PCAN-MIO

Universal Controller for
CAN Applications

Configuration Tutorial

Document version 1.2.0 (2017-11-24)

Relevant products

Product Name	Model	Part Number
PCAN-MIO	Industrial plug connector (Phoenix) Automotive plug connector (Tyco)	IPEH-002187 IPEH-002187-A
PCAN-Explorer 5		IPES-005028
PPCAN-Editor 2, PCAN-View, PEAK-Converter		

PEAK-System Technik GmbH
Otto-Roehm-Straße 69
64293 Darmstadt
Germany

Phone: +49 (0)6151 8173-20
Fax: +49 (0)6151 8173-29

www.peak-system.com
info@peak-system.com

Document version 1.2.0 (2017-11-24)

Contents

1 Introduction

Working successfully with the PPCAN-Editor requires at least some basic understanding by the user regarding hardware knowledge and programming experience.

This tutorial therefore addresses owners of a PCAN-MIO who are trying to do some more complex configurations of the device, using their skills from electronics and informatics education.

At first, you should try to get familiar with the free PPCAN-Editor following the steps of this tutorial. When experiencing more and more difficulties with understanding the matter and proceedings, this may at least serve as an indication for the future use of the PPCAN-Editor: when deciding against the effort, PEAK System offers a configuration service subject to detailed specifications.

1.1 Prerequisites for Operation

For reasonably processing this tutorial and for solving the exercises, a PCAN-MIO (with sufficient power supply) should be at hand. Its CAN busses should be connected to a computer via PEAK interfaces and also should be terminated properly, e.g. using the internal DIP switches (see PCAN-MIO hardware manual for this).

- Two CAN busses are connected to the PC via PEAK interfaces, with 500 kbit/s each
- The PPCAN-Editor software is installed
- As a remote CAN participant, e.g. a PCAN-View (or even better: a PCAN-Explorer part no. IPES-005028) is installed on the PC. Access the interfaces with the mentioned software, thus connecting with one CAN bus each

The device PCAN-MIO offers the following resources for configuration:

- Device ID (4 bit, 0...15 dec) may be adjusted from outside using a rotary switch
- 2 High-speed CAN channels via pluggable transceiver modules. Alternatively, low speed, single wire, and opto-decoupled high speed modules, as well as high-speed modules without wake-up function available
- CAN bit rates[1] (10k; 20k; 33,3k; 47,6k; 50k; 83,3k; 95,2k; 100k; 125k; 250k; 500k; 1M)
- CAN messages (11-bit and 29-bit)
- 1 RS-232 connector (currently works only by using a special firmware)
- 1 output 5 V/200 mA
- 8 digital inputs with switchable pull-ups (low active) or pull-downs (high active)
- 8 digital outputs, 6 of these switchable as low-side, high-side or push-pull, 2 of these with PWM capability (only high-side)
- 6 analog inputs (10 bit, 0 - 10 V)
- 2 analog outputs (10 bit, 0 - 10 V)
- Wake-up function using separate input

[1] Bit rates may be adjusted freely, but actual function is depending on equipped transceiver types

2 The Configuration Concept

Most of the microcontroller-equipped devices from PEAK-System offer possibilities to link any of their internally accessible resources with each other. For this, the firmware allows virtual wiring of the hardware resources by several means, e.g. so called Function Blocks, among others. For creating, editing, and managing configurations, PEAK-System offers the PPCAN-Editor for free download from their website.

Files created this way along with the enclosed configuration are stored to the PC at first, then transferred via CAN to the PCAN device (upload) and stored there non-volatile. Some devices can hold several configurations: the active one is then determined by means of a selector switch.

Project files created with the PPCAN-Editor may contain several configurations. The device ID selects the one to be executed when the device starts. This offers the possibility to wire several devices with different IDs to the same CAN bus and to upload the same multi-configuration file to them all. The different ID of each device will let them load their individual configuration from the nonvolatile memory and therefore executes a different task each.

2.1 Possibilities of Configuration

Linking of internal resources can be done using straight assignment, the simple scaling of values, as well as applying methods "CAN gateway services", "Default values", "Function blocks", "Event based messaging", "Time events", and "Characteristic curves". Devices with only one CAN bus do not support the "Gateway services", and "Time events" may also be missing on some of the smaller platforms. All available resources of a device are reported to the PPCAN-Editor by applying a special file related to that hardware. This so called "hardware profile" lets the PPCAN-Editor allow or restrict configuration possibilities correspondingly. The user instead may refer to the hardware manual of a specific device (see www.peak-system.com for free manual download).

2.2 Scaling

The most elementary means of manipulating values is using the four basic arithmetics. They are controlled with parameters SCALE and OFFSET, taken from mathematics well known linear equation. Here, the parameter SCALE decides on multiplication (if > 1) respectively division (if < 1), whereas parameter OFFSET is responsible for addition (if > 0, positive) respectively subtraction (if < 0, negative). As a neutral setting, SCALE = 1 and OFFSET = 0 are preset by default.

2.3 CAN Gateway Services

Incoming messages on one CAN bus may be (selectively) forwarded to a different CAN bus. Or they may be transmitted on the same CAN bus but with a different ID (e.g. conversion 11-bit <-> 29-bit). Or an incoming message may be used to trigger transmission of a completely different message.

2.4 Default Values

When defining parameter values here, the module's resources may be preset from the start. For example, a non-default bit rate of a CAN bus may be set here, activation of a 5 V supply output for sensors, LEDs, and ports may be switched logically, etc.

2.5 Function Blocks

In the case that simple manipulation of values using SCALE and OFFSET turned out to be insufficient, the firmware offers so called function blocks with even more complex capabilities. Such functions are e.g. value mapping with X/Y tables or matrices, hysteresis functions, delays, counters, timers, low pass filters, a comprehensive collection of mathematical and logical functions up to a complex PIDT1 closed-loop control. Function blocks are processed sequentially or conditionally.

2.6 Event-triggered Transmission of CAN Messages

For CAN messages to be transmitted conditionally, a pool of trigger conditions is available. Also CAN messages can be requested from distant nodes (RTR mechanism supported).

2.7 Characteristic Curves

An incoming X value results in the output of the assigned Y value. Here, 2 to 31 X/Y translation pairs may be defined. X values in between two X/Y pairs will return a Y value linear interpolated from the available Y points. In other words: characteristic curves allow value manipulation in a way like up to 31 different SCALE and OFFSET values would do. Using this, segments of the curve may be influenced in their gradient to define plateaus or discontinuous functions.

3 List of Exercises

An overview on the manifold capabilities of the PCAN hardware (like the PCAN-MIO) may be given when solving the following exercises. For further literature references see appendix A.

- 1a) Forwarding of all messages from CAN-0 to CAN-1 (Gateway)
- 1b) Definition of CAN messages
- 1c) Routing the defined messages from CAN-0 to CAN-1
- 1d) Combination 1a and 1c
- 1e) Cyclic transmission of a message on CAN-1 with signals from CAN-0
- 1f) Scale und Offset
- 1g) Exporting a symbol file
- 1h) Creating a simple instrument panel
- 2a) Reading digital inputs and transmission with a CAN message
- 2b) Setting of digital outputs according to a received CAN message
- 2c) Reading analog inputs and transmission with a CAN message
- 2d) Setting of analog outputs according to a received CAN message
- 2e) Switching the 5 V output
- 3a) Manipulating a CAN signal using the Characteristic Curve function block
- 3b) Threshold with hysteresis
- 3c) Transmission only on signal changes
- 4a) Setting the CAN bit rate (default value)

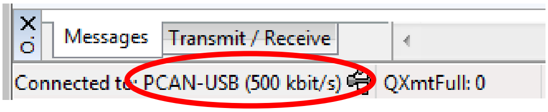
4 Solutions Explained Stepwise

You may find further information on use of the PPCAN-Editor in the online help: to be opened from the program's help menu or by pressing the F1 key.

4.1 Exercise 1a: Forwarding of all Messages from CAN-0 to CAN-1 (Gateway)

1. Start the PPCAN-Editor and proceed as follows.
2. Assign the PPCAN-Editor to a PEAK-Interface, e.g. PCAN-USB. Select the menu item **CAN > Connect** and from there the appropriate CAN interface or CAN network.

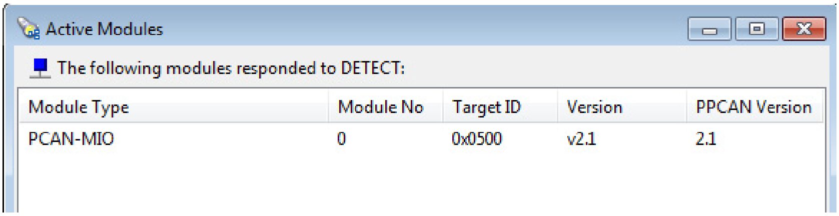
   The selected connection is displayed in the status bar of the PPCAN-Editor (bottom left corner).



Connected to: PCAN-USB (500 kbit/s) QXmtFull: 0

3. Check whether the PCAN-MIO can be found on the CAN network by selecting the menu item **Transmit > Detect Modules**.

   The Active Modules window lists the available devices (here PCAN-MIO) along with some status information.


Active Modules

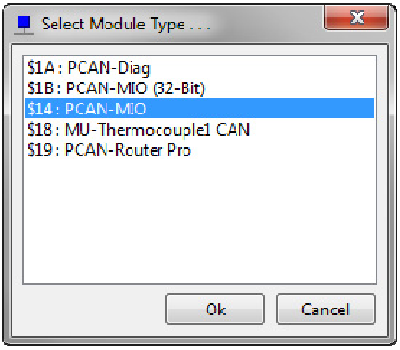
The following modules responded to DETECT:

Module Type	Module No	Target ID	Version	PPCAN Version
PCAN-MIO	0	0x0500	v2.1	2.1

The "Module No" column shows the currently active device ID, 0 in this case. The "Version" column indicates the firmware version.

4. Create a new empty configuration file using the menu item **File > New**.
5. For creating a new device configuration within the configuration file select the menu item **Edit > New Configuration**.

   PPCAN-Editor asks for the type of hardware to be configured. PPCAN-Editor can configure several different PCAN devices, equipped with individual resources each. Therefore, for each type of device a list of available resources is supplied by the manufacturer.


Select Module Type . . .

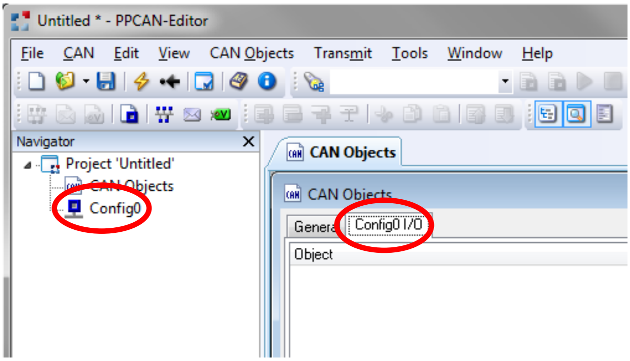
$1A : PCAN-Diag
$1B : PCAN-MIO (32-Bit)
$14 : PCAN-MIO
$18 : MU-Thermocouple1 CAN
$19 : PCAN-Router Pro

6. Select the profile for the PCAN-MIO and confirm with **Ok**.

Note: From serial number 100 onward, PCAN-MIO's work with a different configuration structure. Therefore, when creating a new configuration within the PPCAN-Editor, one must select the appropriate hardware profile: either "PCAN-MIO" up to serial number 99, or "PCAN-MIO (32-Bit)" from serial number 100 onward.

New I/O functions which are only available in the hardware profile "$1B PCAN-MIO (32-Bit)" cannot be copied to configurations based on the profile "$14 PCAN-MIO ".

Besides the **General** tab, a new tab has been created, entitled with the configuration's name Config0 I/O. Also the navigator (at the left window edge) now contains an additional icon named Config0.



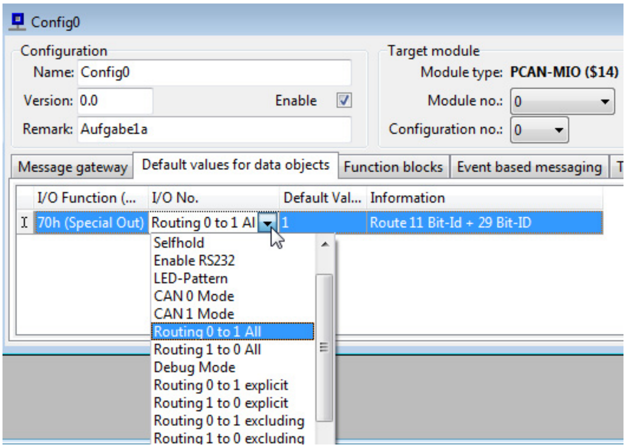
7. Double click on the Icon Config0 at the left window edge.

   It opens a new configuration window. Here more complex linkages of resources can be made which go beyond the simple assignment and scaling.

8. Select the **Default values for data objects** tab, open the context menu (right click), and select **Add Record**.

   The cell content can be edited by either pressing F2, or by a slow double click, or by simply typing the new value.

9. In the table row enter the following values:



Config0

Configuration
Name: Config0
Version: 0.0 Enable ☑
Remark: Aufgabe1a

Target module
Module type: **PCAN-MIO ($14)**
Module no.: 0
Configuration no.: 0

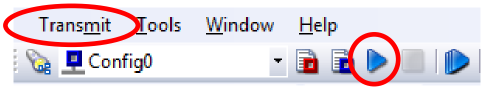
	I/O Function (...	I/O No.	Default Val...	Information
I	70h (Special Out)	Routing 0 to 1 All	1	Route 11 Bit-Id + 29 Bit-ID

- I/O Function: 70h SpecialOut (a virtual module resource)
- I/O No: Routing 0 to 1 All
- Default value: 3 (1=11-bit IDs, 2=29-bit IDs, 3=both types)
- Information: description of what this line does

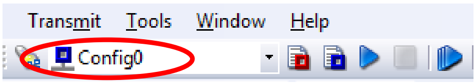
In this case all incoming messages on CAN-0 are forwarded 1:1 to CAN-1. For example, a control unit in the crash area of a vehicle receives the same information, but cannot block the rest of the bus in case of damage.

10. Save the configuration project file (*.ppproj) as Exercise 1a to your PC. To do so, select the menu item **File > Save As.**

11. The configuration must be transmitted to the PCAN-MIO via CAN bus (Upload). For this, select the menu item **Transmit > Send Configuration** or click the corresponding icon on the toolbar.



Important Note: Ensure that the list box in the toolbar shows the name of your configuration Config0.



While uploading, the Output window of the PPCAN-Editor shows several progress messages.

