

MU-Thermocouple1 CAN

PPCAN-Editor Configuration Tutorial



Relevant products

Product Name	Model	Part number
MU-Thermocouple1 CAN (Measuring range J)	Metal-cased measuring unit with 8 measuring channels	IPEH-002205-J
MU-Thermocouple1 CAN (Measuring range K)	Metal-cased measuring unit with 8 measuring channels	IPEH-002205-K
MU-Thermocouple1 CAN (Measuring range T)	Metal-cased measuring unit with 8 measuring channels	IPEH-002205-T
PCAN-Explorer 5		IPES-005028
PPCAN-Editor 2, PCAN-View		

PCAN® is a registered trademark of PEAK-System Technik GmbH.

All other product names mentioned in this document may be the trademarks or registered trademarks of their respective companies. They are not explicitly marked by "™" or "®".

Copyright © 2017 PEAK-System Technik GmbH

Duplication (copying, printing, or other forms) and the electronic distribution of this document is only allowed with explicit permission of PEAK-System Technik GmbH. PEAK-System Technik GmbH reserves the right to change technical data without prior announcement. The general business conditions and the regulations of the license agreement apply. All rights are reserved.

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 1.0.0 (2017-11-06)

Contents

1	Introduction	5
1.1	Prerequisites for Operation	5
2	The Configuration Concept	7
2.1	Possibilities of Configuration	8
2.2	Scaling	8
2.3	CAN Gateway Services	9
2.4	Default Values	9
2.5	Function Blocks	9
2.6	Event-triggered Transmission of CAN Messages	10
2.7	Characteristic Curves	10
3	List of Exercises	11
4	Solutions with Explanations	12
4.1	Exercise 1a: Set the CAN Bit Rate	12
4.2	Exercise 1b: Define CAN messages	14
4.3	Exercise 1c: Temperature Output of the Sensor on Port 1A in °C with max. Resolution	16
4.4	Exercise 1d: Export a Symbol File	20
4.5	Exercise 1e: Create a Simple Instrument Panel	22
4.6	Exercise 1f: Temperature Output in °C with 0.5° Resolution	23
4.7	Exercise 1g: Temperature Output in °F with 1° Resolution (Integer)	24
4.8	Exercise 1h: Use the Characteristic Curve Function	25
4.9	Exercise 2a: Turn on the LED, if Sensor Plugged	30
4.10	Exercise 2b: Slow LED Flashing Below a Threshold	32

4.11	Exercise 2c: Switch on the LED Below a Threshold (with Hysteresis)	35
4.12	Exercise 3a: Send Alarm Message Outside a Region	37
4.13	Exercise 3b: Send Temperature with a Change of at Least 1 ° C	43
4.14	Exercise 4a: Temperature Output of Internal Compensation Sensors	45
4.15	Exercise 4b: Output of the Module ID	47
4.16	Exercise 4c: Output of the Card Type per Slot	49
4.17	Exercise 4d: Set the LEDs External	51
4.18	Exercise 4e: Control the Blink Function	54
4.19	Exercise 5: CAN Messages on Request	57
Anhang A Literature and References		60

1 Introduction

Working successfully with the PPCAN-Editor 2 (instead of Thermocouple Configuration software) requires at least basic understanding by the user regarding hardware knowledge and programming experience.

This tutorial therefore addresses owners of a MU-TC1 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 2 software following the steps of this tutorial. When experiencing more and more difficulties when working through the document, this may at least serve as an indication for the future use of the PPCAN-Editor 2. When deciding against the effort, PEAK-System offers a configuration service according to detailed specifications.

1.1 Prerequisites for Operation

▶ Ensure the following four prerequisites:

1. The device MU-TC1 is supplied with power.
2. The D-Sub connector on the device is connected to another D-Sub connector of a PCAN interface via a terminated CAN cable.
3. The terminated CAN cable must be connected to a computer.
4. A suitable sensor is plugged to **Port 1A**.
5. The **PPCAN-Editor 2** software is installed.
6. As a remote CAN participant, the software **PCAN-View** or even better **PCAN-Explorer** is installed on the PC.

