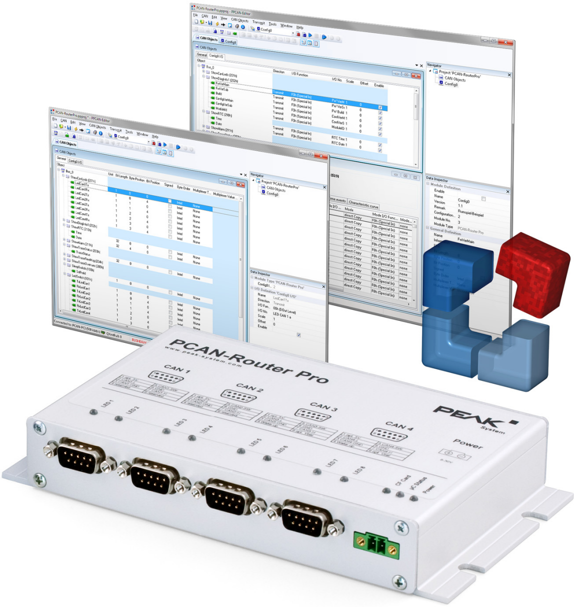
PCAN-Router Pro

4-Channel CAN Router with Data Logger

Configuration Tutorial



Document version 1.2.0 (2020-10-20)

Relevant Products

Product Name	Model	Part No.
PCAN-Router Pro	4 High-speed-CAN channels, Wake-Up capability, other CAN-transceiver modules on request	IPEH-002212
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 (2020-10-20)

Contents

1 Introduction

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

This tutorial therefore addresses owners of a PCAN-Router Pro, 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 Technik GmbH offers to their customers a configuration service subject to detailed specifications.

1.1 Prerequisites for Operation

For reasonably processing this tutorial respectively for solving the exercises, a PCAN-Router Pro (with sufficient power source) should be at hand. Its CAN busses should be connected to a computer via PEAK interfaces and also should be properly terminated e.g. by means of the internal DIP switches.

- At least two CAN busses (e.g. 1 and 4) are connected to the PC via PEAK interfaces, with 500 kbit/s each
- A specially prepared CF card (e.g. 1 GByte, included in shipment) is inserted
- The PPCAN-Editor 2 software is installed

- As a partner CAN participant, e.g. a PCAN-View (or even better: a PCAN-Explorer part no. IPES-005028) is installed on the PC
- Also the included PEAK-Converter software plus a commercial CF card reader (not included) are installed on the PC

The device PCAN-Router Pro offers the following resources for configurations

- Device ID (4 bit, 0..15 dec) may be adjusted within the device using a rotary switch (see user manual PCAN-Router Pro)
- 4 CAN busses (#1 .. #4), with Wake-Up feature
- Different CAN bus transveiver (HS, HS-OPTO[1], LS-DW, LS-SW)
- CAN bus bit rate[2] (10k; 20k; 33.3k; 47.6k; 50k; 83.3k; 95.2k; 100k; 125k; 250k; 500k; 1M)
- CAN messages (11 bit or 29 bit IDs)
- 1 CompactFlash card (serving as virtual CAN bus #5), recording modes configurable
- 2 LEDs per CAN bus (for a total of 8, status can be written or read)
- Time of Day (RTC)
- Beeper
- Software switch for Sleep mode

[1] Please query for availability.

[2] 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 of PEAK-System Technik GmbH 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. Accordingly, a module without configuration just represents a collection of loose ends and is therefore inoperational.

For creating, editing, and managing configurations, PEAK-System Technik GmbH offers the PPCAN-Editor 2 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 valid one is then determined by means of a selector switch.

Project files created with the PPCAN-Editor 2 may contain several configurations. The device ID selects the one to be executed when the device starts (e.g. after being powered). The selector switch simultaneously determines the device ID and the memory slot within the device's non-volatile memory, where the chosen configuration is loaded from. 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 unique ID of each device will let them load their individual configuration from the appropriate memory slot and subsequentially execute 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 advised 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 our website 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, permanent message routings or logging modes may be activated, LEDs and wires 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 vast collection of mathematical and logical functions up to a complex PIDT1 closed-loop control. Function blocks may be 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 Characteristics Curves

Here, 2 to 31 X/Y translation pairs may be defined. An incoming X value results in the output of the assigned Y value. X values in between two X/Y pairs will return an 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 non-continuous functions.

3 List of Exercises

An overview on the vast capabilities of the PCAN hardware (like the PCAN-Router Pro) may be given when solving the following exercises.

- 1a) Forwarding of all messages from CAN-1 to CAN-4
- 1b) Forwarding of defined messages from CAN-1 to CAN-4
- 1c) Forwarding of all messages from CAN-1 to CAN-4 with exceptions
- 2a) Recording of all received messages to a binary Trace file on the CF card
- 2b) Conversion of the binary Trace file from the CF card to a PC
- 3a) Definition of CAN messages (e.g. reading a system variable)
- 3b) Translating CAN ID
- 3c) (Variation 3b) Transmission only if source message was received
- 3d) (Variation 3a) Transmission only on Remote Request
- 4a) Manipulating CAN signals using SCALE and OFFSET
- 4b) Manipulating CAN signals using Function block Characteristic curve
- 5a) LED activity on CAN reception and transmission
- 5b) Controlling LED manually or conditionally
- 5c) Controlling LED externally
- 5d) Controlling the Beeper (continuous tone)
- 5e) Controlling the Beeper (tone sequence)
- 6a) Reading date and time (Hardware Diagnostics)

- 6b) Setting date and time (Hardware Diagnostics)
- 6c) Reading the module ID (Hardware Diagnostics)
- 6d) Reading firmware version and configuration version (Hardware Diagnostics)
- 7a) Sleep and Wake-Up via CAN
- 7b) Sleep and Wake-Up via external pin
- 7c) Sleep and Wake-Up via timed alarm
- 8a) Changing the bit rate
- 9a) Transmitting a Multiplexer Message Automatically
- 9b) Transmitting a Multiplexer Message on Request

4 Solutions and Explanations

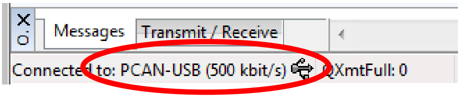
You can find further information about the use of PPCAN-Editor 2 in the help which you can invoke in the program via the menu **Help** or the F1 key.

4.1 Exercise 1a: Forwarding of All Messages from CAN-1 to CAN-4

Activity: Start the PPCAN-Editor with a double click on the icon, or by selecting the PPCAN-Editor from the list of installed programs.

Activity: Connecting the PPCAN-Editor with a PEAK-Interface e. g. PCAN-USB. Select menu item **CAN** -> **Connect** and choose then the appropriate hardware.

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

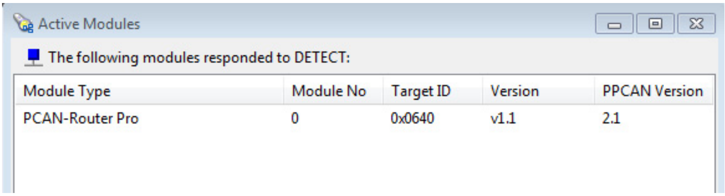


Connected to: PCAN-USB (500 kbit/s)

XmtFull: 0

Activity: Check whether the PCAN-Router Pro can be found on the CAN network by selecting menu item **Transmit -> Detect Modules**.

Reaction: The Active Modules window lists the available devices (here: PCAN-Router Pro) along with some status information.



The following modules responded to DETECT:

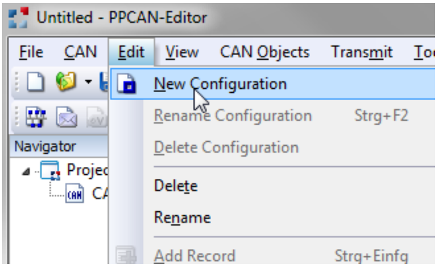
Module Type	Module No	Target ID	Version	PPCAN Version
PCAN-Router Pro	0	0x0640	v1.1	2.1

E. g. column Module No holds the currently adjusted device ID, 0 in this case. The field Version holds the firmware revision.

Activity: Create a new empty configuration file using menu item **File -> New**.

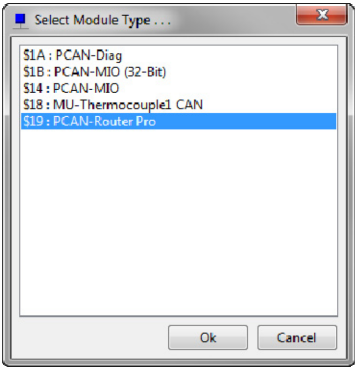
Reaction: A yet empty window appears where global CAN objects can be defined. Instructions on using that window (and editing its content) are given in exercise 3a.

Activity: For creating a new configuration within the configuration file choose menu item **Edit -> New Configuration**:



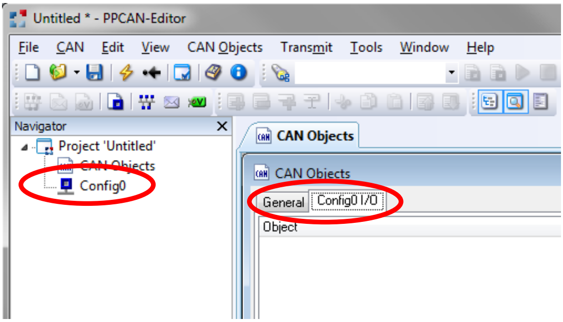
Reaction: PPCAN-Editor asks for the hardware to be configured.

Information: PPCAN-Editor can configure several different PCAN devices, equipped with individual resources each. Therefore with each type of hardware a list of available resources is supplied by the manufacturer for each type of hardware: the so called **hardware profile** file (*.ppprf).

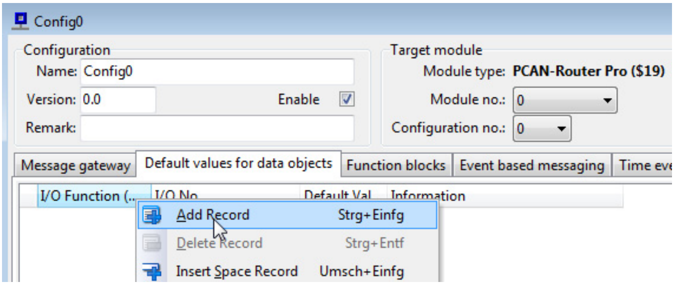


Activity: Choose the profile for the PCAN-Router Pro.

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



Activity: A double-click on this icon will open the configuration window. Here, e.g. default values for data objects can be set, like routing instructions etc. Change to Default values for data objects by selecting that tab. Then a new record must be added here by either selecting menu entry **Edit -> Add record** or selecting **Add record** from the context menu:



Config0

Configuration

Name: Config0

Version: 0.0 Enable ☑

Remark:

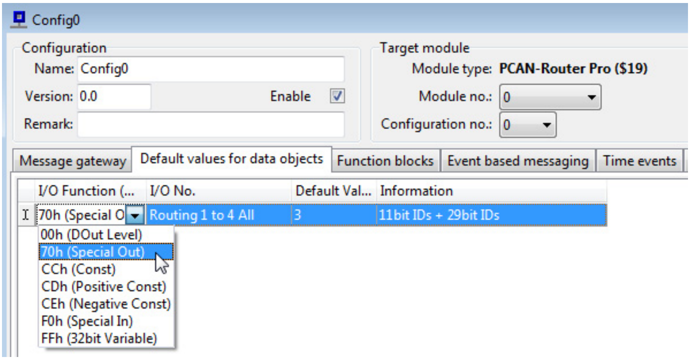
Target module

Module type: PCAN-Router Pro ($19)

Module no.: 0

Configuration no.: 0

Activity: Cell content can be edited by either pressing F2, or by a slow double click, or by simply typing the new value. The entries should be set as follows:



Config0

Configuration

Name: Config0

Version: 0.0 Enable ☑

Remark:

Target module

Module type: PCAN-Router Pro ($19)

Module no.: 0

Configuration no.: 0

I/O Function (...	I/O No.	Default Val...	Information
70h (Special O	Routing 1 to 4 All	3	11bit IDs + 29bit IDs

1. I/O-Function: SpecialOut (one group of resources).

2. I/O-No: Routing 1 to 4 All.

3. Default value: **3** = Sum of **1** (route 11 bit IDs only) and **2** (route 29 bit IDs only).

4. Information: Description of what this line does (helpful in later sessions).

In this example, routing of messages from CAN-1 (source) to CAN-4 (destination) is activated. A value of 3 means, that both the 11 bit IDs (1) as well as the 29 bit IDs (2) are forwarded (parameter values can be combined by addition). The I/O function for routing is located in resource group 0x70 SpecialOut, which (among others) offers the following routing possibilities for CAN messages:

Source is CAN 1	Source is CAN 2	Source is CAN 3	Source is CAN 4
Routing 1 to 2 All	Routing 2 to 1 All	Routing 3 to 1 All	Routing 4 to 1 All
Routing 1 to 3 All	Routing 2 to 3 All	Routing 3 to 2 All	Routing 4 to 2 All
Routing 1 to 4 All	Routing 2 to 4 All	Routing 3 to 4 All	Routing 4 to 3 All
Routing 1 to CF All	Routing 2 to CF All	Routing 3 to CF All	Routing 4 to CF All
Routing 1 to 2 explicit	Routing 2 to 1 explicit	Routing 3 to 1 explicit	Routing 4 to 1 explicit
Routing 1 to 3 explicit	Routing 2 to 3 explicit	Routing 3 to 2 explicit	Routing 4 to 2 explicit
Routing 1 to 4 explicit	Routing 2 to 4 explicit	Routing 3 to 4 explicit	Routing 4 to 3 explicit
Routing 1 to CF explicit	Routing 2 to CF explicit	Routing 3 to CF explicit	Routing 4 to CF explicit
Routing 1 to 2 excluding	Routing 2 to 1 excluding	Routing 3 to 1 excluding	Routing 4 to 1 excluding
Routing 1 to 3 excluding	Routing 2 to 3 excluding	Routing 3 to 2 excluding	Routing 4 to 2 excluding
Routing 1 to 4 excluding	Routing 2 to 4 excluding	Routing 3 to 4 excluding	Routing 4 to 3 excluding
Routing 1 to CF excluding	Routing 2 to CF excluding	Routing 3 to CF excluding	Routing 4 to CF excluding

In this context, explicit means routing of the specified ID only and excluding means routing of everything except that ID. The use of these functions is demonstrated in the following exercises (1b and 1c).

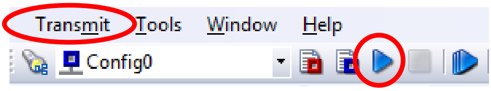
Remark 1: Routing functions explicit und excluding only support 11 bit IDs.

Remark 2: CF card cannot be used as a data source (e.g. for Playbacks).

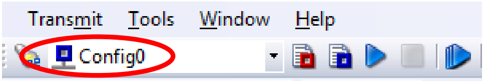
Information: At this point, all the configuration work for solving the exercise is done.

Activity: Enter a title for this configuration in the field **Remark**. The configuration project file (*.ppproj) should be saved as Exercise 1a to your PC. To do so, please select the menu item **File -> Save As**.

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



Important Note: Ensure that the list box in the toolbar (upper window edge) shows the name of your configuration Config0.



Reaction: While uploading, the "Output" window of the PPCAN-Editor shows lots of progress messages regarding the transmission protocol. Their meaning is explained in other documents.

Reaction: The status LED of the PCAN-Router Pro flashes during the transmission and processing of the configuration file randomly. Thereafter, the status LED flashes green at 1 Hz and the PCAN-Router Pro is ready with its new configuration.

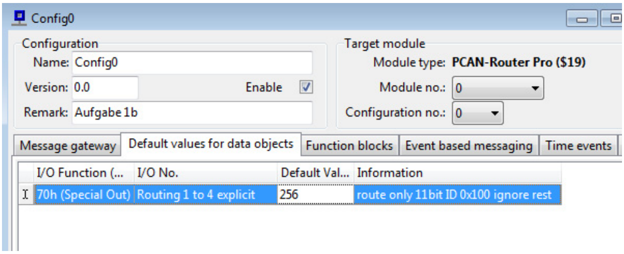
Result: The PCAN-Router Pro will now transfer all incoming messages from CAN-1 unmodified to the CAN-4 (but not in the reverse direction, this must be specified in an additional record line).

