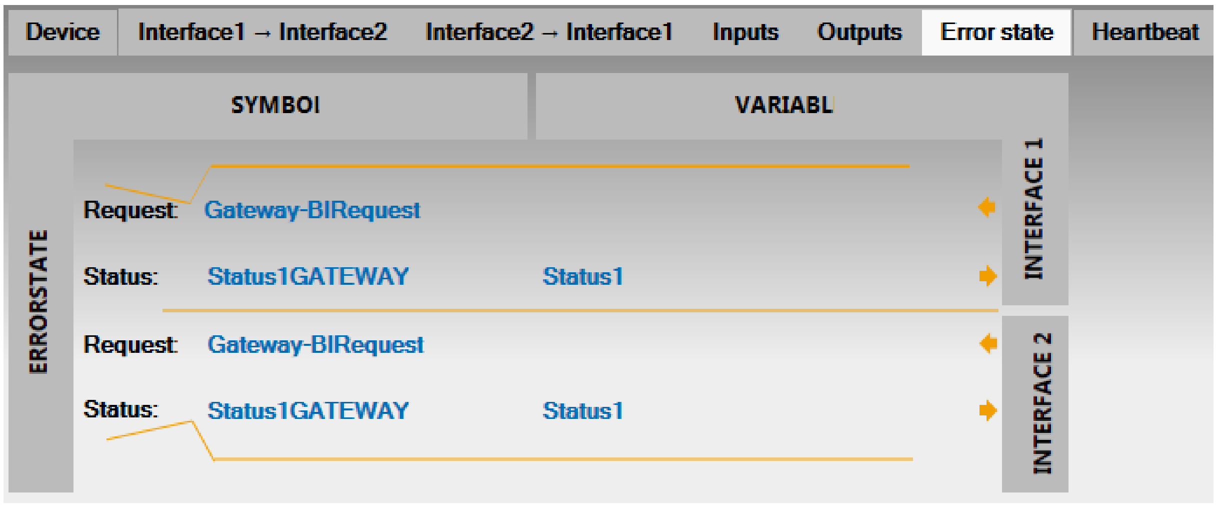
“On” is used to set the state the bits must have to generate a high signal on the output

“Off” is used to set the state the bits must have to generate a low signal on the output.

8.1.6 Tab „Error state“

By receiving a (request-) CAN message from another CAN node, the “CAN Gateway 2G” can transmit an error status message on the same interface from which the request came.

The Symbol / Variables have to be selected via the Symbol Manager.



Request: Gateway-BIRequest
Status: Status1GATEWAY Status1
Request: Gateway-BIRequest
Status: Status1GATEWAY Status1

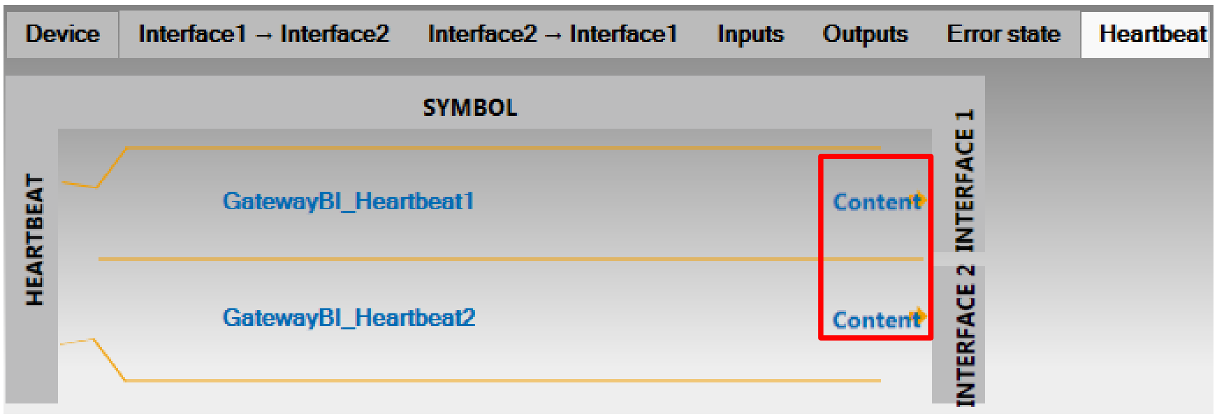
The structure of the error message:

Name	Byte 0	Byte 1		Byte 2		Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
bit position	0..7	8- 14	15	16	17..23	24..39		-	-	-

bit position	Error	Value
0 – 14 = x	Timeout of ID x	0 -> no Timeout 1 -> Timeout
15	Was not able to send a CAN Message	0 -> no error 1 -> error
16	Voltage monitoring	0 -> no undervoltage 1 -> undervoltage
24 - 39	Current voltage when the device is in undervoltage mode.	

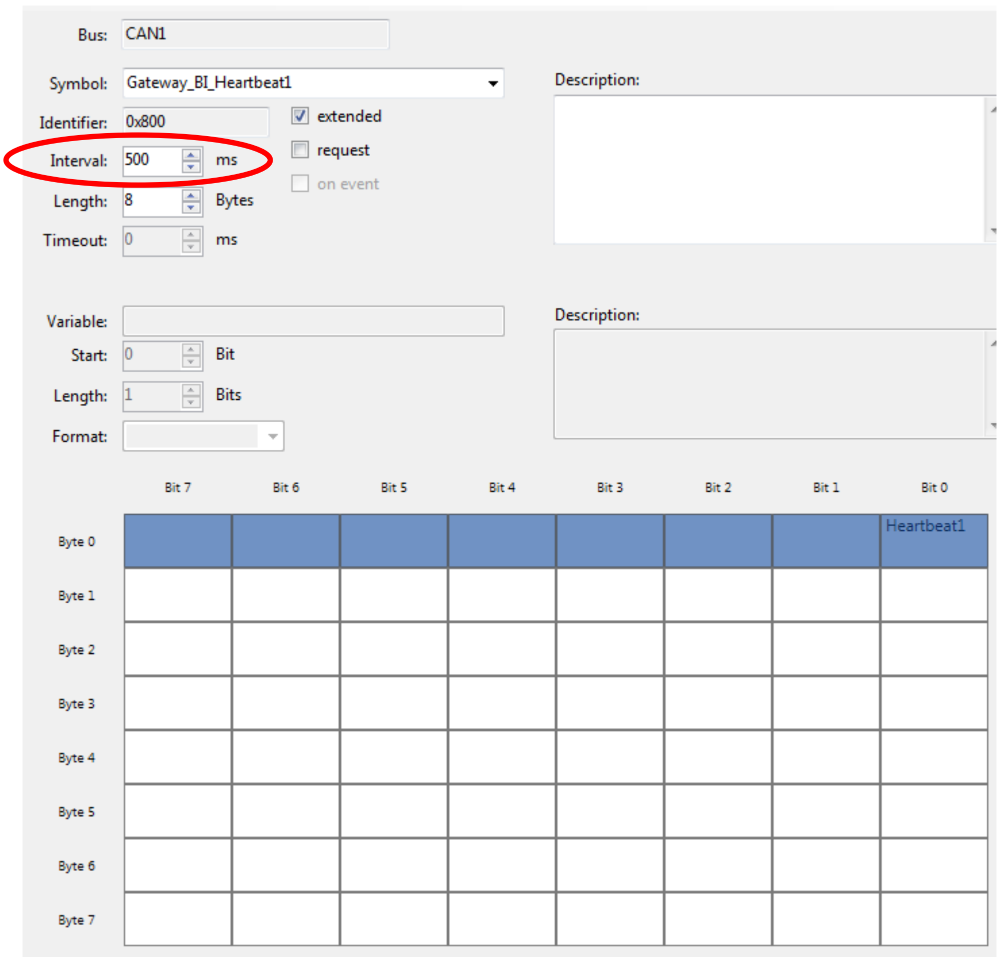
8.1.7 Tab „Heartbeat“

The “CAN Gateway 2G” is capable of transmitting a heartbeat in the form of a CAN message cyclically on both interfaces.



You can define the message content when you click on the label „Content“.

To send this message cyclically, an interval must be entered in the Symbol Manager.



Bus:	CAN1
Symbol:	Gateway_BI_Heartbeat1
Identifier:	0x800
Interval:	500 ms
Length:	8 Bytes
Timeout:	0 ms
extended	✓
request	
on event	
Variable:	
Start:	0 Bit
Length:	1 Bits
Format:	

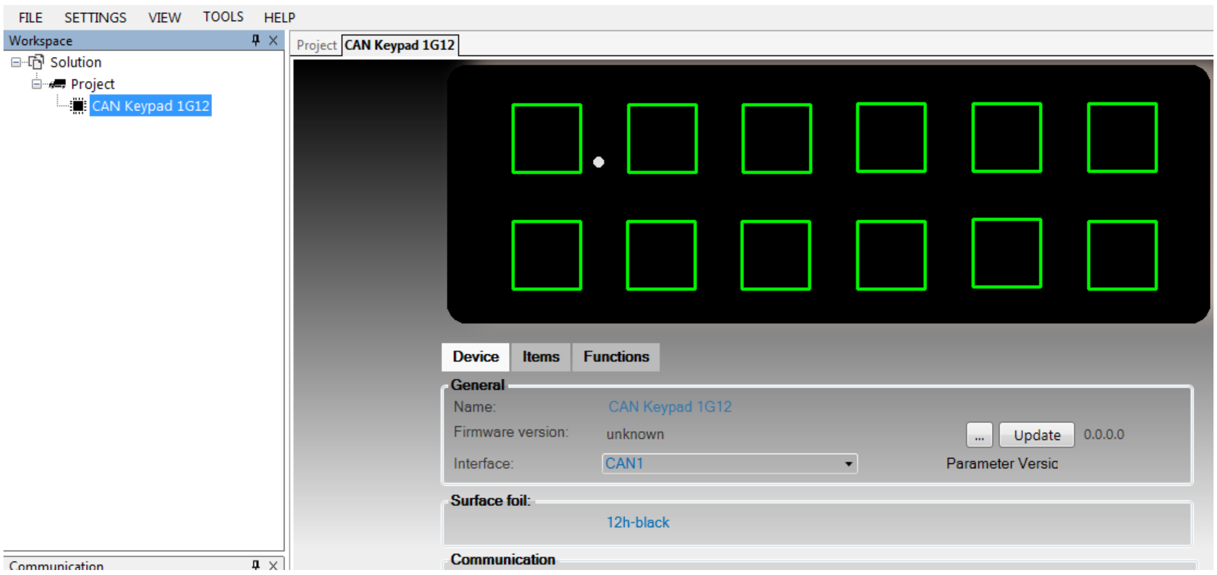
Description:

Description:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0								Heartbeat1
Byte 1								
Byte 2								
Byte 3								
Byte 4								
Byte 5								
Byte 6								
Byte 7								

9 Operation of CAN Keypad 1G 4 | 6 | 12

9.1 Parameterization of CAN Keypad 1G 4 | 6 | 12



General
Name: CAN Keypad 1G12
Firmware version: unknown
0.0.0.0
Interface: CAN1
Parameter Versic
Surface foil: 12h-black

The tab “CAN Keypad 1G 4 | 6 | 12” appears in the workspace.

After clicking on the tab, it opens itself in the middle of the Toolchain.

Now three tabs appear under the keypad schematic:

„Device“

Here, general settings are made which affect the entire device..

„Items“

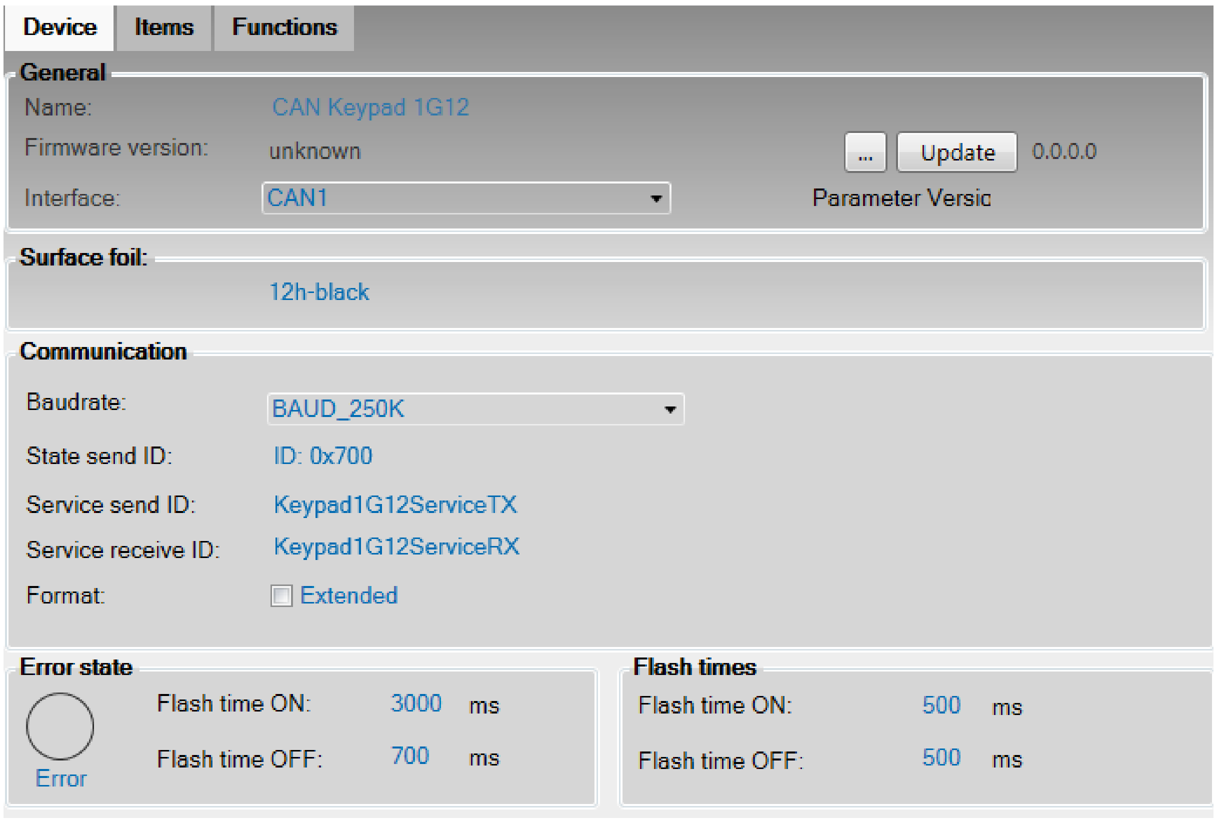
Here you can configure all settings that affect the buttons and displays.

„Functions“

Here are the settings for Location lighting, brightness, CAN-Bus monitoring, Sleep-Mode, CAN-Bus idle.

9.1.1 Tab: „Device“

General settings can be defined here which affect the whole device.



General
Name: CAN Keypad 1G12
Firmware version: unknown

0.0.0.0
Interface: CAN1
Parameter Versic

Surface foil:
12h-black

Communication
Baudrate: BAUD_250K
State send ID: ID: 0x700
Service send ID: Keypad1G12ServiceTX
Service receive ID: Keypad1G12ServiceRX
Format: ☐ Extended

Error state
Error
Flash time ON: 3000 ms
Flash time OFF: 700 ms

Flash times
Flash time ON: 500 ms
Flash time OFF: 500 ms

General

Name	the product name is noted here
Firmware version	is automatically changed to the corresponding firmware version, as soon as a device has been read • With the buttons "..." (Open) and "UPDATE" the firmware of the key-board can be updated
Interface	in the drop-down menu you have to select a CAN-Bus created within the project that will be used by the keypad
Parameter Version	Free versioning as text or number for the data record.

Surface foil

Here you can choose a graphic file that is used as background foil for the keyboard.