The status LED of the PCAN-MIO flashes during the transmission and processing of the configuration file. When the status LED flashes red once, the configuration was processed and an automatic device reset was performed. Thereafter, the status LED blinks slowly green at 1 Hz thus indicating that the PCAN-MIO executes its new configuration.

4.2 Exercise 1b: Definition of CAN Messages

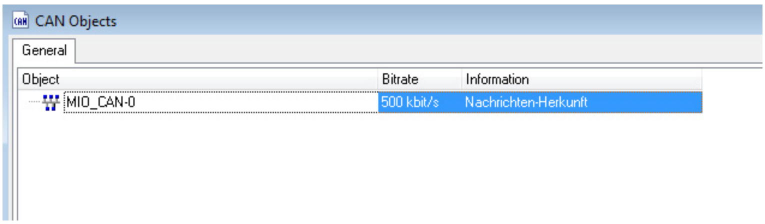
- Create a new empty configuration file.

1. Select the menu item **File > New**.

   An empty window appears where global CAN objects can be defined. If a file contains multiple configurations with different CAN objects, they all must be defined here. Later, they are imported selectively into the different configura-tions.

   In that window a new CAN bus is already created as Bus_0, underneath which global CAN objects can be created hierarchically.

2. Double click the name Bus_0 and enter the new name for it. Select the default bit rate 500 kbit/s from the dropdown list (The choice is informative only and has no effect).

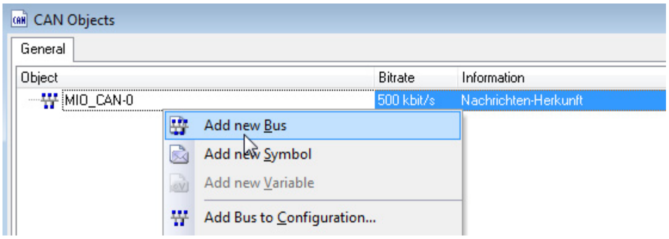


Object	Bitrate	Information
MIO_CAN-0	500 kbit/s	Nachrichten-Herkunft

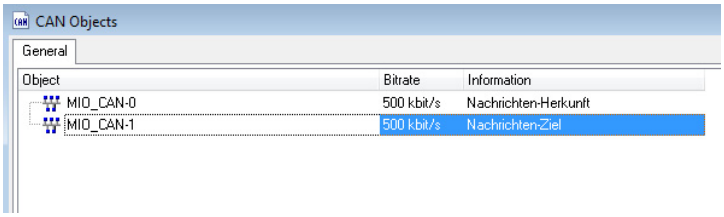
- Bus name: MIO_CAN-0
- Bit rate: 500 kbit/s
- Information: description of what this line does

In order to be able to route, you will also need the second CAN bus of the PCAN-MIO.

3. Open the context menu of the window (by right click on the white background area) and select **Add new Bus**.



4. Double click the name Bus_0 and enter the new name for it. Select the default bit rate 500 kbit/s from the dropdown list (The choice is informative only and has no effect).



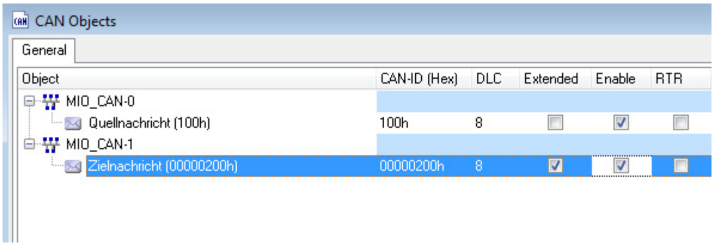
Object	Bitrate	Information
MIO_CAN-0	500 kbit/s	Nachrichten-Herkunft
MIO_CAN-1	500 kbit/s	Nachrichten-Ziel

- Bus name: MIO_CAN-1

- Bit rate: 500 kbit/s
- Information: description of what this line does

Create at least one message on each bus, each with at least one signal.

5. In the context menu (right click) of each bus select **Add new Symbol** (creates a new CAN message) and enter the parameters into the messages:



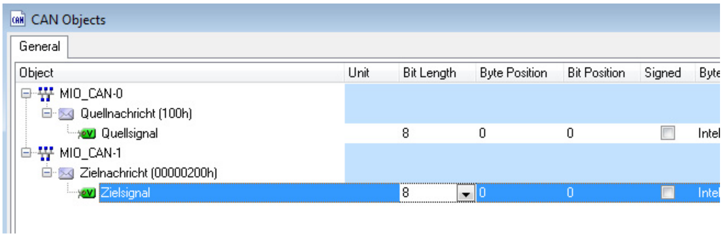
CAN Objects

General

Object	CAN-ID (Hex)	DLC	Extended	Enable	RTR
MIO_CAN-0					
Quellnachricht (100h)	100h	8	☐	✓	☐
MIO_CAN-1					
Zielnachricht (00000200h)	00000200h	8	✓	✓	☐

- Symbol name: Source message and Target message
- CAN ID: 0x100 (11-bit ID) and 0x00000200 (29-bit ID)
- DLC: 8 (message has 8 data bytes)
- Extended: decide on 11-bit ID or 29-bit ID
- Enable: yes, both messages shall be used
- RTR: no, message shall be transmitted always, -not- only on request

6. In the context menu (right click) of each CAN message select **Add new Variable** (creates a new CAN signal) and enter the parameters into the variables:


CAN Objects

General

Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte
MIO_CAN-0						
Quellnachricht (100h)						
Quellsignal		8	0	0	☐	Intel
MIO_CAN-1						
Zielnachricht (00000200h)						
Zielsignal		8	0	0	☐	Intel

- Variable name: Source message and Target message
- Unit: useful with physical quantities, for information only
- Bit length: 8 (both signal each 1 byte wide, value range 0..255)
- Start Byte: 0 (signal starts at the message's edge)
- Start Bit: 0
- Signed: no, can’t be negative
- Byte Order: Intel format (LSB in byte 0/bit 0,
  MSB in byte 1/bit 7)

The empty body of both CAN messages is hereby defined, but not yet assigned to physical data sources. Therefore, a configuration must be created.

4.3 Exercise 1c: Routing the Defined Messages from CAN-0 to CAN-1

Create a new configuration within the configuration file.

1. Select the menu item **Edit > New Configuration**.

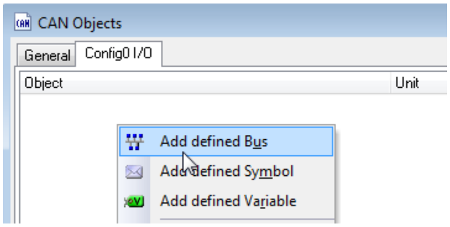
   PPCAN-Editor asks for the hardware to be configured.

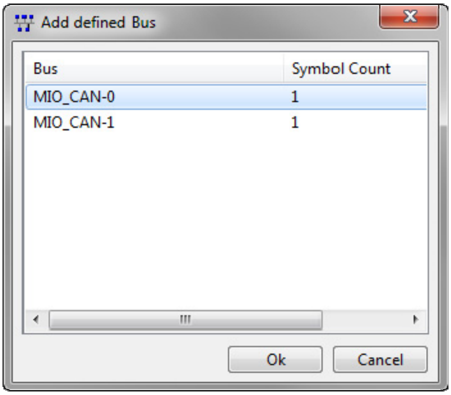
2. Select the profile "$1B PCAN-MIO (32bit)" and confirm with **Ok**.

Besides the **General** tab, a new tab has been created entitled with the configuration's name Config0 I/O. Also the navigator (at the left window edge) now contains an additional icon named Config0.

The globally defined CAN busses, messages, and signals should be used in this configuration. Therefore, they all must be imported.

3. Click on the new **Config0 I/O** tab for bringing it into the foreground. In the context menu (right click) select **Add defined Bus** for each bus.

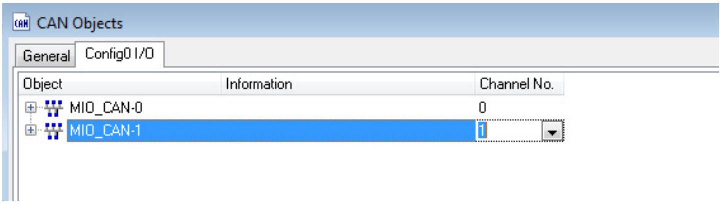




Bus	Symbol Count
MIO_CAN-0	1
MIO_CAN-1	1

The previously defined global CAN busses along with the contained messages and the 8-bit variables will be imported into the configuration.

4. The defined busses must get a number according to the hardware's CAN channels.



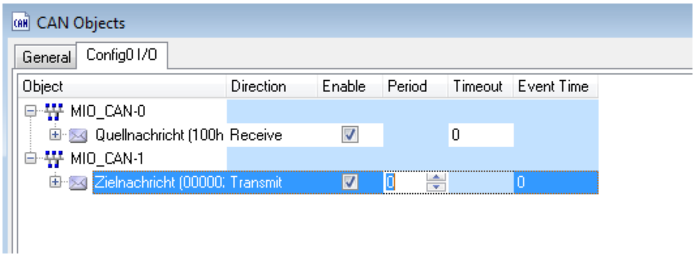
CAN Objects

General | Config0 I/O

Object	Information	Channel No.
MIO_CAN-0		0
MIO_CAN-1		1

- Channel No.: 0 and 1 (assign the hardware CAN channels)

5. Enter the parameters for the messages.



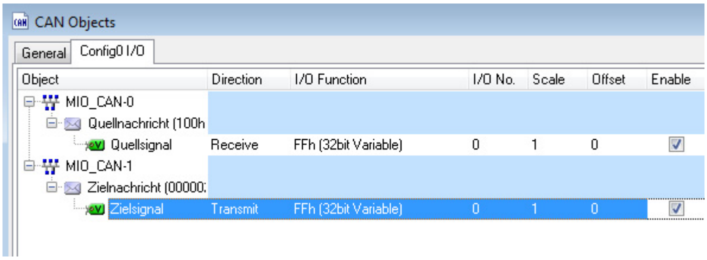
CAN Objects

General | Config0 I/O

Object	Direction	Enable	Period	Timeout	Event Time
MIO_CAN-0					
Quellnachricht (100h	Receive	✓		0	
MIO_CAN-1					
Zielnachricht (00000:	Transmit	✓	0		0

- Direction: Receive (message received from the bus) and Transmit (message transmit to the bus)
- Enable: yes, message should be transmitted
- Period: 0 (only for target message: the message is not transmitted cyclically)

6. Enter the parameters for the signals.



CAN Objects

General | Config0 I/O

Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
MIO_CAN-0						
Quellnachricht (100h						
Quellsignal	Receive	FFh (32bit Variable)	0	1	0	☑
MIO_CAN-1						
Zielnachricht (00000:						
Zielsignal	Transmit	FFh (32bit Variable)	0	1	0	☑

- I/O Function: FFh 32bit Variable (the incoming source signal is written into a 32-bit variable and read later on from there again)
- I/O Number: 0, using the 32-bit variable #0
- Scale: 1 (no scaling at all, like multiplying with 1)
- Offset: 0 (no shifting at all, like addition of 0)
- Enable: yes, this signal (within the message) should be used

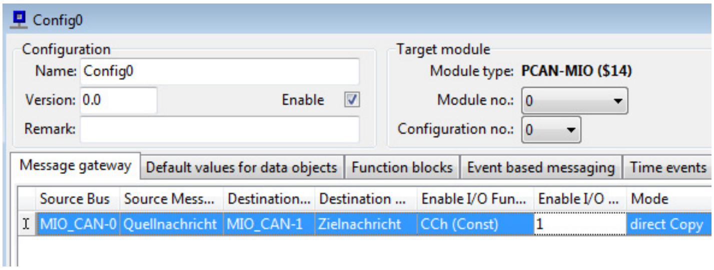
Now the two messages must be transferred to the gateway service, so that it converts from CAN-0 to CAN-1.

7. Double click to the Icon Config0 at the left window edge.

   The new configuration window Config0 opens.

8. Select the **Message gateway** tab, open the context menu (right click) and select **Add Record**.

A table row appears that routes one CAN message. Enter the parameters.



Config0

Configuration

Name: Config0

Version: 0.0 Enable ✓

Remark:

Target module

Module type: PCAN-MIO ($14)

Module no.: 0

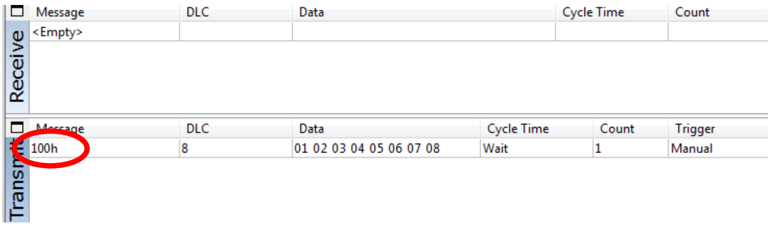
Configuration no.: 0

Message gateway | Default values for data objects | Function blocks | Event based messaging | Time events

Source Bus	Source Mess...	Destination...	Destination ...	Enable I/O Fun...	Enable I/O ...	Mode
MIO_CAN-0	Quellnachricht	MIO_CAN-1	Zielnachricht	CCh (Const)	1	direct Copy

- Source Bus: MIO_CAN-0
- Source Message-ID: ID 0x100 (11-bit format)
- Destination Bus: MIO_CAN-1
- Destination Message-ID: ID 0x00000200 (29-bit format)
- Enable I/O Function: CCh (Const), this routing should always be active
- Enable I/O Number: 1
- Mode: direct copy (transmit whenever something is received)

9. Save the configuration as exercise 1c to your PC.
10. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.

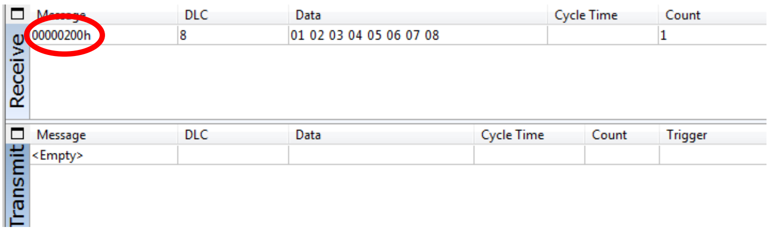


Receive

Message	DLC	Data	Cycle Time	Count
<Empty>				

Transmit

Message	DLC	Data	Cycle Time	Count	Trigger
100h	8	01 02 03 04 05 06 07 08	Wait	1	Manual



Message	DLC	Data	Cycle Time	Count
00000200h	8	01 02 03 04 05 06 07 08		1

Message	DLC	Data	Cycle Time	Count	Trigger
<Empty>					

When transmitting a message with ID 0x100 on CAN-0 to the PCAN-MIO, the contained data is forwarded in message ID 0x000000200 on CAN-1. Other messages are ignored.