4.2 Exercise 1b: Forwarding of Defined Messages from CAN-1 to CAN-4

Information: Only ID 0x100 shall be routed from CAN-1 to CAN-4.

Activity: Open the configuration created in exercise 1a and save as exercise 1b. In the navigator (left edge of the main window), double-click Config0. The dialog box configuration is shown, change to the tab Default values for data objects.

Activity: Modify the existing entry from exercise 1a as shown:



Config0

Configuration

Name: Config0

Version: 0.0 Enable ☑

Remark: Aufgabe 1b

Target module

Module type: PCAN-Router Pro ($19)

Module no.: 0

Configuration no.: 0

I/O Function (...	I/O No.	Default Val...	Information
70h (Special Out)	Routing 1 to 4 explicit	256	route only 11bit ID 0x100 ignore rest

1. I/O-Function: SpecialOut (remains unchanged).
2. I/O-No: Routing 1 to 4 explicit.
3. Default Value: enter here the 11 bit ID to be routed (0..2047dez).
4. Information: Description of what this line does (helpful in later sessions).

Remark: Routing function explicit only support 11 bit IDs!

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 1b to your PC.

Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus. As explained in the first exercise, see previous page for instruction.

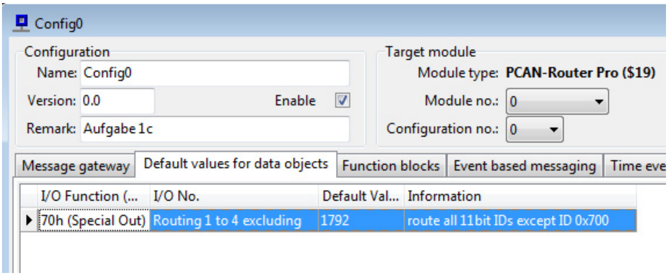
Result: All incoming messages at CAN-1 are ignored, only ID 0x100 is routed to CAN-4.

4.3 Exercise 1c: Forwarding of All Messages from CAN-1 to CAN-4 with Exceptions

Information: Anything except ID 0x700 shall be forwarded from CAN-1 to CAN-4.

Activity: Open configuration from exercise 1a and save as exercise 1c. In the navigator (left edge of the main window), double-click Config0. The dialog box configuration is shown, change to the tab Default values for data objects.

Activity: Modify the existing entry from exercise 1a as shown:



Config0

Configuration
Name: Config0
Version: 0.0
Enable ☑
Remark: Aufgabe 1c

Target module
Module type: PCAN-Router Pro ($19)
Module no.: 0
Configuration no.: 0

I/O Function (...	I/O No.	Default Val...	Information
70h (Special Out)	Routing 1 to 4 excluding	1792	route all 11bit IDs except ID 0x700

1. I/O-Function: SpecialOut (remains unchanged).
2. I/O-No: Routing 1 to 4 excluding.
3. Default Value: enter here the 11 bit ID to be omitted (0..2047dec).

4. Information: Description of what this line does (helpful in later sessions).

Remark: Routing function excluding only supports 11 bit IDs!

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 1c to your PC.

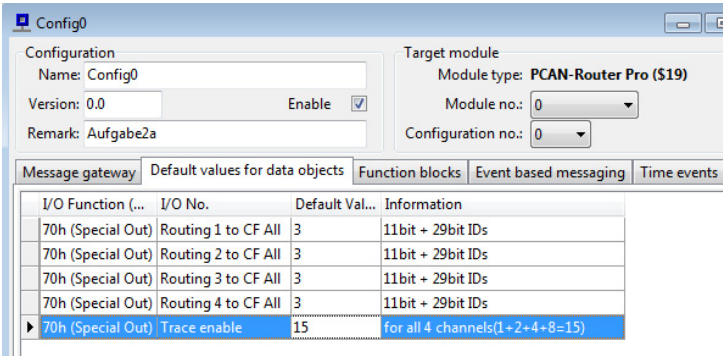
Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

Result: PCAN-Router Pro forwards all 11 bit IDs coming in at CAN-1 to CAN-4 except for ID 0x700, which is discarded (which also applies to all 29 bit IDs).

4.4 Exercise 2a: Recording of all Received Messages to a Binary Trace File on the CF Card

Activity: Open configuration from exercise 1a and save as exercise 2a. In the navigator (left edge of the main window), double-click Config0. The dialog box configuration is shown, change to the tab Default values for data objects.

Activity: Modify the existing entry from exercise 1a and create 4 additional lines as shown:



Configuration

Name: Config0

Version: 0.0 Enable ☑

Remark: Aufgabe2a

Target module

Module type: PCAN-Router Pro ($19)

Module no.: 0

Configuration no.: 0

I/O Function (...	I/O No.	Default Val...	Information
70h (Special Out)	Routing 1 to CF All	3	11bit + 29bit IDs
70h (Special Out)	Routing 2 to CF All	3	11bit + 29bit IDs
70h (Special Out)	Routing 3 to CF All	3	11bit + 29bit IDs
70h (Special Out)	Routing 4 to CF All	3	11bit + 29bit IDs
70h (Special Out)	Trace enable	15	for all 4 channels(1+2+4+8=15)

1. I/O-Function: SpecialOut.
2. Rows 1..4, I/O-No: Routing 1/2/3/4 to CF all.
3. Default Value: **3** = Sum of **1** (route 11 bit IDs only) and **2** (route 29 bit IDs only).
4. Row 5, I/O-No: Trace enable (1 bit for each CAN channel).
5. Default Value: **15** = Sum of **1** (CAN-1), **2** (CAN-2), **4** (CAN-3), and **8** (CAN-4).
6. Information: Description of what this line does (helpful in later sessions).

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 2a to your PC.

Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus).

Result: All messages coming in at the four CAN ports are forwarded to the CF card (logging function). Post processing is done later after transferring the log file to your PC (see exercise 2b).

Remark 1: Also the known routing functions explicit and excluding may be used (only support 11 bit IDs).

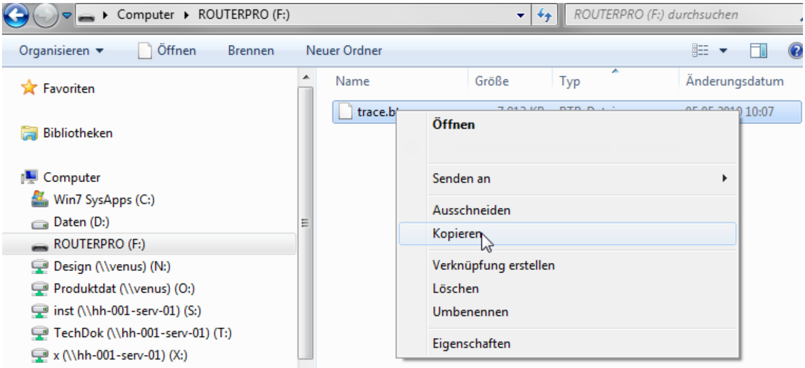
Remark 2: The CF card is also accessible as CAN-5 (virtual CAN channel).

Remark 3: The reverse direction (routing from CF card to the CAN ports, playback) is -not- possible.

4.5 Exercise 2b: Conversion of the Binary Trace File from the CF Card to the PC

Information: Using the CF management tool "PEAK-Converter" (comes along with the PCAN-Router Pro, or to be found on our website or the PEAK product CD), traces can be extracted from the CF card and converted to plain text (*.trc). This format is used by all PEAK-Applications (PCAN-Explorer, PCAN-Trace, etc.) for post processing.

Activity: For data extraction, the CF card must be removed from the PCAN-Router Pro (only to be done when the device is plugged off or in Sleep mode). The card is then inserted into your PC's card reader, and the contained binary file TRACE.BTR should be copied to your PC's local hard disk.



Activity: Now the binary Trace file (respectively the recorded traces) is processed with the CF management tool PEAK-Converter.

Activity: Before re-inserting the CF card into the PCAN-Router Pro any content should be removed in order to have the full capacity available: Simply do a format with Windows Explorer and afterwards copy an empty TRACE.BTR file to the card. Such file is available in different sizes from the PCAN-Router Pro's product CD.

4.6 Exercise 3a: Definition of CAN Messages, e.g. Reading a System Variable

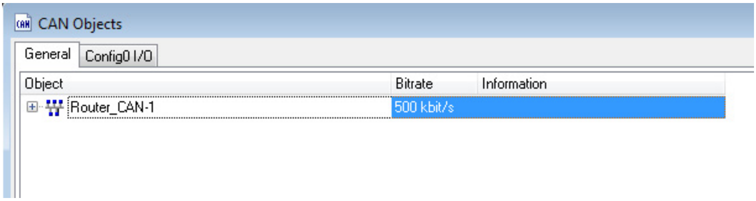
Activity: Create a new empty configuration file using menu item **File -> New**.

Reaction: This will open an (yet empty) window for all the global CAN objects used later within the different configurations. If a file contains multiple configurations with different CAN objects, they all must be defined here. Later, they are imported selectively into the different configurations.

Reaction: In that window, one CAN bus is already defined: Bus_0, below which global CAN objects can be created hierarchically. Remember: This is just an example for a possible configuration.

Definition: Bus 0 will be named Router_CAN-1, the field Bitrate is just for informational purposes. The CAN transceiver TJA1041 (standard model) will work at 500 kbit/s by default.

Activity: Double-click the name Bus_0 and enter a new name for it. Choose the default bit rate 500 kbit/s from the dropdown list (only as an information).

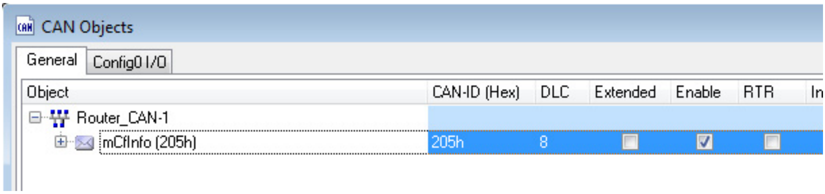


Object	Bitrate	Information
Router_CAN-1	500 kbit/s	

1. Bus name: Router_CAN-1.
2. Bit rate: 500 kbit/s (only as an information).
3. Information: Description of what this line does (helpful in later sessions).

Definition: On this bus, a CAN message mCfInfo shall be transmitted: 8 bytes length, CAN-ID = 0x205.

Activity: Select entry **Add a new Symbol** from the context menu (by right-clicking Router_CAN-1). This defines a new message on the CAN bus Router_CAN-1, parameters of the message must be set as follows:



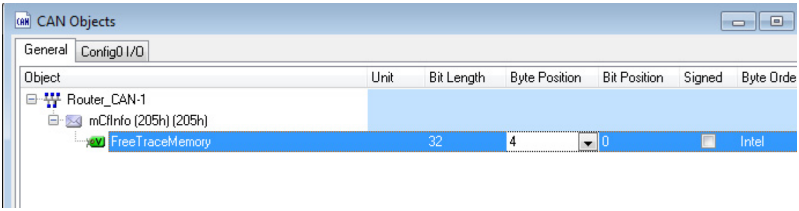
CAN Objects

Object	CAN-ID (Hex)	DLC	Extended	Enable	RTR	In
Router_CAN-1						
mCfInfo (205h)	205h	8	□	☑	□	

1. Symbol name: mCfInfo.
2. ID: 0x205.
3. DLC: 8 (message has 8 data bytes).
4. Extended: No, 11 bit IDs are sufficent.
5. Enabled: Yes.
6. RTR: No, message shall be sent always (-not- only on request).
7. Information: Description of what this line does.

Definition: This message shall contain a 32 bit signal FreeTraceMemory (in the data bytes 5...8), that displays the available number of CAN messages to be stored on the CF card.

Activity: Within the CAN message, a 32 bit wide data object must be created (= CAN signal), that holds the number of free places. From the context menu (right click on CAN message) select
Add a new Variable and then supply the parameters of the signal:



CAN Objects

Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Orde
Router_CAN-1						
mCfInfo (205h) (205h)						
FreeTraceMemory		32	4	0	□	Intel

1. Variable name: FreeTraceMemory.
2. Unit: no physical unit (only as an information).

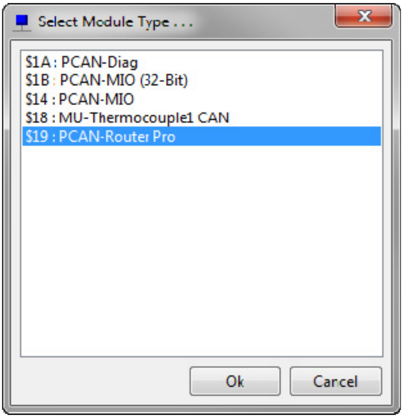
3. Bit length: 32.
4. Start Byte: 4 (by this, the signal is located in data bytes 5..8).
5. Start Bit: 0.
6. Signed: no, can’t be negative.
7. Byte Order: Intel format (LSB in byte 0 bit 0, MSB in byte 3 bit 7).
8. Information: Description of what this line does.

Information: The empty layout of the CAN message is hereby defined, but not yet assigned to a physical data source. Therefore a configuration must be created.

Activity: Create an empty configuration within this file: select the menu item **Edit -> New Configuration**.

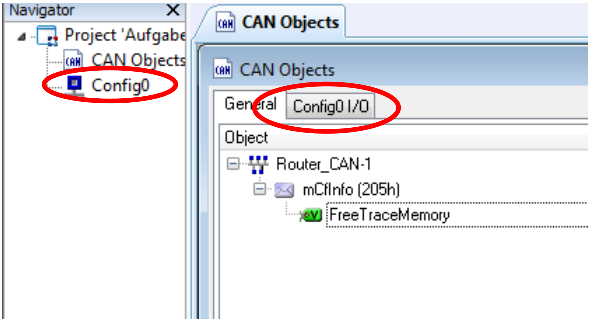
Reaction: You’re asked for the hardware to be configured.

Activity: Select the profile for a PCAN-Router Pro.

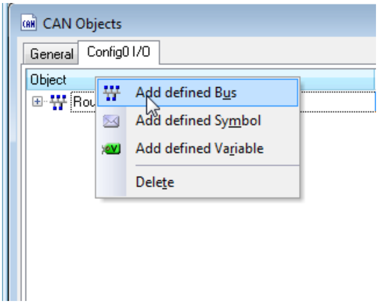


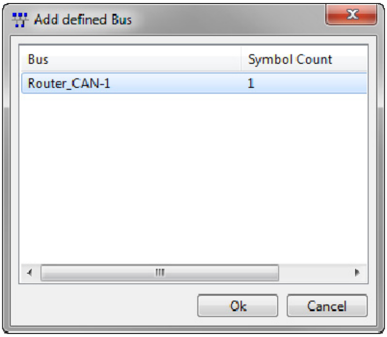
$1A : PCAN-Diag
$1B : PCAN-MIO (32-Bit)
$14 : PCAN-MIO
$18 : MU-Thermocouple1 CAN
$19 : PCAN-Router Pro

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



Activity: The globally defined CAN message shall be used in this configuration here. Therefore it must be imported. Click on the new tab Config0 I/O (for bringing it into the foreground) and select from the context menu (right click) **Add defined Bus**:

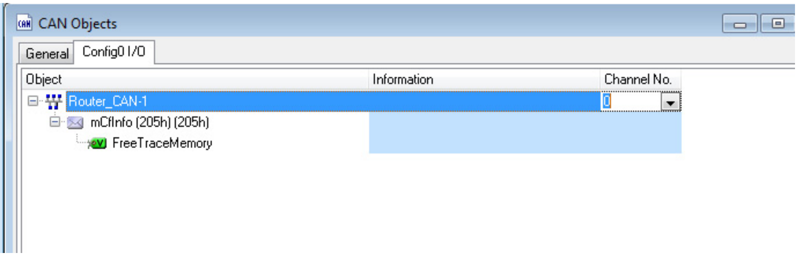




Bus	Symbol Count
Router_CAN-1	1

Reaction: The previously defined global CAN bus Router_CAN-1 (along with his contained message mCfInfo and 32 bit variable FreeTraceMemory) will be imported into the configuration Config0.