7. For PCAN-Explorer users: A Client net is set up, e.g. **Thermo_500k** with 500 kbit/s.

The device **MU-TC1** (Measuring Unit ThermoCouple 1) offers the following resources for configuration:

- └ Device ID (4 bit, 0..15 dec) can be changed inside the device using a switch.
- └ 1 CAN bus (with number **1**, not 0)
- └ CAN bit rates¹ (10k, 20k, 33k3, 47k6, 50k, 83k3, 95k2, 100k, 125k, 250k, 500k, 1M)
- └ CAN messages (11-bit or 29-bit)
- └ Information if and which measurement boards are plugged (5-bit, 0..31 per slot).
 - 0 = no card is plugged (empty slot)
 - 15 = Plugged measuring card for 2 Thermocouples type **K** (green)
 - 16 = Plugged measuring card for 2 Thermocouples type **J** (black)
 - 17 = Plugged measuring card for 2 Thermocouples type **T** (brown)
- └ Temperature of the compensation sensor (reference temperature) of each measuring board (in total 4 values with 13-bit, resolution 1/16°C)
- └ Temperatures of the two measuring sensor per board (in total 8 values with 16-bit, resolution 1/16°C)
- └ 2 LEDs per board (total of 8);
Status can be read and written.

¹ The bit rates can be set freely. The real functionality depends on the transceiver type equipped.

2 The Configuration Concept

The most microcontroller-equipped devices from PEAK-System offer possibilities to link any of their internally accessible resources with each other. For this, the firmware allows virtual wiring of the hardware resources by several means, e.g. Function Blocks, among others. For creating, editing, and managing configurations, PEAK-System offers the PPCAN-Editor 2 free to download.

The created file with the contained configuration is first saved on the PC, then the device is announced via CAN (via upload) and stored there non-volatile.

Some devices can hold several configurations. The valid setting is selected by means of a selector switch. The selector switch simultaneously determines the device ID and the location of the configuration within the non-volatile memory.

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.

This offers the possibility to operate several devices with different IDs on the same CAN bus and to upload the same multi-configuration file to them all.

The different ID ensures that each device loads its specific configuration from the non-volatile memory and performs its task accordingly.

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. Therefore, time events are not possible with any hardware.

All available resources of a device are reported to the PPCAN-Editor 2 by applying a special file related to that hardware. This hardware profile lets the PPCAN-Editor 2 allow or restrict configuration possibilities correspondingly.

2.2 Scaling

The most basic means of manipulating values is using the four basic arithmetics. They are controlled with parameters **SCALE** and **OFFSET**, which are taken from the well known linear equation in mathematics.

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 the one CAN bus can be output selectively on the other CAN bus or on the same CAN bus with a different ID (e.g. conversion 11-bit <-> 29-bit). Or an incoming message can trigger the sending of any other message.

2.4 Default Values

Parameters defined here determine the states of the module after switching on, such as the bit rate of the CAN bus, activation of an external RS232 interface, activation of a 5 V supply output for sensors, LEDs, and ports may be switched logically, etc.

2.5 Function Blocks

If the simple manipulation of measured variables etc. is not sufficient with **SCALE** and **OFFSET**, the firmware offers function blocks with more complex capabilities.

Such functions are, for example, value mapping with X/Y tables or matrices, hysteresis functions, delays, counters, timers, low pass filters, a comprehensive collection of mathematical, and logical functions up to a complex PIDT1 closed-loop control. Function blocks are processed sequentially or conditionally.

2.6 Event-triggered Transmission of CAN Messages

If CAN messages are to be sent only on certain events, a pool of trigger conditions is available. CAN messages can also be externally requested (RTR messages).

2.7 Characteristic Curves

An incoming X value results in the output of the assigned Y value. Here, 2 to 31 X/Y translation pairs may be defined. X values in between two X/Y pairs will return a Y value linear interpolated from the available Y points.

Thus, **SCALE** and **OFFSET** can be set individually for up to 32 sections of a value range (as in the case of scaling). Using this, segments of the curve may be influenced in their gradient to define plateaus or discontinuous functions.

3 List of Exercises

An overview on the manifold capabilities of the PCAN hardware MU-TC1 may be given when solving the following exercises.