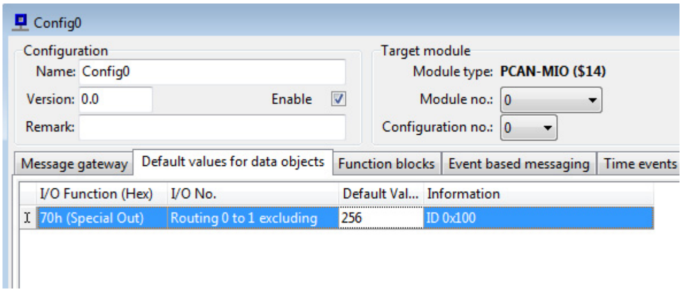
4.4 Exercise 1d: Combination 1a and 1c

All messages should be forwarded from CAN-0 to CAN-1 unchanged, except ID 0x100. This one should still be translated into ID 0x00000200.

1. Open exercise 1c and save as exercise 1d to your PC.
2. Double click to the Icon Config0 at the left window edge.

   It opens the new configuration window Config0.
3. Select the **Default values for data objects** tab, open the context menu (right click) and select **Add Record**.

Routing anything from CAN-0 to CAN-1, except for the specified ID.



Config0

Configuration

Name: Config0

Version: 0.0 Enable ✓

Remark:

Target module

Module type: **PCAN-MIO ($14)**

Module no.: 0

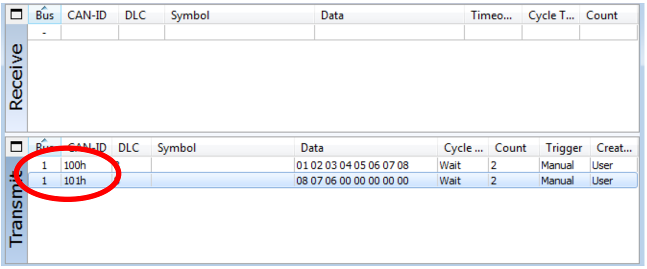
Configuration no.: 0

I/O Function (Hex)	I/O No.	Default Val...	Information
70h (Special Out)	Routing 0 to 1 excluding	256	ID 0x100

- I/O Function: 70h Special Out
- I/O Number: Routing 0 to 1 excluding (routing all from CAN-0 to CAN-1 except 0x100)
- Default Value: 256 (0x100, no routing of this ID. The parameter is represented decimal)

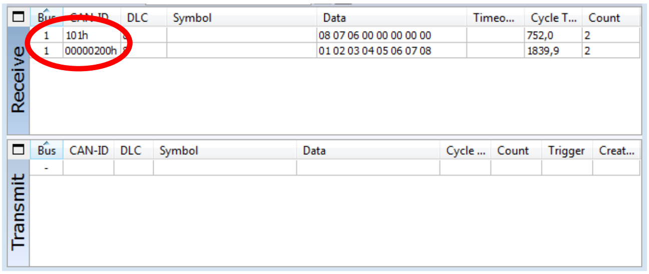
Note: Routing functions `explicit` and `excluding` only support 11-bit IDs. In this context, `explicit` means routing of only the specified messages and `excluding` means routing of everything except the specified messages.

4. Save the configuration as exercise 1d to your PC.
5. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.



Bus	CAN-ID	DLC	Symbol	Data	Timeo...	Cycle T...	Count
-							

Bus	CAN-ID	DLC	Symbol	Data	Cycle ...	Count	Trigger	Creat...
1	100h	[illegible]		01 02 03 04 05 06 07 08	Wait	2	Manual	User
1	101h	[illegible]		08 07 06 00 00 00 00 00	Wait	2	Manual	User



Receive

Bus	CAN-ID	DLC	Symbol	Data	Timeo...	Cycle T...	Count
1	101h	8		08 07 06 00 00 00 00 00		752,0	2
1	00000200h	8		01 02 03 04 05 06 07 08		1839,9	2

Transmit

Bus	CAN-ID	DLC	Symbol	Data	Cycle ...	Count	Trigger	Creat...
-								

When transmitting a message with ID ≠ 0x100 on CAN-0 to the PCAN-MIO, it is copied exactly onto CAN-1.

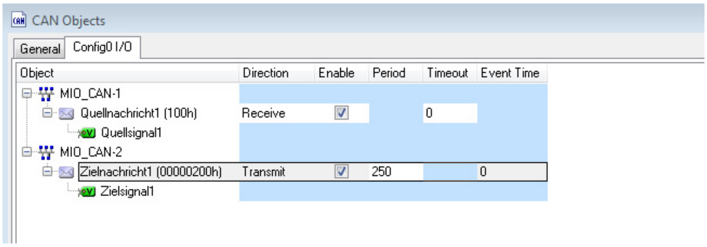
When transmitting a message with ID = 0x100 on CAN-0 to the PCAN-MIO, the contained data is transmitted in message ID 0x000000200 on CAN-1.

Note: This functionality is very useful if a new ECU shall be inserted into an existing vehicle bus and this device requires CAN signals in a different layout (new CAN database). Furthermore the rest of the vehicle bus is copied transparently.

4.5 Exercise 1e: Cyclic Transmission of a Message on CAN-1 with Signals from CAN-0

By setting a transmission period for the message 0x00000200, cyclic transmission of that ID takes place on CAN-1. The required data are still taken from the 32-bit variable #0, which is filled by CAN-0. On source message timeout, the last known content will be repeated.

1. Open exercise 1c and save as exercise 1e to your PC.
2. Open the **Config0 I/O** tab and set a cycle time for the transmit message.


CAN Objects

General | Config0 I/O

Object	Direction	Enable	Period	Timeout	Event Time
MIO_CAN-1					
Quellnachricht1 (100h)	Receive	✓		0	
Quellsignal1					
MIO_CAN-2					
Zielnachricht1 (00000200h)	Transmit	✓	250		0
Zielsignal1					

- Period: 250 ms (the message is now transmitted cyclically)

3. Save the configuration as exercise 1e to your PC.
4. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.

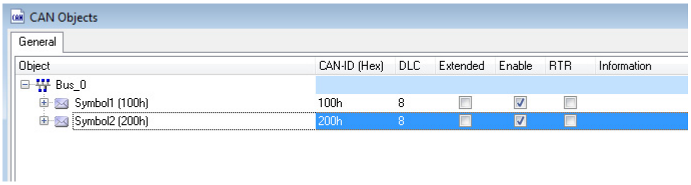
4.6 Exercise 1f: Scale and Offset

With Scale and Offset, values from a CAN bus can be manipulated like using the four basic arithmetics, all this without using a function block. For example, an increasing 8-bit value (rising ramp) can be inverted (falling ramp) 0 > 0xFF, 0xFF > 0.

To do so, the original value from the incoming message is written into a 32-bit variable (there is no smaller type), and when writing to the output message, this value is processed with Scale = -1 and Offset = 255. As an alternative, the incoming value can be processed immediately (before writing it to the 32-bit variable), and is then passed on to the CAN message directly.

Important Note: Make sure that the manipulated result value – under all conditions - fits into the 32-bit size.

1. Create an empty configuration file using the menu item **File > New**.
2. Add two symbols (CAN messages) to the already existing CAN bus.

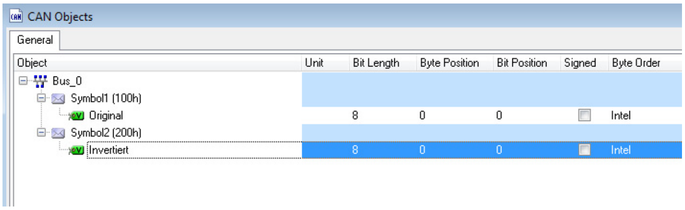


CAN Objects

General

Object	CAN-ID (Hex)	DLC	Extended	Enable	RTR	Information
Bus_0						
Symbol1 (100h)	100h	8	☐	✓	☐	
Symbol2 (200h)	200h	8	☐	✓	☐	

3. Each message will get an 8-bit variable (CAN signal).

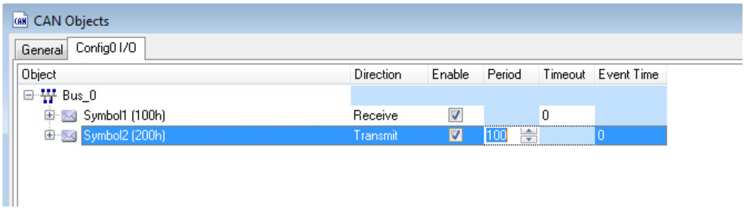


CAN Objects

General

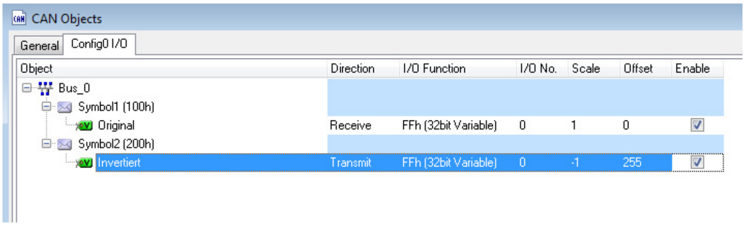
Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order
Bus_0						
Symbol1 (100h)						
Original		8	0	0	☐	Intel
Symbol2 (200h)						
Invertiert		8	0	0	☐	Intel

Create a new configuration within the file and import all CAN objects. The parameters then should be entered to meet this scenario: A message 0x100 is received, the contained signal is written into a 32-bit variable #0. A message 0x200 is transmitted cyclically (100 ms), the contained signal is taken from the 32-bit variable #0, inverted (Scale = -1) and lifted (Offset = 255).


CAN Objects

General | Config0 I/O

Object	Direction	Enable	Period	Timeout	Event Time
Bus_0					
Symbol1 (100h)	Receive	✓		0	
Symbol2 (200h)	Transmit	✓	100		0


CAN Objects

General | Config0 I/O

Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
Symbol1 (100h)						
Original	Receive	FFh (32bit Variable)	0	1	0	✓
Symbol2 (200h)						
Invertiert	Transmit	FFh (32bit Variable)	0	-1	255	✓

4. Save the configuration as exercise 1f to your PC.
5. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.

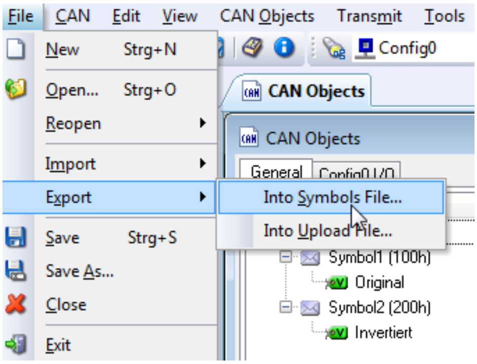
The data contained in the receive message 0x100 (data byte 0) is transmitted as y = (-1)*x+255 resp. y = 255-x in message 0x200. A rising ramp (increasing x values) is converted into a falling ramp (decreasing y values).

4.7 Exercise 1g: Exporting a Symbol File

For users of the PCAN Explorer (Part No IPES-005028) it is comfortable to display the content of CAN messages decoded to plain text. For this, a symbol file must be imported, created and saved to file by the PPCAN-Editor. It is based on the project from exercise 1f.

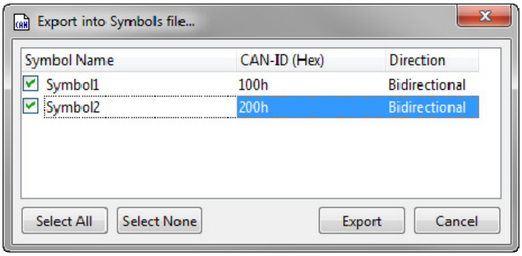
PPCAN-Editor:

1. Select the **General** tab to focus the data base.
2. Select the menu item **File > Export > Into Symbol File**.



3. Save the symbol file as exercise 1g to your PC.

   A window appears offering CAN objects to be exported. Click on those objects you wish to export, or click **Select All**.

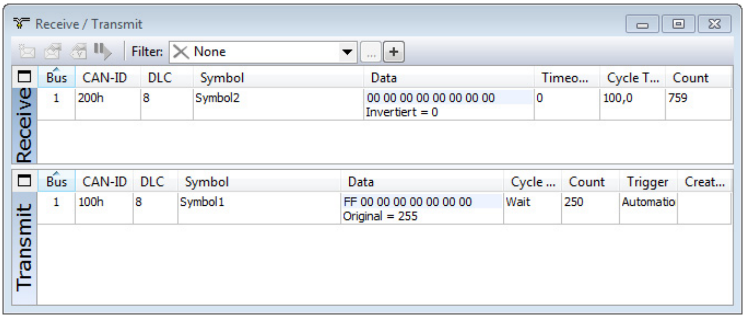

Export into Symbols file...

Symbol Name	CAN-ID (Hex)	Direction
✓ Symbol1	100h	Bidirectional
✓ Symbol2	200h	Bidirectional

4. Click the **Export** button. The symbol file is now created.

PCAN-Explorer:

5. The symbol file may be loaded with **File > Open** and activated with **File > Apply**.



Bus	CAN-ID	DLC	Symbol	Data	Timeo...	Cycle T...	Count
1	200h	8	Symbol2	00 00 00 00 00 00 00 00 Invertiert = 0	0	100,0	759

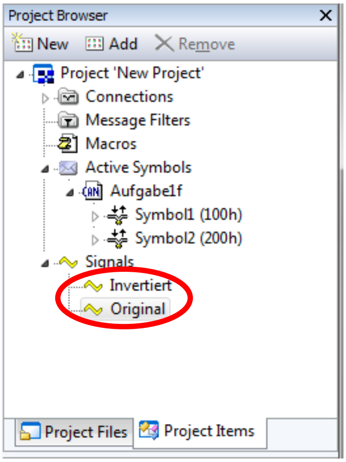
Bus	CAN-ID	DLC	Symbol	Data	Cycle ...	Count	Trigger	Creat...
1	100h	8	Symbol1	FF 00 00 00 00 00 00 00 Original = 255	Wait	250	Automatio	

The PCAN-Explorer will now show (in column Data) the transferred byte in decimal notation with the correct variable name.

4.8 Exercise 1h: Creating a Simple Instrument Panel

Using a PCAN-Explorer (Part No. IPES-005028) with installed **Instrument Panel** Add-in (Part No. IPES-005088), the CAN values may be represented and modified graphically.