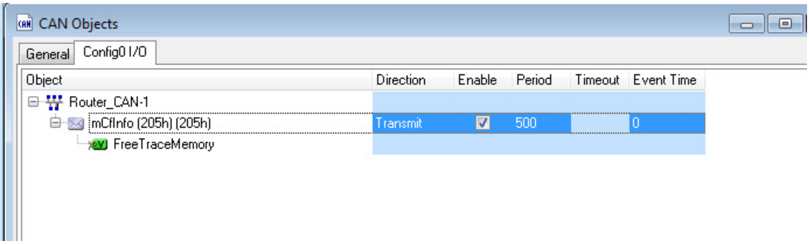
Activity: The defined bus Router_CAN-1 must get a CAN channel of the hardware (here: CAN channel #0):



Object	Information	Channel No.
Router_CAN-1		0
mCfInfo (205h) (205h)		
FreeTraceMemory		

1. Channel-Number: 0 (the hardware CAN channels are numbered 0-3 internally).

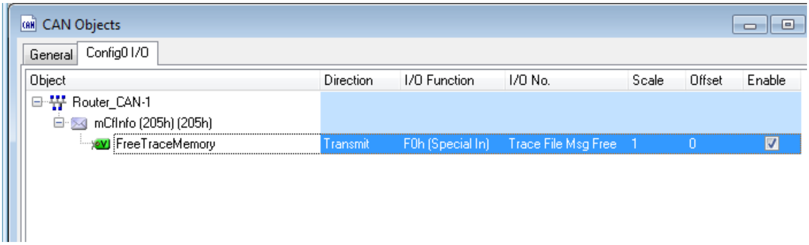
Activity: Now the message must be supplied with physical parameters:



Object	Direction	Enable	Period	Timeout	Event Time
Router_CAN-1					
mCfInfo (205h) (205h)	Transmit	☑	500		0
FreeTraceMemory					

1. Direction: Transmit (the PCAN-Router Pro shall be transmitter).
2. Enable: Yes, this is the message to be transmitted.
3. Period: 500 (the transmission cycle time in ms).

Activity: The contained CAN signal must get a data source:



Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Router_CAN-1						
mCfInfo (205h) (205h)						
FreeTraceMemory	Transmit	F0h (Special In)	Trace File Msg Free	1	0	☑

1. I/O-Function: F0-Special In (this is the source: an internal firmware variable).
2. I/O-Number: Trace File Msg Free (which is the name of the variable).
3. Scale: 1 (no scaling at all, like multiplying with 1).
4. Offset: 0 (no shifting at all, like addition of 0).
5. Enable: Yes, this is the signal (within the message) to be used.

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 3a to your PC.

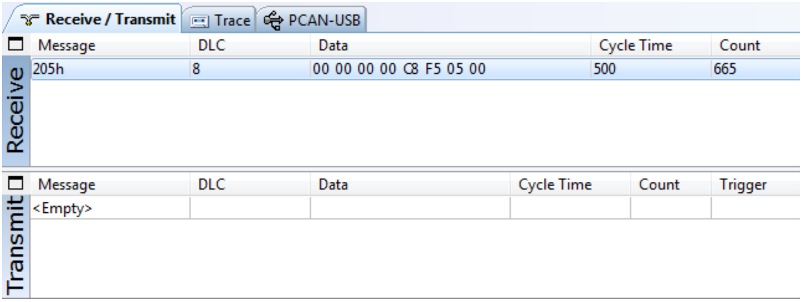
Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus as shown in exercise 1a.

Reaction: While uploading, the "Output" window of the PPCAN-Editor shows lots of progress messages regarding the transmission protocol. Their meaning is explained in other documents.

Reaction: While transmitting and processing of the configuration, the status LED of the PCAN-Router Pro is blinking randomly. After automatic restart, the status LED blinks with 1 Hz and the PCAN-Router Pro has successfully started with your new configuration.

Result: PCAN-Router Pro now transmits via CAN-1 a message with ID0x205, length 8 carrying the 32 bit value TraceFileMsgFree (which is an internal variable of the PCAN-Router Pro).

With a PCAN-View connected to the same CAN network this message can be watched.



Message	DLC	Data	Cycle Time	Count
205h	8	00 00 00 00 C8 F5 05 00	500	665

Message	DLC	Data	Cycle Time	Count	Trigger
<Empty>					

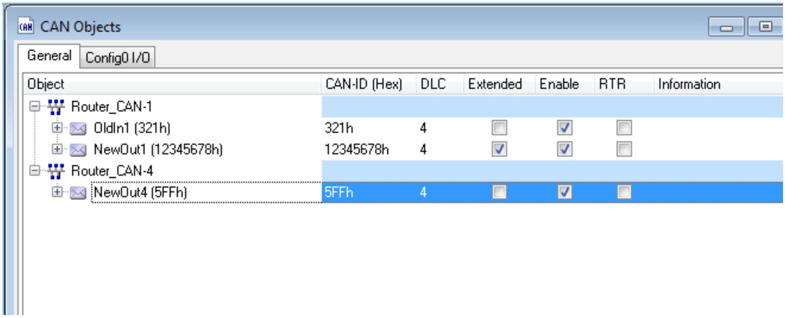
The shown message shown here contains information that the trace memory of the CompactFlash card has still room for 390600 CAN message (= 0x05F5C8).

4.7 Exercise 3b: Translating a CAN ID

Information: For the conversion of a CAN message to a different ID first two messages must be globally defined and imported into a new configuration. The buttons, menus, and clicks for doing this are already known from the previous exercises.

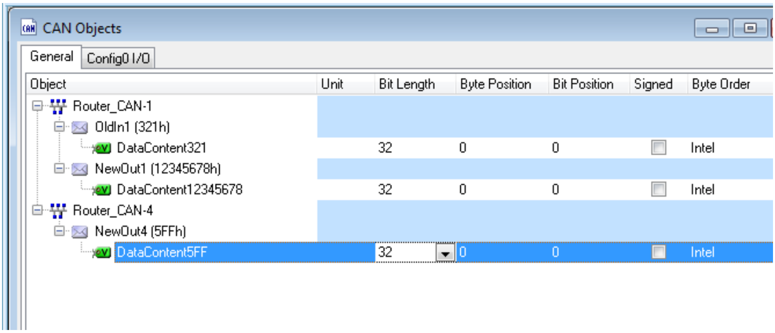
Definition: The contents of the incoming message 0x321 shall be transferred unchanged to the outgoing message 0x12345678. This transmit message will additionally be sent with ID 0x5FF on CAN-4.

Activity: Global definition of two CAN busses Router_CAN-1 and Router_CAN-4, as well as 3 messages (**Add a new Symbol**):



Object	CAN-ID (Hex)	DLC	Extended	Enable	RTR	Information
Router_CAN-1						
OldIn1 (321h)	321h	4	□	☑	□	
NewOut1 (12345678h)	12345678h	4	☑	☑	□	
Router_CAN-4						
NewOut4 (5FFh)	5FFh	4	□	☑	□	

Activity: Global definition of a 32 bit signal (**Add a new Variable**) within each of the 3 messages:

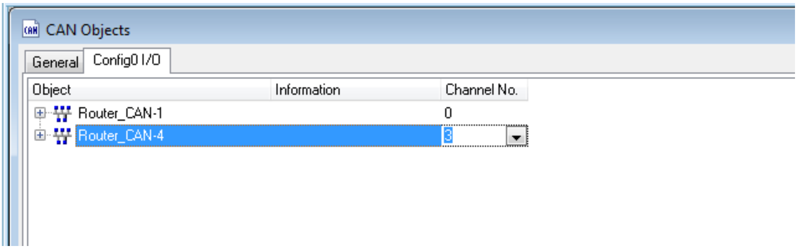


Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order
Router_CAN-1						
OldIn1 (321h)						
DataContent321		32	0	0	□	Intel
NewOut1 (12345678h)						
DataContent12345678		32	0	0	□	Intel
Router_CAN-4						
NewOut4 (5FFh)						
DataContent5FF		32	0	0	□	Intel

Activity: Creating a new configuration with **Edit -> New Configuration** (incl. selecting the hardware profile PCAN-Router Pro).

Activity: Import of the globally defined CAN objects into the tab Config0 I/O. To do so, open (right click) the context menu and select **Add defined Bus.** In the selection window choose the busses Router_CAN-1 and Router_CAN-4 one by one.

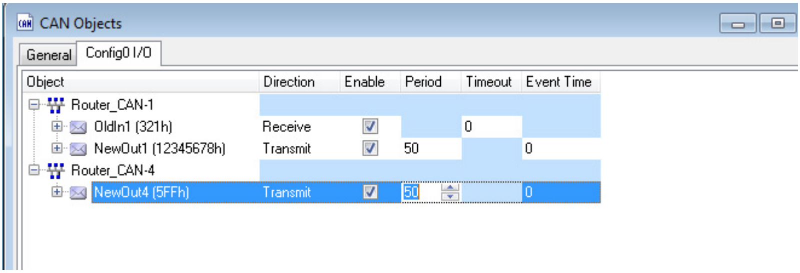
Activity: Assigning hardware channels to the busses:



Object	Information	Channel No.
Router_CAN-1		0
Router_CAN-4		3

1. Channel-Number: 0 (for Router_CAN-1) and 3 (for Router_CAN-4).

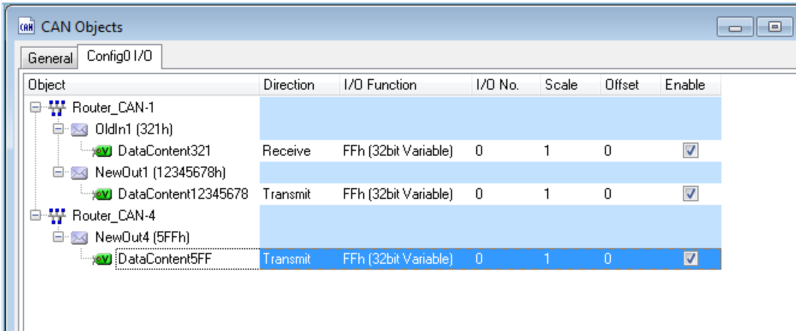
Activity: Enter the parameters for the 3 messages:



Object	Direction	Enable	Period	Timeout	Event Time
Router_CAN-1					
OldIn1 (321h)	Receive	☑		0	
NewOut1 (12345678h)	Transmit	☑	50		0
Router_CAN-4					
NewOut4 (5FFh)	Transmit	☑	50		0

1. Direction: one incoming message 0x321 (Receive), two outgoing messages 0x12345678 und 0x5FF (Transmit).
2. Enable: Yes, all these messages shall be used.
3. Period: 50 (transmission cycle time in ms).

Activity: Enter the parameters for the 3 signals:



Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Router_CAN-1						
OldIn1 (321h)						
DataContent321	Receive	FFh (32bit Variable)	0	1	0	☑
NewOut1 (12345678h)						
DataContent12345678	Transmit	FFh (32bit Variable)	0	1	0	☑
Router_CAN-4						
NewOut4 (5FFh)						
DataContent5FF	Transmit	FFh (32bit Variable)	0	1	0	☑

1. I/O-Function: FFh 32 bit variable (used for temporary storage of the content).
2. I/O-Number: 0 (256 of these RAM variables are available).
3. Scale: 1 (no scaling at all, like multiplying with 1).
4. Offset: 0 (no shifting at all, like addition of 0).
5. Enable: Yes, all these signals (within the messages) shall be used.

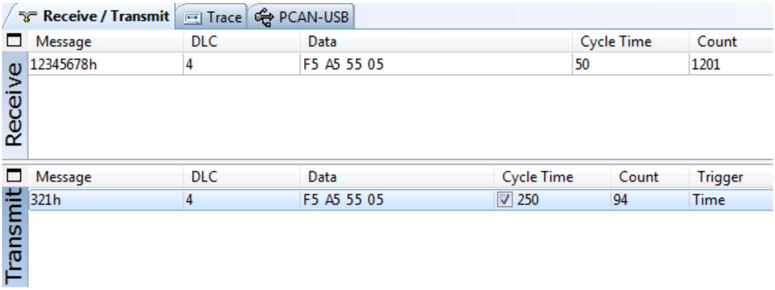
Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 3b to your PC.

Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

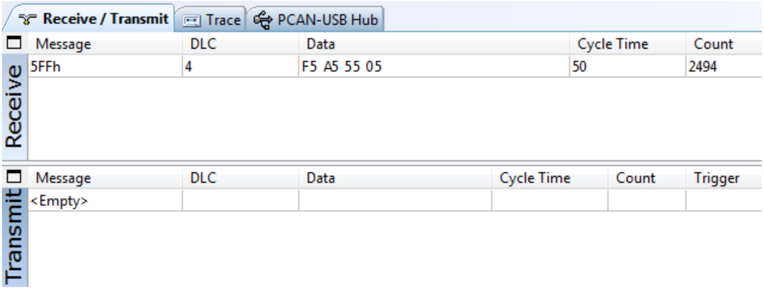
Activity: From PCAN-View or PCAN-Explorer, a 4 byte CAN message with ID 0x321 is sent into PCAN-Router Pro's CAN-1 with data bytes containing an eye-catching pattern.

Result: CAN-1 and CAN-4 will transmit messages 0x12345678 and 0x5FF cyclically with the same data pattern.



Message	DLC	Data	Cycle Time	Count
12345678h	4	F5 A5 55 05	50	1201

Message	DLC	Data	Cycle Time	Count	Trigger
321h	4	F5 A5 55 05	☑ 250	94	Time



Message	DLC	Data	Cycle Time	Count
5FFh	4	F5 A5 55 05	50	2494

Message	DLC	Data	Cycle Time	Count	Trigger
<Empty>					

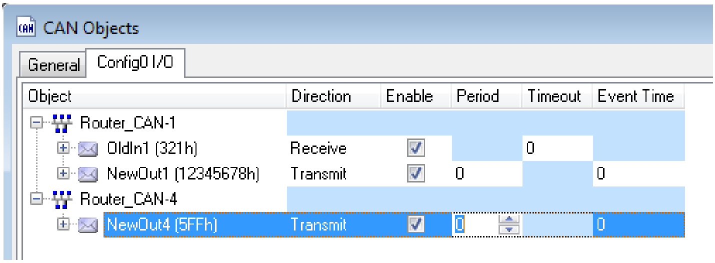
Information: Receive message 0x321 included a 32 bit value (0x0555A5F5) which is assigned to RAM variable # 0 within the PCAN-Router Pro. When transmitting message 0x12345678 and

0x5FF, the content of this variable # 0 is read back and stored into the transmit message. It may be stored on any position within the messages and may previously be scaled or modified by mathematical means.

4.8 Exercise 3c: (Variation 3b) Transmission Only if Source Message was Received

Information: Another feature is sending CAN messages, e. g. only if message 0x321 was really received, or its content has changed. As a variation of the previous exercise, the messages 0x5FF and 0x12345678 are transmitted only if a source message 0x321 was received.