Communication

Baudrate	setting of the CAN-Bus bit rate used by the keypad • the maximum speed is 1000 kbit/s • the slowest speed is 20 kbit/s
State send ID	on this CAN-Bus ID the information of all the key states is transmitted

Byte	Bit	Description
0	7 - 0	Bit 7 = Key 1 Bit 0 = Key 8
1	15 - 12	Bit 15 = Key 9 Bit 12 = Key 12
1	11 - 8	Bit 11 = LED Key 1 Bit 8 = LED Key 4
2	16 - 23	Bit 23 = LED Key 5 Bit 16 = LED Key 12
3	31 – 24	Digital value for state ON value range 0 = MIN value range 255 = MAX
4	39 - 32	Digital value for state OFF value range 0 = MIN value range 255 = MAX
5	47 - 40	Digital value of the light sensor value range 0 = MIN value range 255 = MAX
6	55 - 48	Digital value of the measuring voltage „Umess[bit]“, measures the applied supply voltage via a voltage divider Ubat, calculation: Ubat[V] = 7 * Umess[bit] * 5V / 255bit + 0,7V, value range 0 = MIN value range 255 = MAX
7	63 - 56	errorstate, Bit56 = Digital status of input Bit57..60 = unused, Bit61 = voltage monitoring error, if the undervoltage limit is undershot Bit62 = light sensor, light sensor Hardware defect Bit63 = LED-Driver-Error, Driver Hardware defect

Service send ID	this send ID is responsible for finding the keypad on the CAN-Bus in order to update its parameters
Service receive ID	this receive ID is responsible for finding the keypad on the CAN-Bus in order to update its parameters
Format	Here you can switch between extended or standard format for „Service send ID“ and „Service receive ID“ • Standard → ID length is 11 bits • Extended → ID length is 29 bits

Error state

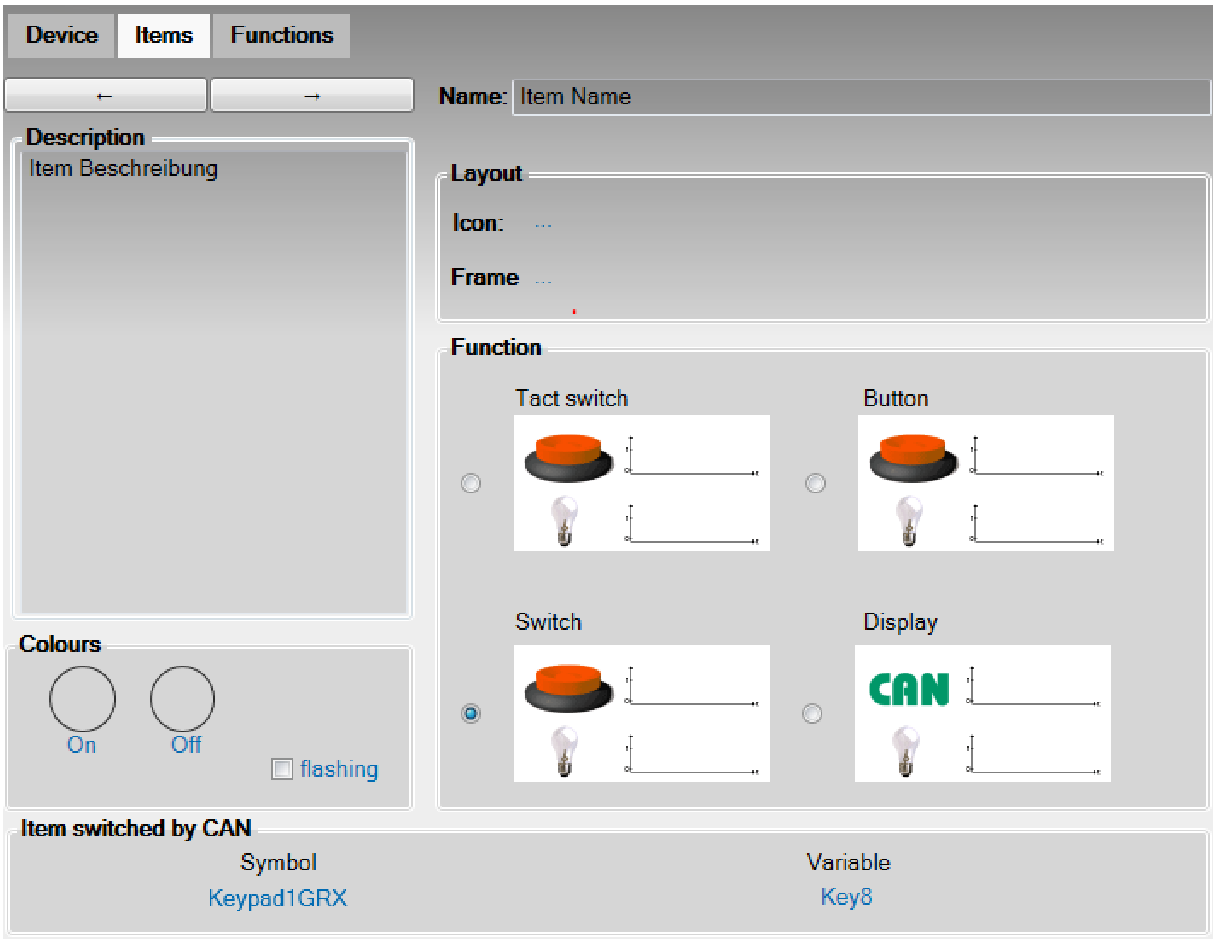
Error	When you click on the circle you can change the flash color that is shown on all switched keys when an error occurs.
Flash time ON	time in which the error color is displayed
Flash time OFF	time in which the keypad illumination is off during the error

Flash times

Flash time ON	time in which the flashing buttons show the color for the ON state
Flash time OFF	time in which the flashing buttons show the color for the OFF state

9.1.2 Tab „Items“

Here you can change all settings for each switching or display element. You can open the settings for a key, when you click on the keyboard.



Name: Item Name

Item Beschreibung

Function: Tact switch ( ) Button ( ) [✓ Switch] Display ( )

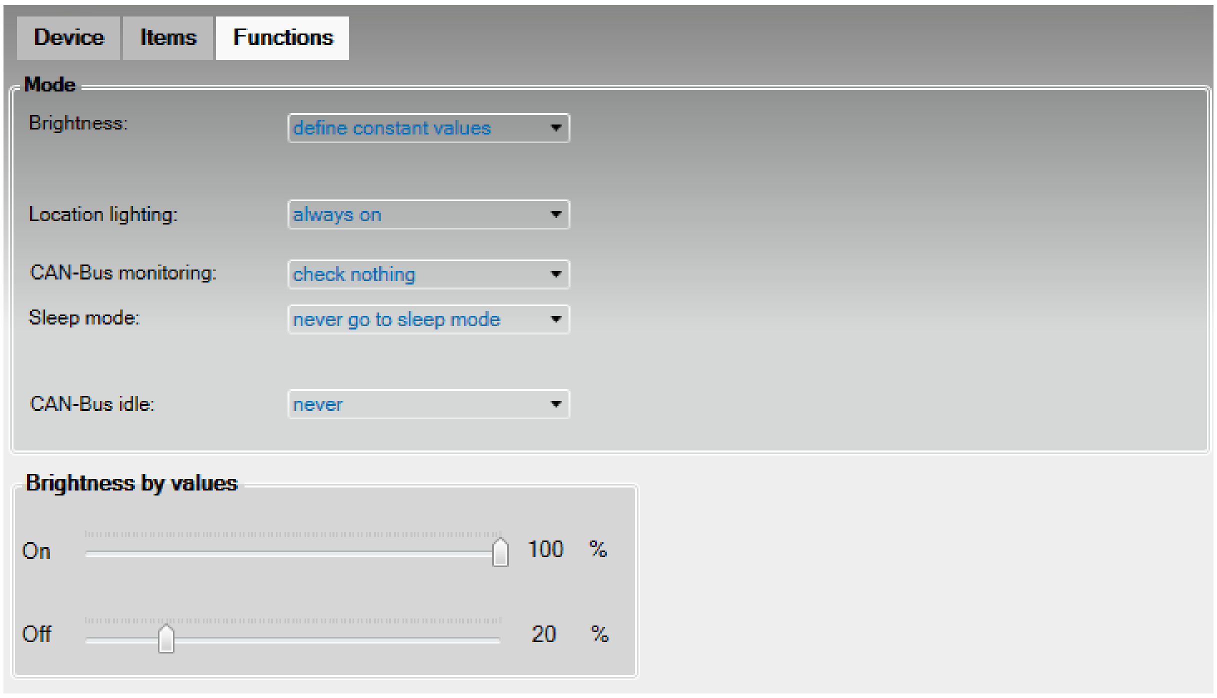
Colours: [ ] flashing

Symbol	Variable
Keypad1GRX	Key8

Name	for each key a name can be entered here (stored on the PC only and cannot be written on the keypad)
Description	a description field for each key (stored on the PC only and cannot be written on the keypad)
Layout	
Icon	When you click on it with the left mouse button, you can choose an icon from the icon folder.
Frame	You can choose a frame for every icon
Colors	
On	color which can be seen if the button is in the ON state
Off	color which can be seen if the button is in the OFF state
flashing	Set if the LED should switch between off and on (Settings in tab Device „Flash times“)
Function	
Tact switch	only a short pulse is sent, no matter how long the button is pressed
Button	as long as the button is pressed, the key is considered on
Switch	once pressed the switch is ON until it is pressed again, then it is OFF
Display	serves as a purely optical element
Item switched by CAN	
Symbol	ID of the CAN message that switches the LED on and off
Variable	Bit that switches the key on and off (in the Symbol Manager)

9.1.3 Tab „Functions“

In this tab, the parameters of functions can be changed, affecting the whole keypad.



Brightness:	define constant values
Location lighting:	always on
CAN-Bus monitoring:	check nothing
Sleep mode:	never go to sleep mode
CAN-Bus idle:	never

On	100	%
Off	20	%

Mode

Brightness	via the combo box one can control the brightness of the keyboard, different modes can be selected, depending on which input is used

by CAN message:	in the Symbol Manager one can select a 8 Bit variable for the ON and the OFF state that defines the brightness (Range: 0 - 255)
define constant values:	a percentage value will be set (100% equals 255), that can only be changed through writing the parameters on the keypad once again
adjust by brightness sensor:	the sensor control range determines at what ambient brightness the brightness of the fields should be changed; the ON / OFF control range defines how much the external change should influence the brightness of the keypad

Location lighting select the trigger that enables the location lighting

always on:	the location lighting cannot be switched off
on HIGH-Level at Input 1:	the location lighting is switched on by a rising edge (level changes from „0“ to „1“) on the digital input
on LOW-Level at Input 1:	the location lighting is switched on by a falling edge (level changes from „1“ to „0“)on the digital input
Level changed on Input1:	the location lighting changes its state when the signal at the digital input changes (level changes from „1“ to „0“ or „0“ to „1“)
if this BIT = 1:	by selecting a variable in the Symbol Manager the location lighting is turned on as long as the respective bit is set to 1
if this BIT = 0:	by selecting a variable in the Symbol Manager the location lighting is turned on as long as the respective bit is set to 0
BIT is changed:	by selecting a variable in the Symbol Manager the location lighting changes its state when the respective bit changes

CAN-Bus monitoring select whether and what should be checked on the CAN-Bus

check nothing:	the keypad sends its CAN messages and switches directly without feedback from the CAN-Bus
check CAN-Bus access:	the keypad checks whether other CAN nodes are available and if this is not the case, the defined error state is displayed
switch by CAN message:	the keypad waits for an acknowledgment of its sent message in form of another CAN message before it switches the respective key

Sleep mode select the trigger that enables the sleep mode

never go to sleep mode:	the keyboard never enters the sleep mode
go to sleep after this time:	the keypad goes into sleep mode after the set time without in- or outputting a signal
on HIGH-Level at Input 1:	the keypad enters the sleep mode when a rising edge (level changes from “0” to “1”) is read on the digital input 1
on LOW-Level at Input 1:	the keypad enters the sleep mode when a falling edge (level changes from “1” to “0”) is read on the digital input 1
Level changed on Input 1:	the keypad enters or exits the sleep mode when the signal at the digital input 1 changes

CAN-Bus idle	select the trigger that enables a kind of standby mode in which the keypad does not send any CAN messages

never:	the keypad never goes into CAN-Bus idle
on HIGH-Level at Input 1:	the keypad goes into CAN-Bus idle when a rising edge (level changes from “0” to “1”) is read on the digital input 1
on LOW-Level at Input 1:	the keypad goes into CAN-Bus idle when a falling edge (level changes from “1” to “0”) is read on the digital input 1
Level changed od Input1:	the keypad enters or exits the CAN-Bus idle mode when the signal at the digital input 1 changes
if this BIT = 1:	by selecting a variable in the Symbol Manager the CAN-Bus idle mode is turned on as long as the respective bit is set to 1
if this BIT = 0:	by selecting a variable in the Symbol Manager the CAN-Bus idle mode is turned on as long as the respective bit is set to 0
BIT is changed:	by selecting a variable in the Symbol Manager the key-pad enters or exits the CAN-Bus idle mode when the re-spective bit changes

9.2 Integrating own icons

The Toolchain already contains a large icon library out of the box.
In case the right icon for your application is not available, it is possible to create the icon yourself and add it to the library.

An icon can be created with the free Windows tool called "Paint" for example.

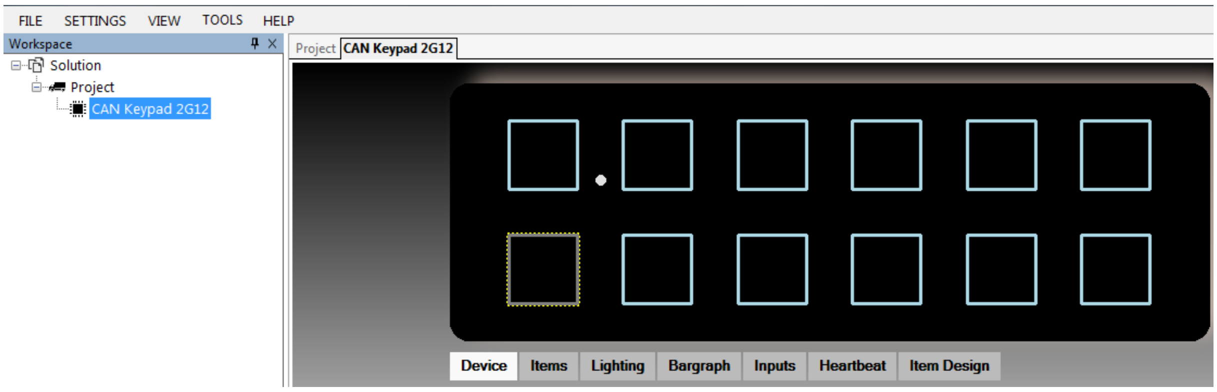
The icon must have the following parameters:
- Background must be white or transparent
- Size 65x65 pixels
- File format "PNG"
-

The foil must have the following parameters:
- Background must be white or transparent
- 12 Keys size 700x237 Pixel
- 6 Keys size 384x237 Pixel
- 4 Keys size 277x237 Pixel
- File format „PNG“

When the icon or foil fulfills the requirements, you can move it in the installation folder of the Tool-chain. After that you can use your icon or foil in the Toolchain.

10 Operation of CAN Keypad 2G 4 | 6 | 12

10.1 Parameterization of CAN Keypad 2G 4 | 6 | 12



The tab “CAN Keypad 2G(4 | 6 | 12)” appears in the workspace.

After clicking on the tab, it opens itself in the middle of the Toolchain.

Now seven tabs appear under the keypad schematic:

„Device“

Here, general settings are made which affect the entire device.

„Items“

Here you can configure all settings that affect the buttons and displays.