- └ Exercise 1a: Set the CAN Bit Rate
- └ Exercise 1b: Define CAN messages
- └ Exercise 1c: Temperature Output of the Sensor on Port 1A in °C with max. Resolution
- └ Exercise 1d: Export a Symbol File
- └ Exercise 1e: Create a Simple Instrument Panel
- └ Exercise 1f: Temperature Output in °C with 0.5° Resolution
- └ Exercise 1g: Temperature Output in °F with 1° Resolution (Integer)
- └ Exercise 1h: Use the Characteristic Curve Function
- └ Exercise 2a: Turn on the LED, if Sensor Plugged
- └ Exercise 2b: Slow LED Flashing Below a Threshold
- └ Exercise 2c: Switch on the LED Below a Threshold (with Hysteresis)
- └ Exercise 3a: Send Alarm Message Outside a Region
- └ Exercise 3b: Send Temperature with a Change of at Least 1 °C
- └ Exercise 4a: Temperature Output of Internal Compensation Sensors
- └ Exercise 4b: Output of the Module ID
- └ Exercise 4c: Output of the Card Type per Slot
- └ Exercise 4d: Set the LEDs External
- └ Exercise 4e: Control the Blink Function
- └ Exercise 5: CAN Messages on Request

4 solutions with Explanations

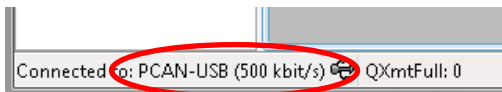
You may find further information on use of the PPCAN-Editor 2 in the online help: to be opened from the programs **Help** menu or by pressing the **F1** key.

4.1 Exercise 1a: Set the CAN Bit Rate

▶ Do the following to define the CAN bus:

1. Start the PPCAN-Editor 2.
2. Connect the PPCAN Editor 2 to the CAN bus that is connected to the MU-Thermocouple1.
3. Select the menu item **CAN > Connect**.
4. Select the appropriate **CAN hardware** and **Bitrate**.

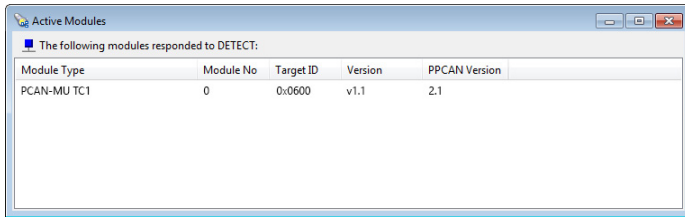
The selected connection is displayed in the status bar of the PPCAN-Editor 2 (bottom left corner).



Example of a Connection

5. Select the **Transmit > Detect Modules** menu item to check whether the MU-TC1 can be found in the CAN network.

The **Active Modules** dialog box appears. Status information of the MU-TC1 is displayed.



In the **Module No** field, the device ID is displayed (here 0).
 In the **Version field**, the firmware version is specified.

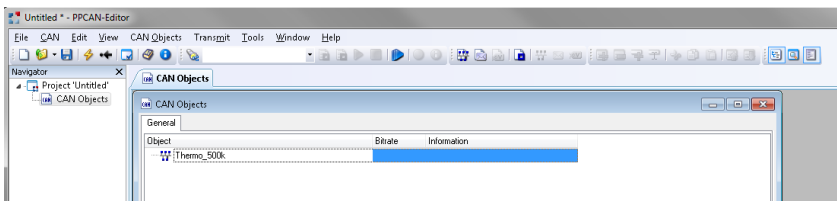
6. Create a new empty configuration file using the menu item **File > New**.

An empty window with the global CAN objects for all configurations contained later in the file appears.

Note: 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.

In that window a new CAN bus is already created as **Bus_0**, underneath which global CAN objects can be created hierarchically. A configuration is therefore not created.

7. The **Bus_0** is named **Thermo_500k** for this example.
8. Double click the name **Bus_0** and enter the new name.



- **Object:** Thermo_500k (bus name)
- **Bit rate:** only informative
- **Information:** description of the line

4.2 Exercise 1b: Define CAN messages

In order to transmit the required temperature, a CAN message with a length of 2 bytes suffices, since the corresponding measured value requires 16 bits.