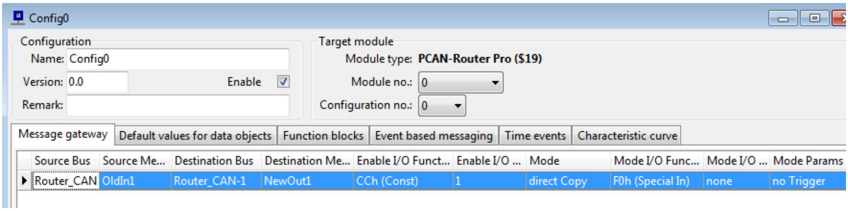
Activity: First, in the window CAN Objects, tab Config0 I/O set period values to 0, thus switching off cyclic transmission:



Object	Direction	Enable	Period	Timeout	Event Time
Router_CAN-1					
OldIn1 (321h)	Receive	☑		0	
NewOut1 (12345678h)	Transmit	☑	0		0
Router_CAN-4					
NewOut4 (5FFh)	Transmit	☑	0		0

1. Period: 0 (no cyclic transmission).

Activity: Instead, create two new entries in window Config0, tab Message Gateway: To do so, open the context menu (right click) and select **Add Record**. Enter the parameters of the ID 0x12345678 as follows:



Config0

Configuration

Name: Config0

Version: 0.0 Enable ☑

Remark:

Target module

Module type: **PCAN-Router Pro ($19)**

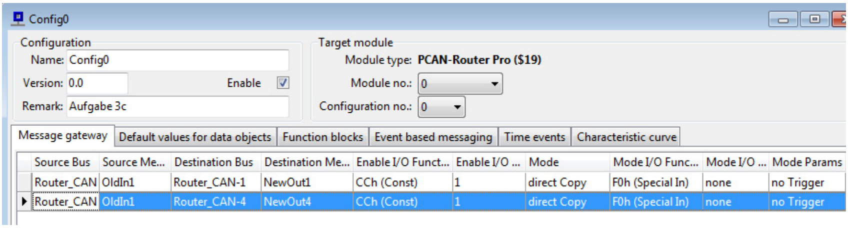
Module no.: 0

Configuration no.: 0

Source Bus	Source Me...	Destination Bus	Destination Me...	Enable I/O Funct...	Enable I/O ...	Mode	Mode I/O Func...	Mode I/O ...	Mode Params
Router_CAN	OldIn1	Router_CAN-1	NewOut1	CCh (Const)	1	direct Copy	F0h (Special In)	none	no Trigger

1. Source-Bus: Bus#0 (= Router_CAN-1).
2. Source-Message-ID: ID 0x321 (in 11 bit format).
3. Destination-Bus: Bus#0 (= Router_CAN-1).
4. Destination-Message-ID: ID 0x12345678 (in 29 bit format).
5. Enable I/O-Function: this routing shall -always- be active, so set it constantly.
6. Enable I/O-Number: set to value 1.
7. Mode: Direct copy (whenever something is received).
8. Mode I/O Function: not implemented yet, do not change.
9. Mode I/O Number: not implemented yet, do not change.
10. Mode Params: not implemented yet, do not change.
11. Information: Description of what this line does.

Activity: Then enter the parameters for the ID 0x5FF (on the 4th bus) as follows:



Config0

Configuration

Name: Config0

Version: 0.0

Enable ☑

Remark: Aufgabe 3c

Target module

Module type: **PCAN-Router Pro ($19)**

Module no.: 0

Configuration no.: 0

Source Bus	Source Me...	Destination Bus	Destination Me...	Enable I/O Funct...	Enable I/O ...	Mode	Mode I/O Func...	Mode I/O ...	Mode Params
Router_CAN	OldIn1	Router_CAN-1	NewOut1	CCh (Const)	1	direct Copy	F0h (Special In)	none	no Trigger
Router_CAN	OldIn1	Router_CAN-4	NewOut4	CCh (Const)	1	direct Copy	F0h (Special In)	none	no Trigger

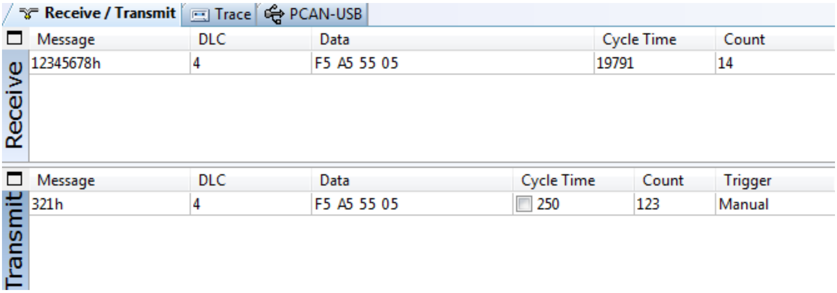
1. Source-Bus: Bus#0 (= Router_CAN-1).
2. Source-Message-ID: ID 0x321 (in 11 bit format).
3. Destination-Bus: Bus#3 (= Router_CAN-2).
4. Destination-Message-ID: ID 0x5FF (in 11 bit format).

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 3c to your PC.

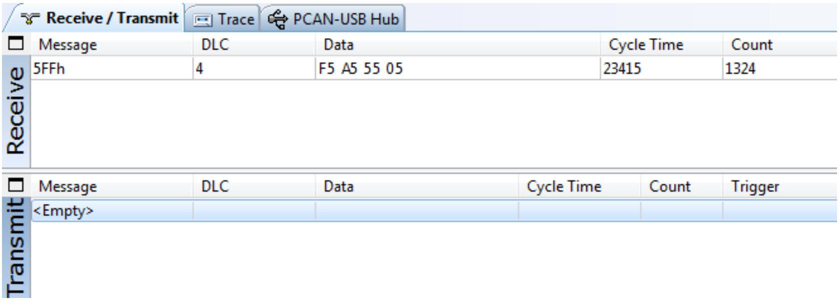
Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

Result: When sending a message with ID 0x321 to the PCAN-Router Pro, the contained data is forwarded in message ID 0x12345678 on CAN-1 and also in message ID 0x5FF on CAN-4:



Message	DLC	Data	Cycle Time	Count
12345678h	4	F5 A5 55 05	19791	14

Message	DLC	Data	Cycle Time	Count	Trigger
321h	4	F5 A5 55 05	□ 250	123	Manual



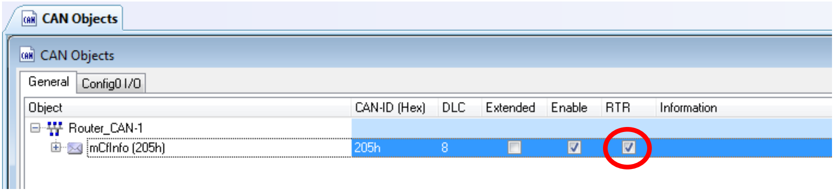
Message	DLC	Data	Cycle Time	Count
5FFh	4	F5 A5 55 05	23415	1324

Message	DLC	Data	Cycle Time	Count	Trigger
<Empty>					

4.9 Exercise 3d: (Variation 3a) Transmission Only on Remote Request

Information: Based on exercise 3a the internal variable FreeTraceMemory shall be transmitted on external request (RTR = Remote Transmission Request).

Activity: Open the configuration from exercise 3a and save it as exercise 3d. In tab General modify the CAN message to Transmission Request by setting the RTR check for that symbol:



CAN Objects

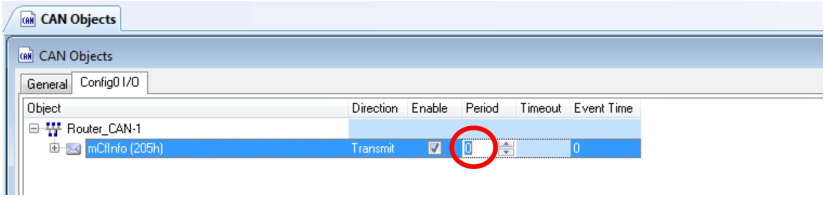
CAN Objects

General | Config0 I/O

Object	CAN-ID (Hex)	DLC	Extended	Enable	RTR	Information
Router_CAN-1						
mCfInfo (205h)	205h	8	□	☑	☑	

1. RTR: activate Remote Transmission Request.

Activity: In tab Config0 I/O set period value 0, thus switching off cyclic transmission:



CAN Objects

CAN Objects

General | Config0 I/O

Object	Direction	Enable	Period	Timeout	Event Time
Router_CAN-1					
mCfInfo (205h)	Transmit	☑	0		0

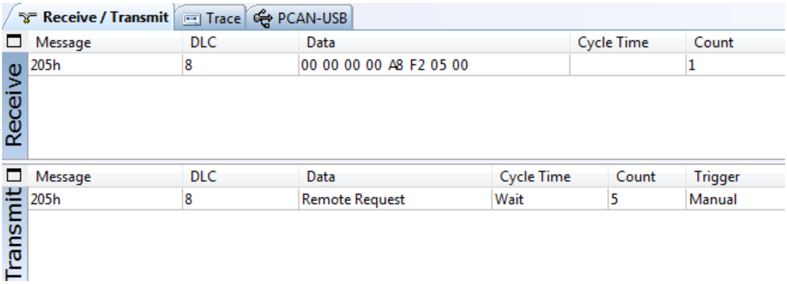
1. Period: 0 (no more cyclic transmission).

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 3d to your PC.

Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

Result: CAN message mCfInfo with ID 0x205 is transmitted only, if previously a request with ID 0x205 and length = 0 was received.



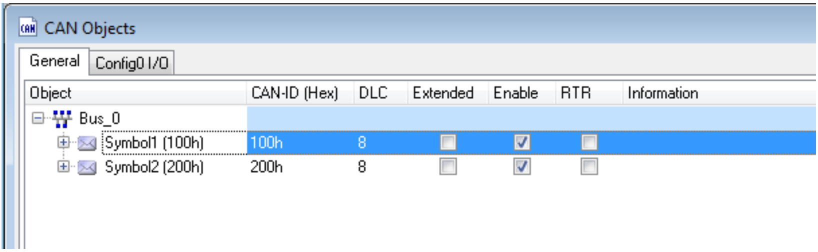
Message	DLC	Data	Cycle Time	Count
205h	8	00 00 00 00 A8 F2 05 00		1

Message	DLC	Data	Cycle Time	Count	Trigger
205h	8	Remote Request	Wait	5	Manual

4.10 Exercise 4a: Manipulating CAN Signals Using SCALE and OFFSET

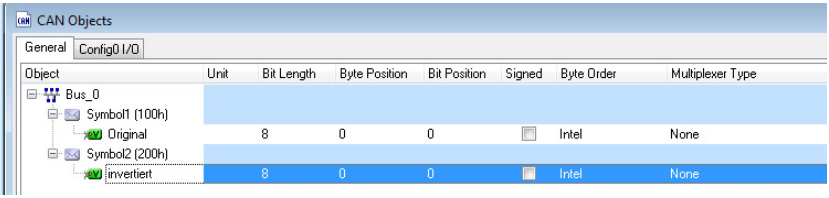
Information: With the parameters Scale and Offset values from a CAN bus can be manipulated like using the four basic arithmetics, all without a definition of a function block. For example, an increasing 8 bit value (rising ramp) can be inverted (falling ramp): 0x00..0xFF > 0xFF..0x00. To do so, the original value from the incoming message is written into a 32 bit variable (there is not a 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: You have to take care, that the manipulated result value – under all conditions - fits into the 32 bit size.

Activity: Create an empty configuration file using menu item **File -> New**. In the General tab add two symbols (CAN messages) to the already existing bus 0:



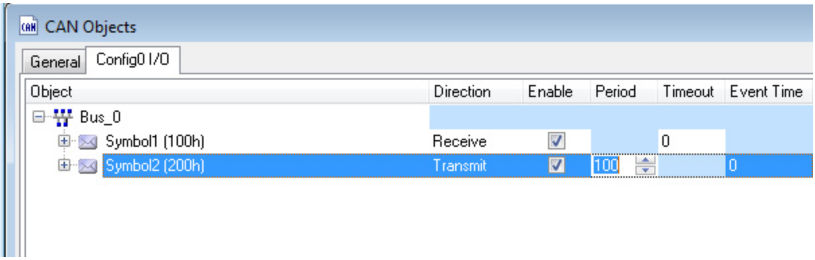
Object	CAN-ID (Hex)	DLC	Extended	Enable	RTR	Information
Bus_0						
Symbol1 (100h)	100h	8	□	☑	□	
Symbol2 (200h)	200h	8	□	☑	□	

Then, each message will get a 8 bit variable (= CAN signal) using the context menu:



Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order	Multiplexer Type
Bus_0							
Symbol1 (100h)							
Original		8	0	0	□	Intel	None
Symbol2 (200h)							
invertiert		8	0	0	□	Intel	None

Activity: Create a new configuration within the file (HW profile: PCAN-Router Pro) and import all CAN objects from the General tab, e.g. by dragging a complete bus with all underlying objects from the General tab and drop it onto the configuration tab's name "Config I/O". 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 msec), the contained signal is taken from the 32 bit variable #0, inverted (Scale = -1) and lifted (Offset=255):

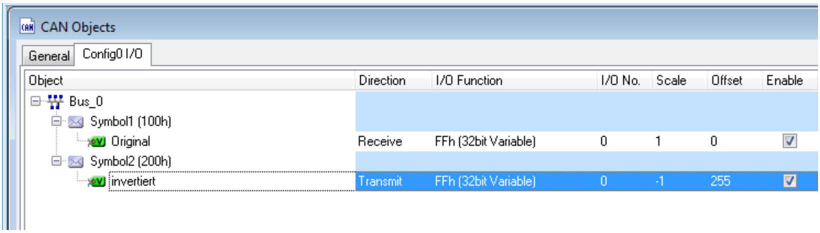


Object	Direction	Enable	Period	Timeout	Event Time
Bus_0					
Symbol1 (100h)	Receive	☑		0	
Symbol2 (200h)	Transmit	☑	100		0

1. Direction: 0x100 is received, 0x200 is transmitted by the PCAN-Router Pro.

2. Enable: Yes.

3. Period: 100 (transmission cycle time in ms).



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	☑

Information: At this point, all the configuration work for solving the exercise is done.

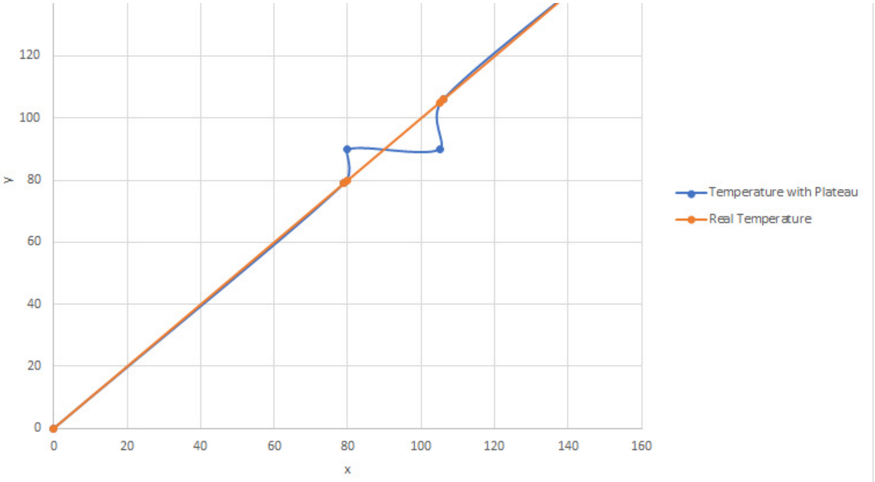
Activity: Save the configuration file as project exercise 4a to your PC.

Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

Result: 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 (increasing x values) is converted into a falling ramp (decreasing y values).

4.11 Exercise 4b: Manipulating CAN Signals Using Function Block Characteristic Curve

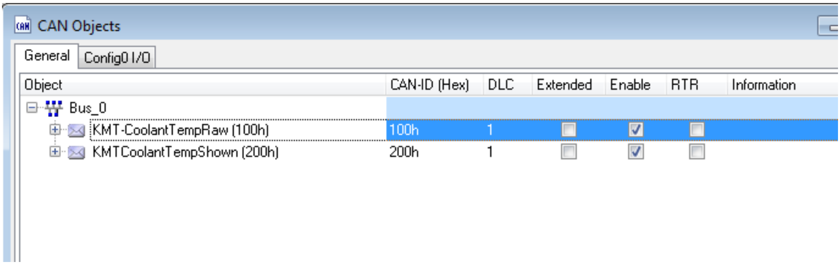
Definition: The temperature of a motor’s cooling fluid shall be represented as constantly 90 °C, even if the real coolant temperature varies between 80 °C and 105 °C. Such plateau is often implemented for smoothing of analog meters. In case the real temperature leaves the specified range (e.g. motor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



Information: For implementing such a behavior, the function block "characteristic curve" is suitable. It recalculates any incoming value by means of a X->Y list (thus creating that plateau).