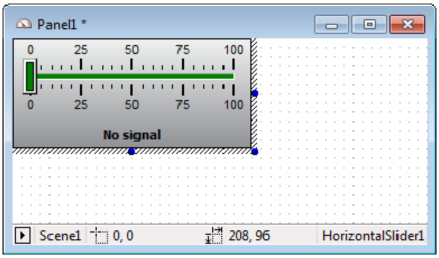
The variables from exercise 1g also appear as CAN objects in the PCAN explorer's Project Browser on the left screen edge. Here they may be used e.g. in connection with an instrument/control for simple editing resp. visualization of signals.



For example, the CAN signal "Original" can be manipulated easily using a slider. Also the use in scripts for test automation is a possible application of these objects.

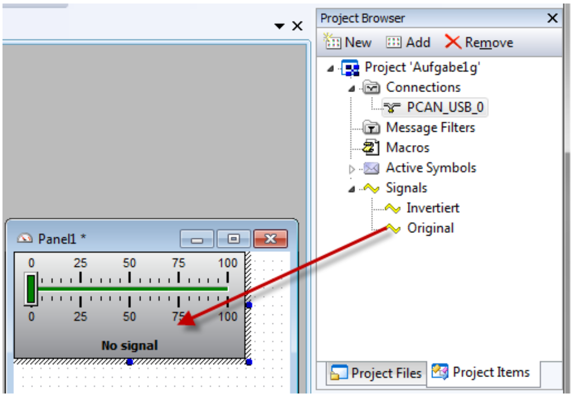
1. Select the menu item **Tools > Instruments Panel > Create horizontal Slider Control**.

This creates a new instrument panel with a horizontal slider.

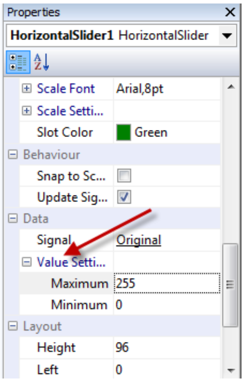


2. Assign a CAN variable to this slider control by drag and drop.

   Simply drag the variable over the instrument/control and drop it there.

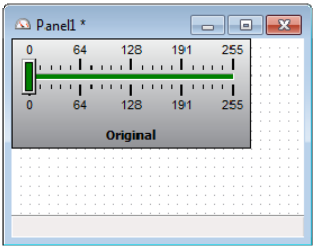


By this, the instrument/control also gets appropriate settings for value range etc. Nevertheless, settings can be edited manually with **Properties > Data > Value Setting**.


Properties

HorizontalSlider1 HorizontalSlider

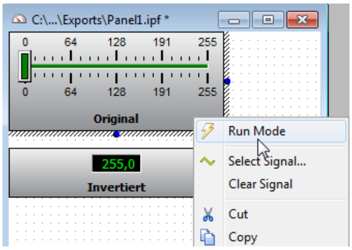
⊞ Scale Font	Arial,8pt
⊞ Scale Setti...	
Slot Color	🟩 Green
⊟ Behaviour	
Snap to Sc...	☐
Update Sig...	☑
⊟ Data	
Signal	Original
⊟ Value Setti...	
Maximum	255
Minimum	0
⊟ Layout	
Height	96
Left	0

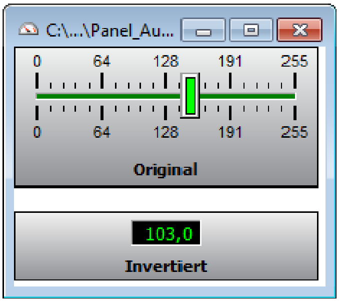


For also displaying the result signal "Invertiert" comfortably, add a display instrument and seize it with the CAN variable.

3. Select the menu item **Tools > Instruments Panel > Create Value Indicator** to get a display instrument. Place this e.g. below the slider, and finally assign the CAN variable "Invertiert" by drag & drop.

4. Right click somewhere in the instruments panel and choose **Run Mode** to make use of the instruments.





Moving the slider manipulates the first byte of the transmit message 0x100 (values between 0 and 255). In the receive message 0x200 the resulting byte value displays vice versa. The values contained in the receive message 0x100 (data byte 0) are transmitted as y = (-1)*x+255 resp. y = 255-x in message 0x200. A rising ramp ("Original", increasing x values) is converted into a falling ramp ("Invertiert", decreasing y values).

5. Save the panel as exercise 1h.ipf to your PC.

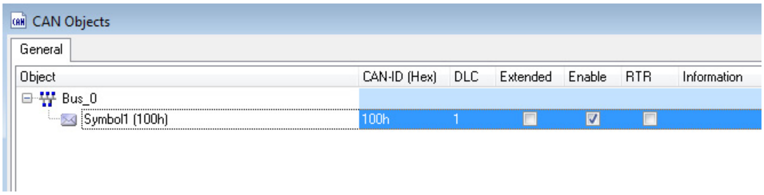
4.9 Exercise 2a: Reading Digital Input State and Transmitting this in a CAN Message

The PCAN-MIO is equipped with various hardware inputs and outputs, which can be linked to other resources.

To set the logical state of a digital input, make a connection with either GND or Vbat to a Din.

1. Create an empty configuration using the menu item **File > New**.

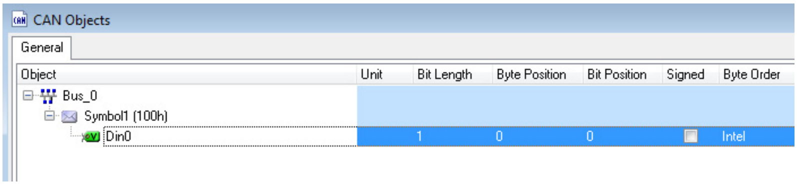
2. Add a symbol (CAN messages) to the already existing CAN bus.


CAN Objects

General

Object	CAN-ID (Hex)	DLC	Extended	Enable	RTR	Information
Bus_0						
Symbol1 (100h)	100h	1	☐	☑	☐	

3. The message gets a 1-bit variable (CAN signal for the digital input Din-0).

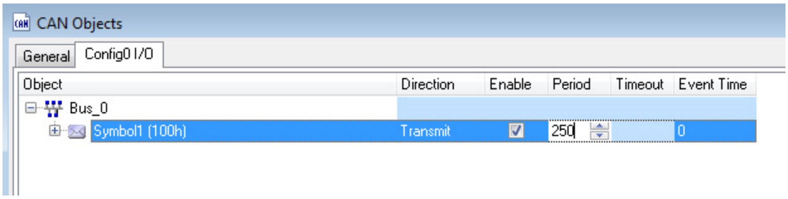

CAN Objects

General

Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order
Bus_0						
Symbol1 (100h)						
Din0		1	0	0	☐	Intel

4. Create a new configuration and import the CAN objects.

Enter the parameters for the message 0x100 as shown.



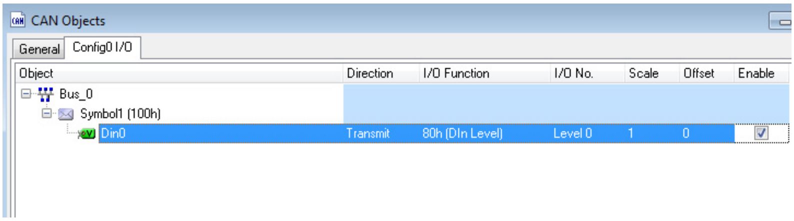
CAN Objects

General | Config0 I/O

Object	Direction	Enable	Period	Timeout	Event Time
Bus_0					
Symbol1 (100h)	Transmit	✓	250		0

- Direction: Transmit
- Enable: yes
- Period: 250

The contained signal is assigned to the state of digital input Din-0.



CAN Objects

General | Config0 I/O

Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
Symbol1 (100h)						
Din0	Transmit	80h (DIn Level)	Level 0	1	0	✓

- Direction: Transmit (the PCAN-MIO transmits the status of digital input DIN-0)
- I/O Funktion: 80h DIn Level
- I/O Number: 0
- Scale: 1
- Offset: 0
- Enable: yes

5. Save the configuration as exercise 2a to your PC.
6. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.

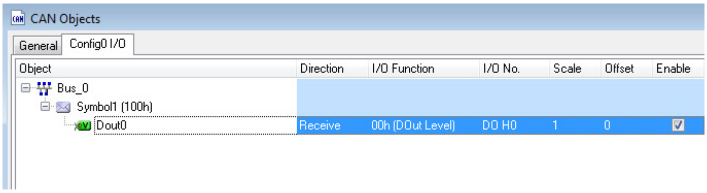
The CAN message with ID 0x100 is transmitted cyclically and the first byte reflects the logical state of digital input Din-0.

If this input is not connected, the transmitted value depends on whether the internal PullUp or PullDown resistor is activated. This is also part of your configuration (in tab **Default values for data objects**). If the input is connected to a voltage source, the transmitted value is 1 when connected to Vbat, or the transmitted value is 0 when connected to GND.

4.10 Exercise 2b: Setting Digital Output State According to a Received CAN Message

For supervising the digital output Dout-0, a voltage meter (voltmeter respectively DMM) should be connected to this output. Alternatively, a LED with a matching series resistor (e.g. about 1.8 kOhm) may be connected from here towards GND.

1. Open exercise 2a and save as exercise 2b to your PC.
2. In the **General** tab rename the CAN signal Din0 to Dout0.
3. In the **Config0 I/O** tab enter the following values for the CAN message and the CAN signal.



Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
Symbol1 (100h)						
Dout0	Receive	00h (DOut Level)	DO H0	1	0	☑

- Direction: Receive
- I/O Function: 00h DOut Level
- Enable: yes

The PCAN-MIO is equipped with configurable drivers for the digital outputs:

- a) The outputs 0...7 can drive Vbat level or an open state (High-side driver). For this purpose the I/O numbers Do H0 up to Do H7 should be chosen
- b) The outputs 0...7 can sink GND level or an open state (Low-side driver). For this purpose the I/O numbers Do L2 up to Do L7 should be chosen
- c) The outputs 0...7 can drive Vbat level or sink GND level (Push-Pull driver). For this purpose the I/O numbers Do HL0 up to Do HL7 should be chosen

4. Save the configuration as exercise 2b to your PC.
5. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.

When a message 0x100 transmitted to the PCAN-MIO with Byte0/Bit0 = 1, the digital output Dout-0 will be activated.

- In case a) obtains a Vbat level
- In case b) a GND level
- In case c) a Vbat level

When a message 0x100 transmitted to the PCAN-MIO with Byte0/Bit0 = 0, the digital output Dout-0 will be deactivated.

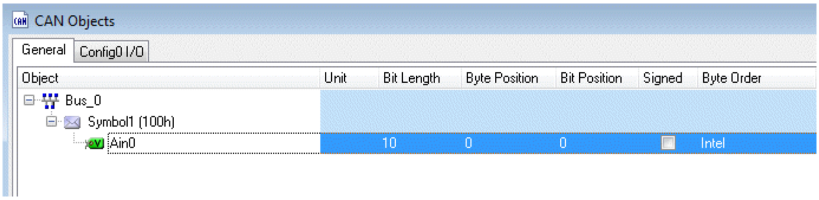
- In case a) obtains an open state
- In case b) an open state
- In case c) a GND level

4.11 Exercise 2c: Reading Analog Input State and Transmitting this in a CAN Message

For comprehensible results in the following example, please connect a voltage source to the analog input Ain-0. For this example a 10K Ohm potentiometer can be connected between GND and Vbat, so that a variable voltage is available at the pickup. The resolution of the A/D converter is 10 bit. Accordingly the CAN signal must be two bytes wide (when using Scale = 1).

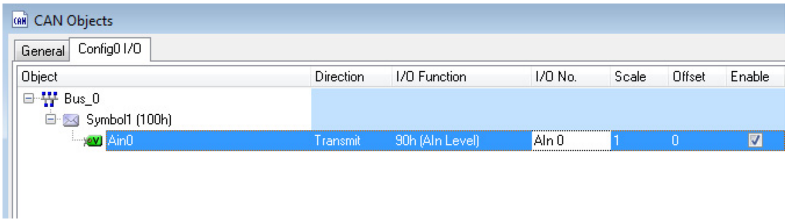
1. Open the configuration exercise 2a and save as exercise 2c to your PC.

2. On the **General** tab change the length of the CAN message to 2 byte, rename the CAN signal to Ain0 and increase the bit length to 10.



Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order
Bus_0						
Symbol1 (100h)						
Ain0		10	0	0	☐	Intel

3. In the **Config0 I/O** tab assign the following values to the CAN signal.



Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
Symbol1 (100h)						
Ain0	Transmit	90h (AIn Level)	AIn 0	1	0	☑

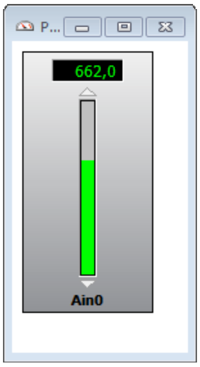
- I/O Function: 90h AIn Level
- I/O Number: AIn 0

- Enable: yes

4. Save the configuration as exercise 2c to your PC.
5. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.

The message 0x100 is transmitted cyclically to CAN-1 (every 250 ms). When a variable voltage is applied to the input Ain-0, the contents of the message is changed in bytes 0 and 1 (10-bit value), whereas byte 0/bit 0 is the LSB, byte 1/bit 1 is the MSB (Intel format).

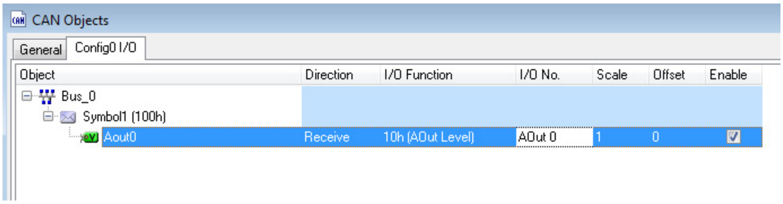
With the help of the PCAN-Explorer Add-in **Instrument Panel** (Part No IPES-005088), the values can be represented graphically, e.g. using a bar graph control (see exercise 1h).



4.12 Exercise 2d: Setting of Analog Outputs According to a Received CAN Message

For supervising the analog output Aout-0, a voltage meter (voltmeter respectively DMM) should be connected to this output.

1. Open exercise 2c and save as exercise 2d to your PC.
2. In the **General** tab rename the CAN signal to Aout0.
3. In the **Config0 I/O** tab change the CAN message from transmit to receive, and assign the CAN signal to Aout-0.