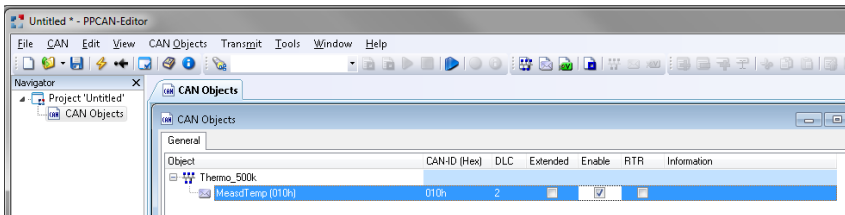
Solution process: The **MeasdTemp** CAN message, with the ID 0x010 and the length of 2 bytes, is to be periodically sent on the **Thermo_500k** bus every 300 ms.

▶ Do the following to define the CAN message:

1. Open the context menu of **Thermo_500k** with a right click.
2. Select **Add a new Symbol**.

This creates a new CAN message on the **Thermo_500k** bus.

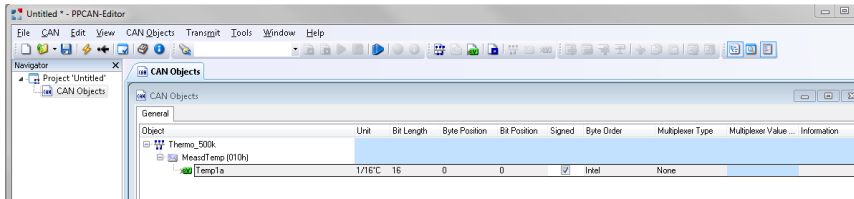
3. Assign the following values to its parameters:



- **Object:** MeasdTemp (symbol name)
- **CAN ID (Hex):** 0x010
- **DLC:** 2
- **Extended:** no, a 11-bit CAN ID is enough.
- **Enable:** yes
- **RTR:** no, the message should not be transmitted on request only.
- **Information:** description of the line

Within the CAN message a 16-bit CAN signal must be applied, which includes the measured temperature.

4. Open the context menu of the CAN message with a right click and select **Add a new Variable**.
5. Assign the following values to its parameters:



- **Object:** Temp1A (variable name)
- **Unit:** 1/16 °C (only informative)
- **Bit Length:** 16
- **Byte Position:** 0 (start byte)
- **Bit Position:** 0 (start bit)
- **Signed:** yes, signed (32767 is the largest positive value, -32768 is the greatest negative value)
- **Byte Order:** intel format (LSB in byte 0 bit 0, MSB in byte 1 bit 7)
- **Information:** description of the line

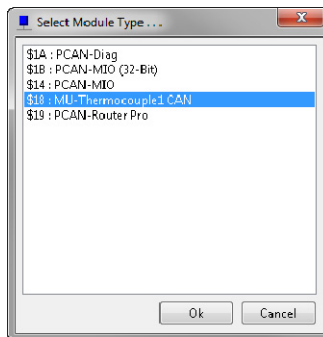
The empty body of both CAN messages is hereby defined, but not yet assigned to physical data sources. Therefore, a configuration must be created.

4.3 Exercise 1c: Temperature Output of the Sensor on Port 1A in °C with max. Resolution

▶ Do the following to create a new configuration within the configuration file:

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

A selection window of the hardware to be configured appears.



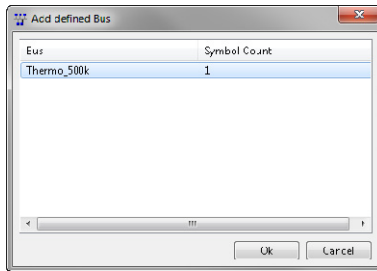
2. Select the profile **MU-Thermocouple1 CAN** and confirm with **Ok**.

Besides the **General** tab, a new tab has been created entitled with the configuration's name **Config0 I/O**.

An icon named **Config0** is now visible in the navigation window on the left.

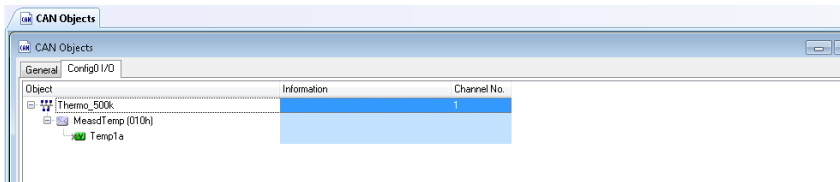
The globally defined CAN busses, messages, and signals should be used in this configuration. Therefore, they all must be imported.