„Lighting“

To configure keypad illumination to different terms of operation

„Bargraph“

In case of additional indicator-LEDs in several items, configurations can be set here

„Inputs“

The two hardware inputs can be configured individual

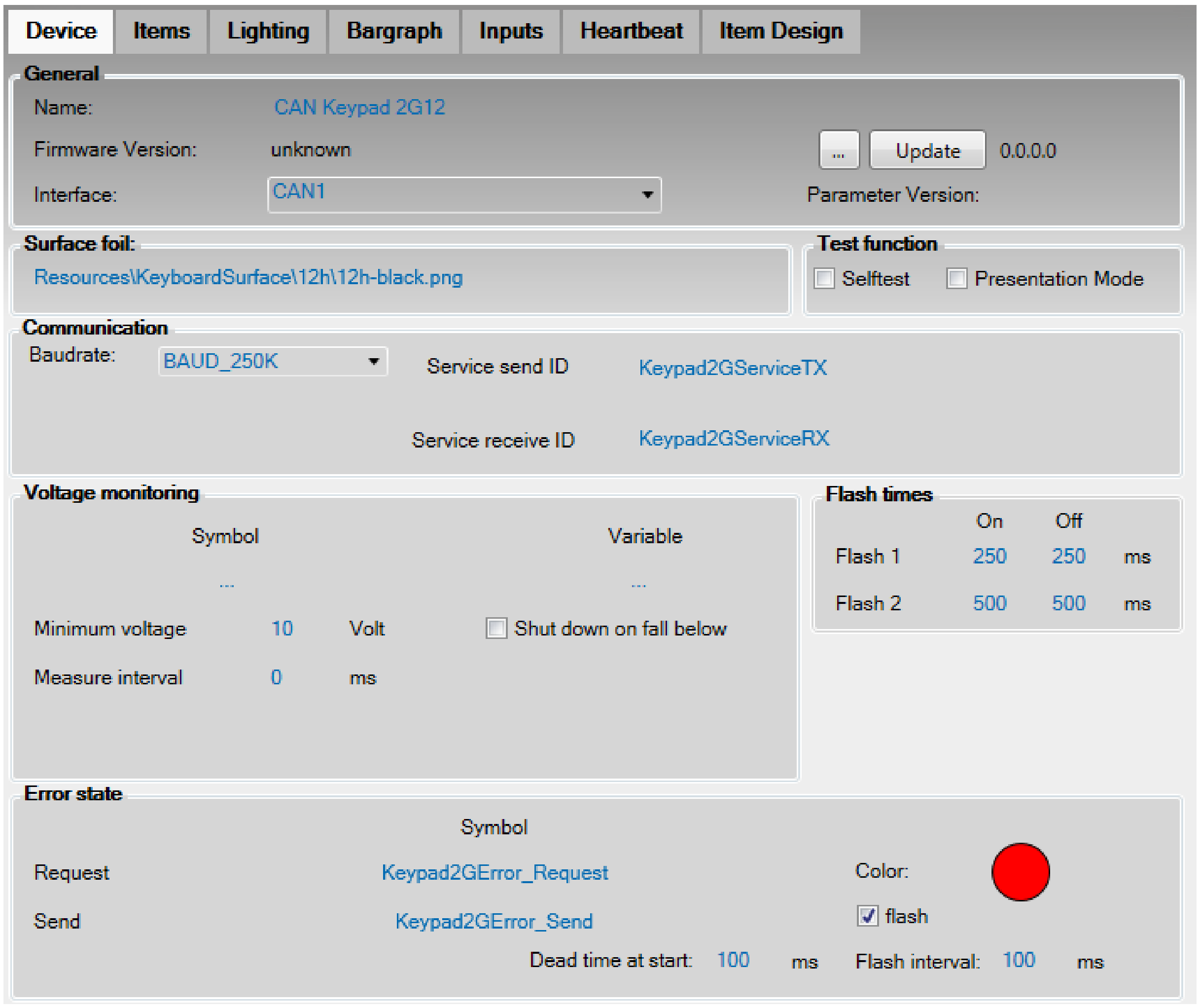
„Heartbeat“

to configure under which conditions „heartbeat“ will be send on CAN-Bus

„Item Design“

to configure the graphic design of the keypad`s surface. These settings will not be backed up on the keypad. This function also helps the miunske team to realize the foil design.

10.1.1 Tab „Device“



General

Name: CAN Keypad 2G12
Firmware Version: unknown ... Update 0.0.0.0
Interface: CAN1 Parameter Version:

Surface foil:

Resources\KeyboardSurface\12h\12h-black.png

Test function

Selftest ☐ Presentation Mode ☐

Communication

Baudrate: BAUD_250K
Service send ID Keypad2GServiceTX
Service receive ID Keypad2GServiceRX

Voltage monitoring

Symbol ... Variable ...
Minimum voltage 10 Volt ☐ Shut down on fall below
Measure interval 0 ms

Flash times

	On	Off	
Flash 1	250	250	ms
Flash 2	500	500	ms

Error state

	Symbol
Request	Keypad2GError_Request
Send	Keypad2GError_Send

Color: ●
☑ flash
Dead time at start: 100 ms
Flash interval: 100 ms

General settings can be defined here which affect the whole device.

General

Name	the name of the product
Firmware Version	is automatically changed to be corresponding firmware version, as soon as a device has been read • With the buttons **„…“** (open) and **„Update“** the firmware of the keypad can be updated
Interface	in the drop-down menu a CAN-Bus created during the project used by the keypad can be choose
Parameter Version	Free versioning as text or number for the data record.

Surface foil

A graphic file (.png) can be used as a skin-design for the keypad surface.

Test function	
Selftest	If a positive operating voltage is applied, all mounted RGB LEDs as well as additional LEDs were tested by the keypad.
Presentation Mode	The presentation mode is used to directly switch the LEDs on push-button printing, without sending a message by CAN-Bus. **This is only true for the RGB LEDs.**
Communication	
Baudrate	setting of the CAN-Bus bit rate used by the keypad • the maximum speed is 1000 kbit/s • the minimum speed is 20 kbit/s
Service send ID	this send ID is responsible for finding the keypad on the CAN-Bus in order to update its parameters
Service receive ID	this receive ID is responsible for finding the keypad on the CAN-Bus in order to update its parameters
Voltage monitoring	
Symbol	ID on which the CAN message is send with the supply voltage value.
Variable	16 Bit Value on which the supply voltage is send in **mV**.
Minimum voltage	defines the limit in supply voltage forcing the keypad in stand-by
Measure interval	after the time set here, the supply voltage is measured cyclically
Shut down on fall below	The "sleep mode" function is activated when the undervoltage limit is reached. To wake up the keypad from "sleep mode", terminal 30 has to be re-applied. Alternatively, the terminal 15 can be controlled via the hardware input 1. The prerequisite is that hardware input 1 has been parameterized as "sleep mode".
Flash times	2 independent blinking times with individual on / off ratio can be selected for the device
Flash 1	the “on” / “off” color of the LED, which is preset in “Items”, is displayed in the flashing rhythm
Flash 2	the “on” / “off” color of the LED, which is preset in “Items”, is displayed in the flashing rhythm

Error state

Color	A RGB color value can be set, which is displayed (flashing) on all connected keys when a connection error occurs.
Request	CAN-ID, which transmits any errors (as a CAN message).
Send	CAN-ID to which the error condition is sent.
flash	when the button "Flash" is activated, the set color will flash in the flashing interval if an error occurs
Flash interval	Defines the flashing frequency in the case of error of the respective button.
Dead time at start	Time that elapses until indicating a present error.

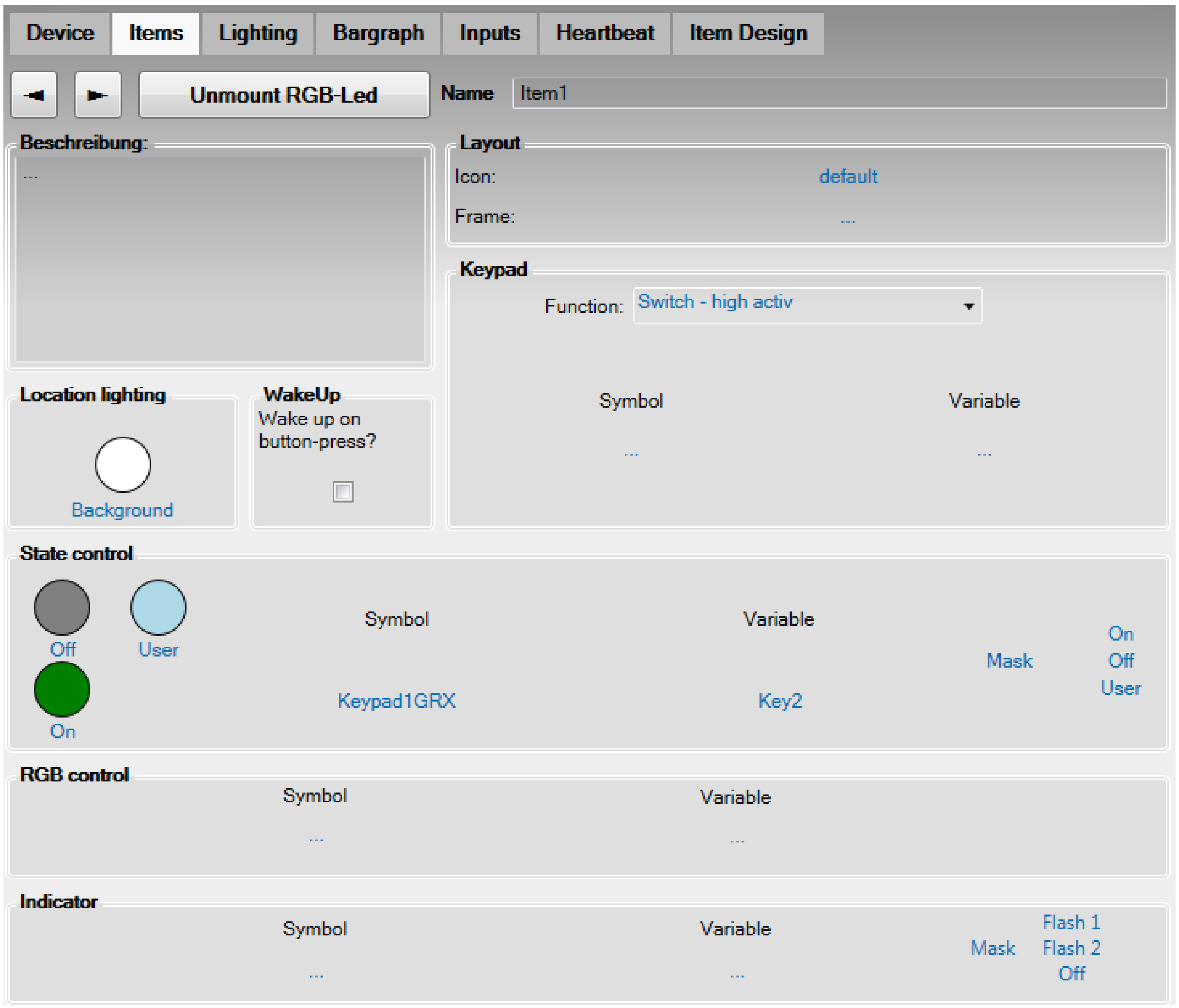
structure of the CAN message for requesting the fault condition:

Name	**Byte 0**	**Byte 1**	**Byte 2**		**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
bit position	0..14	15	16	17..23	24..39		-	-	-

bit position	**Error**	**Value**
0 – 14 = x	Timeout of ID x	0 -> no Timeout 1 -> Timeout
15	Was not able to send a CAN Message.	0 -> no error 1 -> error
16	Voltage monitoring	0 -> no undervoltage 1 -> undervoltage
24 - 39	Current voltage when the device is in undervoltage mode.	

10.1.2 Tab „Items“

Here are all settings for each of the keys. You can click on a rectangle on the keyboard to choose a key for the configuration.



Name: Item1

Layout
Icon: default
Frame: ...

Keypad
Function: Switch - high activ
Symbol: ... Variable: ...

State control
Symbol: Keypad1GRX Variable: Key2
Mask: On / Off / User

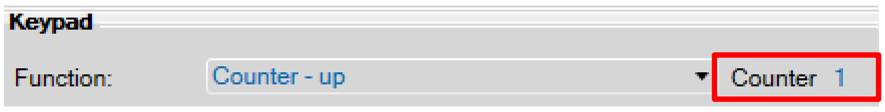
RGB control
Symbol: ... Variable: ...

Indicator
Symbol: ... Variable: ...

Name	A name can be assigned for each item. (This is only stored on the PC and cannot be written to the keypad.)
Description	This is a description field for each key. (It is only stored on the PC and can-not be written to the keypad)
Layout	
Icon	Icon Manager, that contains a large icon library, is selected. An icon for the selected key can be choose.
Frame	for each icon a frame. can be added

Keypad

Function	different functions be can be assigned to the keys
no function	works as a display element
Switch high active	Contains a latching function: once push "on", Another time "off".
Switch low aktiv	Contains a latching function: once push "off", Another time "on".
Tactswitch high active	A short pulse is send, no matter how long the button is pressed. Value changes from "0" to "1".
Tactswitch low active	A short pulse is send, no matter how long the button is pressed. Value changes from "1" to "0".
Button high active	As long as the button is pushed, the button is considered "on". Value changes from "0" to "1".
Button low active	As long as the button is pushed, the button is considered "on". Value changes from "1" to "0".
Counter up	A value is added by each key pushing until the maximum counter value is reached. After that starting again at the initial value. It is important to ensure that the variable for the counter value is long enough.



Keypad

Function: Counter - up Counter 1

Counter down	A value is subtracted from the counter value by every key-pushing until the counter value is 0. Then it starts again at the initial value. It´s important to ensure that the variable for the counter value is long enough.
Symbol	ID on which state CAN message is send.
Variable	Bit indicating state (in the symbol manager).

Location lighting

Background	Color, that appears by activated hardware input (parameter assignment of inputs).

WakeUp

Wake up on button-press?	end sleep mode by pushing the respective key (parameterization sleep mode is in the chapter "Inputs")

State control

Color Off	Color that appears when the button is switched off. (CAN message)
Color On	Color that appears when the button is switched on. (CAN message)
Color User	Color that appears when the button is switched on. (CAN message)
Symbol	ID, on which CAN message for control of colors is send.
Variable	Bit which is responsible for controlling the luminous color of respective state (can be set in the symbol manager).
On	Bit used for "on" state.
Off	Bit, used for "off" state.
User	Bit used for "user" state.
Mask	Disjunction of several bits to CAN message that causes a masking of the CAN signal.

RGB control

Symbol	ID used for sending CAN message for triggering the color.
Variable	64-bit message containing the following definition:

description	key	ID	message
color red	any	any	xx 00 00 00 00 00 00 00
color green	any	any	00 xx 00 00 00 00 00 00
color blue	any	any	00 00 xx 00 00 00 00 00
addition LED 1	any	any	00 00 00 xx 00 00 00 00
addition LED 2	any	any	00 00 00 00 xx 00 00 00
addition LED 3	any	any	00 00 00 00 00 xx 00 00
addition LED 4	any	any	00 00 00 00 00 00 xx 00
addition LED 5	any	any	00 00 00 00 00 00 00 xx

Each LED can take a different brightness value, which is determined by the value xx in the CAN message.