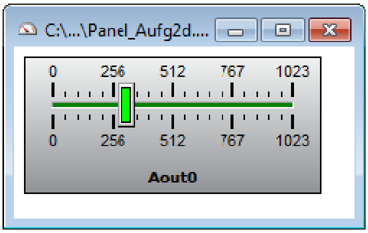


Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
Symbol1 (100h)						
Aout0	Receive	10h (AOut Level)	AOut 0	1	0	✓

4. Save the configuration as exercise 2d to your PC.
5. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.

When a message 0x100 is sent to the PCAN-MIO, with a 10-bit value encoded in the correct position (Intel format), you can measure an equivalent voltage 0 - 10 V at the analog output Aout-0.

By means of the PCAN-Explorer Add-in **Instrument Panel** (Part No IPES-005088), the values can be set using a slider control (see exercise 1h).

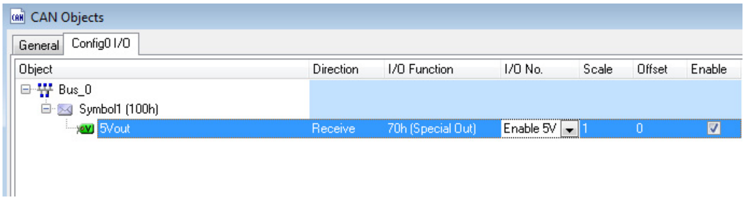


4.13 Exercise 2e: Switching the 5 V Output

For the supply of sensors the PCAN-MIO provides a switchable auxiliary voltage of 5 V which can drive loads up to 200 mA.

For supervising the state of the auxiliary voltage, a voltage meter (voltmeter respectively DMM) should be connected to this output. Alternatively, a LED with a matching series resistor (e.g. about 390 Ohm) may be connected from here towards GND.

1. Open exercise 2b and save as exercise 2e to your PC.
2. In the **General** tab rename the CAN signal to 5Vout.
3. In the **Config0 I/O** tab assign the following values to the CAN signal.



Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
Symbol1 (100h)						
5Vout	Receive	70h (Special Out)	Enable 5V	1	0	✓

I/O Function: 70h Special Out

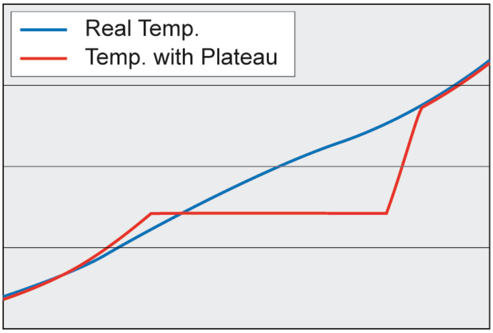
- I/O Number: Supply 5 V

4. Save the configuration as exercise 2e to your PC.
5. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.

When a message 0x100 transmitted to the PCAN-MIO carries byte0/bit0 = 1, the 5 V output will be activated, if byte0/bit0 = 0, the 5 V output will be deactivated.

4.14 Exercise 3a: Manipulating a CAN Signal using Function Block Characteristic Curve

The temperature of a motor's cooling fluid shall be represented as constantly 90 °C, even if the real coolant temperature varies between 80 °C and 105 °C. Such plateau is often implemented for smoothing of analog meters. In case the real temperature leaves the specified range (e.g. motor defect), it shall be displayed directly.

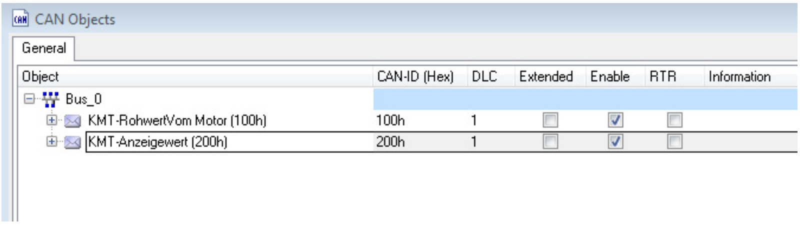


➡ For implementing such behavior, the function block "characteristic curve" is suitable, converting the incoming measured temperature from the engine (raw value) into an artificial curve containing a plateau for the analog instrument.

That smoothed value is then transmitted in a separate CAN message.

1. Create an empty configuration file using the menu item **File > New**.

2. Define two CAN messages with an 8-bit CAN signal each.

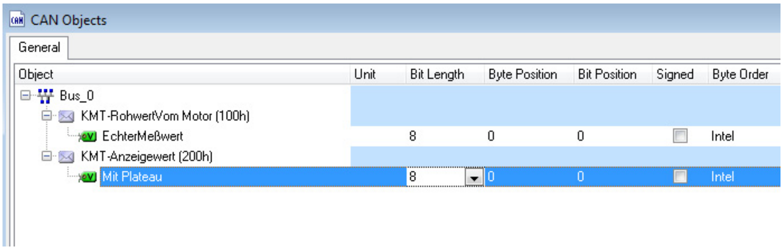


CAN Objects

General

Object	CAN-ID (Hex)	DLC	Extended	Enable	RTR	Information
Bus_0						
KMT-RohwertVom Motor (100h)	100h	1	☐	✓	☐	
KMT-Anzeigewert (200h)	200h	1	☐	✓	☐	

- CAN-ID: 0x100 for the incoming raw value; 0x200 for the outgoing display value (with plateau)
- DLC: 1 (both CAN messages are 1 byte of length)
- Enable: yes



CAN Objects

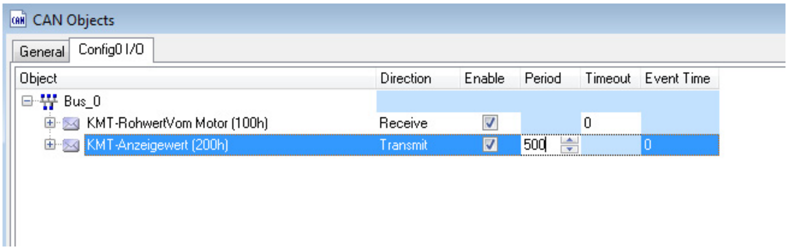
General

Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order
Bus_0						
KMT-RohwertVom Motor (100h)						
EchterMeßwert		8	0	0	☐	Intel
KMT-Anzeigewert (200h)						
Mit Plateau		8	0	0	☐	Intel

- Bit Length: 8
- Signed: no, always positive

3. Create a new configuration within the configuration file by selecting the menu item **Edit > New Configuration**.

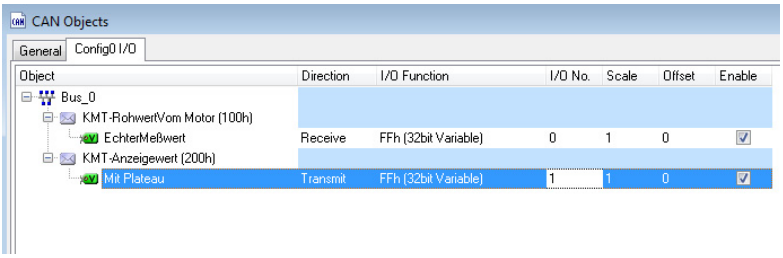
4. Import the globally defined CAN busses with all CAN messages and CAN signals. Open the context menu (right click) and select **Add defined Bus**.


CAN Objects

General | Config0 I/O

Object	Direction	Enable	Period	Timeout	Event Time
Bus_0					
KMT-RohwertVom Motor (100h)	Receive	✓		0	
KMT-Anzeigewert (200h)	Transmit	✓	500		0

- Direction: 0x100 is received, 0x200 is transmitted.
- Enable: yes
- Period: 500 ms (cycle time for the transmit message carrying the converted coolant temperature signal).


CAN Objects

General | Config0 I/O

Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
KMT-RohwertVom Motor (100h)						
EchterMeßwert	Receive	FFh (32bit Variable)	0	1	0	✓
KMT-Anzeigewert (200h)						
Mit Plateau	Transmit	FFh (32bit Variable)	1	1	0	✓

- I/O Function, I/O Number: The signal RealTemperature from receive message 0x100 is transferred into variable #0, the modified result WithPlateau is transferred into transmit message 0x200.

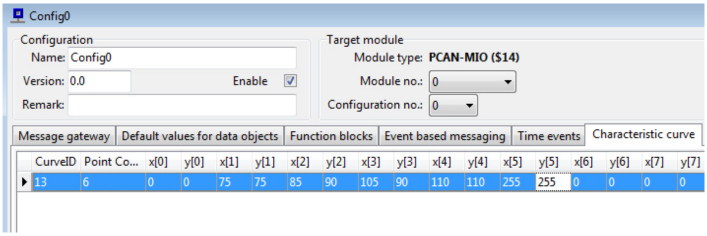
The assignment of the raw value (variable #0 to the smoothed result variable #1) is defined in a characteristic curve (a list of X/Y pairs). A function block “characteristic curve” is needed to manage the conversion using this list. The following table defines all points needed to create the mentioned plateau (X values other than the listed ones are linearly interpolated):

	X	Y
Curve point 0	X=0	Y=0
Curve point 1	X=75	Y=75
Curve point 2	X=85	Y=90
Curve point 3	X=105	Y=90
Curve point 4	X=110	Y=110
Curve point 5	X=255	Y=255

This conversion table is now keyed as a characteristic curve.

5. Double click on the Icon Config0 at the left window edge. Select the **Characteristic curve** tab. Open the context menu (right click) and select **Add Record**.

   A new table row appears, representing a characteristic curve. This curve must be filled with the mentioned values.



Config0

Configuration
Name: Config0
Version: 0.0   Enable ☑
Remark:

Target module
Module type: **PCAN-MIO ($14)**
Module no.: 0
Configuration no.: 0

CurveID	Point Co...	x[0]	y[0]	x[1]	y[1]	x[2]	y[2]	x[3]	y[3]	x[4]	y[4]	x[5]	y[5]	x[6]	y[6]	x[7]	y[7]
13	6	0	0	75	75	85	90	105	90	110	110	255	255	0	0	0	0

- CurveID: 13 (an arbitrary number)
- Point Count: 6 (number of X/Y points on the curve)
- Pairs of values 0... 5: the characteristic curve (values taken from the table above). Further entries are not used and contain 0

Important Note: X value must be entered in strictly ascending order!

Finally, you must manage the assignment of the incoming raw value to the characteristic curve’s X axis and also of the resulting Y value (= result) to variable #1, which is subsequently transmitted onto CAN. For this, a special function block Characteristic curve is needed, which handles that conversion.

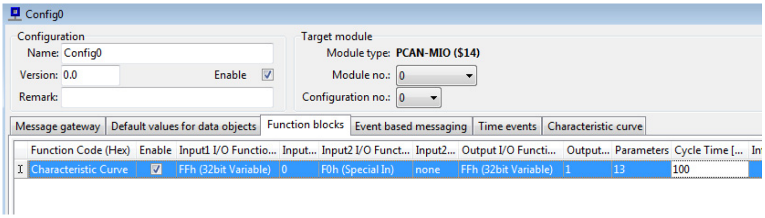
6. Click to the **Function blocks** tab to create a new function block. Open the context menu (right click) and select **Add Record**.

   A new line appears, representing a function block. This line must be supplied with values.

   Basically, each function block has two inputs (operands) and one output (result), each of these consisting of an I/O type and an I/O number. Additional there is a main switch (enable) and a cycle time (How often is this block re-calculated, in ms).

Note: In the function block Characteristic curve the second input is always unused.

7. Enter the following parameters into the function block.



Config0

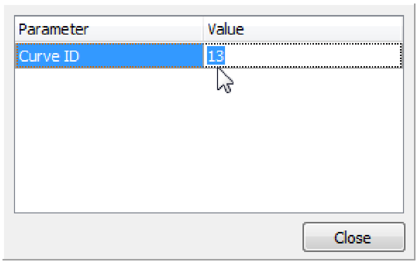
Configuration
Name: Config0
Version: 0.0 Enable ✓
Remark:

Target module
Module type: **PCAN-MIO ($14)**
Module no.: 0
Configuration no.: 0

Function Code (Hex)	Enable	Input1 I/O Functio...	Input...	Input2 I/O Funct...	Input2...	Output I/O Functi...	Output...	Parameters	Cycle Time [...	In
Characteristic Curve	✓	FFh (32bit Variable)	0	F0h (Special In)	none	FFh (32bit Variable)	1	13	100	

- Function Code: Characteristic Curve (a special function block for this purpose)
- Enable: yes, this function block shall be active
- Input1 I/O Function: FFh 32bit Variable
- Input1 I/O Number: 0 (X value comes from 32-bit variable #0)
- Input2 I/O Function: F0h Special In (not used)
- Input2 I/O Number: none (not used)

- Output I/O Function: FFh 32bit Variable
- Output I/O Number: 1 (Y result is written to 32-bit variable #1)
- Parameters: 13 (number of the already defined characteristic curve). When clicking the field, the following dialog window appears:



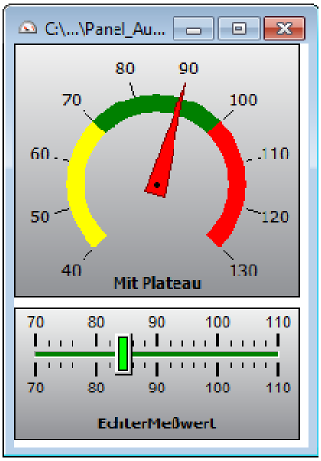
Parameter	Value
Curve ID	13

- Enter the number of the already defined characteristic curve in column Value. Confirm with **Close**
- Cycle time: 100 (calculation of the raw value takes place every 100 ms)

8. Save the configuration as exercise 3a to your PC.
9. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.

A continuously rising input value (from message 0x100) is superimposed with a plateau and then forwarded (to message 0x200).

With the help of the PCAN-Explorer Add-in **Instrument Panel** (Part No IPES-005088), the values can be represented graphically (see exercise 1h).

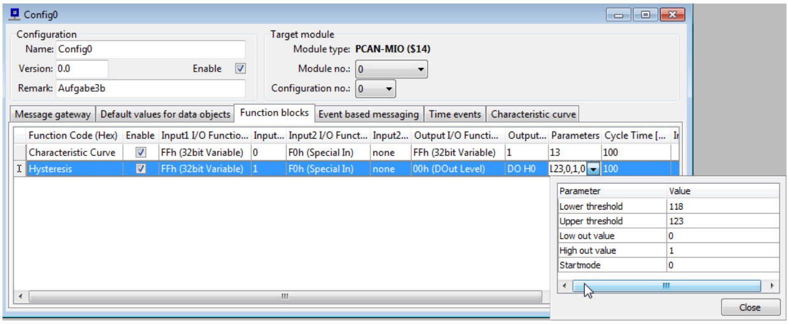


4.15 Exercise 3b: Threshold with Hysteresis

Thinking of an extension for exercise 3a, a warning lamp comes in mind which shall be lit when temperature exceeds 123°C and goes dark when temperature falls below 118°C. For such behavior a function block Hysteresis is suitable.