3. Click on the new **Config0 I/O** tab.
4. Open the context menu with a right click and select **Add defined bus**.



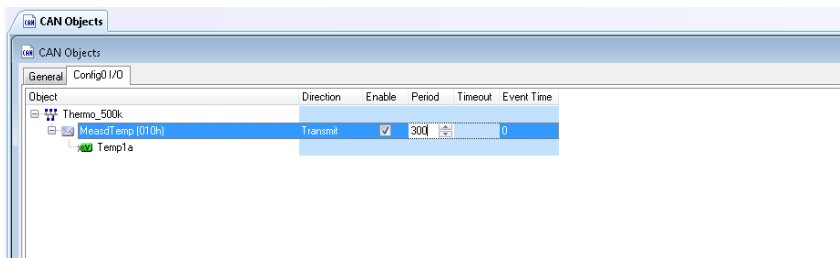
The previously defined global CAN bus **Thermo_500k** along with the contained messages and the 16-bit variables will be imported into the configuration.

Important Note: the channel number for MU-TC1 must be set to **1**, which sometimes can not be selected in the list box (Channel 0 does not exist at the MU-TC1).



— **Channel No.:** 1 (assign the hardware CAN channels)

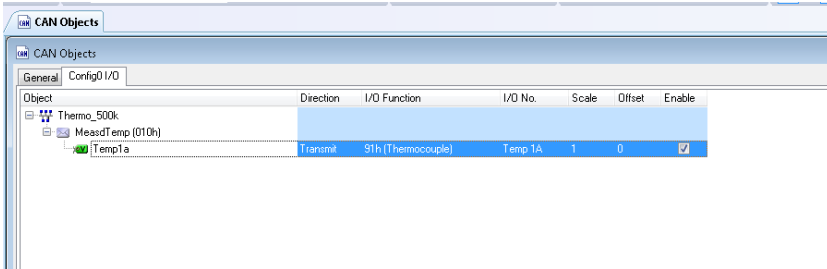
5. Enter the parameters for the messages:



— **Direction:** Transmit (the MU-TC1 should be transmitter)

- **Enable:** yes, message should be transmitted.
- **Period:** 300 (the transmission cycle time in ms)

6. Enter the parameters for the signals:



- **I/O-Function:** 91-Thermocouple
(this is the data source: a Thermocouple sensor)
- **I/O-Number:** Temp1A
(this is the sensor port, providing the data)
- **Scale:** 1 (no scaling at all, like multiplying with 1)
- **Offset:** 0 (no shifting at all, like addition of 0)
- **Enable:** yes, this signal (within the message) should be used.

7. Save the configuration as `exercise 1c` to your PC.

8. Transfer (upload) the configuration file to the MU-TC1 via the CAN bus.

Select the menu item **Transmit > Send Configuration** (or click the corresponding icon on the toolbar).

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

The power LED of the MU-TC1 flashes unrhythmical during the transmission and processing of the configuration file. As soon as

the LEDs of the sensor ports flash shortly, the configuration was processed and an automatic device reset was performed.

Now the MU-TC1 is ready for operation with its new configuration. The power LED flashes at 1 Hz. The measured temperature is transmitted (in 1/16 ° C).

4.4 Exercise 1d: Export a Symbol File

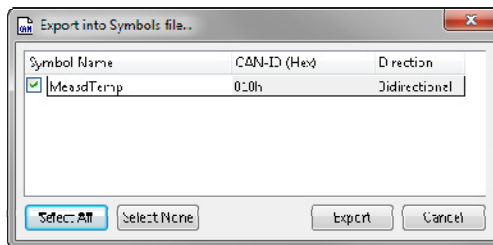
With the PCAN-Explorer you can simply decode the read value into plain text using a symbol file. The symbol file is generated in the PPCAN-Editor 2 from the CAN data.

▶ Do the following to export a symbol file in PPCAN-Editor 2:

1. Select the **General** tab to select the data base.
2. Select the menu item **File > Export into Symbol File**.
3. Save the file as `Exercise 1d` to your PC.

A selection window with CAN objects appears.

4. Select the CAN objects to be exported.



5. Click the **Export** button.

The symbol file is now created.