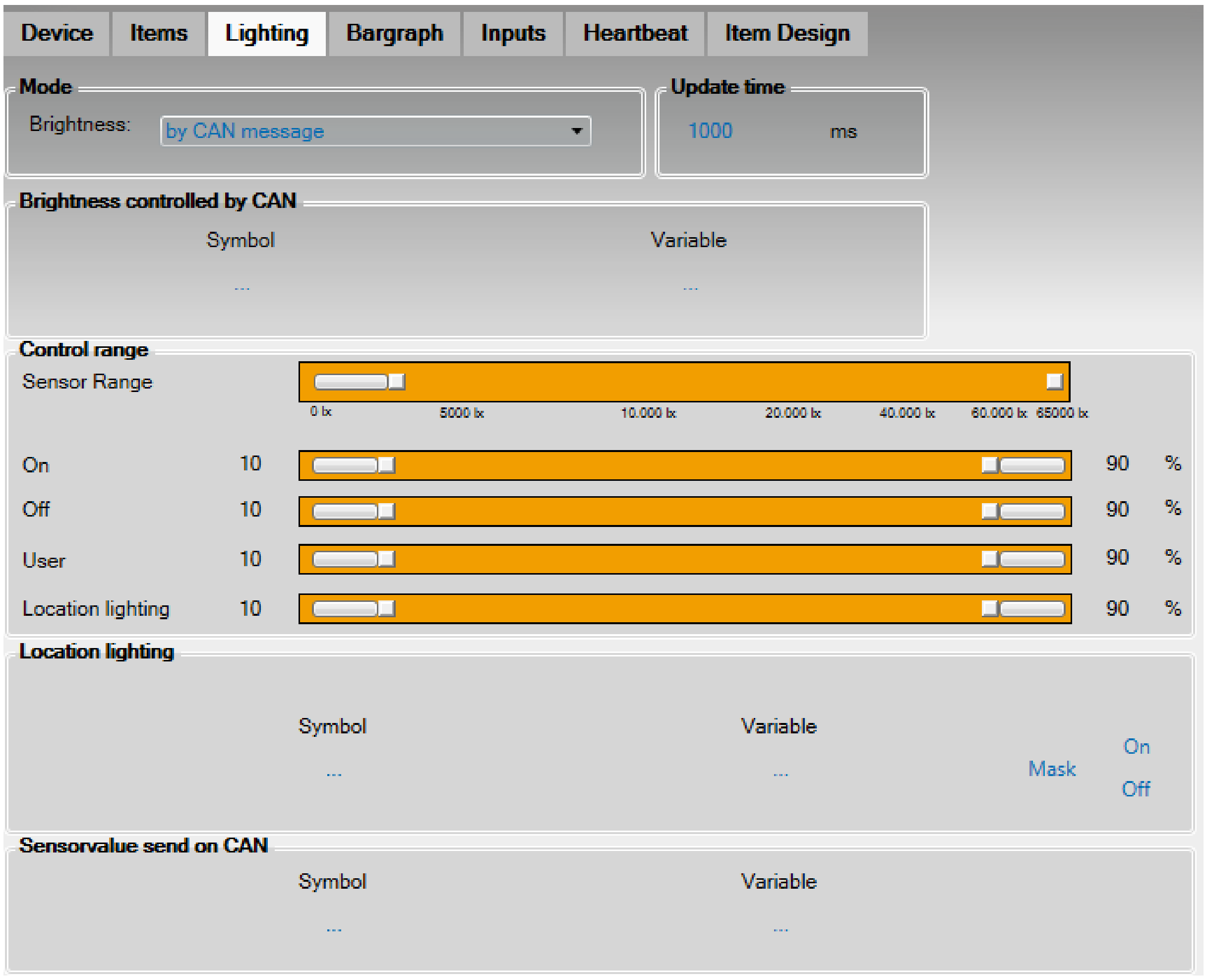
$FF_h = 255_d$ --> 100%

$20_h = 25,5_d$ --> 10%

Indicator	Message of two bits can be defined which causes the RGB LED to flash between "Off" state and "On" state. Furthermore it is possible setting two different blinking times used for representing different situations.
Symbol	ID on which CAN message is send for the control of the flashing times.
Variable	Bit responsible for status change. (In the symbol manager)
Flash 1	Bit responsible for setting state. Flash duration is configured in tab device.
Flash 2	Bit responsible for setting state. Flash duration is configured in tab device.
Mask	Disjunction of several bits to CAN message that causes a masking of the CAN signal.
Off	Bit responsible for setting state.

10.1.3 Tab „Lighting“

Brightness of the entire keypad is parameterized in the respective states.



Mode
Brightness: by CAN message

Update time
1000 ms

Control range

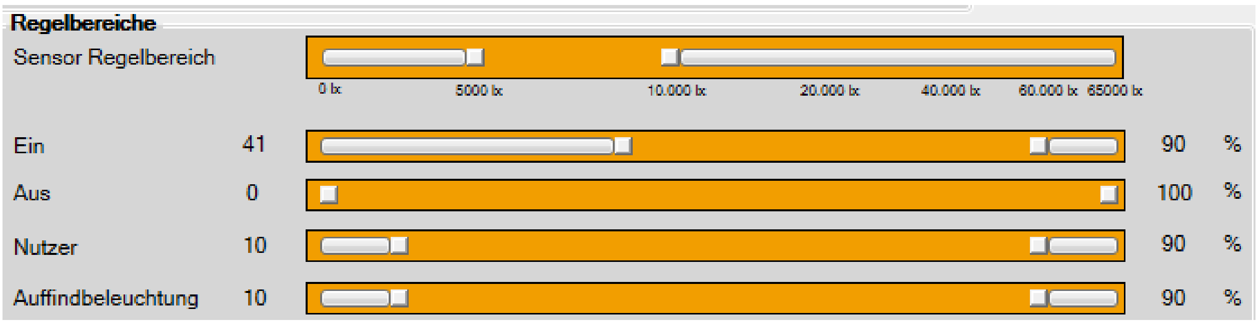
On	10	90	%
Off	10	90	%
User	10	90	%
Location lighting	10	90	%

Mode

Brightness	The combination panel is used for controlling input that determines brightness of the keypad. Settings can be specialized after the selection.
by CAN message:	A variable of 8 bits can be selected for brightness control in the symbol manager (Range of values: 0d - 255d).).
define constant values:	a percentage value is set for every state (100% means 255d)
adjust by brightness sensor:	The "Sensor range" area determines at which environment lightness a brightness-adjustment of the keypad-illumination is necessary. For each state a percentage control range is set that affects the keypad brightness.
Update time	Time elapses between detecting the sensor value up to a visible change in brightness at the keypad. The default value is set to 1000 milliseconds
Brightness controlled by CAN	A second node in the CAN-Bus is responsible for sending required brightness values
Symbol	ID, on which the CAN message for brightness control is send.
Variable	two bytes, that contain brightness value (in the symbol manager)

Control range

functional example



Regelbereiche

Sensor Regelbereich

0 lx 5000 lx 10.000 lx 20.000 lx 40.000 lx 60.000 lx 65000 lx

Ein	41	90	%
Aus	0	100	%
Nutzer	10	90	%
Auffindbeleuchtung	10	90	%

E.g. A Sensor Range of 5000lx - 10000lx is set. That means: 5000lx corresponds to a value of 0d and 10000lx of 255d on the CAN bus.

The "Off" brightness control range operates exactly in the "Sensor Range" control range 5000lx - 10000lx.

On the other hand, the "on" brightness control range only works from 41% to 90%. When 5000lx is reached, the lower brightness control range of 41% is reached.

The same is valid for 10000lx and 91% brightness.

As soon as the set “Sensor Range” value is exceeded or undershot, no change of brightness of the respective state appears anymore.

Examples of brightness values:

bright sunny day	100.000	lx
overcast summer day	20.000	lx
elite football stadium	1.400	lx
TV-studio lighting	1.000	lx
office / room lighting	500	lx
hall lighting	100	lx
living room	50	lx
street lighting	10	lx
candle in approx. 1 meter distance	1	lx

Location lighting

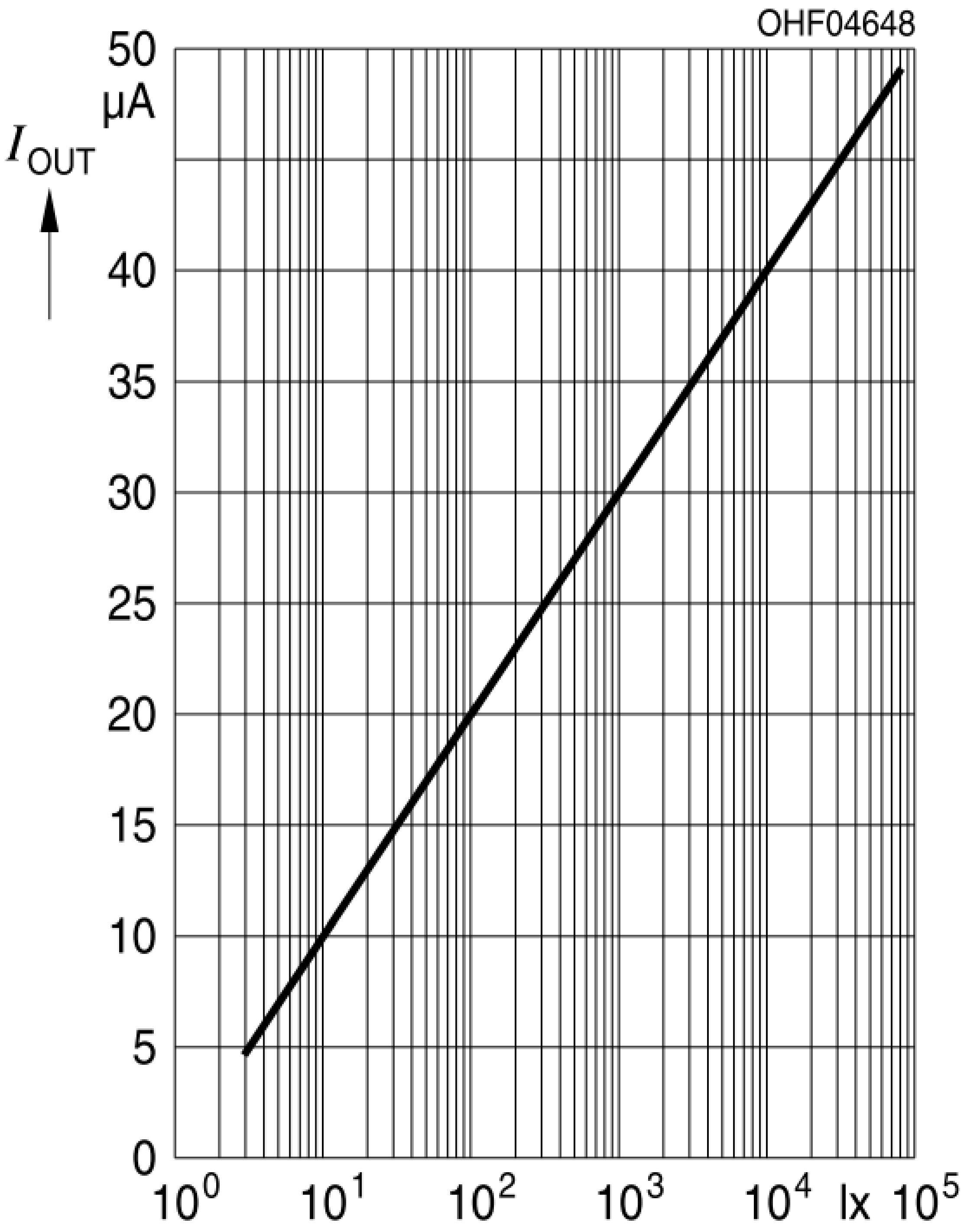
Symbol	ID, on which the CAN message is send for switching detection light.
Variable	Bit, which activates the detection light (in the symbol manager)
Mask	disjunction of several bits to CAN message that causes a masking of the CAN signal.
On	Bit responsible for setting state.
Off	Bit responsible for setting state.

Sensorvalue send on CAN

Symbol	ID, on which the CAN message for sensor value output is send.
Variable	Two bytes, which represent the sensor brightness change in the CAN message. (Setting in the symbol manager)

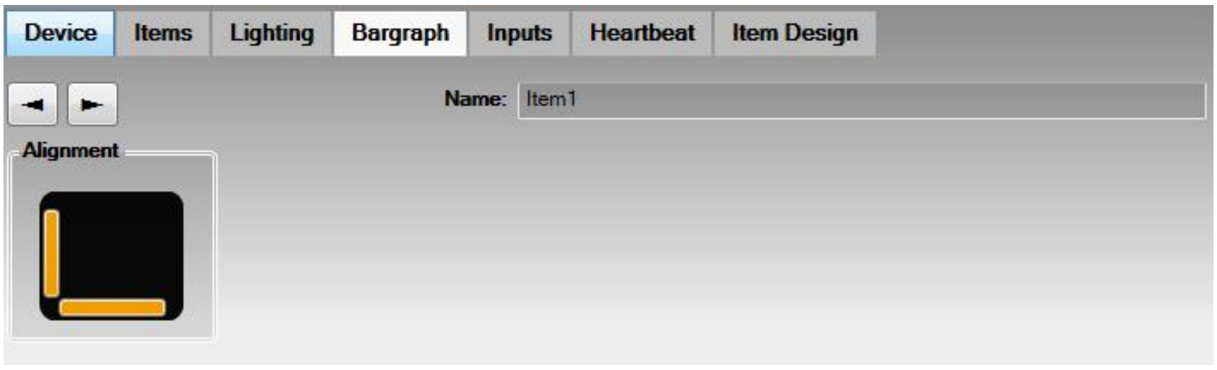
Calculate the values using the following formula (diagram):

$$\frac{65535}{X * 100000} = I_{OUT}$$

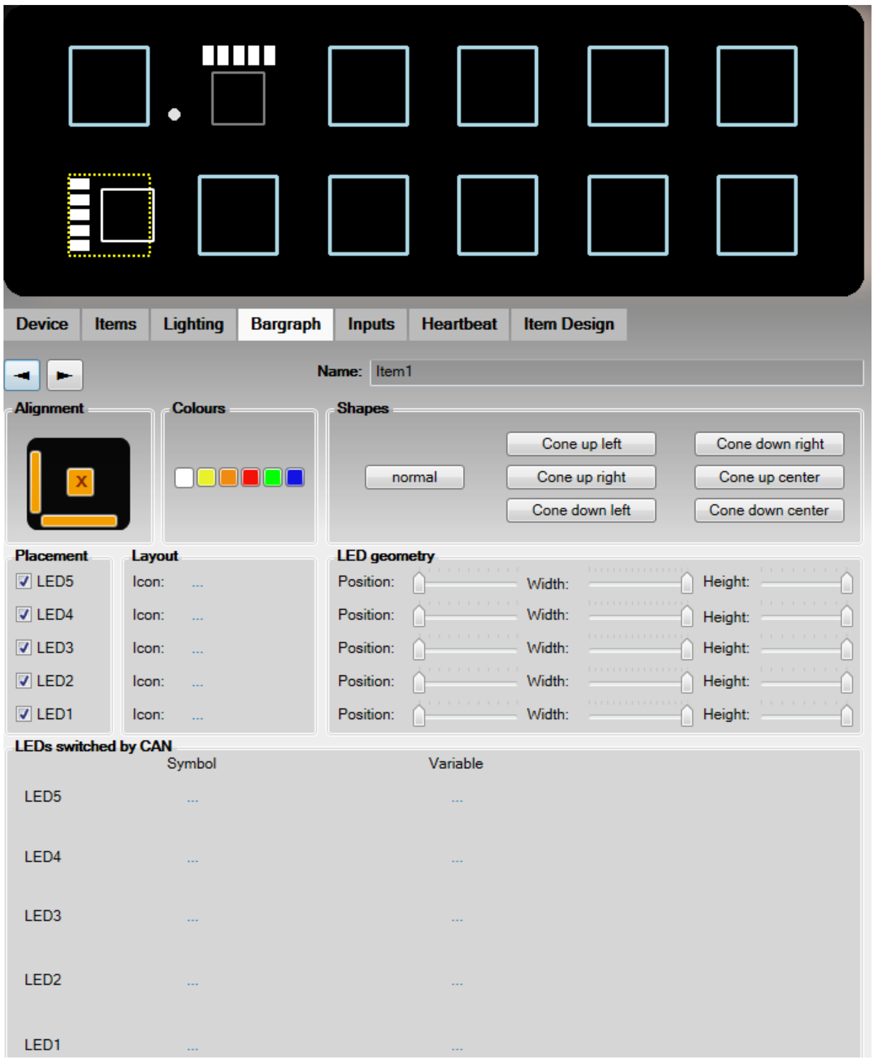


10.1.4 Tab „Bargraph“

If additional LEDs are selected for the respective keypad, their parameters are set in this tab. First of all the position-type must be selected in the graphic under "Alignment". **Only one alignment, either "horizontal" or "vertical", is possible.**



After a variant has been selected, it is shown for the selected item in the keypad-image above the several tabs and the interface for parameterization appears.