➡ The warning lamp is connected to Dout-0. Maybe use a LED with a matching series resistor (e.g. 1.5 kOhm) towards GND.

1. Open exercise 3a and save as exercise 3b to your PC.
2. To create another function block, click on the **Function blocks** tab. Open the context menu (right click) and select **Add Record**.
3. Enter the following parameters into the function block.



Configuration
Name: Config0
Version: 0.0
Enable ✓
Remark: Aufgabe3b

Target module
Module type: PCAN-MIO ($14)
Module no.: 0
Configuration no.: 0

Message gateway | Default values for data objects | Function blocks | Event based messaging | Time events | Characteristic curve

Function Code (Hex)	Enable	Input1 I/O Functio...	Input...	Input2 I/O Funct...	Input2...	Output I/O Functi...	Output...	Parameters	Cycle Time [...
Characteristic Curve	✓	FFh (32bit Variable)	0	F0h (Special In)	none	FFh (32bit Variable)	1	13	100
Hysteresis	✓	FFh (32bit Variable)	1	F0h (Special In)	none	00h (DOut Level)	DO H0	123,0,1,0	100

Parameter	Value
Lower threshold	118
Upper threshold	123
Low out value	0
High out value	1
Startmode	0

- Function Code: Hysteresis (the threshold switch with hysteresis function)
- Enable: yes, this function block shall be active
- Input1 I/O Function: FFh 32bit Variable
- Input1 I/O Number: 1 (you can also use 0, because in this temperature range the curves run identically)
- Input2 I/O Function: F0h Special In (not used)
- Input2 I/O Number: none (not used)
- Output I/O Function: Dout-0 (a hardware output with High-side driver)
- Parameter: Lower Threshold: 118 °C (switch off point); Upper Threshold: 123 °C (switch on point); below Lower Threshold (LowOutValue) the output Dout-0 will be inactive (0); above Upper Threshold (HighOutValue) the output Dout-0 will be active (1)
- Cycle time: 100 (calculation of the raw value takes place every 100 ms)

4. Save the configuration as exercise 3b to your PC.
5. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.

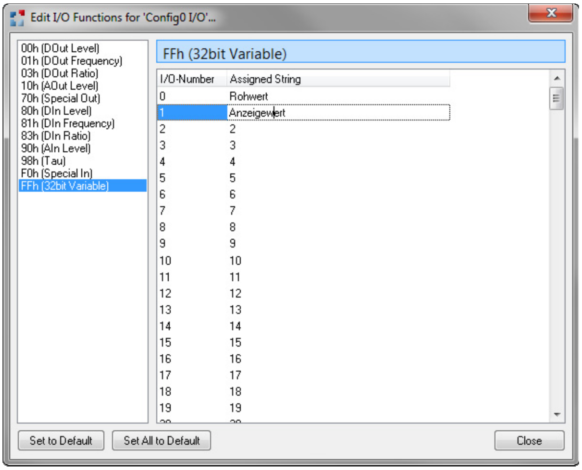
When CAN message 0x100 transmits a temperature > 123 °C to the PCAN-MIO, the output Dout-0 changes to 1 (LED on).

When CAN message 0x100 transmits a temperature < 118 °C to the PCAN-MIO, the output Dout-0 changes to 0 (LED off).

Note: A readable name can be assigned to each 32-bit variable (and other resources). To do this, follow these steps:

1. Starting from the **Function blocks** tab, select the menu item **Edit > I/O-Funktion**.

   The following dialog window appears:



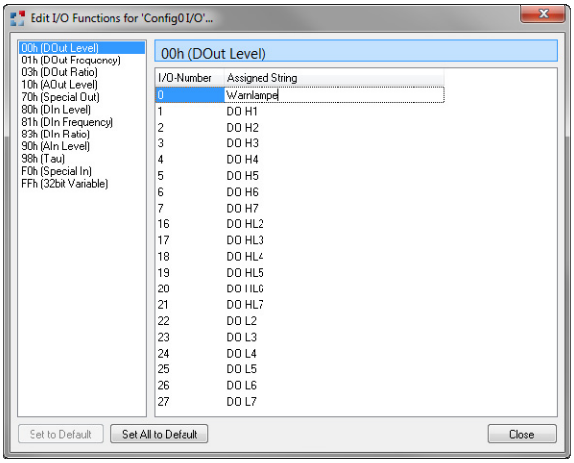
Edit I/O Functions for 'Config0 I/O'...

- 00h (DOut Level)
- 01h (DOut Frequency)
- 03h (DOut Ratio)
- 10h (AOut Level)
- 70h (Special Out)
- 80h (DIn Level)
- 81h (DIn Frequency)
- 83h (DIn Ratio)
- 90h (AIn Level)
- 98h (Tau)
- F0h (Special In)
- FFh (32bit Variable)

FFh (32bit Variable)

I/O-Number	Assigned String
0	Rohwert
1	Anzeigewert
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19

   Rename the 32-bit variable #0 into “Rohwert” and the variable #1 into “Anzeigewert”.



I/O-Number	Assigned String
0	Warnlampe
1	DO H1
2	DO H2
3	DO H3
4	DO H4
5	DO H5
6	DO H6
7	DO H7
16	DO HL2
17	DO HL3
18	DO HL4
19	DO HL5
20	DO HL6
21	DO HL7
22	DO L2
23	DO L3
24	DO L4
25	DO L5
26	DO L6
27	DO L7

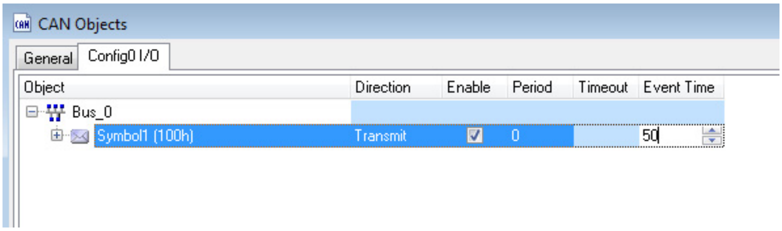
Rename the variable DOut Level #0 into “Warnlampe”.

From here you can work with these names in the configuration, which will reduce errors caused by confusion of a variable number significantly.

4.16 Exercise 3c: Transmission only on Signal Changes

1. Open exercise 2c and save as exercise 3c to your PC.
2. In the **Config0 I/O** tab switch off the periodic transmission of the CAN message 0x100 (change cycle time from 250 ms to 0 ms).

Thus, the CAN message is not longer transmitted cyclically.

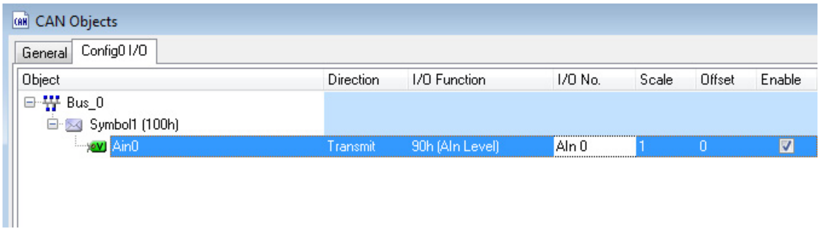

CAN Objects

General | Config0 I/O

Object	Direction	Enable	Period	Timeout	Event Time
Bus_0					
Symbol1 (100h)	Transmit	✓	0		50

- Period: 0 ms
- Event Time: 50 ms (if the message is triggered constantly, this is the minimum transmission distance. So the bus load will be limited.)

The CAN signal in this message remains unchanged.

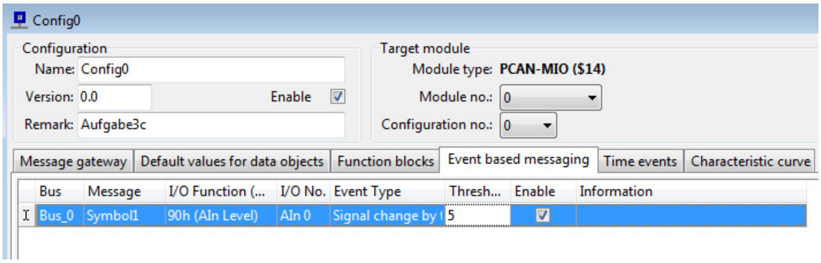

CAN Objects

General | Config0 I/O

Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
Symbol1 (100h)						
Ain0	Transmit	90h (AIn Level)	AIn 0	1	0	✓

3. The condition which triggers the transmission, is defined in the configuration window.
4. Double click on the Icon Config0 at the left window edge. Select the **Event based messaging** tab. Open the context menu (right click) and select **Add Record**.

   If the trigger hits, the defined message 0x100 should be transmitted onto CAN-0. Trigger should be an analog level, which is applied to Ain 0. If this level changes by an amount (parameter threshold), then ID 0x100 will be transmitted.



Config0

Configuration

Name: Config0

Version: 0.0 Enable ☑

Remark: Aufgabe3c

Target module

Module type: **PCAN-MIO ($14)**

Module no.: 0

Configuration no.: 0

Message gateway | Default values for data objects | Function blocks | Event based messaging | Time events | Characteristic curve

	Bus	Message	I/O Function (...	I/O No.	Event Type	Thresh...	Enable	Information
I	Bus_0	Symbol1	90h (AIn Level)	AIn 0	Signal change by t	5	☑	

- Bus: Bus_0
- Message: 0x100
- I/O Function: 90h Ain Level
- I/O No: AIn-0
- Event Type: Signal change by threshold
- Threshold: 5
- Enable: yes

5. Save the configuration as exercise 3c to your PC.
6. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.

If a potentiometer (as described in exercise 2c) is connected to Ain-0 and turned, then the PCAN-MIO will transmit a message 0x100 containing the analog value.

The transmit frequency can be varied with the parameter threshold. This threshold however should be greater than 1 because the LSB of the A/D converter may flicker, if the pot value is on a boundary between two digits.

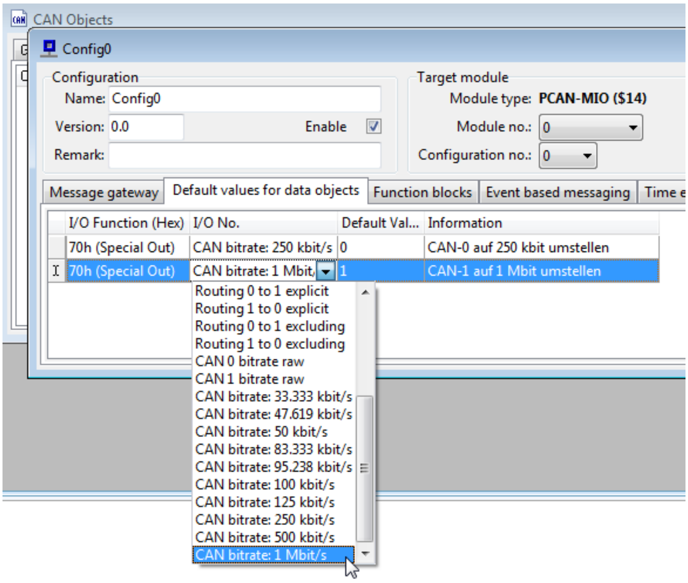
The trigger signal (here: Ain-0) -must- be included in the transmit message, otherwise the message is not transmitted. If it shall be invisible, its length may be set to 0.

4.17 Exercise 4a: Setting the CAN Bit Rate (Default Value)

Depending on the installed CAN transceivers, PCAN-MIO sets the following bit rates by default:

Interface-Type	CAN-Transceiver	Default bit rate	WakeUp-capability
HS	TJA-1041 (default)	500 kbit/s	yes
HS opto	TJA-1040	500 kbit/s	no
HS	TJA-1040	500 kbit/s	no
LS-DW	TJA-1054	125 kbit/s	no
LS-SW	TH-8056	33,3 kbit/s	yes

When starting the PCAN-MIO these values can be overwritten by appropriate entries in the **Default values for data objects** tab of your configuration.



Name: Config0
Version: 0.0
Enable ✓
Remark:
Module type: **PCAN-MIO ($14)**
Module no.: 0
Configuration no.: 0

I/O Function (Hex)	I/O No.	Default Val...	Information
70h (Special Out)	CAN bitrate: 250 kbit/s	0	CAN-0 auf 250 kbit umstellen
70h (Special Out)	CAN bitrate: 1 Mbit/	1	CAN-1 auf 1 Mbit umstellen

Routing 0 to 1 explicit
Routing 1 to 0 explicit
Routing 0 to 1 excluding
Routing 1 to 0 excluding
CAN 0 bitrate raw
CAN 1 bitrate raw
CAN bitrate: 33.333 kbit/s
CAN bitrate: 47.619 kbit/s
CAN bitrate: 50 kbit/s
CAN bitrate: 83.333 kbit/s
CAN bitrate: 95.238 kbit/s
CAN bitrate: 100 kbit/s
CAN bitrate: 125 kbit/s
CAN bitrate: 250 kbit/s
CAN bitrate: 500 kbit/s
CAN bitrate: 1 Mbit/s

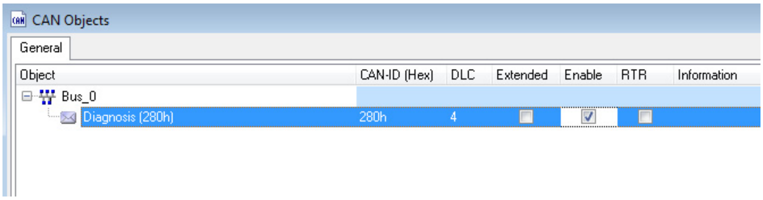
Note: As PPCAN-Editor (running on your PC) doesn't know about the transceiver types installed in your PCAN MIO, it will offer all usual bit rates. Please take care that the equipped transceivers support the settings, e.g. LS-SW does not support bit rates beyond 83.3 kbit/s, whereas TJA-1040 does not support bit rates below 40 kbit/s.

4.18 Exercise 5a: Reading the Module ID (Diagnostics)

The module ID is a 4-bit value, which is set to 0 by default, but can be changed inside of the PCAN-MIO by means of a rotary switch. The ID has several functions. For example, it selects one from several configurations contained in a PPCAN project file according to the switch position. When experiencing strange behavior of your currently uploaded configuration, one of the first steps in trouble-shooting is determination of the module ID. It sometimes may happen that a configuration is edited again and again without success, since each time a different one is executed by the PCAN-MIO.