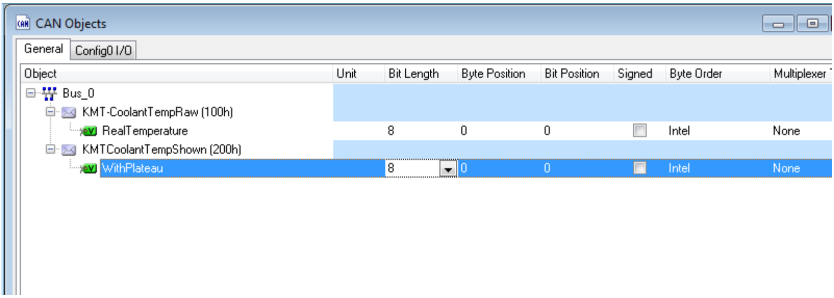
In the following example, 32 bit variable #0 holds the raw input value, which is given to the characteristic curve as input, whereas 32 bit value #1 gets the result of the conversion. That smoothed value is then transmitted in a separate CAN message.

Activity: As with the previous example, create an empty configuration file using menu item **File -> New,** then add two symbols (= CAN messages) with an 8 bit variable (= CAN signal) each:



Object	CAN-ID (Hex)	DLC	Extended	Enable	RTR	Information
Bus_0						
KMT-CoolantTempRaw (100h)	100h	1	□	☑	□	
KMTCoolantTempShown (200h)	200h	1	□	☑	□	

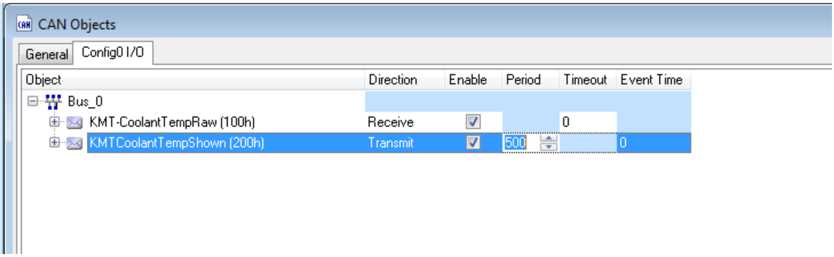
1. CAN-ID: 0x100 for the incoming raw value, 0x200 for the outgoing display value (with plateau).
2. DLC: Both CAN messages are 1 byte of lenght.
3. Enable: Yes.



Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order	Multiplexer
Bus_0							
KMT-CoolantTempRaw (100h)							
RealTemperature		8	0	0	□	Intel	None
KMTCoolantTempShown (200h)							
WithPlateau		8	0	0	□	Intel	None

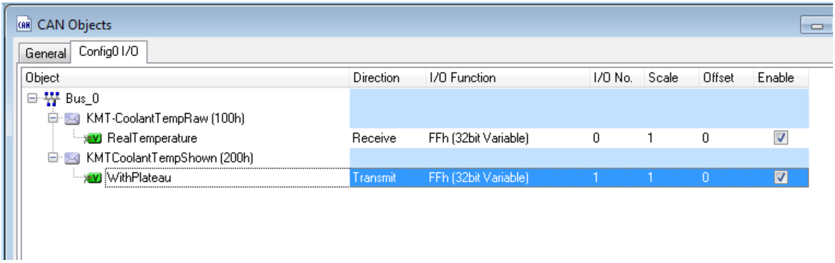
1. Bit Length: 8 (value range 256 sufficient).
2. Signed: no, always positive.

Activity: As in the previous example, please create a new configuration within the file (HW profile: PCAN-Router Pro) and import all CAN objects from the General tab:



Object	Direction	Enable	Period	Timeout	Event Time
Bus_0					
KMT-CoolantTempRaw (100h)	Receive	☑		0	
KMTCoolantTempShown (200h)	Transmit	☑	500		0

1. Direction: 0x100 is received, 0x200 is transmitted.
2. Enable: Yes.
3. Period: 500 ms (= cycle time for the transmit message carrying the converted coolant temperature signal).



Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
KMT-CoolantTempRaw (100h)						
RealTemperature	Receive	FFh (32bit Variable)	0	1	0	☑
KMTCoolantTempShown (200h)						
WithPlateau	Transmit	FFh (32bit Variable)	1	1	0	☑

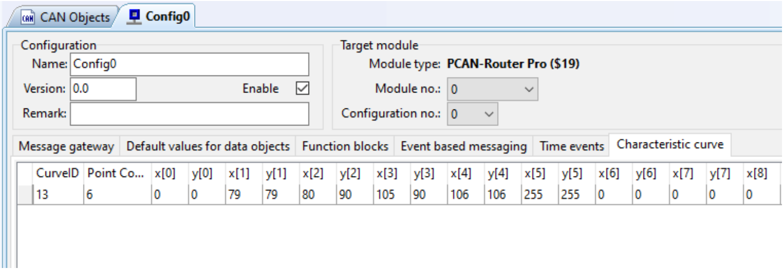
I/O-Function, I/O-Number: The signal RealTemperature from message 0x100 is transferred into variable #0, the modified result WithPlateau is transferred with message 0x200.

Information: 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=79	Y=79
Curve point 2	X=85	Y=90
Curve point 3	X=105	Y=90
Curve point 4	X=106	Y=106
Curve point 5	X=255	Y=255

Activity: These values are now entered as a characteristic curve. To do so, open the configuration window in the navigator (left edge of the main window), double-click Config0, then change to tab Characteristic curve, open the context menu there (right click) and choose menu item **Add Record.**

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



Configuration

Name: Config0

Version: 0.0 Enable ☑

Remark:

Target module

Module type: **PCAN-Router Pro ($19)**

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]	x[8]
13	6	0	0	79	79	80	90	105	90	106	106	255	255	0	0	0	0	0

1. Curve-ID: 13 (a number chosen at random).
2. Point Count: 6 (number of X/Y points on the curve).
3. Pairs of values 0..5: The Characteristic curve (values taken from the table above). Further entries are not used and contain 0 / 0.

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

Information: 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 subsequentially transmitted onto CAN. For this, a special function block Characteristic curve is needed, which handles that conversion.

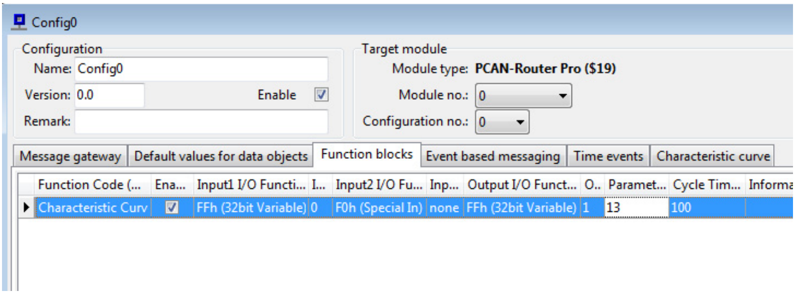
Activity: For creating a new function block, focus on tab Function blocks, open context menu (right click), and choose menu item **Add Record**.

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

Information: Basically, each function block has two inputs (operands) and one output (result), each of the three consisting of an I/O type and a I/O number. Additional there is a main switch (enable) and a cycle time (With what frequency is this block re-calculated, in ms).

Remark: In the function block Characteristic Curve the second input is always unused.

Activity: The function block is supplied as follows:



Config0

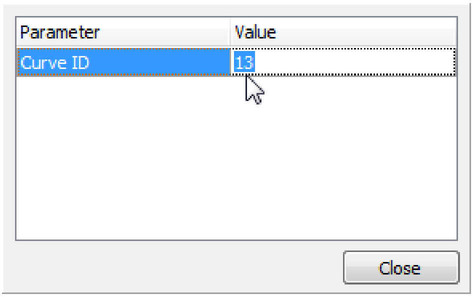
Configuration
Name: Config0
Version: 0.0 Enable ☑
Remark:

Target module
Module type: **PCAN-Router Pro ($19)**
Module no.: 0
Configuration no.: 0

Function Code (...	Ena...	Input1 I/O Functi...	I...	Input2 I/O Fu...	Inp...	Output I/O Funct...	O..	Paramet...	Cycle Tim...	Informa
Characteristic Curv	☑	FFh (32bit Variable)	0	F0h (Special In)	none	FFh (32bit Variable)	1	13	100	

1. Function Code: Characteristic curve (a special function block for this purpose).
2. Enable: Yes, this function block shall be active.

3. Input1: FF-32 bit variable, 0 (X value comes from 32 bit variable #0).
4. Input2: F0-Special In, none (= unused).
5. Output: FF-32 bit variable, 1 (Y result is written to 32 bit variable #1).
6. Parameter: 13 (number of the already defined characteristic curve). When clicking the field, the following dialog window appears.



Parameter	Value
Curve ID	13

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

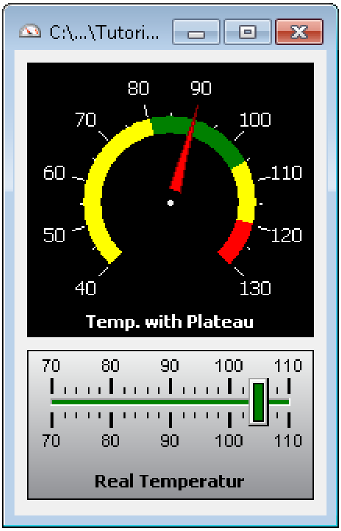
Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 4b to your PC.

Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

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

Information: With the help of a PCAN-Explorer (Part No. IPES-005028) with an **Instrument Panel** Add-in (Part No. IPES-005028), the values can be represented graphically.



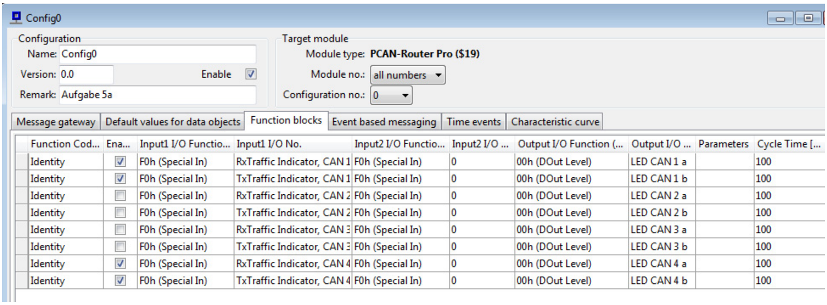
4.12 Exercise 5a: LED Activity on CAN Reception and Transmission

Information: Reception and transmission of CAN messages can be visualized. For this, the PCAN-Router Pro is equipped with two LEDs per channel. In this exercise, the LEDs shall indicate CAN activity, separated by direction Rx and Tx.

Activity: Create an empty configuration project file by using menu item **File -> New**.

Activity: Create a new configuration within the configuration project file using menu item **Edit -> New Configuration**.

Activity: Activate the configuration window (by clicking the icon Config0 at the left window edge). Select the tab Function Blocks and create a new entry for each LED. To do so, use menu item **Add Record** from the context menu:



Config0

Configuration
Name: Config0
Version: 0.0 Enable ☑
Remark: Aufgabe 5a

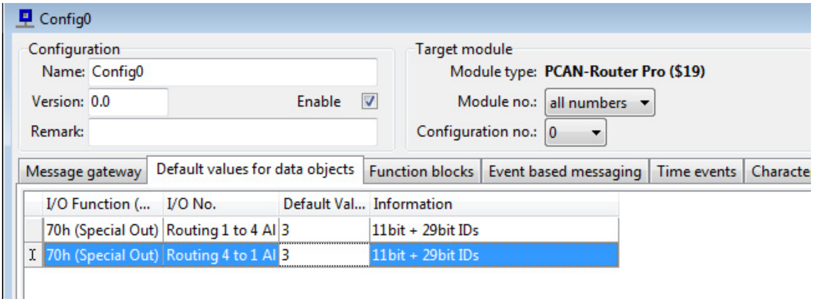
Target module
Module type: **PCAN-Router Pro ($19)**
Module no.: all numbers
Configuration no.: 0

Function Cod...	Ena...	Input1 I/O Functio...	Input1 I/O No.	Input2 I/O Functio...	Input2 I/O ...	Output I/O Function (...	Output I/O ...	Parameters	Cycle Time [...
Identity	☑	F0h (Special In)	RxTraffic Indicator, CAN 1	F0h (Special In)	0	00h (DOut Level)	LED CAN 1 a		100
Identity	☑	F0h (Special In)	TxTraffic Indicator, CAN 1	F0h (Special In)	0	00h (DOut Level)	LED CAN 1 b		100
Identity	□	F0h (Special In)	RxTraffic Indicator, CAN 2	F0h (Special In)	0	00h (DOut Level)	LED CAN 2 a		100
Identity	□	F0h (Special In)	TxTraffic Indicator, CAN 2	F0h (Special In)	0	00h (DOut Level)	LED CAN 2 b		100
Identity	□	F0h (Special In)	RxTraffic Indicator, CAN 3	F0h (Special In)	0	00h (DOut Level)	LED CAN 3 a		100
Identity	□	F0h (Special In)	TxTraffic Indicator, CAN 3	F0h (Special In)	0	00h (DOut Level)	LED CAN 3 b		100
Identity	☑	F0h (Special In)	RxTraffic Indicator, CAN 4	F0h (Special In)	0	00h (DOut Level)	LED CAN 4 a		100
Identity	☑	F0h (Special In)	TxTraffic Indicator, CAN 4	F0h (Special In)	0	00h (DOut Level)	LED CAN 4 b		100

1. Function Code: Identity (a variable’s content is copied into another).
2. Enable: Yes, this function block shall be active.
3. Input 1: System variable Special In, TrafficIndicator (active for 100 ms).
4. Input 2: F0-Special In and none (= unused).
5. Output: Dout-Level and the appropriate LED number.
6. Parameter: not necessary.
7. Cycle time: 100 (refresh of LED status every 100 ms, it is not the lucent period).

Remark: The LEDs lucent period of 100 ms is hardcoded witin the firmware and can’t be changed. The identity Function blocks cycle time therefore cannot be used to affect the LED behavior.

Activity: Last thing to do is the declaration of routing all the messages incoming at CAN-1 to CAN-4 and vice versa. The appropriate settings are done in tab Default values for data objects:



Configuration

Name: Config0

Version: 0.0 Enable ☑

Remark:

Target module

Module type: **PCAN-Router Pro ($19)**

Module no.: all numbers

Configuration no.: 0

I/O Function (...	I/O No.	Default Val...	Information
70h (Special Out)	Routing 1 to 4 Al	3	11bit + 29bit IDs
70h (Special Out)	Routing 4 to 1 Al	3	11bit + 29bit IDs

1. I/O-Function: SpecialOut (one group of device functions).
2. I/O-No: Routing 1 to 4 All and Routing 4 to 1 All.
3. Default value: 3 = Sum of 1 (only routes 11 bit IDs) and 2 (only routes 29 bit IDs).
4. Information: Description of what this line does.

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 5a to your PC.

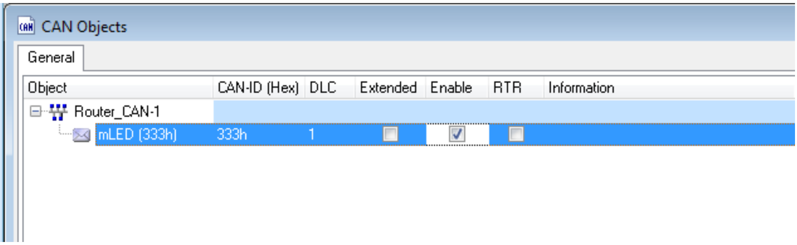
Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

Result: At reception or transmission of a CAN message the assigned LED will be lit for 100 ms.

4.13 Exercise 5b: Controlling LED Manually or Conditionally

Information: From bus Router_CAN-1 a 1-byte message mLED shall be received, containing an analog value LedByte (0..255). When the analog value exceeds 126, the LED should be on. If the value is below 127, the LED remains off.