Name	The function name for the respective keypad is displayed here.
Alignment	This determines the position of the additional LEDs in horizontal or vertical direction.
Colors	By clicking on the respective additional LED in the graphic, a color for the LED can be defined. However, this is only relevant for the graphic design, since the additional LEDs can only be illuminated as unicolor LEDs in the color variant fitted.
Shapes	Different design variants are used for the additional LEDs. These are only for the surface layout relevant.
Placement	Type of additional LEDs on the CAN keypad is selected here.
Layout	A separate symbol can be defined for each additional LED. Note, that this is only relevant for graphic design
LED geometry	Beyond the predefined shapes, further variants can be defined for each LED.

LEDs switched by CAN

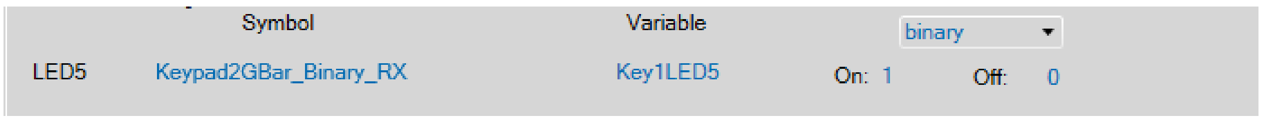
LED1 – LED5	For each additional LED, click on the **"..."**

LED4 ... ...

A separate message with ID and BIT can be defined.

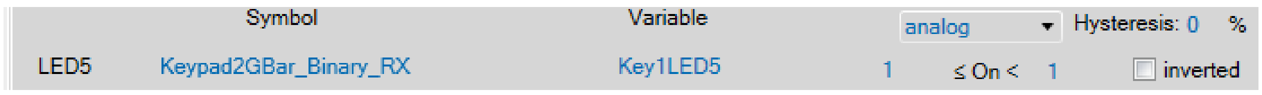
Symbol	ID, on which the CAN message for the control of the additional LED is send.
Variable	Bit responsible for status change. (In the symbol manager)

Once a message has been defined, the switch-on-threshold can be assigned.
This can be done binary or analog.



	Symbol	Variable	binary	
LED5	Keypad2GBar_Binary_RX	Key1LED5	On: 1	Off: 0

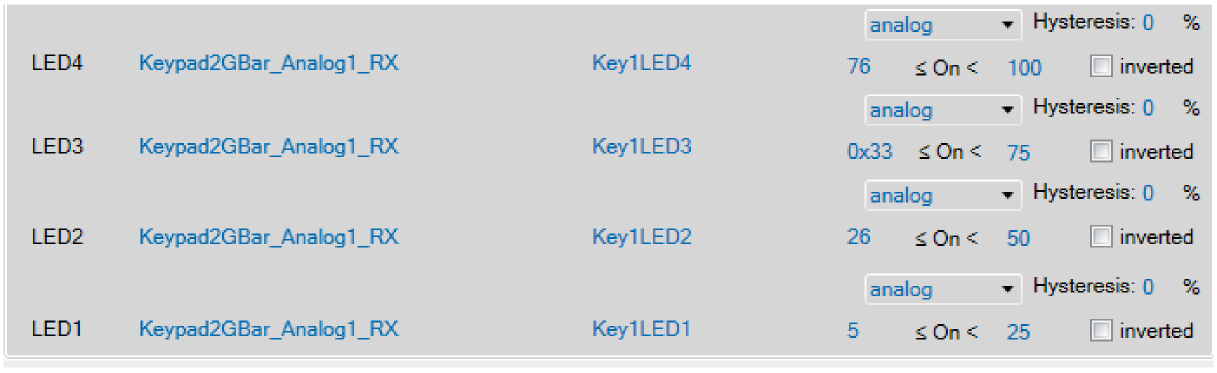
Binary	The state of the LED is between two defined Bits.
On	Value responsible for the "on" state. maximum value: 65535
Off	Value responsible for the "off" state. maximum value 65535



	Symbol	Variable	analog	Hysteresis: 0 %
LED5	Keypad2GBar_Binary_RX	Key1LED5	1 ≤ On < 1	☐ inverted

Analog	Status of the LED can be defined between two ranges. The scaling of the measuring range gives the number of bits that are necessary for the control.

Exemplary parameterization:

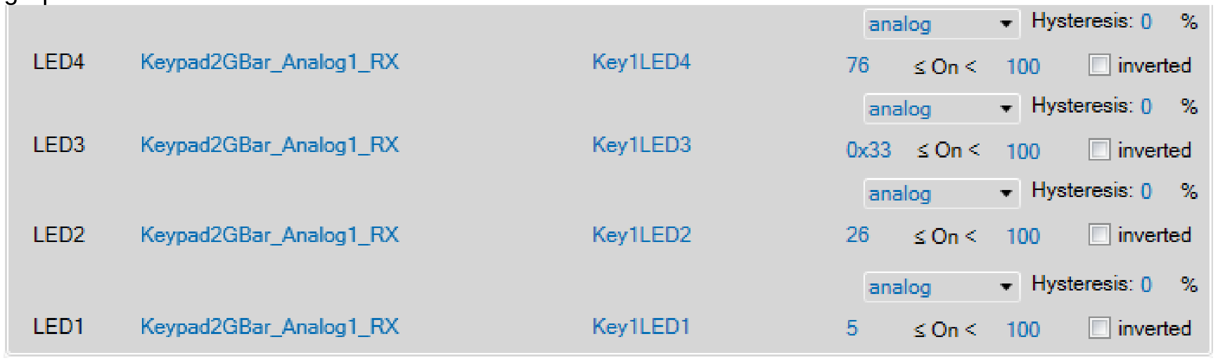


			analog			Hysteresis: 0 %	
LED4	Keypad2GBar_Analog1_RX	Key1LED4	76	≤ On <	100	☐ inverted	
			analog			Hysteresis: 0 %	
LED3	Keypad2GBar_Analog1_RX	Key1LED3	0x33	≤ On <	75	☐ inverted	
			analog			Hysteresis: 0 %	
LED2	Keypad2GBar_Analog1_RX	Key1LED2	26	≤ On <	50	☐ inverted	
			analog			Hysteresis: 0 %	
LED1	Keypad2GBar_Analog1_RX	Key1LED1	5	≤ On <	25	☐ inverted	

The value range is 2 Byte, in decimal 0-65535. You can set the limits in decimal and hexadecimal (0x… prefix).

LED1 is defined from 5 to 25, which means LED1 is on when the value is between 5 and 25 else it is off. LED2 is on when the value is between 26 and 50 else it is off.
LED3
State „On“= 51 (0x33)
State „Off“= 75
LED4
State „On“= 76
State „Off“= 100

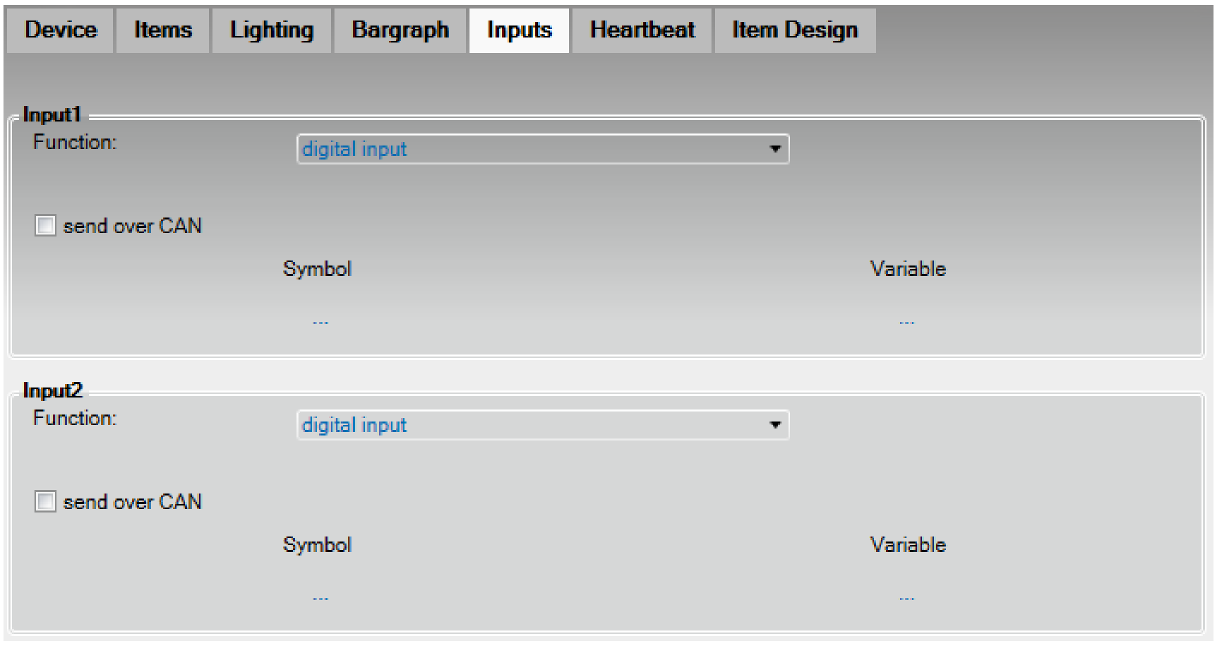
When you want to turn on the LED’s one after the other, you have to configure as in the following graphic:



			analog			Hysteresis: 0 %	
LED4	Keypad2GBar_Analog1_RX	Key1LED4	76	≤ On <	100	☐ inverted	
			analog			Hysteresis: 0 %	
LED3	Keypad2GBar_Analog1_RX	Key1LED3	0x33	≤ On <	100	☐ inverted	
			analog			Hysteresis: 0 %	
LED2	Keypad2GBar_Analog1_RX	Key1LED2	26	≤ On <	100	☐ inverted	
			analog			Hysteresis: 0 %	
LED1	Keypad2GBar_Analog1_RX	Key1LED1	5	≤ On <	100	☐ inverted	

10.1.5 Tab „Inputs“

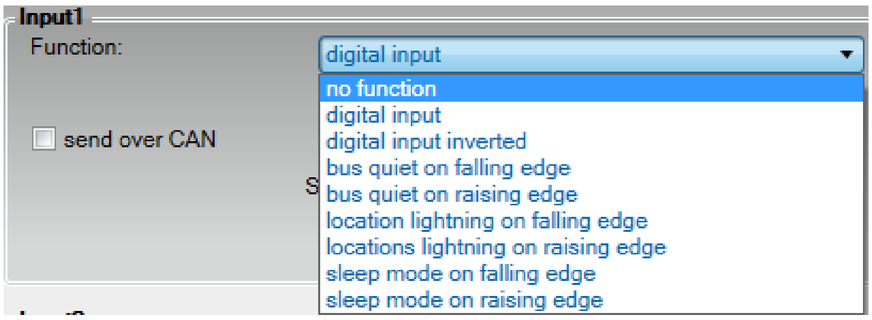
Using this tab, certain states can be defined for the two inputs on the hardware.



Input1
Function: digital input
send over CAN
Symbol ...
Variable ...

Input2
Function: digital input
send over CAN
Symbol ...
Variable ...

Input 1 & Input 2	These are hardware-inputs. The corresponding technical specifications are defined in the data sheet
Function	The "drop-down" field is used to select the respective functions of the hardware-input.


Input1
Function: digital input
send over CAN

no function
digital input
digital input inverted
bus quiet on falling edge
bus quiet on raising edge
location lightning on falling edge
locations lightning on raising edge
sleep mode on falling edge
sleep mode on raising edge

no function	The input is not used in the controller processing.
digital input	The input responds to a high signal.
digital input inverted	The input responds to a low signal.
bus quiet on falling edge	When the level is changed from "High" to "Low", no message is send on the CAN.
bus quiet on raising edge	When the level is changed from "Low" to "High", no message is send on the CAN.
location lightning on falling edge	When the level is changed from "High" to "Low" the location lightning is activated.

location lightning on raising edge	When switching from "Low" to "High" the location lightning is activated
sleep mode on falling edge	When switching from "High" to "Low", the keypad switches to power saving mode. The input must have a new level change from "Low" to "High" for returning to normal state. This can also be triggered by a parametrized key on the keyboard (tab „Items“).
sleep mode on raising edge	When switching from "Low" to "High", the keypad switches to power saving mode. The input must have a new level change from "High" to "Low" for returning to normal state This can also be triggered by a parametrized key on the keyboard (tab „Items“).

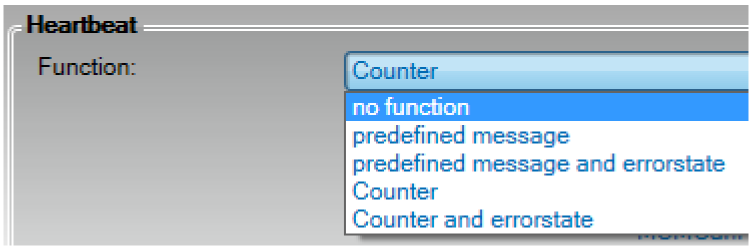
send over CAN	The respective input state can be applied to the CAN bus.
Symbol	ID, on which the CAN message for the evaluation of the input is made.
Variable	Bit, responsible for the evaluation of the input (can be set in the symbol manager).

10.1.6 Tab „Heartbeat“

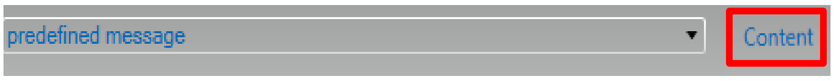
This tab defines under which conditions "Heartbeat" should be send on the CAN-Bus.

Heartbeat

Function	Use the drop-down list to select the respective heartbeat- function.



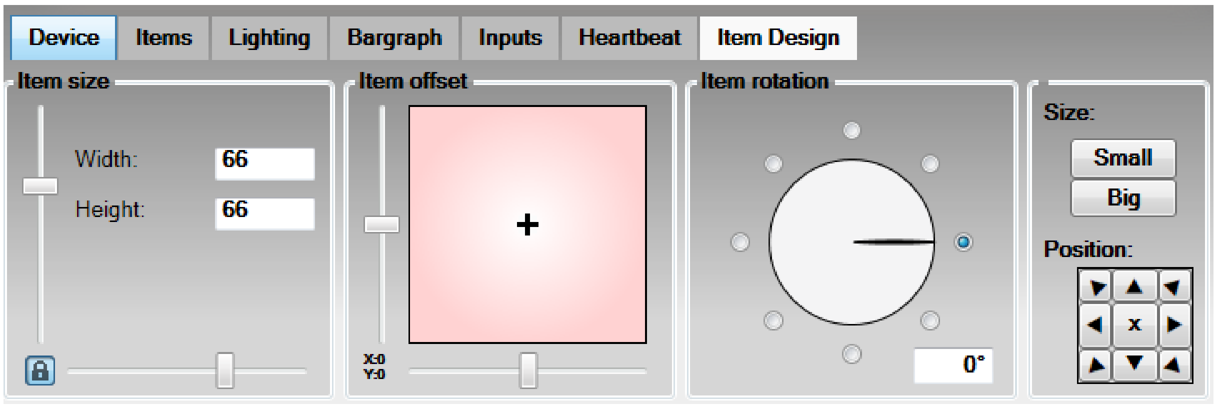
no function	Heartbeat is not included in the controller processing
predefined message	Message can be freely defined in the content.



predefined message and errorstate	Message can be freely defined in the content. In addition, one byte can be selected for error status output. Error status only indicates that an error has occurred. This must then be queried by request. Setting error state in tab “Device”.
counter	One byte can be configured as a counter. When the transmission cycle time is reached, the defined byte is increased by value “1”.
counter and error-state	In addition, one byte can be configured as a counter and one byte as error status output. When the transmission cycle time is reached, the defined counter byte is increased by value “1”. Error status only indicates that an error has occurred. This must then be queried by request. Setting error state in tab “Device”.
Symbol	ID on which the keypad sends the heartbeat. (Can be set in the symbol manager)
Variable	Bit which defines the state change. (Can be set in the symbol manager)

10.1.7 Tab „Item Design“

Settings for the graphic design of the slide are done under this tab. These settings are only for graphical orientation and are not stored on the keyboard.