As an example, we now transmit a CAN message "Diagnosis" with ID 0x280, length 8 bytes on bus CAN-0. This message contains the 4-bit signal ModuleID, which displays the current position of the module ID switch.

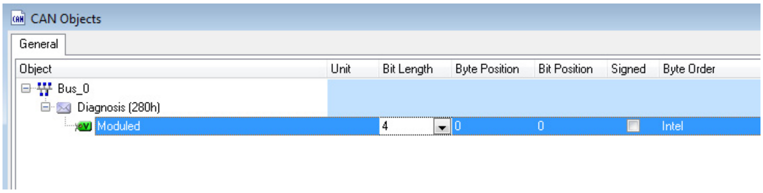
1. Create a new configuration and define a Transmit message Diagnosis, that contains a CAN signal with the module ID.



Object	CAN-ID (Hex)	DLC	Extended	Enable	RTR	Information
Bus_0						
Diagnosis (280h)	280h	4	☐	✓	☐	

- CAN-ID: 0x280
- DLC: 4
- Enable: yes

2. Create a new CAN signal ModuleID.



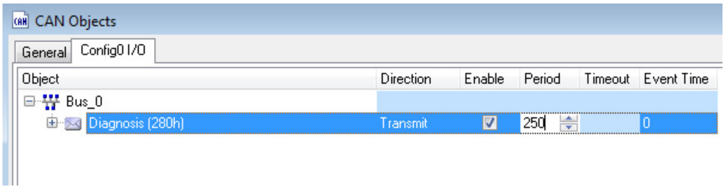
CAN Objects

General

Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order
Bus_0						
Diagnosis (280h)						
Moduled		4	0	0	☐	Intel

- Bit Length: 4

3. Importing of the message and CAN signal into the configuration and entering the parameter.



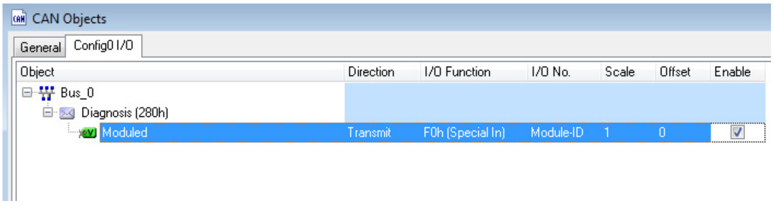
CAN Objects

General | Config0 I/O

Object	Direction	Enable	Period	Timeout	Event Time
Bus_0					
Diagnosis (280h)	Transmit	☑	250		0

- Direction: Transmit
- Enable: yes
- Period: 250 ms

4. Supplying the signal with internal variable ModuleID of the PCAN-MIO.



CAN Objects

General | Config0 I/O

Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
Diagnosis (280h)						
Moduled	Transmit	F0h (Special In)	Module-ID	1	0	☑

- I/O Function: F0h Special In
- I/O Number: Module-ID
- Enable: yes

5. Save the configuration as exercise 5a to your PC.
6. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.

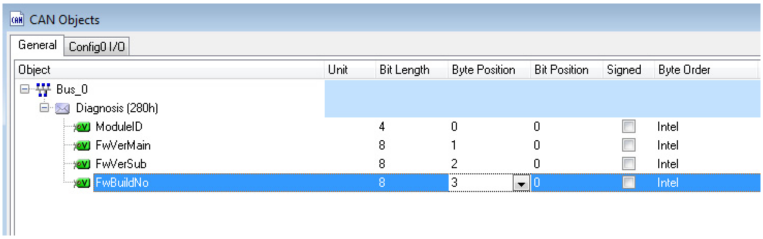
A cyclic message Diagnosis (with ID 0x280) will be transmitted every 250 ms, which carries the signal ModuleID.

Note: Changes of the module ID (e.g. by turning the rotary switch) is shown immediately, but it becomes effective only after a restart of the module (e.g. Power Off/On).

4.19 Exercise 5b: Reading the Firmware Version (Diagnostics)

The firmware version of PCAN-MIO can be read for diagnostic purposes.

1. Open exercise 5a and save as exercise 5b to your PC.
2. Add another three CAN signals (FirmwareVersionMain, FirmwareVersionSub, FirmwareBuildNo) and enter the parameters as follows.

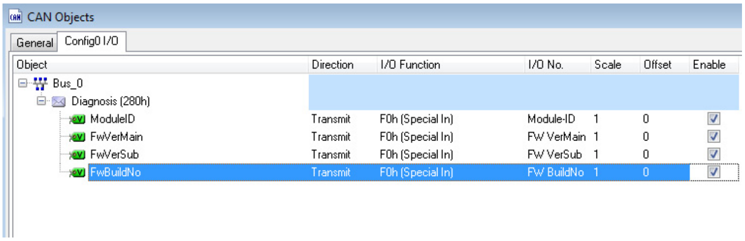


Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order
Bus_0						
Diagnosis (280h)						
ModuleID		4	0	0	☐	Intel
FwVerMain		8	1	0	☐	Intel
FwVerSub		8	2	0	☐	Intel
FwBuildNo		8	3	0	☐	Intel

- In the second message byte (byte 1), the 8-bit value FwVersionMain shall be transmitted
- In the third message byte (byte 2), the 8-bit value FwVersionSub shall be transmitted
- In the fourth message byte (byte 3), the 8-bit value FwBuildNumber shall be transmitted

3. Import the three CAN signals into the configuration as follows: Open the context menu (right click) and select **Add defined Variable.**

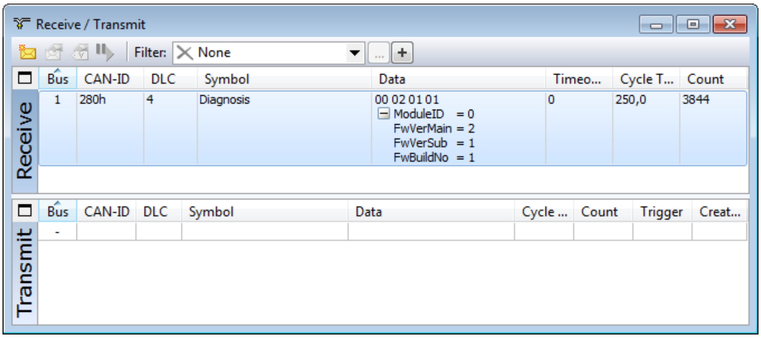
   Supply the signals with values (assign virtual module resources).



Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
Diagnosis (280h)						
ModuleID	Transmit	F0h (Special In)	Module-ID	1	0	✓
FwVerMain	Transmit	F0h (Special In)	FW VerMain	1	0	✓
FwVerSub	Transmit	F0h (Special In)	FW VerSub	1	0	✓
FwBuildNo	Transmit	F0h (Special In)	FW BuildNo	1	0	✓

4. Save the configuration as exercise 5b to your PC.
5. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.

The cyclic message Diagnosis with ID 0x280 now also includes information on the firmware version.



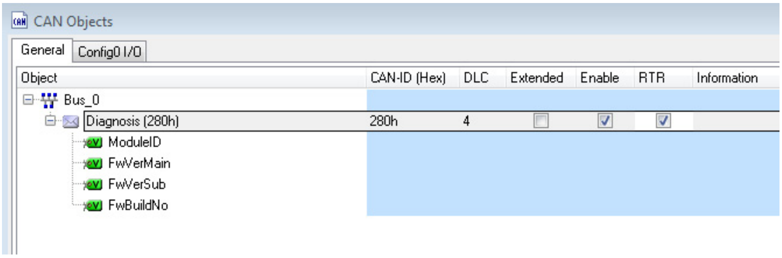
Bus	CAN-ID	DLC	Symbol	Data	Timeo...	Cycle T...	Count
1	280h	4	Diagnosis	00 02 01 01 ModuleID = 0 FwVerMain = 2 FwVerSub = 1 FwBuildNo = 1	0	250,0	3844

Bus	CAN-ID	DLC	Symbol	Data	Cycle ...	Count	Trigger	Creat...
-								

4.20 Exercise 6: CAN Messages on Request

Based on exercise 5b the internal variable Diagnosis shall be transmitted on remote request (RTR = Remote Transmission Request).

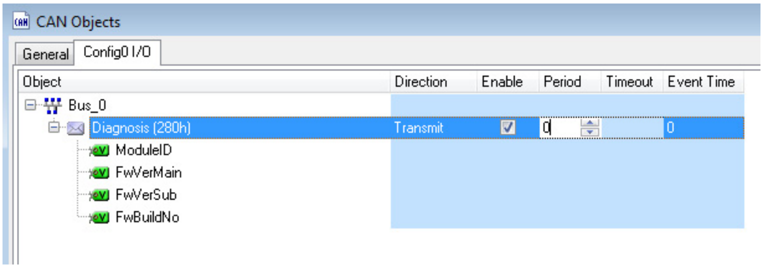
1. Open exercise 5a and save as exercise 5b to your PC.
2. Modify the CAN message to a transmission request by setting the RTR check for that symbol.



Object	CAN-ID (Hex)	DLC	Extended	Enable	RTR	Information
Bus_0						
Diagnosis (280h)	280h	4	☐	✓	✓	
ModuleID						
FwVerMain						
FwVerSub						
FwBuildNo						

RTR: yes (activate Remote Transmission Request)

3. Set period value 0, thus switching off cyclic transmission.



CAN Objects

Object	Direction	Enable	Period	Timeout	Event Time
Bus_0					
Diagnosis (280h)	Transmit	✓	0		0
ModuleID					
FwVerMain					
FwVerSub					
FwBuildNo					

- Period: 0 (no more cyclic transmission)

4. Save the configuration as exercise 5b to your PC.
5. Transmit (upload) the configuration to the PCAN-MIO via CAN bus.

CAN message Diagnosis with ID 0x280 is transmitted only if a request with ID 0x280 and length = 0 was received previously.

4.21 Exercise 7: Sleep Mode (Energy Saving)

The PCAN-MIO can switch off itself (configurable) e.g. for saving battery power. This so called sleep mode occurs:

- if (with installed wake-up-capable CAN transceivers) no further CAN message is received that could awaken the transceiver
- and the hardware input WakeUp does -not- detect a Vbat level (e.g. clamp 15)
- and the internal variable SELFHOLD is set to 0

After waking up either by reception of an adequate CAN message (with installed wake-up-capable CAN transceivers) or with Vbat level at the hardware input WakeUp, the internal variable SELFHOLD is set to 1 and keeps the module alive from here.

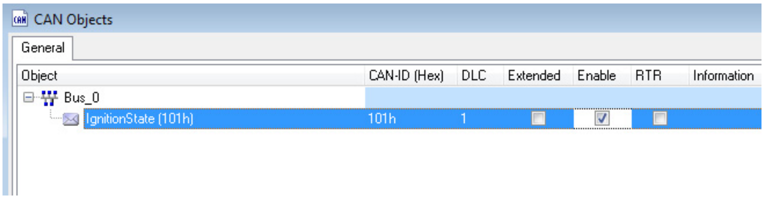
Prerequisite for the awakening is, that the appropriate CAN bus is equipped with a wake-up-capable transceiver, e.g. LS-SW.

Interface-Type	CAN-Transceiver	Default bit rate	WakeUp-capability
HS	TJA-1041 (default)	500 kbit/s	yes
HS opto	TJA-1040	500 kbit/s	no
HS	TJA-1040	500 kbit/s	no
LS-DW	TJA-1054	125 kbit/s	yes
LS-SW	TH-8056	33,3 kbit/s	yes

In case a CAN message is no longer received and a timeout value is specified, all contained CAN signals reset to a default value. This default signal value is 0, unless specified otherwise in the tab Default values for data objects.

On most vehicle CAN busses, there is usually a cyclic signal that reflects the position of the ignition key (clamp 15 and power mode). By receiving a WakeUp frame from the vehicle (e.g. high voltage levels) the PCAN-MIO wakes up and then reacts to the receipt of the CAN signal IgnitionOn. As long as the CAN signal IgnitionOn = 1, the module should stay awake, when IgnitionOn = 0 the PCAN-MIO should go asleep (delayed by 6 seconds), independent from other transmitted messages. The signal IgnitionOn = 1 triggers/retriggers a Monoflop, that expires 6 seconds after the last trigger and sets subsequentially SELFHOLD = 0.

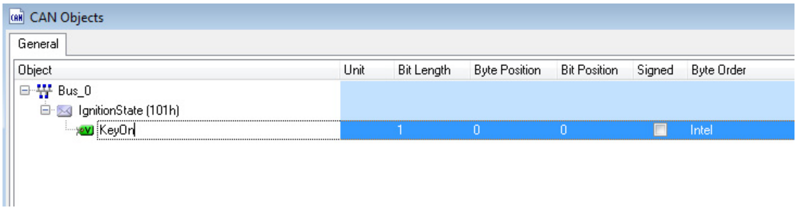
1. Create a new configuration and define a transmit message that contains a CAN signal ignition key on.



Object	CAN-ID (Hex)	DLC	Extended	Enable	RTR	Information
Bus_0						
IgnitionState (101h)	101h	1	☐	✓	☐	

- Message name: IgnitionState
- CAN-ID: 0x101
- DLC: 1 byte

- Enable: yes

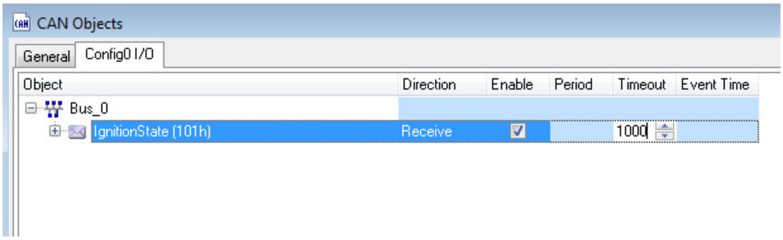

CAN Objects

General

Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order
Bus_0						
IgnitionState (101h)						
KeyOn		1	0	0	☐	Intel

- Signal name: KeyOn
- Bit Length: 1 bit
- Byte/Bit Position: 0, 0 (in the first byte right)
- Signed: no, only 1 bit

2. Create a new configuration within the configuration file. Import the globally defined CAN busses with all CAN messages and CAN signals.