▶ Do the following to load and activate the symbol file in the PCAN-Explorer:

1. Select **File > Open** from the menu.
2. Click **File > Apply** to activate the file.

In the PCAN-Explorer, the two transmitted bytes of the measured value are not displayed in the Data field. In this case, the measured value is not as desired.

3. Right-click the symbol file in the **Project Browser** and open the file.

The editor opens in the PCAN-Explorer.

4. Locate the following line:

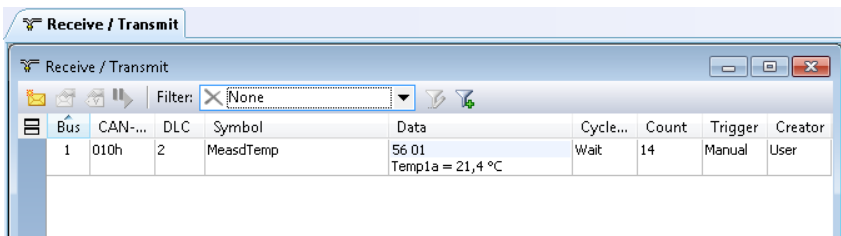
```
a=Temp1A      signed
```

5. Complete the following parameters:

```
a=Temp1A      signed /u:°C /f:0,0625
```

This is necessary because, on the one hand, a physical measured value is transmitted with the unit °C, and on the other hand, because this measured value is resolved in 1/16 degree increments (= 0.0625).

After the manual correction of the symbol file (with saving and applying), the current measurement is displayed in °C in the **Data** field.



4.5 Exercise 1e: Create a Simple Instrument Panel

Using a PCAN-Explorer with installed Instrument Panel Add-in, the CAN values can be represented and modified graphically.

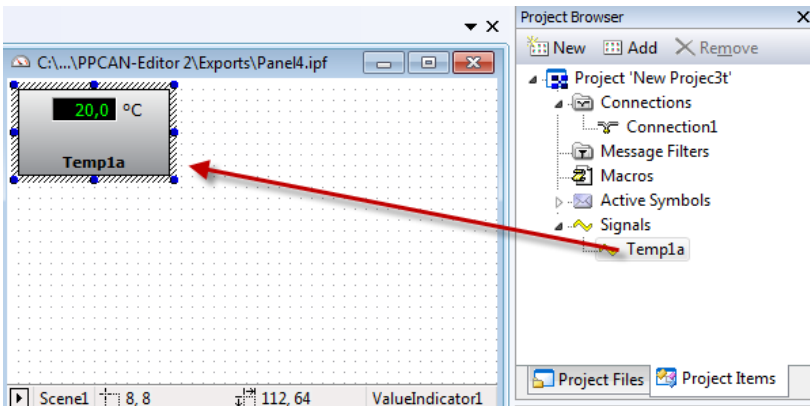
▶ Do the following to create an instrument panel in the PCAN-Explorer:

1. Select the menu item **Tools > Instruments Panel > Create Value Indicator**.

An empty panel with a digital display appears, to which a data source must be assigned.

2. Use the left mouse button to drag the variable **Temp1A** from the **Project Browser** to the instrument.

The instrument now displays the name of the variable and receives the appropriate formatting.



3. Save the file as `exercise_1e` on your PC.

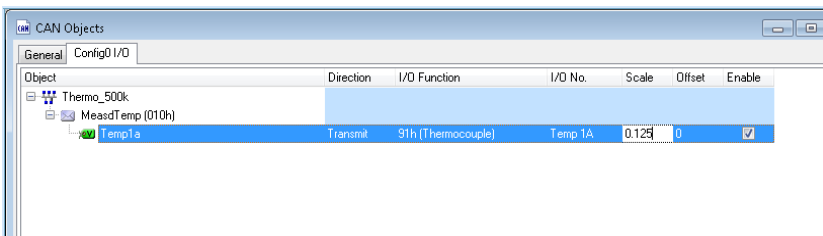
4.6 Exercise 1f: Temperature Output in °C with 0.5° Resolution

The resolution $1/16$ °C is indeed the maximum possible, but its use is often impractical. Instead for example, you could measure in 0.5 °C increments. For this, the manipulation of the scale value ($1/16 * 2$) is sufficient, a multiplication by a factor 0.125 . Of course, hardware still measures in $1/16$ °C steps, only the scaling of the transmitted data is changed.