Width: 66
Height: 66
0°

Item size

Width	The respective symbol can be adapted to the entire field
Height	The respective symbol can be adapted to the entire field
Item offset	Symbols can be moved to a specific position in the field
Item rotation	Symbols can be rotated in degrees around center axis
Size	Adjustment of the symbol to the size of the field or half field, in case of additional LEDs are used
Position	Symbols can be aligned quickly

10.2 Integrating own icons

The Toolchain already contains a large icon library out of the box.
In case the right icon for your application is not available, it is possible to create the icon yourself and add it to the library.

An icon can be created with the free Windows tool called "Paint" for example.

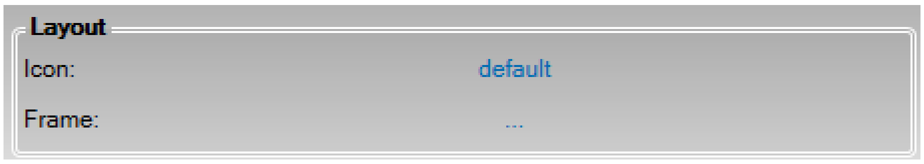
The icon must have the following parameters:
- Background must be white or transparent
- Size 65x65 pixels
- File format "PNG"

The foil must have the following parameters:
- Background must be white or transparent
- 12 Keys size 700x237 Pixel
- 6 Keys size 384x237 Pixel
- 4 Keys size 277x237 Pixel
- File format „PNG“

When the icon or foil fulfills the requirements, you can move it in the installation folder of the Tool-chain. After that you can use your icon or foil in the Toolchain.

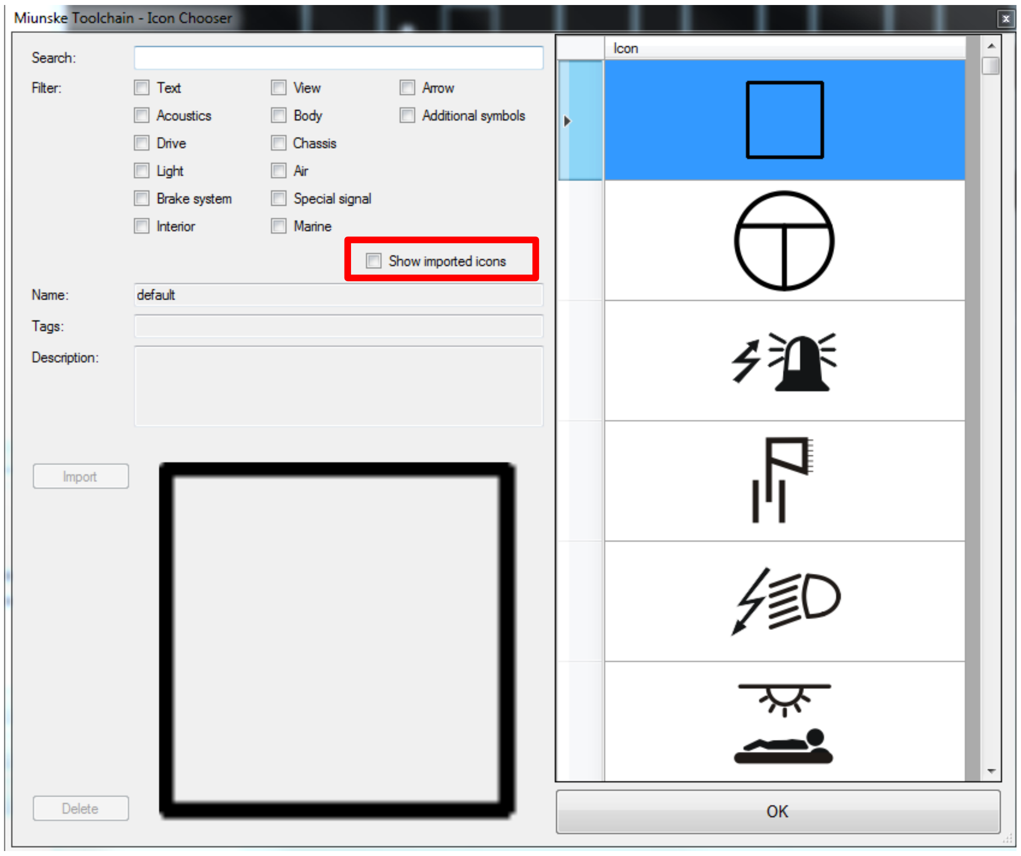
This is done as follows:

1. locate the icon library (icon chooser)



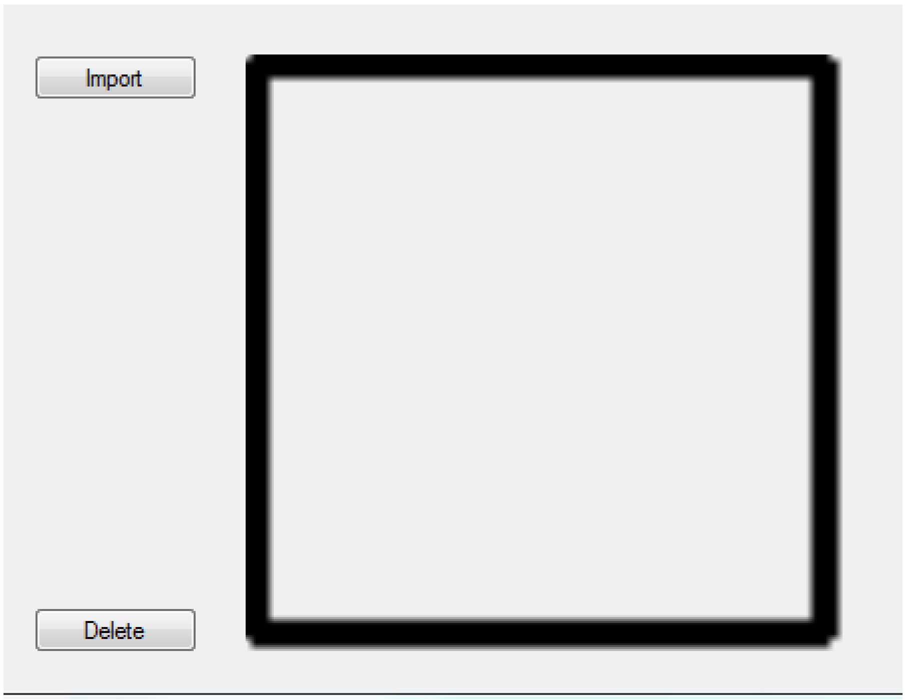
Layout
Icon: default
Frame: ...

2. In the icon chooser, click the "Show imported icons" check box



Name: default

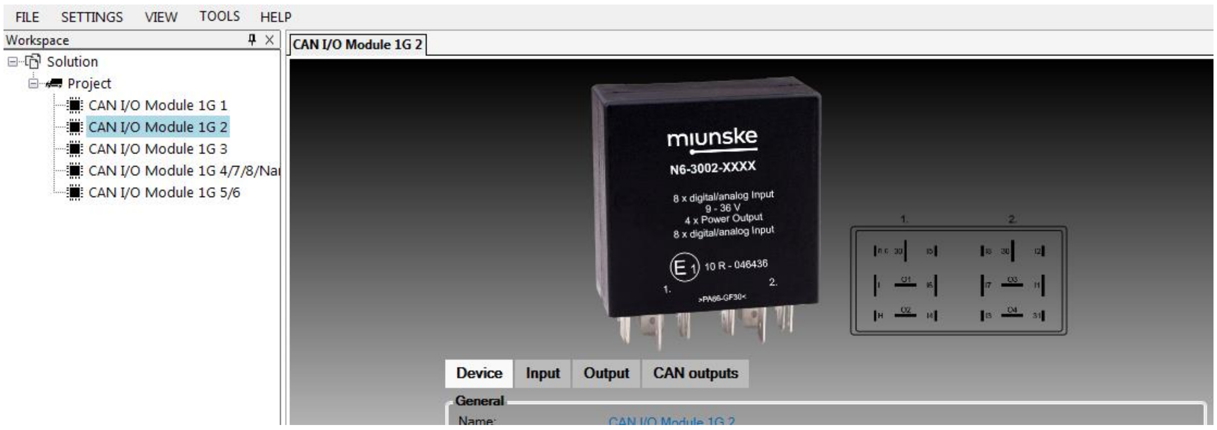
3. The icon can then be selected using the "Import" button



11 CAN I/O-Modules 1GX

11.1 Parameterization of CAN I/O-Modules 1GX

This chapter will explain how the I/O modules are parameterized. The I/O modules differ only in the number of inputs and outputs. Accordingly, the approaches are identical.



Name: CAN I/O Module 1G 2

The tab „CAN I/O Module 1G…“appears in the workspace.

After clicking the tab, it opens in the middle of the Toolchain.

There are four tabs below the graphic:

„Device“

Here, general settings are made which affect the entire device.

„Input“

Configuration of the hardware inputs

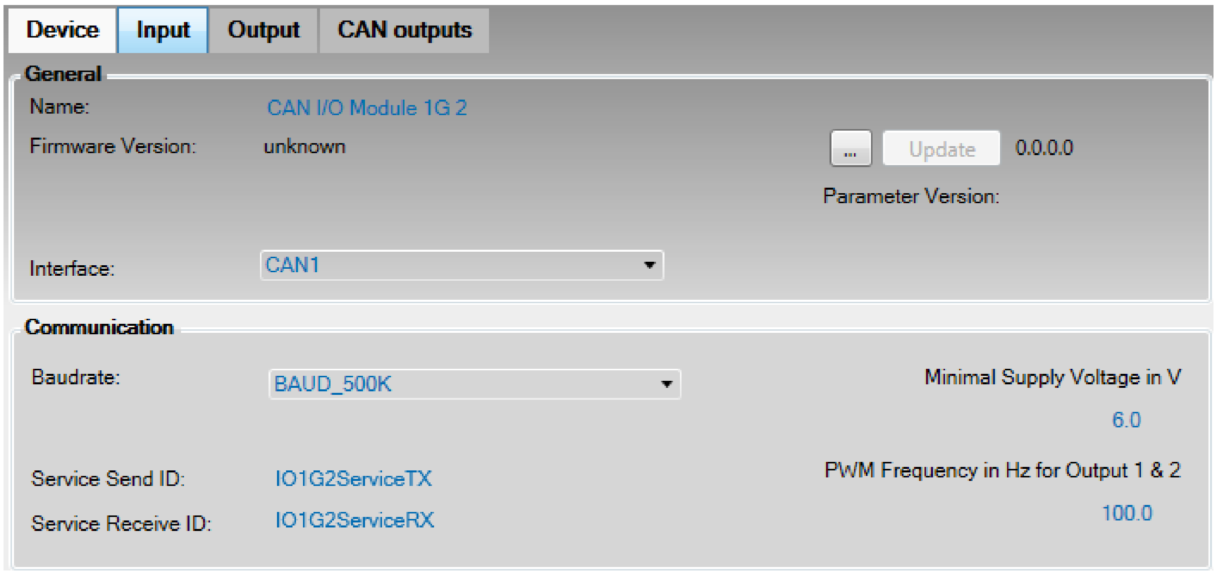
„**Output“**

Configuration of the hardware outputs

„CAN outputs“

State changes are output to the CAN bus after a predefined bit arrangement

11.1.1 Tab „Device“

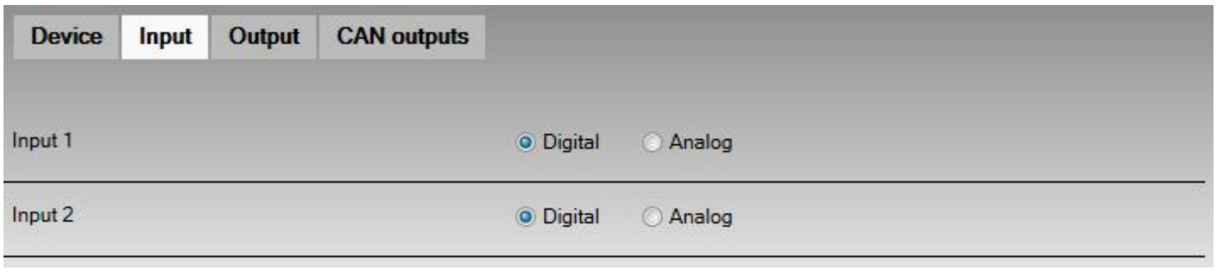


Name: CAN I/O Module 1G 2
Firmware Version: unknown
0.0.0.0
Parameter Version:
Interface: CAN1
Baudrate: BAUD_500K
Minimal Supply Voltage in V: 6.0
Service Send ID: IO1G2ServiceTX
Service Receive ID: IO1G2ServiceRX
PWM Frequency in Hz for Output 1 & 2: 100.0

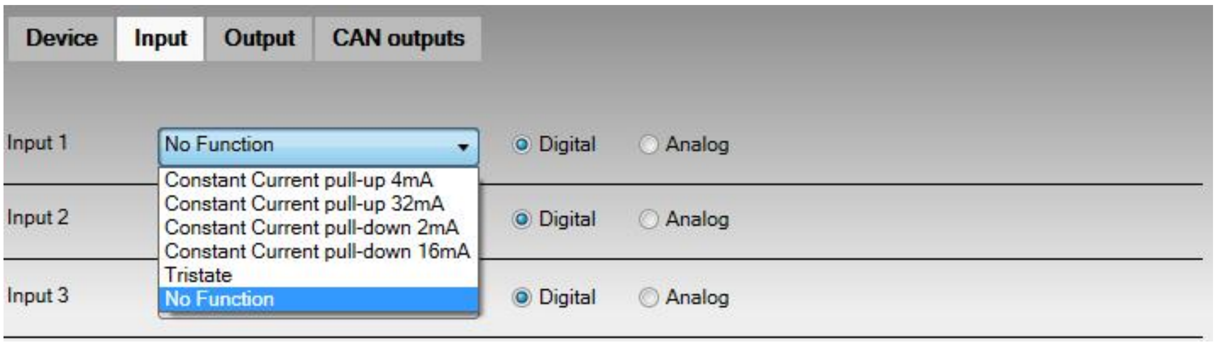
General	
Name	the name of the product
Firmware Version	is automatically changed to be corresponding firmware version, as soon as a device has been read With the buttons „…“ (open) and „**Update**“ the firmware of the I/O-Module can be updated
Interface	in the drop-down menu a CAN-Bus created during the project used by the I/O-Module can be choose
Parameter Version	Free versioning as text or number for the data record.
Communication	
Baudrate	setting of the CAN-Bus bit rate used by the I/O-Module • the maximum speed is 1000 kbit/s • the minimum speed is 100 kbit/s
Service send ID	this send ID is responsible for finding the I/O-Module on the CAN-Bus in order to update its parameters
Service receive ID	this receive ID is responsible for finding the I/O-Module on the CAN-Bus in order to update its parameters
Minimal Supply Voltage in V	defines the limit in supply voltage forcing the keypad in stand-by
PWM Frequency in Hz for Output 1 & 2	Setting the PWM frequency between 2 Hz and 1 kHz. All other outputs can-not be parameterized. PWM frequency is fixed there to 1kHz

11.1.2 Tab „Input“

Parameterization of the hardware inputs. Each I/O module has a different number of inputs, which can only be parameterized by digital / analog. The I/O module 1G2 and 1G3, have more setting options.