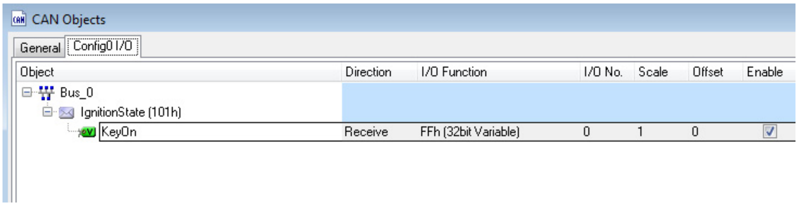


CAN Objects

General | Config0 I/O

Object	Direction	Enable	Period	Timeout	Event Time
Bus_0					
IgnitionState (101h)	Receive	✓		1000	

- Direction: Receive
- TimeOut: 1000 ms (afterwards the signal goes to the default value 0)



CAN Objects

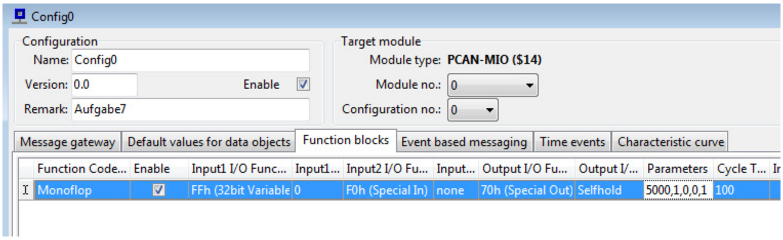
General | Config0 I/O

Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
IgnitionState (101h)						
KeyOn	Receive	FFh (32bit Variable)	0	1	0	✓

- I/O Function: FFh 32bit Variable
- I/O Number: 0
- Scale: 1
- Offset: 0
- Enable: yes

3. Double click on the Icon Config0 at the left window edge, and click to the **Function blocks** tab to create a new function block.

4. Create a function block Monoflop, which deactivates SELFHOLD (5 seconds after the variable 0 has changed to value 0).



Config0

Configuration
Name: Config0
Version: 0.0 Enable ☑
Remark: Aufgabe7

Target module
Module type: **PCAN-MIO ($14)**
Module no.: 0
Configuration no.: 0

Message gateway | Default values for data objects | Function blocks | Event based messaging | Time events | Characteristic curve

Function Code...	Enable	Input1 I/O Func...	Input1...	Input2 I/O Fu...	Input...	Output I/O Fu...	Output I/...	Parameters	Cycle T...	Ir
Monoflop	☑	FFh (32bit Variable	0	F0h (Special In)	none	70h (Special Out)	Selfhold	5000,1,0,0,1	100	

- Function Code: Monoflop
- Enable: yes
- Input1 I/O-Function: FFh 32bit Variable
- Input1 I/O-Number: 0
- Input2 I/O-Function: not used
- Input2 I/O-Number: not used
- Output I/O-Function: 70h SpecialOut (a virtual module resource)
- Output I/O-Number: Selfhold
- Parameter: 5 s, everything else can be looked up in the reference
- Cycle Time: 100 ms

5. Save the configuration as exercise 5b to your PC.
6. Transmit (Upload) the configuration to the PCAN-MIO via CAN bus.

5 seconds after the signal changes from 1 to 0 (or 6 seconds after the message fails), SELFHOLD turns off the power supply and the module goes into sleep mode. With a wake-up capable transceiver equipped, the PCAN-MIO can be reactivated by a CAN message. Alternatively, the WakeUp line can be supplied with high-level.

4.22 Exercise 8a: Transmitting a Multiplexer Message Automatically

Information: The example transmits a CAN message with varying variables that are controlled by a multiplexer (intermediate addressing).

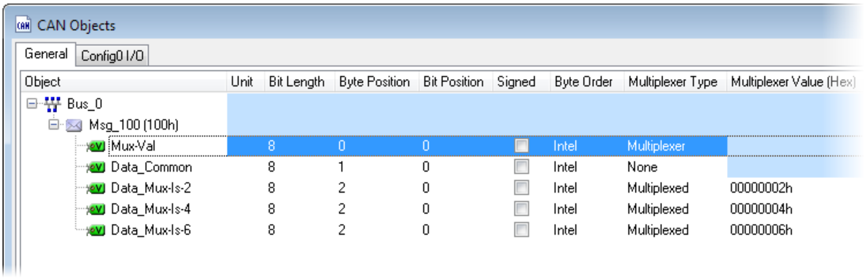
Definition: The example message has the following key parameters:

- CAN ID: 100h
- Length: 3 bytes
- Bit assignment (variables):

Byte no./ Start bit	Bits	Designation	Use
0/0	8	Mux-Val	Multiplexer
1/0	8	Data_Common	Variable independent of multiplexer (always used)
2/0	8	Data_Mux-is-2 Data_Mux-is-4 Data_Mux-is-6	Changing variable depending on the multiplexer value in data byte 0 (here: 2, 4 or 6)

Action: In the **CAN Objects** windows on the **General** tab, create the new CAN message 100h with the key parameters above. For the variables enter the following in the columns **Multiplexer Type** and **Multiplexer Value**:

Designation	Multiplexer Type / Value	Explanation
Mux-Val	Multiplexer	Value determines which Multiplexed variable is used.
Data_Common	None	Variable is always used in this CAN message, independently of the Multiplexer.
Data_Mux-Is-2	Multiplexed / 2	On Multiplexer values 2, 4, and 6, the corresponding variable is used.
Data_Mux-Is-4	Multiplexed / 4	
Data_Mux-Is-6	Multiplexed / 6	


CAN Objects

General | Config0 I/O

Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order	Multiplexer Type	Multiplexer Value (Hex)
Bus_0								
Msg_100 (100h)								
Mux-Val		8	0	0	☐	Intel	Multiplexer	
Data_Common		8	1	0	☐	Intel	None	
Data_Mux-Is-2		8	2	0	☐	Intel	Multiplexed	00000002h
Data_Mux-Is-4		8	2	0	☐	Intel	Multiplexed	00000004h
Data_Mux-Is-6		8	2	0	☐	Intel	Multiplexed	00000006h

Remark: The setting in the **Multiplexer Type** column has the following possibilities:

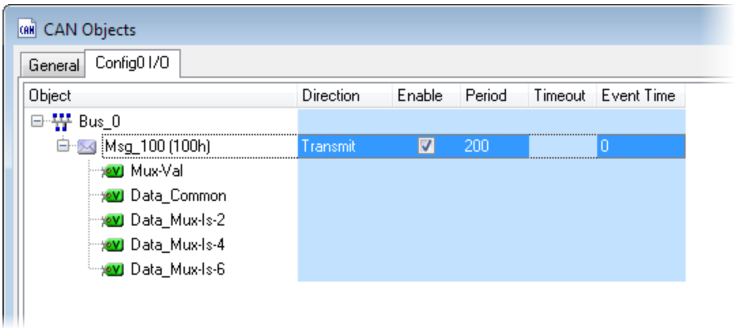
- **Multiplexer**: The variable contains the multiplexer value (data type Unsigned). This multiplexer type may only be used once within a message and must be placed before the variable definitions of the Multiplexed type in the list.
- **None**: This variable is used in all transmit messages, independently of the multiplexer value.
- **Multiplexed**: This variable is only transmitted if the given **Multiplexer Value** fits the current multiplexer value from Mux-Val.

Information: In the following, fixed test values are assigned to the Data variables to be transmitted. Furthermore, the message is to be transmitted every 200 ms. Definition is done in the device-specific configuration (here: Config0 I/O).

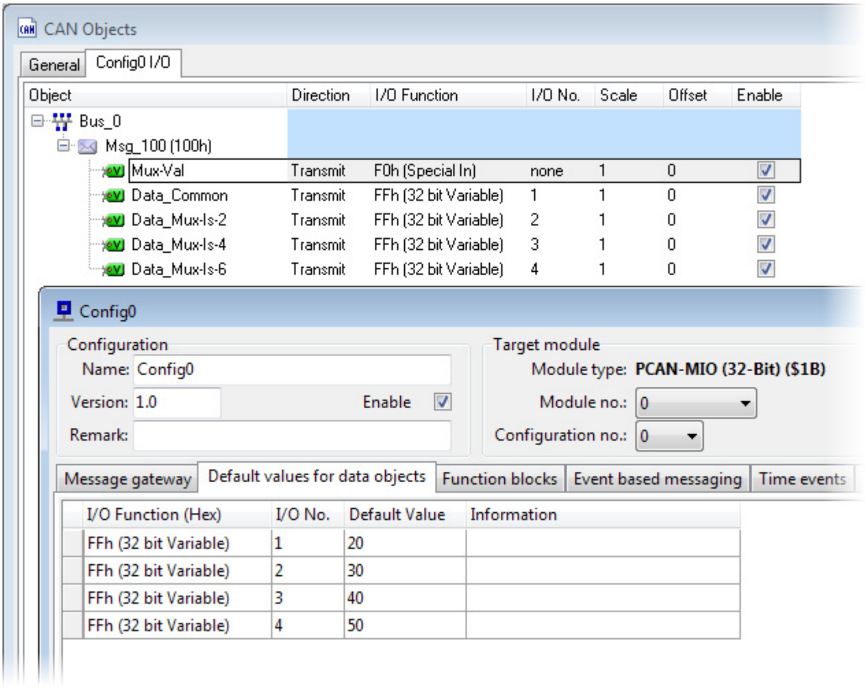
Action: If not already done, add a new configuration to the PPCAN-Editor project with **Edit** > **New Configuration**. Select the module type **PCAN-Router Pro** (see also on page 12).

In the context menu (right-click) of the CAN message **Msg_100**, select the **Add Symbol to Configuration** entry and then **Config0 I/O**.

Definition: In the configuration Config0, now some settings for the CAN message and the contained variables must be done, in the **CAN Objects** window as well as in the **Config0** window.



Object	Direction	Enable	Period	Timeout	Event Time
Bus_0					
Msg_100 (100h)	Transmit	✓	200		0
Mux-Val					
Data_Common					
Data_Mux-Is-2					
Data_Mux-Is-4					
Data_Mux-Is-6					



Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
Msg_100 (100h)						
Mux-Val	Transmit	F0h (Special In)	none	1	0	✓
Data_Common	Transmit	FFh (32 bit Variable)	1	1	0	✓
Data_Mux-Is-2	Transmit	FFh (32 bit Variable)	2	1	0	✓
Data_Mux-Is-4	Transmit	FFh (32 bit Variable)	3	1	0	✓
Data_Mux-Is-6	Transmit	FFh (32 bit Variable)	4	1	0	✓

Name: Config0
Version: 1.0
Enable ✓
Remark:
Module type: PCAN-MIO (32-Bit) ($1B)
Module no.: 0
Configuration no.: 0

I/O Function (Hex)	I/O No.	Default Value	Information
FFh (32 bit Variable)	1	20	
FFh (32 bit Variable)	2	30	
FFh (32 bit Variable)	3	40	
FFh (32 bit Variable)	4	50	

Remark: The resource Special In (F0h)/none is assigned to the multiplexer variable Mux-Val. This means that the variable is set to the next Multiplexed variable with each transmission of the CAN message (here every 200 ms). Thus, the Multiplexed variables are transmitted in sequence.

Information: The configuring process for this example is finished and the project can be transferred to the PCAN-Router Pro.

4.23 Exercise 8b: Transmitting a Multiplexer Message on Request

Information: In alternative to the automatic iteration of the Multiplexed variables, another resource can be used, e.g. a 32-bit variable that is changed via CAN. In this way the multiplexer value is determined from the outside.

Application possibility: Several parameters are to be polled via CAN, but only a single CAN ID is available for transmission. A multiplexer value is assigned to each parameter. The multiplexer value is determined by a separate receive message, as answer the multiplexer CAN message is transmitted.

In this exercise, the reception of the CAN message 1FFh (1 byte) triggers the transmission of the already existing 100h message. The data byte of 1FFh is used as multiplexer value in 100h.

Remark: This exercise is based on the previous 8a.

Action: In the **CAN Objects** window on the **General** tab, create the additional CAN message 1FFh with the following properties:

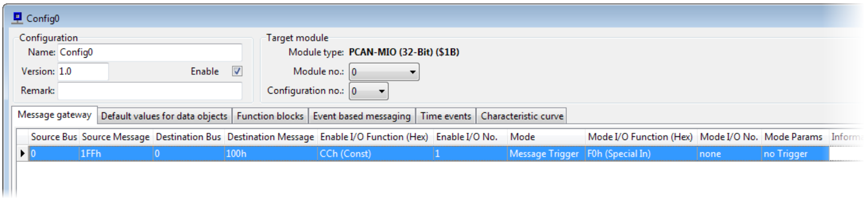
- CAN ID: 1FFh (Trigger_Msg_100)
- Length: 1 byte
- Bit assignment (variables):

Byte no./ Start bit	Bits	Designation	Use
0/0	8	Request_MuxData	Value for the multiplexer in ID 100h

Under **Config0 I/O**, set the message to **Receive** and assign the **Request_MuxData** variable to the internal 32-bit variable 255 (I/O function FFh).

Also under **Config0 I/O**, alter the already existing CAN message **Msg_100** (100h) that it is not transmitted periodically anymore by setting the **Period** to 0. Alter the **Mux-Val** variable that it receives its value from the internal 32-bit variable 255 (I/O function FFh).

Like in exercise 1c (on page 19), a new entry is inserted in the **Message Gateway** of **Config0** so that the reception of **Trigger_Msg_100** triP2E9A5ggers the transmission of **Msg_100**.


Config0

Configuration
Name: Config0
Version: 1.0 Enable ✓
Remark:

Target module
Module type: PCAN-MIO (32-Bit) ($1B)
Module no.: 0
Configuration no.: 0

Message gateway | Default values for data objects | Function blocks | Event based messaging | Time events | Characteristic curve

Source Bus	Source Message	Destination Bus	Destination Message	Enable I/O Function (Hex)	Enable I/O No.	Mode	Mode I/O Function (Hex)	Mode I/O No.	Mode Params	Inform
0	1FFh	0	100h	CCh (Const)	1	Message Trigger	F0h (Special In)	none	no Trigger	

Information: The configuring process for this example is finished and the project can be transferred to the PCAN-Router Pro.

4.24 Exercise 9: Bonus Exercises

- One could develop a configuration, which issues a pulse wave on Dout-0 with a defined frequency and duty cycle. This may be verified using a suitable measuring instrument (frequency meter, oscilloscope). The frequency or duty cycle may be controlled via CAN message
- One could develop a configuration, which evaluates an incoming pulse wave on Din-0 regarding frequency and duty cycle. This may be verified using a suitable pulse generator (function generator, signal generator). The measured frequency and duty cycle may be transmitted onto CAN bus, or 2 proportional analog voltages may be issued at Aout-0 and Aout-1
- One could develop a configuration, which (depending on a Din) routes a 32-bit signal through one of two characteristic curves. There is also a "set of characteristic curves" forming a 3 dimensional translation area controlled by an additional parameter Z coming in via CAN. If Z is between 2 curves, the result will be interpolated linearly.
- Also think of feedback control systems: a vast area ...