Activity: Create an empty configuration project file by using menu item **File -> New**. Globally define a receive message mLED (0x333) containing an 8 bit CAN signal LedByte:

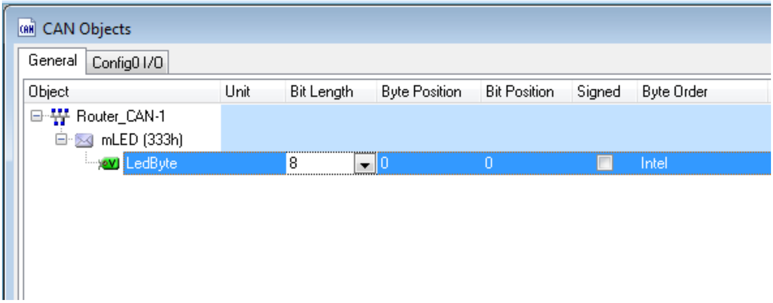


CAN Objects

General

Object	CAN-ID (Hex)	DLC	Extended	Enable	RTR	Information
Router_CAN-1						
mLED (333h)	333h	1	□	☑	□	

1. CAN-ID: 0x333.
2. DLC: 1 (the CAN messages has a length of 1 byte).
3. Enable: Yes.



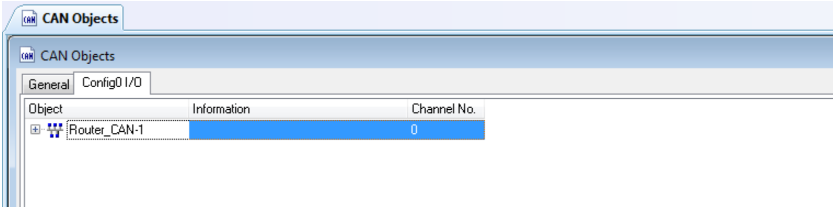
CAN Objects

General | Config0 I/O

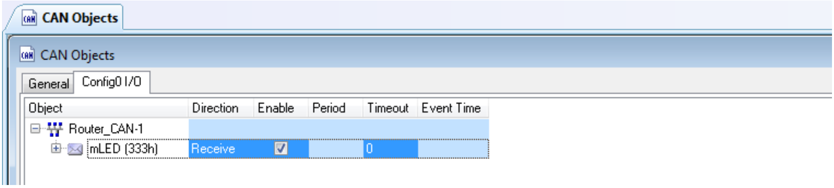
Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order
Router_CAN-1						
mLED (333h)						
LedByte		8	0	0	□	Intel

1. Bit Length: 8 (256 values sufficient).
2. Signed: no, never negative.

Activity: Create a new configuration within the project file: for this, use menu item **Edit -> New Configuration**. Then import the receive message into the new configuration and finally enter the parameters:

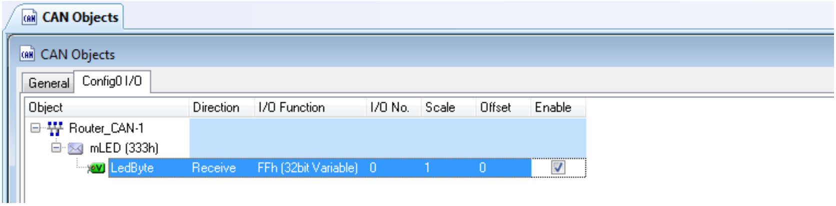


Object	Information	Channel No.
Router_CAN-1		0



Object	Direction	Enable	Period	Timeout	Event Time
Router_CAN-1					
mLED (333h)	Receive	☑		0	

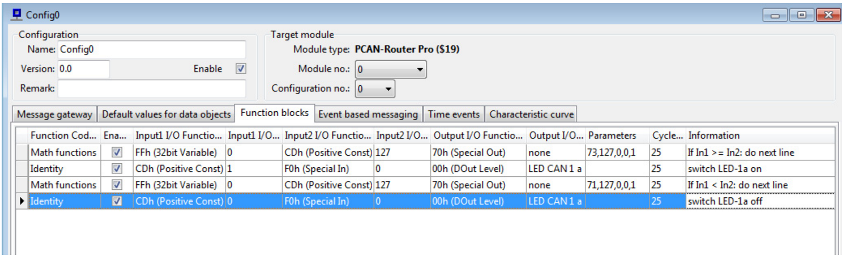
1. Direction: 0x333 will come in (Receive).
2. Enable: Yes.



Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Router_CAN-1						
mLED (333h)						
LedByte	Receive	FFh (32bit Variable)	0	1	0	☑

I/O-Function, I/O-Number: The signal LedByte from the message 0x333 is transferred into the variable #0.

Activity: Open the configuration window (by clicking icon Config0 at the left window edge) and select tab Function blocks. From the context menu, use **Add Record** to create 4 new Function blocks:



Config0

Configuration

Name: Config0

Version: 0.0 Enable ☑

Remark:

Target module

Module type: **PCAN-Router Pro ($19)**

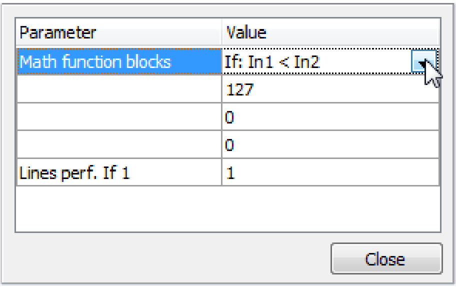
Module no.: 0

Configuration no.: 0

Function Cod...	Ena...	Input1 I/O Functio...	Input1 I/O...	Input2 I/O Functio...	Input2 I/O...	Output I/O Functio...	Output I/O...	Parameters	Cycle...	Information
Math functions	☑	FFh (32bit Variable)	0	CDh (Positive Const)	127	70h (Special Out)	none	73,127,0,0,1	25	If In1 >= In2: do next line
Identity	☑	CDh (Positive Const)	1	F0h (Special In)	0	00h (DOut Level)	LED CAN 1 a		25	switch LED-1a on
Math functions	☑	FFh (32bit Variable)	0	CDh (Positive Const)	127	70h (Special Out)	none	71,127,0,0,1	25	If In1 < In2: do next line
Identity	☑	CDh (Positive Const)	0	F0h (Special In)	0	00h (DOut Level)	LED CAN 1 a		25	switch LED-1a off

Line 1:

1. Function block: Math Function (IF).
2. Enable: Yes.
3. Input#1 I/O-Function: 32 bit variable.
4. Input#1 I/O-Number: 0.
5. Input#2 I/O-Function: constant.
6. Input#2 I/O-Number: value is **127.**
7. Output I/O-Function: unused (do not change).
8. Output I/O-Number: unused (do not change).
9. Parameter: Compare, whether **In1 is greater equal 127**: If YES, then execute next line (switch on LED). If NO, then skip 1 line. Following dialog window appears:



Parameter	Value
Math function blocks	If: In1 < In2
	127
	0
	0
Lines perf. If 1	1

The type of Math function block may chosen from a list: click – slowly - the Value field twice for opening the list.

Parameters below must be entered directly. Confirm with **Close**.

10. Cycle: This Function block is processed every 25 ms.

Line 2:

1. Function block: Identity (copy 1:1).
2. Enable: Yes.
3. Input#1 I/O-Function: constant.
4. Input#1 I/O-Number: value is **1.**
5. Input#2 I/O-Function: unused (do not change).
6. Input#2 I/O-Number: unused (do not change).
7. Output I/O-Function: is written to the following hardware resource.
8. Output I/O-Number: LED CAN-1a.
9. Parameter: none.
10. Cycle: this Function block is processed every 25 ms.

Line 3:

1. Function block: Math Function (ELSE respectively an IF with reversed condition).
2. Enable: Yes.
3. Input#1 I/O-Function: 32 bit variable.
4. Input#1 I/O-Number: #0.
5. Input#2 I/O-Function: constant.
6. Input#2 I/O-Number: value is **127.**
7. Output I/O-Function: unused (do not change).
8. Output I/O-Number: unused (do not change).

9. Parameter: Compare, whether **In1 is less than 127**: If YES, then execute the next line (switch off LED). If NOT, then skip 1 line.
10. Cycle: this Function block is processed every 25 ms.

Line 4:

1. Function block: Identity (copy 1:1).
2. Enable: Yes.
3. Input#1 I/O-Function: constant.
4. Input#1 I/O-Number: value is **0**.
5. Input#2 I/O-Function: unused (do not change).
6. Input#2 I/O-Number: unused (do not change).
7. Output I/O-Function: is written to the following hardware resource.
8. Output I/O-Number: LED CAN-1a.
9. Parameter: none.
10. Cycle: this Function block is processed every 25 ms.

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 5b to your PC.

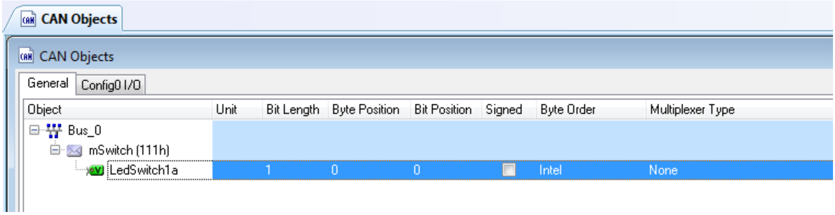
Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

Result: When a value up to **126 (0x7E)** is received, LED-1a remains off. From a value greater than or equal to **127 (0x7F)** the LED is switched on.

4.14 Exercise 5c: Controlling LED Externally

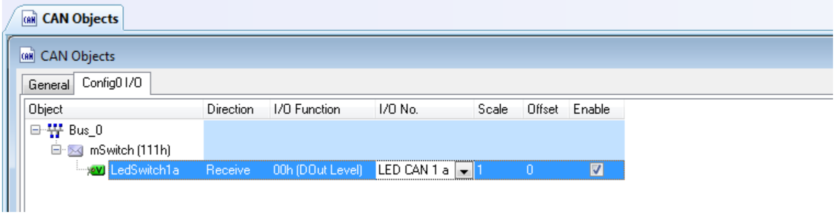
Information: For remote switching of a LED it is sufficient to receive a 1 bit signal from CAN. This signal is then assigned to the internal resource Dout-Level -> LED CAN-1a.

Activity: Create an empty configuration file using menu item **File -> New**. Define a reception message mSwitch (0x111) carrying a 1 bit CAN signal:



Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order	Multiplexer Type
Bus_0							
mSwitch (111h)							
LedSwitch1a		1	0	0	□	Intel	None

Activity: Create a new configuration within the file, import the globally defined CAN message and enter the parameters:



Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
mSwitch (111h)						
LedSwitch1a	Receive	00h (DOut Level)	LED CAN 1 a	1	0	☑

1. Direction: Receive (since switch value comes in from CAN).
2. I/O-Function: DOut-Level (internal hardware resource).
3. I/O-Number: LED CAN-1a.

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 5c to your PC.

Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

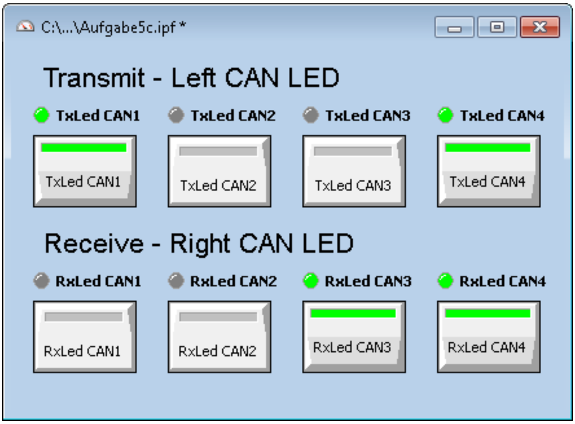
Result: Via transmitting the corresponding bit the LED can be switched on or off.

Tip: When expanding the example to 8 LEDs and you own a PCAN-Explorer (Part No. IPES-005028), a symbol file can be created (see the appropriate manual) containing each switch value in symbolic form.

```
{SEND}

[mSwitch]
ID=111h
Picture=---b---a ---d---c ---f---e ---h---g
a=TxLedCan1 bit
b=RxLedCan1 bit
c=TxLedCan2 bit
d=RxLedCan2 bit
e=TxLedCan3 bit
f=RxLedCan3 bit
g=TxLedCan4 bit
h=RxLedCan4 bit
```

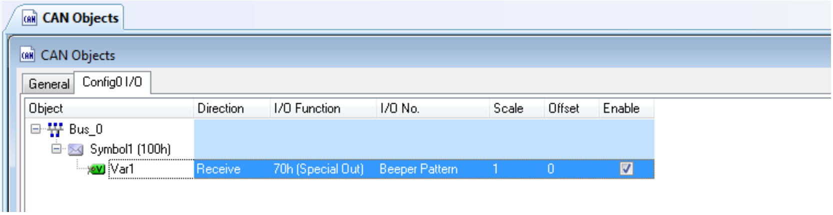
Remark: With the help of a PCAN-Explorer (Part No. IPES-005028) and the Instruments Panel Add-in (Part No. IPES-005028), a graphical interface for the visualization of the switch can easily be created.



4.15 Exercise 5d: Controlling Beeper (Continuous Tone)

Information: Similar to the activation of a LED, a continuous tone (or a short pattern) from the internal beeper can be also be controlled externally.

Activity: Configuring of the PCAN-Router Pro in a way that the 32 bit receive signal (from CAN) is written directly to firmware resource BeeperPattern:



Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
Symbol1 (100h)						
Var1	Receive	70h (Special Out)	Beeper Pattern	1	0	☑

1. I/O Function: Special Out.

2. I/O Number: Beeper Pattern.

3. Enable: Yes.

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 5d to your PC.

Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

Information: When creating a tone pattern, user must calculate a 32 bit value first.

Structure of this value is as follows:

> The upper 24 bit will define the tone pattern.
> The lower 5 bit will declare, how many of these bits are already played.
> In between there are 2 bits unused and 1 bit decides whether the pattern is played endlessly (1=continuous) or only once (0=one-shot).

Playback speed is 100 ms per bit.

Example: Programming of a continuous tone:

```
10000000.00000000.00000000.00100001
mmmmmmmm.mmmmmmmm.mmmmmmmm          = 1* 100ms tone
                          .uu       = unused
                             w      = 0=OneShot; 1=endless repeat
                              lllll = Pattern length: 1 bit
```

Duration: 24 *100 ms = **2,4 s**

Pattern: *0x80. 0x00. 0x00. 0x21*

Activity: Send this pattern to the PCAN-Router Pro using a PCAN-View or PCAN-Explorer:
0x100-8-"21 00 00 80 00 00 00 00", 0

For switching it off, send a pattern with all bits set to 0:
0x100-8-"00 00 00 00 00 00 00 00", 0

4.16 Exercise 5e: Controlling the Beeper (Tone Sequence)

Information: Based on the configuration from exercise 5d, different tone pattern may be generated.

Example: Programming the SMS Morse code:

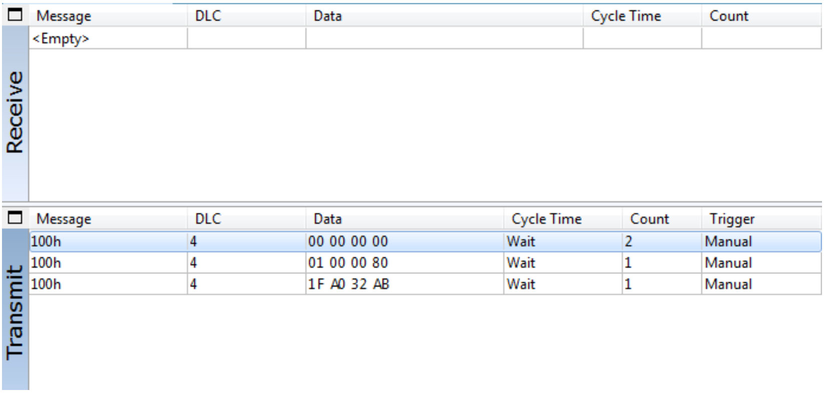
```
10000000.00000000.00000000.00100001
mmmmmmm                             = S (6 bit used)
      mm.mmmmmmm                    = M (8 bit used)
               mm.mmmmm             = S (6 bit used)
                      mmmmm.        = Pause=4*100 ms (4 bit used)
                           uu       = unused
                             w      = 0=OneShot, 1=endless repeat
                              lllll = Pattern length: 24 bit
```

Duration: 24 *100 ms = 2,4 s

Pattern: *0xAB. 0x32. 0xA0. 0x1F*

Activity: Send this pattern to the PCAN-Router Pro using a PCAN-View or PCAN-Explorer:
0x100-8-"1F A0 32 AB 00 00 00 00", 0

For switching it off, send a pattern with all bits set to 0:
0x100-8-"00 00 00 00 00 00 00 00", 0



Receive

Message	DLC	Data	Cycle Time	Count
<Empty>				

Transmit

Message	DLC	Data	Cycle Time	Count	Trigger
100h	4	00 00 00 00	Wait	2	Manual
100h	4	01 00 00 80	Wait	1	Manual
100h	4	1F A0 32 AB	Wait	1	Manual

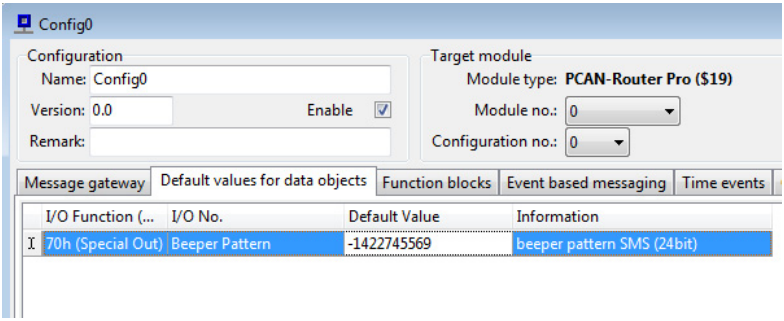
Tip: The 32 bit value for a beep pattern (like the one from exercise 5e) can be stored as a constant value in tab Default values for data objects. To do so, a 32 bit variable is supplied with the calculated bit pattern and remains unchanged further on. When defining several different patterns here, then incoming CAN selector or a calculation result can decide which pattern to be played.