➡ Do the following to get the temperature with 0.5° resolution:

1. Click the **Config0 I/O** tab on the CAN signal **Temp1A** and change the **Scale** value from 1 to 0.125 .

Thus, the number of steps per degree is reduced to $16 * 0.125 = 2$.



2. Save the configuration as `exercise 1f` on your PC.
3. Transfer (upload) the configuration file to the MU-TC1 via the CAN bus.



Tip: Only one-eighth of the value range is required in the CAN message. 3 bits could be saved. Then the variable **Temp1A** would only be 13 bits long (instead of 16 bits).

```
Picture=aaaaaaaa aaaaaaaaa // original
Picture=aaaaaaaa ---aaaaa // modified
a=Temp1A signed /u:°C /f:0,0625 // original
a=Temp1A signed /u:°C /f:c // modified
```

4.7 Exercise 1g: Temperature Output in °F with 1° Resolution (Integer)

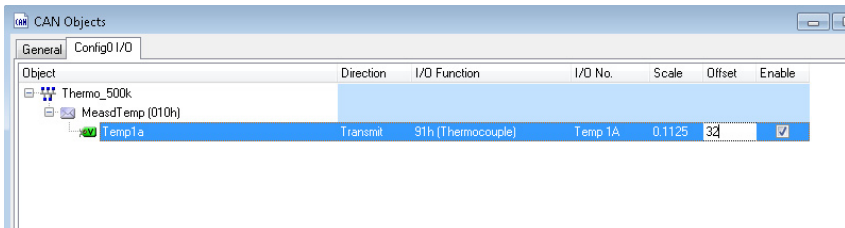
The parameters **Scale** and **Offset** show the temperature in whole ° **Fahrenheit**. The device still measures internally in 1/16 ° Celsius. The conversion factor is as follows:

$$T_{Fahrenheit} = ((T_{Celsius} \times 9) / 5) + 32$$

From this, a **Scale** value of $1/16 * 9/5 = \mathbf{0.1125}$ is calculated. The **Offset** value of 32 is accepted because the transfer was converted to full degree. At a resolution of 1/4 degrees, a factor of 4 would have to be included in both scale and offset (ie Scale: $0.1125 * 4 = \mathbf{0.45}$ and Offset: $32 * 4 = \mathbf{128}$).

▶ Do the following to get the temperature in Fahrenheit:

1. Click on **Config0 I/O** tab at the CAN signal **Temp1A**.
2. Change the **Scale** value from **1** to **0.1125**.
3. Change the **Offset** value from **0** to **32**.



4. Save the configuration as `exercise 1f` on your PC.
5. Transfer (upload) the configuration file to the MU-TC1 via the CAN bus.

```
a=Temp1A    signed /u:°C /f:0,0625 // original
a=Temp1A    signed /u:°F           // modified
```


4.8 Exercise 1h: Use the Characteristic Curve Function

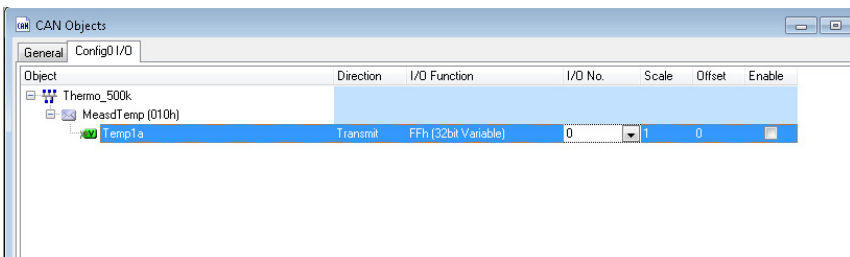
The coolant temperature should be displayed as constant 90 ° C between 80 ° C and 105 ° C (to smooth out fluctuations or not to disturb the driver). Or the advance of a tachometer is to grow gradually with speed.

But since you do not want to experiment with high temperatures or speeds on your desk, another example has been chosen, which makes at least the mechanism comprehensible. The transmitted measuring value (= temperature) of the sensor of Port 1A should be inverted in the range 0..50 °C.

The measured temperature has been passed directly to the CAN signal and was possibly formatted via Scale and Offset. For more complex calculations, a function block must be used from the firmware.