Input 1	[✓ Digital]	Analog
Input 2	[✓ Digital]	Analog


Device
Input
Output
CAN outputs
Constant Current pull-up 4mA
Constant Current pull-up 32mA
Constant Current pull-down 2mA
Constant Current pull-down 16mA
Tristate
No Function

Input 1	No Function	[✓ Digital]	Analog
Input 2		[✓ Digital]	Analog
Input 3		[✓ Digital]	Analog

Constant Current pull-up 4mA	the input can carry a current of 4mA to ground potential
Constant Current pull-up 32mA	the input can carry a current of 32mA to ground potential
Constant Current pull-down 2mA	the input can drive a current of 2mA against operating voltage
Constant Current pull-down 16mA	the input can drive a current of 16mA against operating voltage
Tristate	The use of a high-ohm input as a digital input is defined as follows:

input voltage	digital signal
< 4V	Low = 0
> 4V	High = 1

No Function	No signal evaluation of the respective input

Digital	The input is set to 1 when the operating voltage is reached. When reaching ground potential it is set again to 0.

input voltage	digital signal
< 4V	Low = 0
> 4V	High = 1

Analog	Analogue input signals are digitized with a resolution of 12 bits. The measuring range without upstream voltage divider depends on the respective hardware.

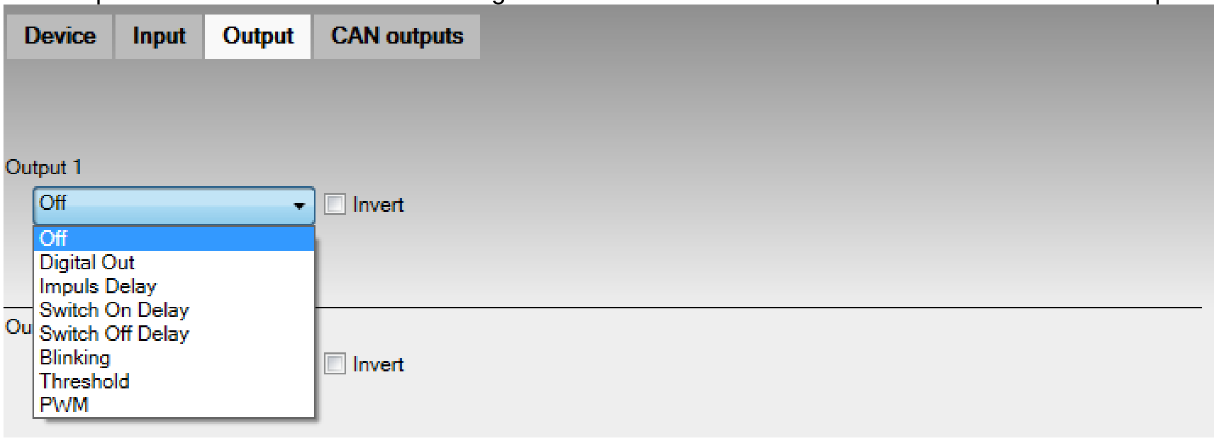
Hardware	measuring range	resolution	Volts per bit
1G1	0 - 30 V	12Bit	0,00732V
1G2	0 – 5 V	12Bit	0,00122V
1G3	0 – 5 V	12Bit	0,00122V
1G4	0 – 30 V	12Bit	0,00732V
1G5	0 – 36 V	12Bit	0,00879V
1G6	0 – 36 V	12Bit	0,00879V
1G7	0 – 36 V	12Bit	0,00879V
1G8	0 – 36 V	12Bit	0,00879V

Calculation example:

12 Bit	=	FFF		
FFF	=	4095		
1G1	=	30V / 4095	=	0,00732V per Bit

11.1.3 Tab „Output“

The outputs of the I/O modules are configured here. Each module has a different number of outputs.

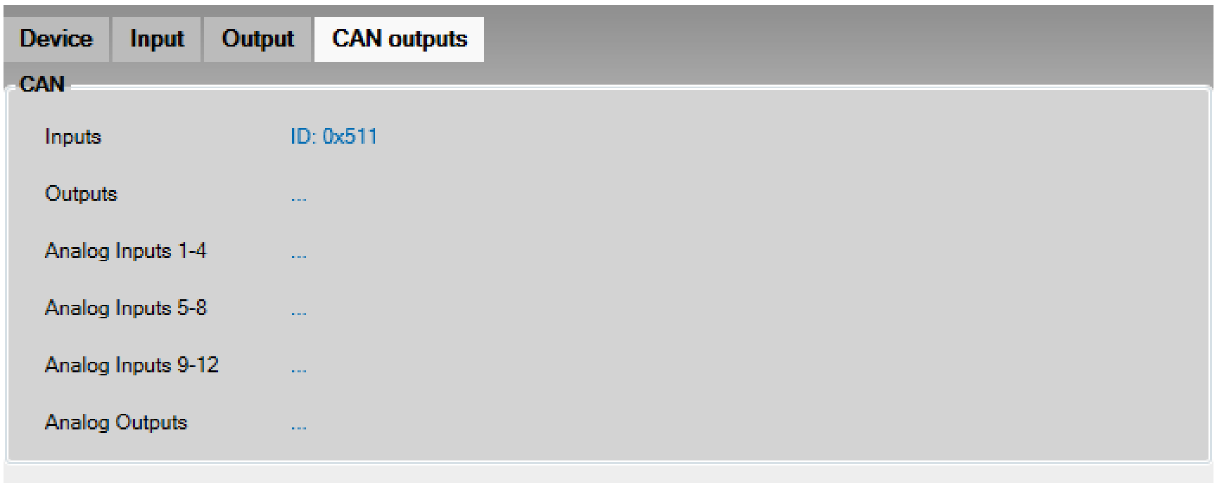


Off	The output is not used
Digital Out	Output has only two states of operating voltage or ground potential
Impulse Extension	At each rising edge of the input, the output is switched on for the specified time. The total switching time is extended by the specified time duration for each further rising edge at the input. If the Impulse Extension is activated via a predefined CAN message, it begins with the first received high level of the parameterized data bit.
Switch On Delay	The output is switched when a permanent high level is applied to the selected input for a fixed time. If the switch-on delay is controlled via a CAN input, the last received signal value is used between two consecutive messages. When the switch-on delay time expires within a CAN cycle, the output is switched on.
Switch Off Delay	The output is switched off when a permanent low level has been applied to the selected input for a fixed time. If the switch-off delay is controlled via a CAN input, the last received signal value is used between two consecutive messages. When the switch-off delay time expires within a CAN cycle, the output is switched off.

Blinking	At the output, the level changes between high and low, as long as the input signal is recognized as high level. When switching to low level, the output switches off immediately. If the flashing function is controlled via a CAN input, the last received signal value is used between two consecutive messages. When the flashing time expires within a CAN cycle, the output is switched off.
Threshold	The output is switched when the analog input value exceeds the parameterized high threshold. The output is switched off when the analogue input value falls below the set minimum threshold value. The "analog threshold" function can only be connected to a input of the module or to a CAN input.
PWM	The output outputs a PWM signal whose pulse width is dependent on an analog input value. An analog input value can be a hardware input or a CAN input. The adjustable frequency (2 - 1000Hz) of the PWM is only valid for the outputs OUT1 and OUT2. All other outputs have a fixed frequency of 1 kHz and cannot be parameterized. The maximum signal value (corresponds to 100% pulse width) is specified over the value range. If a higher value is present at the selected input, a high level (100%) is output. At each falling edge of the output signal, the newly calculated pulse width is assumed. If a short circuit is detected, this output is deactivated until the device is restarted.
Trigger	Determines the input which is responsible for the evaluation of the signal.
Symbol	ID on which the module sends the input results. (Can be set in the symbol manager)

11.1.4 Tab „CAN outputs“

Each state of the inputs (digital / analog) as well as the outputs can be output to the CAN. The CAN messages are parameterized under this tab.



Inputs	ID: 0x511
Outputs	...
Analog Inputs 1-4	...
Analog Inputs 5-8	...
Analog Inputs 9-12	...
Analog Outputs	...

Input	The allocation of the hard-ware-side inputs to the individual data bits of the CAN message is **fixed**. A switched-on input is signaled with the value "1", "0" is not switched. The maximum of 12 available inputs are distributed to the first two bytes, starting with Byte0 Bit7. The cycle time is adjustable in 10ms steps up to a maxi-mum of 2s in the Symbol Manager.

Name	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Inputs	IN1..IN12		-	-	-	-	-	-

Output

For I/O modules with power outputs, the status information of the power drivers can be output. All CAN nodes in the network thus receive information about the status of the other CAN nodes.

The assignment of the status information of the outputs is **set** to the individual data bits of the CAN message. The cycle time is adjustable in 10ms steps up to a maximum of 2s in the Symbol Manager.

Name	**Byte 0**		**Byte 1**		**Byte 2**		**Byte 3**		**Byte 4**		**Byte 5**		**Byte 6**	**Byte 7**
bit position	7..0		15..8		23..16		31..24		39..32		-		-	-
Outputs	01	02	03	04	05	06	07	08	09	10	-	-	-	-

reserved [1Bit] Bit 7	state [2Bit] Bit 6..5	state output [1Bit] Bit 4
	00: no error 01: overtemperature 10: idle 11: short circuit	0: output = OFF 1: output = ON

Analog inputs 1-4

In the messages for analog values 1-12, the voltage values of the respective inputs are output. The analog values are always output as 16-bit signed in the big-endian format. For all I/O modules, the assignment of the analog inputs to the individual data bits of the CAN message is **fixed**. The cycle time is adjustable in 10ms steps up to a maximum of 2s in the Symbol Manager.

Name	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
Analog inputs 1-4	IN1		IN2		IN3		IN4	
Analog inputs 5-8	IN5		IN6		IN7		IN8	
Analog inputs 9-12	IN9		IN10		IN11		IN12	

Analog Outputs

The message of the analog outputs indicates the switched load currents of the individual power outputs. The measuring range of the load current is 0 ... 25,5A with a resolution of 0.1A / bit, while the resolution of 0.1A / bit at currents> 3A is usable. For all I/O modules with power outputs, the assignment of the load current outputs to the individual data bytes of the CAN message is **fixed**. The cycle time is adjustable in 10ms steps up to a maximum of 2s in the Symbol Manager.

Name	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
Analog outputs	OUT1	OUT2	OUT3	OUT4	OUT5	OUT6	-	-

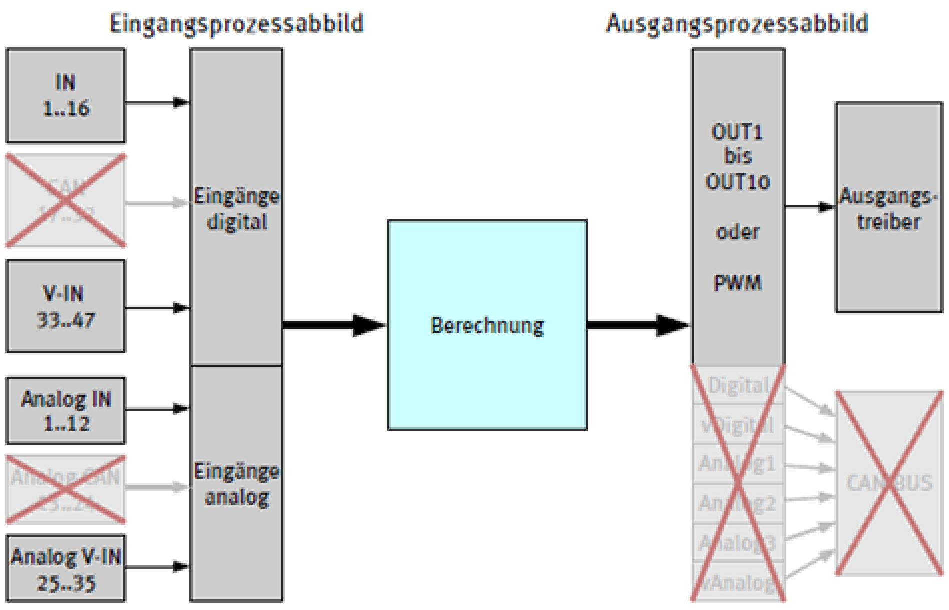
11.2 Special Functions CAN I/O-Module

If the Toolchain settings for your application are not sufficient, the I/O module can be freely programmed via the API interface. The I/O modules use a Freescale MC9S08DV32 controller for which a suitable compiler must be used.

11.2.1 Signal processing

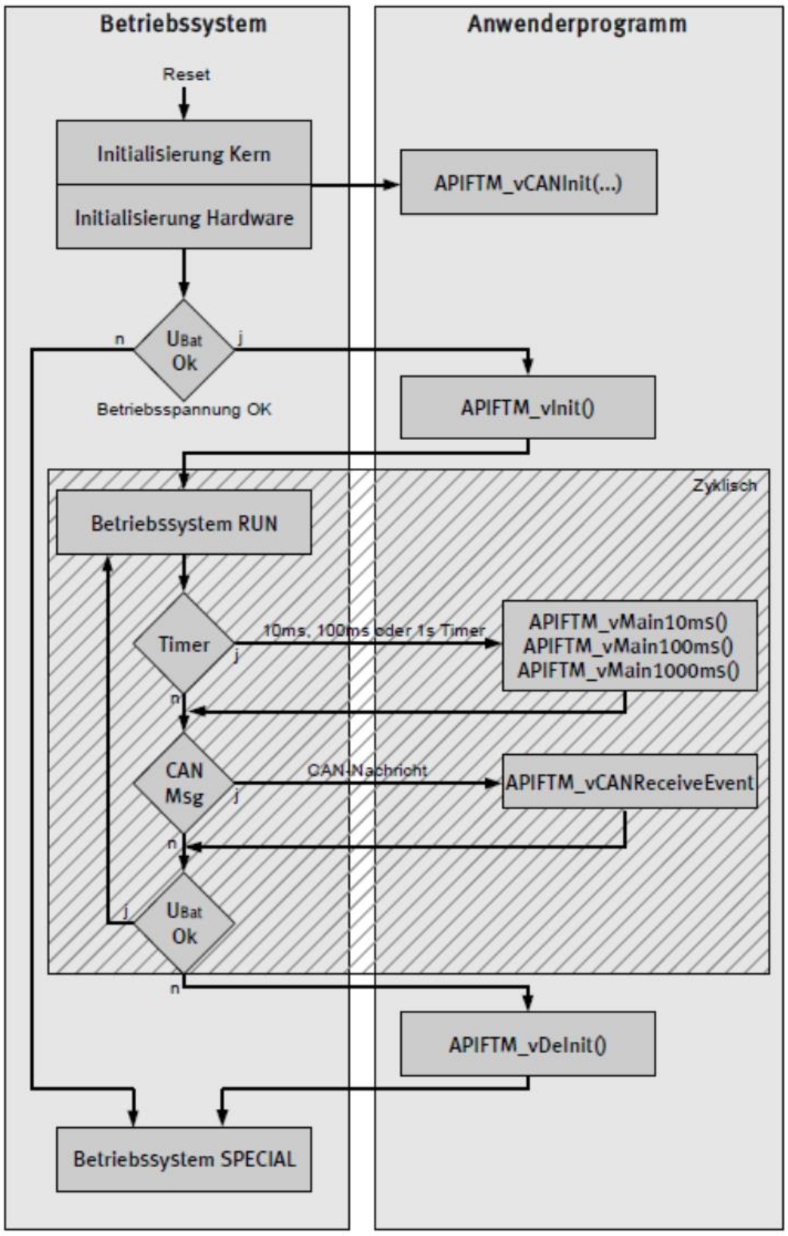
The processing of the data is similar to the FTM configuration software. With the following changes:

- The user program does the calculation of the output process image.
- The user program must trigger the reading and outputting of the images.
- CAN messages are not stored in the input process image.
- There are no cyclic CAN messages sent.