Remark 1: As the current pattern is bigger than 31 bit, the assignment to a variable must be done in **2's complement** (since the PPCAN-Editor only accepts SIGNED variables in tab Default values for data object).

Open Windows accessory Calculator:

- Switch display mode to **HEX**
- Adjust width to **Dword** (=32 Bit)
- Enter the number: **AB32A01F**
- Change the sign: **+/-**
- Switch display mode to **DEZ**
- Again, change the sign: **+/-**
- Result = **-1422745569** (Enter this value into the PPCAN-Editor)

Open the configuration window in the navigator (left edge of the main window) by double-clicking Config0. Change to the tab Default values for data objects:



Config0

Configuration

Name: Config0

Version: 0.0 Enable ☑

Remark:

Target module

Module type: **PCAN-Router Pro ($19)**

Module no.: 0

Configuration no.: 0

I/O Function (...	I/O No.	Default Value	Information
70h (Special Out)	Beeper Pattern	-1422745569	beeper pattern SMS (24bit)

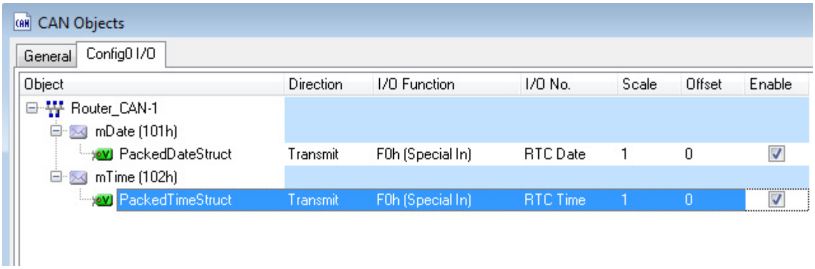
1. I/O-Function: Special Out.
2. I/O-Number: Beeper Pattern.
3. Default Value: -1422745569.

Remark 2: The continuous tone as explained in exercise 5d must be entered as value **-2147483615** in the PPCAN-Editor’s default values.

4.17 Exercise 6a: Reading Date and Time (Hardware Diagnostics)

Information: Date and time in the PCAN-Router Pro are supplied by a hardware RealTimeClock. The values can be read from the internal variables and subsequently be transmitted via CAN, e.g. for display purpose.

Activity: Definition of two transmit messages (length = 4 each), which are cyclically filled by the PCAN-Router Pro with date and time values and transmitted on CAN bus Router_CAN-1.



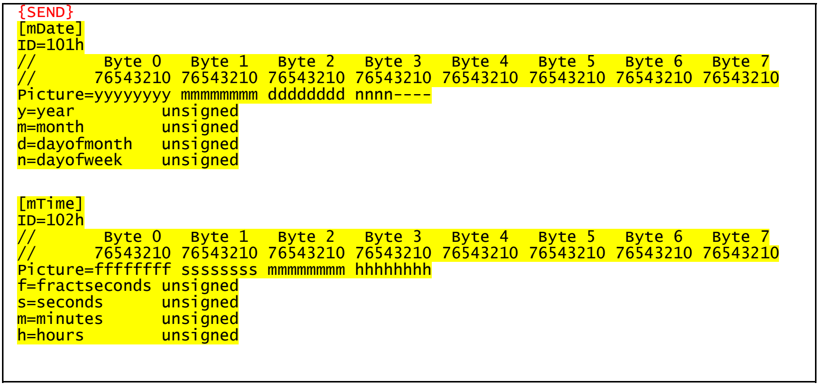
Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Router_CAN-1						
mDate (101h)						
PackedDateStruct	Transmit	F0h (Special In)	RTC Date	1	0	☑
mTime (102h)						
PackedTimeStruct	Transmit	F0h (Special In)	RTC Time	1	0	☑

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 6a to your PC.

Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

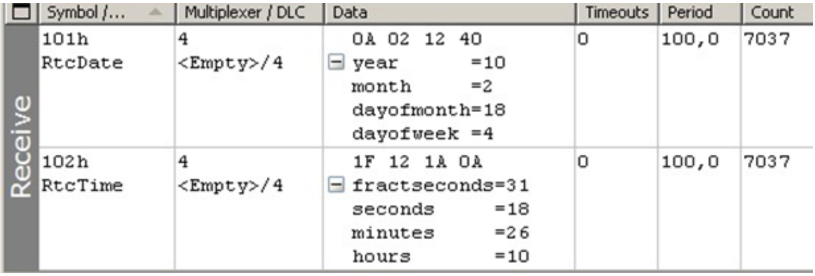
Information: With the help of a PCAN-Explorer (Part No. IPES-005028), the packed structures of the RTC can be displayed in plain text, when decoded by means of a symbol file.



```
{SEND}
[mDate]
ID=101h
//       Byte 0   Byte 1   Byte 2   Byte 3   Byte 4   Byte 5   Byte 6   Byte 7
//      76543210 76543210 76543210 76543210 76543210 76543210 76543210 76543210
Picture=yyyyyyyy mmmmmmmm dddddddd nnnn----
y=year         unsigned
m=month        unsigned
d=dayofmonth   unsigned
n=dayofweek    unsigned

[mTime]
ID=102h
//       Byte 0   Byte 1   Byte 2   Byte 3   Byte 4   Byte 5   Byte 6   Byte 7
//      76543210 76543210 76543210 76543210 76543210 76543210 76543210 76543210
Picture=ffffffff ssssssss mmmmmmmm hhhhhhhh
f=fractseconds unsigned
s=seconds      unsigned
m=minutes      unsigned
h=hours        unsigned
```

By applying this symbol file, messages transmitted from the PCAN-Router Pro are decoded as follows:



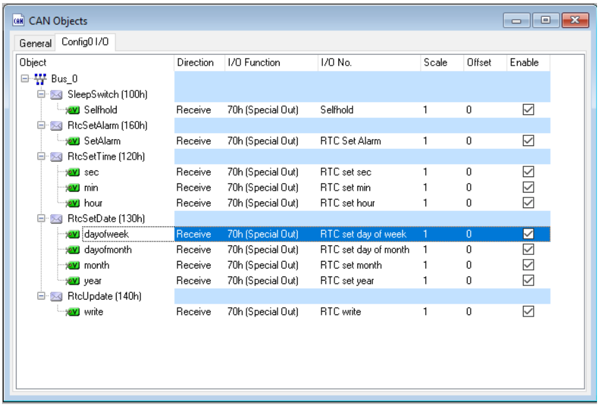
Receive

Symbol /...	Multiplexer / DLC	Data	Timeouts	Period	Count
101h RtcDate	4 <Empty>/4	0A 02 12 40 ⊟ year =10 month =2 dayofmonth=18 dayofweek =4	0	100,0	7037
102h RtcTime	4 <Empty>/4	1F 12 1A 0A ⊟ fractseconds=31 seconds =18 minutes =26 hours =10	0	100,0	7037

4.18 Exercise 6b: Setting Date and Time (Hardware Diagnostics)

Information: For adjusting the RealTimeClock, all elements for date and time are supplied in separate values, there is no packed structure like when reading time and date (see exercise 6a).

Definition: 3 separate messages are to be implemented for date, time, and the activation command. Each firmware variables therein is 8 bit wide:



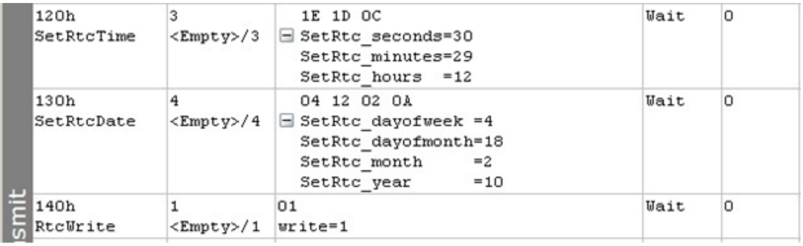
Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
SleepSwitch (100h)						
Selfhold	Receive	70h (Special Out)	Selfhold	1	0	☑
RtcSetAlarm (160h)						
SetAlarm	Receive	70h (Special Out)	RTC Set Alarm	1	0	☑
RtcSetTime (120h)						
sec	Receive	70h (Special Out)	RTC set sec	1	0	☑
min	Receive	70h (Special Out)	RTC set min	1	0	☑
hour	Receive	70h (Special Out)	RTC set hour	1	0	☑
RtcSetDate (130h)						
dayofweek	Receive	70h (Special Out)	RTC set day of week	1	0	☑
dayofmonth	Receive	70h (Special Out)	RTC set day of month	1	0	☑
month	Receive	70h (Special Out)	RTC set month	1	0	☑
year	Receive	70h (Special Out)	RTC set year	1	0	☑
RtcUpdate (140h)						
write	Receive	70h (Special Out)	RTC write	1	0	☑

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 6b to your PC.

Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

Result: If these (properly supplied) CAN messages are now sent to PCAN-Router Pro, the internal RealTimeClock may be adjusted to an actual value:



120h
SetRtcTime
	3
	<Empty>/3
		1E 1D 0C
		⊟ SetRtc_seconds=30
		SetRtc_minutes=29
		SetRtc_hours =12
			Wait	0

130h
SetRtcDate
	4
	<Empty>/4
		04 12 02 0A
		⊟ SetRtc_dayofweek =4
		SetRtc_dayofmonth=18
		SetRtc_month =2
		SetRtc_year =10
			Wait	0

140h RtcWrite	1 <Empty>/1	01 write=1	Wait	0

Remark: The shown message will set the RTC to
Thursday, 18. February 2010, 12:29:30.

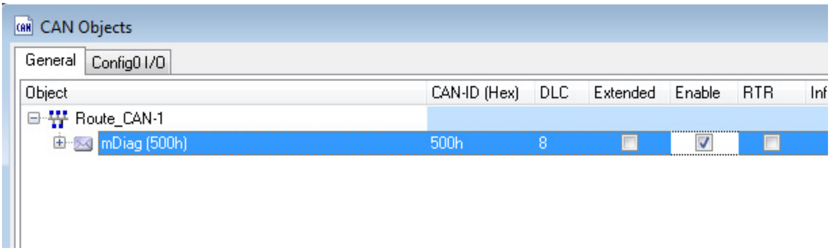
4.19 Exercise 6c: Reading the Module ID (Hardware Diagnostics)

Information: The module ID is a 4 bit value, which is set to 0 by default, but can be changed inside of the PCAN-Router Pro 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 unexpected behavior of your freshly edited configuration, one of the first steps in trouble-shooting is determination of the module ID. It sometimes happens that a configuration is edited again and again without success, since each time a different one is executed by the PCAN-Router Pro.

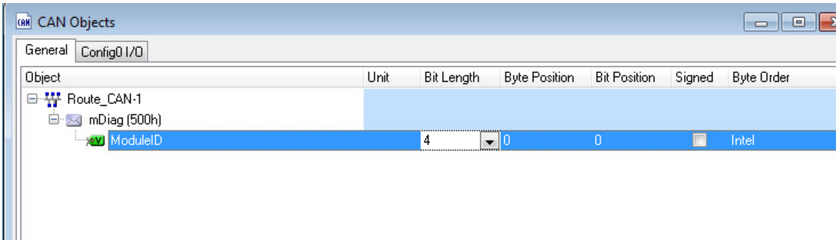
Definition: Transmitting a CAN message mDiag with ID 0x500, length 8 bytes on the bus Router_CAN-1. This message contains the 4 bit signal ModuleID, which displays the current position of the module ID rotary switch.

Activity: Create a new CAN message mDiag at the General tab (length = 8 bytes):



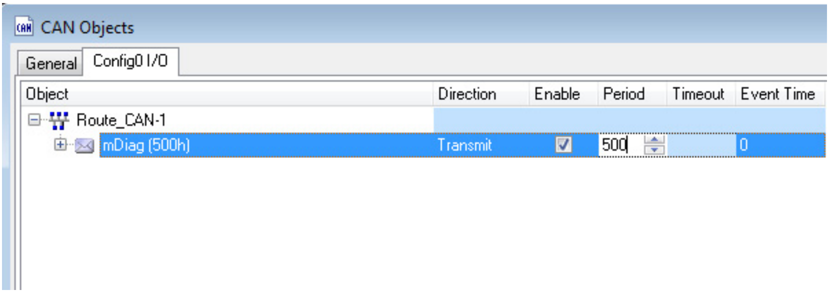
Object	CAN-ID (Hex)	DLC	Extended	Enable	RTR	Inf
Route_CAN-1						
mDiag (500h)	500h	8	□	☑	□	

Creating the new CAN signal ModuleID (4 Bit, unsigned) at the General tab:



Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order
Route_CAN-1						
mDiag (500h)						
ModuleID		4	0	0	□	Intel

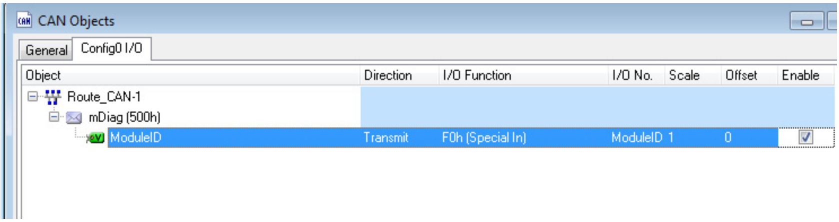
Importing of the message and signal into the configuration (cycle time 500 ms):



CAN Objects

Object	Direction	Enable	Period	Timeout	Event Time
Route_CAN-1					
mDiag (500h)	Transmit	☑	500		0

Supplying the signal with internal variable ModuleID of the PCAN-Router Pro:



CAN Objects

Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Route_CAN-1						
mDiag (500h)						
ModuleID	Transmit	F0h (Special In)	ModuleID	1	0	☑

1. I/O-Function: F0-Special In.
2. I/O-Number: ModuleID.
3. Enable: Yes.

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 6c to your PC.

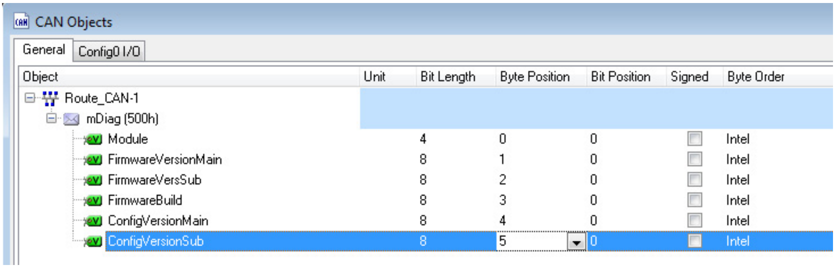
Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

Reaction: The CAN message mDiag (with ID 0x500) carrying the module ID is transmitted.

Information: 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).

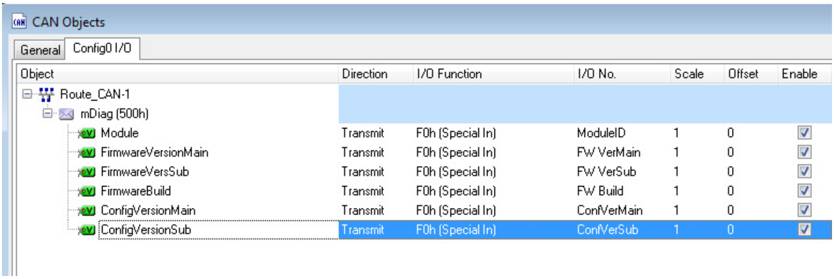
4.20 Exercise 6d: Reading Firmware Version and Configuration Version (Hardware Diagnostics)

Activity: The mDiag message will get 5 additional 8 bit variables: Firmware version (3 numbers) and configuration version (2 numbers):



Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order
Route_CAN-1						
mDiag (500h)						
Module		4	0	0	☐	Intel
FirmwareVersionMain		8	1	0	☐	Intel
FirmwareVersSub		8	2	0	☐	Intel
FirmwareBuild		8	3	0	☐	Intel
ConfigVersionMain		8	4	0	☐	Intel
ConfigVersionSub		8	5	0	☐	Intel

Assignment of data sources:



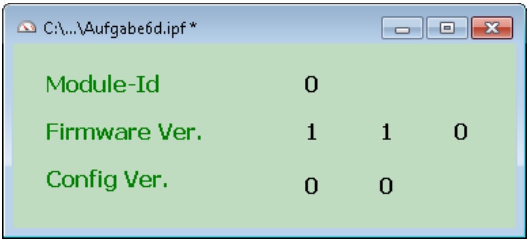
Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Route_CAN-1						
mDiag (500h)						
Module	Transmit	F0h (Special In)	ModuleID	1	0	☑
FirmwareVersionMain	Transmit	F0h (Special In)	FW VerMain	1	0	☑
FirmwareVersSub	Transmit	F0h (Special In)	FW VerSub	1	0	☑
FirmwareBuild	Transmit	F0h (Special In)	FW Build	1	0	☑
ConfigVersionMain	Transmit	F0h (Special In)	ConfVerMain	1	0	☑
ConfigVersionSub	Transmit	F0h (Special In)	ConfVerSub	1	0	☑

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 6d to your PC.

Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

Remark: Using a PCAN-Explorer (Part No. IPES-005028) with the Instruments Panel Add-in (Part No. IPES-005028) installed, you can visualize all signals received from the PCAN-Router Pro by means of a symbol file:

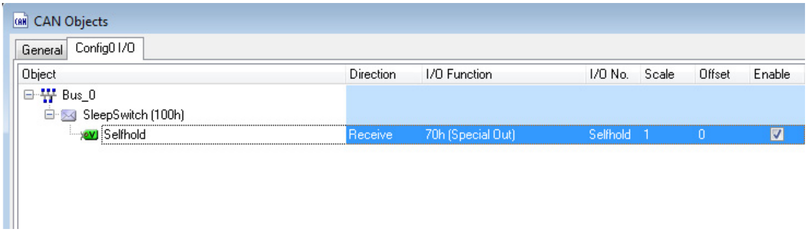


Module-Id	0		
Firmware Ver.	1	1	0
Config Ver.	0	0	

4.21 Exercise 7a: Sleep and Wake-Up via CAN

Information: PCAN-Router Pro is usually equipped with 4 transceivers capable of waking the device. If the device is in Sleep mode, any incoming CAN message will wake the device and set it to the normal operation mode.

Activity: Definition of a receive message, which sets the internal variable SELFHOLD to 0. When receiving this message, the PCAN-Router Pro enters the so called Sleep Mode (e.g. useful for saving vehicle’s battery capacity):



CAN Objects

General | Config0 I/O

Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
SleepSwitch (100h)						
Selfhold	Receive	70h (Special Out)	Selfhold	1	0	☑

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 7a to your PC.

Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN bus.

Result: If a message ID 0x100 carrying the signal Selfhold = 0 is received, then the PCAN-Router Pro enters Sleep mode immediately. Any subsequent message will cause a Wake-Up of the device (Status LED is blinking).

Information: After waking up, the internal variable SELFHOLD is initialized to "1". Therefore the PCAN-Router Pro stays awake all the time (until configured otherwise).

4.22 Exercise 7b: Sleep and Wake-Up via External Pin

Information: A second method for waking up the PCAN-Router Pro is connecting pin 4 of the D-Sub sockets 3 or 4 with Vbat (8..26 V). The vehicle's wire ignition (carrying that voltage when driver's key is present and turned) is suitable for this purpose.

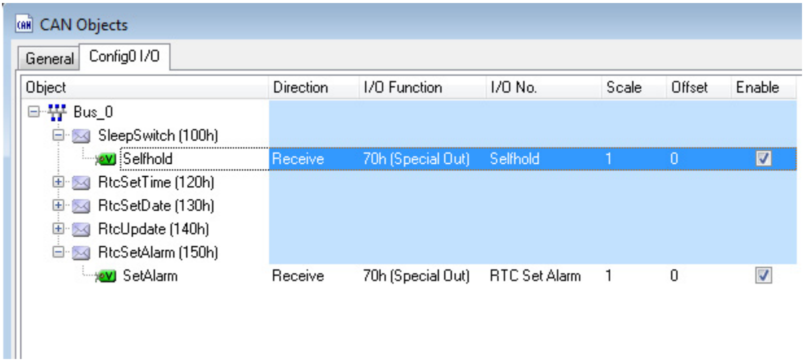
4.23 Exercise 7c: Sleep and Wake-Up via Timed Alarm

Information: A third method for waking up the PCAN-Router Pro is setting the alarm clock (feature of the internal real time clock RTC).

Remark: Assuming the current time to be **Thursday, February 18th, 2010, 12:29:30** o'clock, then a suitable alarm time could be **Thursday, February 18th, 2010, 12:30:00** o'clock for example. Setting the clock was already demonstrated in *4.18 Exercise 6b: Setting Date and Time (Hardware Diagnostics) on page 66*.

Information: For this task the receive messages **RtcSetAlarm** (ID = 0x160) for setting the wake-up time (length = 4 bytes) and **SleepSwitch** (ID = 0x100) for setting the PCAN-Router Pro to sleep mode are required (see *Exercise 7a: Sleep and Wake-Up via CAN on page 71*).

Activity: Load the configuration file **7c.** If you want to set a different wake-up time for testing, please refer to *4.18 Exercise 6b: Setting Date and Time (Hardware Diagnostics) on page 66*.



Object	Direction	I/O Function	I/O No.	Scale	Offset	Enable
Bus_0						
SleepSwitch (100h)						
Selfhold	Receive	70h (Special Out)	Selfhold	1	0	☑
RtcSetTime (120h)						
RtcSetDate (130h)						
RtcUpdate (140h)						
RtcSetAlarm (150h)						
SetAlarm	Receive	70h (Special Out)	RTC Set Alarm	1	0	☑

Information: At this point, all the configuration work for solving the exercise is done.

Activity: Save the configuration file as project exercise 7c to your PC.

Activity: Transmit (upload) the configuration to the PCAN-Router Pro via CAN.

Information: After setting the alarm time 12:30:00 o'clock PCAN-Router Pro must be set to Sleep mode first by setting Selfhold = 0 (send message **0x100** for this).

Result: As soon as the message ID 0x100 with signal Selfhold = 0 is received, the PCAN-Router Pro enters Sleep mode immediately. When reaching the programmed alarm time, the device wakes up.

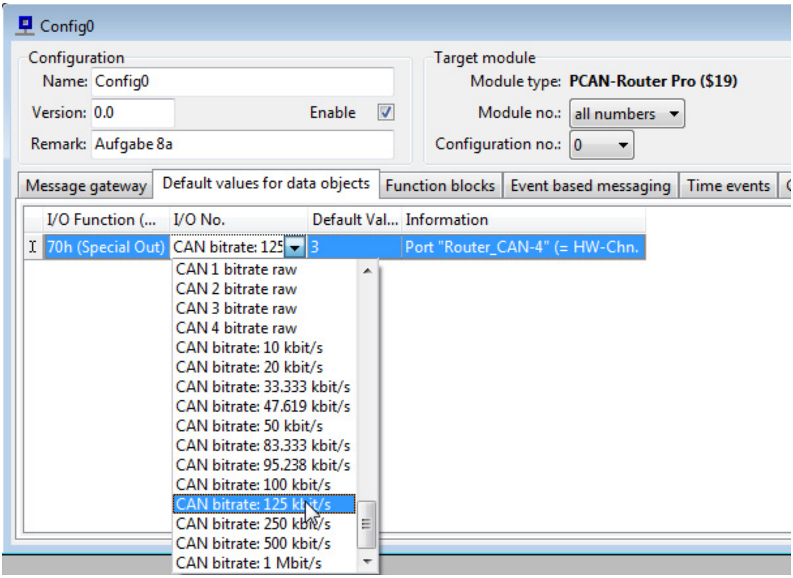
4.24 Exercise 8a: Changing the Bit Rate

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

Interface-Type	CAN-Transceiver	Default bit rate	Wake-Up capability
HS	TJA-1041 (default)	500 kbit/s	yes
HS opto[3]	TJA-1040	500 kbit/s	no
HS	TJA-1040	500 kbit/s	no
HS	82C251	500 kbit/s	no
LS-DW	TJA-1054	125 kbit/s	yes
LS-SW	TH-8056	33,3 kbit/s	yes

These default bit rates may be changed by adding appropriate entries in the configuration. On the **Default values for data objects** tab, add one record per channel to be modified and fill in the desired transmission speeds:

[3] Query for availability.



Configuration
Name: Config0
Version: 0.0 Enable ☑
Remark: Aufgabe 8a

Target module
Module type: PCAN-Router Pro ($19)
Module no.: all numbers
Configuration no.: 0

I/O Function (...	I/O No.	Default Val...	Information
70h (Special Out)	CAN bitrate: 125	3	Port "Router_CAN-4" (= HW-Chn.

- **I/O No.**: Enter the desired bit rate here.
- **Default Value**: CAN channel number (0…3).

Remark: As PPCAN-Editor (running on your PC) doesn't know about the transceiver types installed in your PCAN-module, it will offer all usual bit rates. Please take care that the equipped transceivers support the settings. E.g. TH-8056 does not support bit rates beyond 83.3 kbit/s, whereas TJA-1040 does not support bit rates below 40 kbit/s.

4.25 Exercise 9a: Transmitting a Multiplexer Message Automatically

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

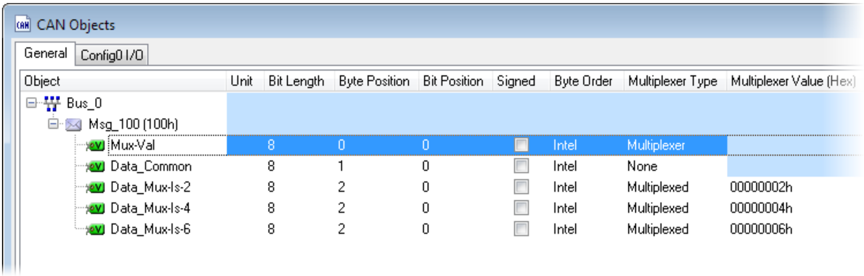
Definition: The example message has the following key parameters:

- 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	



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:

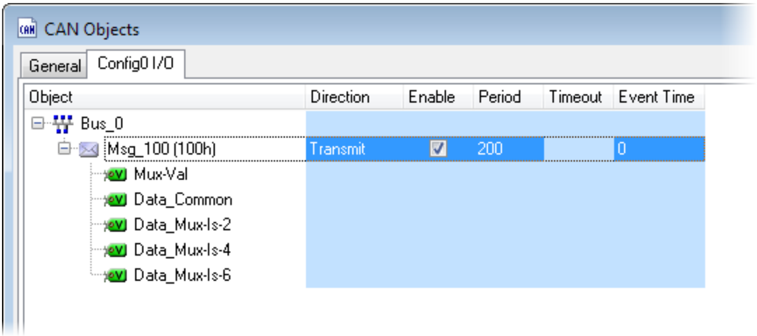
- **Multiplexer**: The variable contains the multiplexer value (data type Unsigned). This multiplexer type may only be used once within a message and must be placed before the variable definitions of the Multiplexed type in the list.
- **None**: This variable is used in all transmit messages, independently 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. The 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 4.1 on page 13).

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

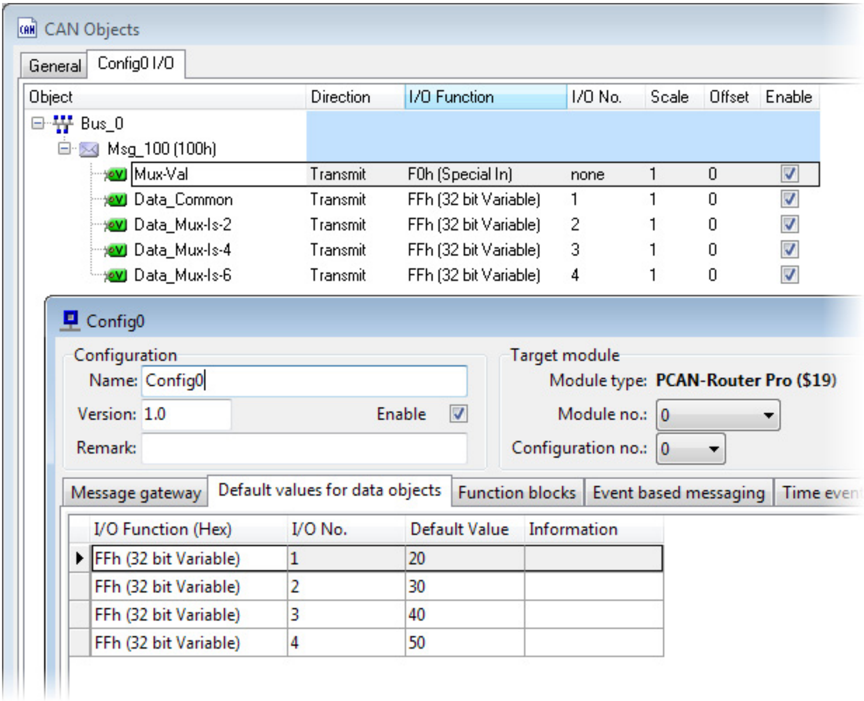
Definition: In the configuration Config0, now some settings for the CAN message and the contained variables must be done, in the **CAN Objects** window as well as in the **Config0** window.



CAN Objects

General | Config0 I/O

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					



CAN Objects

General | Config0 I/O

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	☑

Config0

Configuration

Name: Config0

Version: 1.0 Enable ☑

Remark:

Target module

Module type: **PCAN-Router Pro ($19)**

Module no.: 0

Configuration no.: 0

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

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.

4.26 Exercise 9b: 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 9a.

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

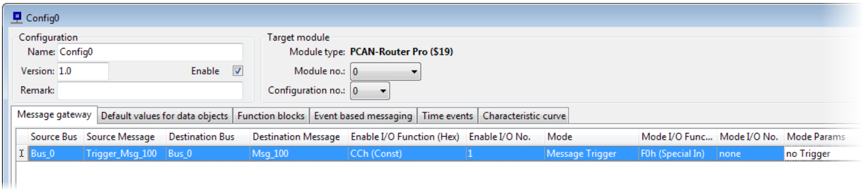
- CAN ID: 1FFh (Trigger_Msg_100)
- Length: 1 byte
- Bit assignment (variables):

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 3c (on page 36), 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-Router Pro ($19)**

Module no.: 0

Configuration no.: 0

Source Bus	Source Message	Destination Bus	Destination Message	Enable I/O Function (Hex)	Enable I/O No.	Mode	Mode I/O Func...	Mode I/O No.	Mode Params
Bus_0	Trigger_Msg_100	Bus_0	Msg_100	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.