11.2.2 Operating system and application software

The operating system is compiled along with the user program and installed in the form of a binary file. It will not run without the user program. The operating system calls different functions of the application program (once or periodically). These must always be present. All functions of the application program start in these calls.



The user program can call custom and given (API) functions.

The operating system provides functions to control the FTM module. These functions are explained in detail in the section about the API.

<u>**Note:**</u> All functions are not allowed to exceed the maximum processing time of 1ms.

Infinite loops and longer-term calculations are not allowed. If, however, extensive calculations need to be performed, the processing needs to be distributed over multiple time slices.

11.3 API types and variables

11.3.1 “Typ API_tstCANData”

The type “API_tstCANData” represents a CAN message.

Type	Name	Description
tWord	dwIdentifier	CAN ID, up to 29 bits
tByte	bIs29Bit	ID format, whether message is sent / received with 11 or 29 bit ID
tByte	bLength	DataLength code, number of data bytes, length of the array ab-Data[]
tByte	abData[]	up to 8 bytes of data

11.3.2 Enumeration “API_tenCAN_Baudrate”

The enumeration type “API_tenCAN_Baudrate” represents the possible CAN messages.

Value	Name	Description
0	API_eneCANBaudrate1M	Baud rate 1MBaud
1	API_eneCANBaudrate500k	Baud rate 500 kBaud
2	API_eneCANBaudrate250k	Baud rate 250 kBaud
3	API_eneCANBaudrate125k	Baud rate 125 kBaud
4	API_eneCANBaudrate100k	Baud rate 100 kBaud

11.3.3 Type “API_tstCANInitParams”

The type “API_tstCANInitParams” sets the initial values of the CAN interface.

Type	Name	Description
tWord	dwServiceRxId	The CAN module accepts CAN messages with this ID as a service message
tWord	dwServiceTxId	The CAN module transmits CAN messages with this ID in response to service messages
tByte	boServiceRxIdType	ID format, whether service message is received with 11 or 29 bit
tByte	boServiceTxIdType	ID format, whether service message is sent with 11 or 29 bit
API_tenCAN_Baudrate	eneCANBaudrate	Baud rate

11.3.3.1 Enumeration “API_tenMC33PortParam”

The enumeration type “API_tenMC33PortParam” represents the possible settings for the input driver component of the 1G2/1G3 CAN module. Here, each input can be configured individually.

Value	Name	Description
0	API_eneMC33PortPU4mA	Measuring current of 4 mA at input to ground
1	API_eneMC33PortPU32mA	Measuring current of 32mA at input to ground
2	API_eneMC33PortPD2mA	Measuring current of 2mA at input to UBat
3	API_eneMC33PortPD16mA	Measuring current of 16mA at input to UBat
4	API_eneMC33PortTristate	No measuring current, analog input

11.3.4 Variable “API_VERSION”

The variables “API_VERSION_1” and “API_VERSION_2” determine the software version. These variables must be present in the application program.

Type	Name	Description
const tByte	API_Version_1	This variable is required as part 2 (2.x.0) of the software version
const tByte	API_Version_2	This variable is required as part 3 (2.0.x) of the software version

The major version number (x.0.0) is set by the operating system.

11.4 Required API functions

11.4.1 Function “APIFTM_vCANInit”

This function is called once during initialization of the CAN driver. A pointer is passed, where the application can store their parameters before the hardware is initialized. If the user program does not write values into the structure, the following default values are used:

Name	IO1	IO2	IO3	IO4 / Nano	IO5
dwServiceRxId	0x0E	0x16	0x1E	0x26	0x2E
dwServiceTxId	0x0F	0x17	0x1F	0x27	0x2F
boServiceRxIdType	0	0	0	0	0
boServiceTxIdType	0	0	0	0	0
eneCANBaudrate	500 kBaud	500 kBaud	500 kBaud	500 kBaud	500 kBaud

This function is called by the system before “APIFTM_vInit”.

11.4.2 Function “APIFTM_vInit”

This function is called once by the operating system during initialization. Here the user program should initialize the attached hardware and process images. It is recommended to use the following API functions here:

- APIFTM_bSetMC33PortParameter(...)
- APIFTM_vInitDIN()
- APIFTM_bInitDOUT(..)

11.4.3 Function “APIFTM_vDeInit”

This function is called, if the operating system leaves the "Run" state. Reasons for leaving the this state are:

- Undervoltage

In this case, the application program should shut down the connected hardware safely and/or disable it. It is recommended to use the following API function:

- “APIFTM_vStopDOUT”

11.4.4 Function “APIFTM_vMain10ms”

This function is called cyclically every 10 ms, if the operating system is in the “Run” state.

11.4.5 Function “APIFTM_vMain100ms”

This function is called cyclically every 100 ms, if the operating system is in the “Run” state.

11.4.6 Function “APIFTM_vMain1000ms”

This function is called cyclically every second, if the operating system is in the “Run” state.

11.4.7 Function “APIFTM_vStateSpecialInit”

This function is called once, when the operating system switches to the special condition.

Reasons for the change into the special condition:

- Undervoltage

11.4.8 Function “APIFTM_vStateSpecial_DeInit”

This function is called once when the operating system exits the special state and changes back to the "Run" state. Reasons for the changes in the "Run" state:

- Power supply is OK again

11.4.9 Function “APIFTM_vStateSpecialMain10ms”

This function is called cyclically every 10 ms, if the operating system is in the special condition.

11.4.10 Function “APIFTM_vStateSpecialMain100ms”

This function is called cyclically every 100 ms, if the operating system is in the special condition.

11.4.11 Function “APIFTM_vCANReceiveEvent”

This function is called by the operating system, if any CAN telegram was received. A pointer is passed, where the application can retrieve the data of the CAN message. The data of the received telegram is only available within this function and needs to be stored in the user program, in case it has to be used later.

11.5 API-Functionen

11.5.1 “APIFTM_vInitDIN”

The function “APIFTM_vInitDIN()” initializes the input process image and the hardware connected to the controller. It should be called within the function “APIFTM_vInit()”.

This function has no parameters.

Note: If the parameters of the input driver component (1G2/1G3) are set by the function “APIFTM_bSetMC33PortParameter(...)”, this has to be done before calling the function “APIFTM_vInitDIN()”.

11.5.2 “APIFTM_bInitDOUT”

The function “APIFTM_bInitDOUT()” initializes the process output image and the hardware connected to the controller. This function has a parameter of the type Tword that sets the frequency of the PWM outputs 1 and 2 from 2 Hz to 1000 Hz in 0.1 Hz steps.

This function returns zero if the initialization was successful. If another value is returned, an error has occurred and the hardware has not been initialized.

11.5.3 “APIFTM_vStopDOUT”

The function “APIFTM_vStopDOUT()” turns all outputs off. This function has no parameters.

11.5.4 “APIFTM_bSetMC33PortParameter”

The function “APIFTM_bSetMC33PortParameter(...)” determines how the input driver component of the 1G2/1G3 behaves. By default each port is set to 4mA pullup. Changes only take effect after the function “APIFTM_vInitDIN()” is called. It is recommended to set the MC33 parameters in the function “API_vInit()” and then execute “APIFTM_vInitDIN()”.

This function has the following parameters:

- tByte: Port of MC33 975, 0 to (8 or 12) depending on the hardware
- “API_tenMC33PortParam”: list of possible functions

This function returns zero if the parameters have been saved. If another value is returned, at least one parameter is incorrect.

Note: Changing the parameters at runtime requires reinitialization of the input process image. This can lead to data loss.

11.5.5 “APIFTM_bSetAnalogSamples”

The function “APIFTM_bSetAnalogSamples(...)” determines how often an analog value is converted before it is stored in the analog process image.

By default there are zero samples set for each channel. Nothing is converted.

This function has the following parameters:

- tByte: input channel for which the value is set; 0 corresponds to IN1, 1 to IN2, etc.
- tByte: number of conversions, ranging from 0 (no conversion) to a maximum of 15

Changes to this parameter take effect immediately and can be changed as desired.

This function returns zero if the parameters have been saved. If another value is returned, at least one parameter is incorrect.

Note: The A/D conversion requires computing time, thus unused inputs should not be converted.

Note: When using the input driver component of the 1G2/1G3 module, only one input is converted every 10 ms. Therefore, at the beginning of the process there is no data for the inputs. If several in-puts are converted from analog to digital, the result of all inputs is available after the conversion of all ports (t = n * 10ms).

11.5.6 “APIFTM_bSetPowerThreshold”

The function “APIFTM_bSetPowerThreshold(...)” sets the threshold voltage in 100mV steps, under which the operating system changes to the special (error) state. All outputs are disabled.

Changes to this parameter take effect immediately.

This function has the following parameters:

- tByte: voltage ranging from 50 to 255 in 100mV steps

This function returns zero if the parameters have been saved. If another value is returned, at least one parameter is incorrect.

11.5.7 “APIFTM_vPAIn”

The function “APIFTM_vPAIn()” reads the analog and digital process image from the hardware and stores the data in the input process image. This function has no parameters.

11.5.8 “APIFTM_vPAOut”

The function “APIFTM_vPAOut()” writes the analog and digital process image to the hardware.

This function has no parameters.

11.5.9 “APIFTM_bGetPAAnalog”

The function “APIFTM_bGetPAAnalog(...)” returns an entry from the digital input process image. This function has the following parameters:

- tByte: input 0 to 15, or virtual input 32 to 47
- tByte *: pointer to a byte with the return value

This function returns zero if the parameters were found. If another value is returned, at least one parameter is incorrect.

Note: The virtual inputs provide additional information about the state of the output driver.

Reserved [1Bit] Bit 3	Status [2Bit] Bit 1..2	State output [1Bit] Bit 0
	00: no error 01: overtemperature 10: idle 11: short circuit	0: output = OFF 1: output = ON

Special status “no load”: the CAN I/O Module can detect whether a load is connected to a low current output. For all other CAN I/O Modules, this feature is not available.

11.5.10 “APIFTM_bGetPADigital”

The function “APIFTM_bGetPADigital(...)” returns an entry from the analog input process image.

This function has the following parameters:

- tByte: input 0 to 12, or virtual input 24 to 31
- tWord *: pointer to a word with the return value

This function returns zero if the parameters were found. If another value is returned, at least one parameter is incorrect.

11.5.11 “APIFTM_vSetPADigital”

The function “APIFTM_bSetPADigital(...)” stores a digital value 0 or 1 in the output process image. This function has the following parameters:

- tByte : output 0 to 10
- tByte : 0 = off, otherwise on

This function returns zero if the value has been stored. If another value is returned, at least one parameter is incorrect.

11.5.12 “APIFTM_vSetPAAnalog”

The function “APIFTM_vSetPAAnalog(...)” stores an analog value from 0 to 100% in the output process image. This function has the following parameters:

- tByte : output 0 to 6 (depending on the hardware used)
- tByte : analog value from 0 to 100%

This function returns zero if the value has been stored. If another value is returned, at least one parameter is incorrect.

11.5.13 “APIFTM_bSendCANMessage”

The function “APIFTM_bSendCANMessage” sends a CAN message. Therefore, a pointer to the CAN data that has to be sent needs to be passed. This function has the following parameters:

- API_tstCANData * : pointer to an existing structure in the user program that needs to be sent

This function returns zero if the message has been sent. If another value is returned, at least one parameter is incorrect.

11.5.14 APIFTM_vSleep[1]

The function “APIFTM_vSleep()” puts the controller to sleep. This function has the following parameters:

- tDWord : time to sleep in ms

This function returns zero if the controller was asleep.

11.5.15 APIFTM_vStandby[2]

The function “APIFTM_vStandby()“ puts the controller to sleep. This function has the following parameters:

- tDWord : time to sleep in ms

This function returns zero if the controller was asleep.

11.5.16 APIFTM_vReset

The function “APIFTM_vReset” will reset the controller. This function has no parameters.

Note: The reset is performed immediately. This function does not return.

[1] not implemented

[2] not implemented

11.6 Development environment

The development environment describes all the tools that are needed to create an executable binary file from the operating system libraries and the user program and to transfer it to the hardware.

Development process:

1. Compiling the source files

2. Linking libraries and objects

3. Creating the firmware binary file

4. Flashing and debugging the software

The individual steps are discussed in the following sections.

11.6.1 Compiling the source files

The application software consists of at least one file *.c file. When delivered, this is the file “Template_IOx.c”. It is located in the project directory in the “src” folder. Other source files must be placed in this folder. The header files are located in the “inc” folder. Also “cdef.h” and “APIFTM.h” are located there. Other header files must be placed in this folder. Shall own source files be added, they must be included in the following files:

- div/makeglobal : add the files (name) to a new line in the section "CFILES"
- div/make.IOx : add the files (name) to a new line in the section "CFILES" if the file is only needed for this hardware
- div/CANIO_IOx.lkf : add the name below the line „out/IOx/IOx.o“

For Compiling the following compiler is required:

- “Cosmic compiler” for HC08 version 4.x

The path to the installation directory of the compiler must be set in the file “div/makeglobal”.

In the project directory, the file “make.exe” is located. It is required for automated compilating and linking. It will use the script called "makefile".

The following project-specific data can be set:

- PRJNAME : Project name CANIO
- SubSystems : Name of the subsystems, the different types of hardware (1G1 to 1G5)

To start compiling you can either execute “Make_all.bat” or alternatively run “make.exe” with specified parameters:

- make all : compile all hardware systems
- make clean : remove all object and binaries files
- make SubSysforDevelopment : compile selected hardware systems, refer the makefile for the list of subsystems

Note: The path to the compiled code cannot contain spaces or special characters.

11.6.2 Linking libraries and objects

The linking of the libraries is done automatically while creating the software. Therefore the operating system libraries have to be located in lib/IOx/. The following files are needed:

- lib/IOx/CANIO_IOx.lib
- lib/IOx/CANIO_IOx_FWL.lib
- lib/IOx/CANIO_IOx_RAM.lib

11.6.3 Creating the firmware binary file

The firmware file is automatically generated when you create the software and stored in the folder out/IOx. Several different files will be generated, required for different debuggers.

- *.elf : Executable and Linkable Format
- *.s19 : Motorola S19 file
- *.map : overview of memory usage
- *.afw : software file that is required by the CAN I/O Module. Update over the Toolchain

11.6.4 Flashing and debugging the software

There are two ways to flash the generated software on the hardware:

- Debugger “NoICE”
- Toolchain (Firmware Update)

Debugger „NoICE“

With this tool, the developed software can be flashed on the hardware, where it can then be executed and debugged. For easy starting, *.noi files exist in the project folder.

When opening the file, the debugger is started and the specified software is immediately loaded and flashed. It is possible to write variables to these files that should be constantly monitored.

Note: If software is installed with the “NoICE” debugger the FTM module is not secured.

Note: The “NoICE” debugger software can also upload software that does not fit the hardware used. This can damage the hardware!

Firmware Update mit Toolchain

With the help of the tool “Toolchain” *.afw files can be transferred to the CAN I/O Modules. These *.afw files are used to update software to the next version. The Toolchain checks the existing software and the hardware identifier, thus only the matching files will be transferred.

The firmware update uses the CAN bus but there should be only one device connected. After the update the CAN-I/O-Module is secured.

11.6.5 Secure und unsecure

To prevent unauthorized access to the software on the CAN I/O Modules by third parties, each module must be “secured”.

To reuse an inadvertently secured CAN I/O Module, there is an unsecuring tool. The manufacturer P&E has a “P&E Embedded Multilink Toolkit” including the “Unsecure Utility”. It deletes the entire contents of the flash and allows reflashing and debugging the firmware.

For technical support, please contact:

solution@miunske.com

Information about updates and new developments are maintained on the miunske homepage

https://miunske.com/entwicklung