In this case, one that transforms the measured value via a characteristic curve (assigns each X value a new Y value) and writes the result in a 32-bit variable. Its new value (the transformed measurement) can be inserted into the CAN signal now.

Starting with the last step, in the CAN database the measured value from Port 1A is not transmitted anymore, instead the content of the 32-bit variable #0. Based on the example 1c the CAN signal, the I/O Function and the I/O Number have to be modified as follows:



- └ I/O Function: FF 32bit Variable
- └ I/O Number: 0

A falling characteristic curve (e.g. number #7) with 2 points has to be set up next:

	X	Y
Point 1	X = 0	Y = 800 (equivalent 50°C)
Point 2	X = 800	Y = 0 (equivalent 0°C)

▶ Do the following to create a characteristic curve:

1. Double click the icon **Config0** at the left side of the navigation window.

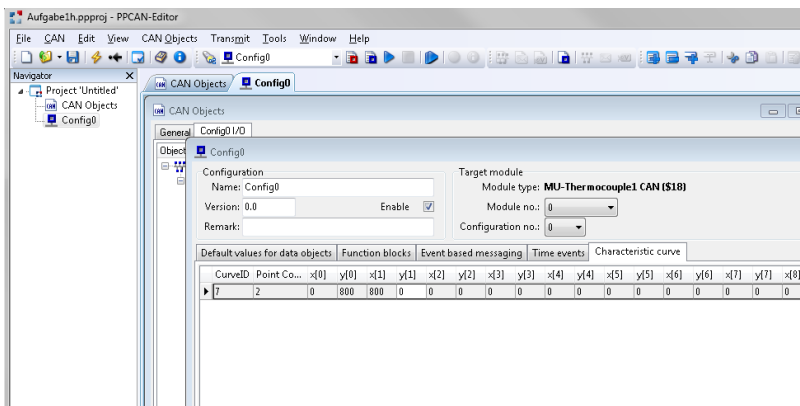
A new window titled **Config0** opens.

Here complex links of resources can be made.

2. Select the **Characteristic curve** tab.
3. Open the context menu with a right mouse click and select **Add Record**.

A table line appears, which represents a characteristic curve.

4. Enter the following parameters.

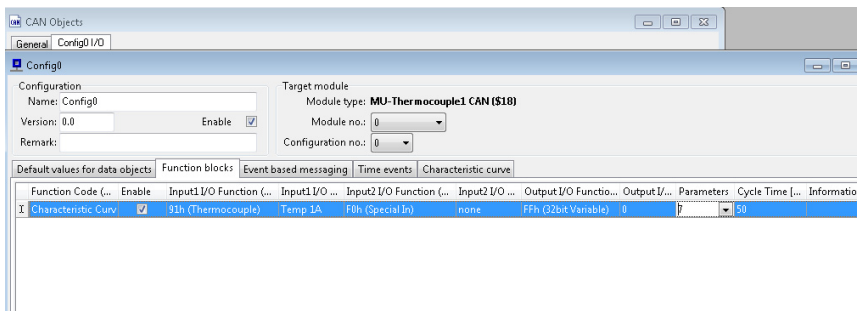


- └─ **CurveID:** 7 (an arbitrary number)
- └─ **Point Count:** 2 (number of X/Y value pairs)
- └─ **Pair of values 1:** $x[0] = 0$ should result in $y[0] = 800$ result
- └─ **Pair of values 2:** $x[1] = 800$ should result in $y[1] = 0$ result (thus you obtain a falling line)
- └─ **Information:** description of the line

Finally, you must manage the assignment of the incoming raw value to the characteristic curves X axis and also of the resulting Y value to variable #0, which is subsequently transmitted onto CAN. For this, the special function block **Characteristic curve** is needed, which handles that conversion.

▶ Do the following to create the required function block:

1. Select the **Function blocks** tab and create a new function block.
2. Open the context menu (right click) and select **Add Record**.
A table line appears, which is a function block.
3. Enter the following parameters:



- └─ **Function Code:** Characteristic Curve (a special function block for this purpose)

- └ **Enable:** yes, this function block should be active.
 - └ **Input1:** "91-Thermocouple" and "Temperature1a"
(the source of the X values)
 - └ **Input2:** "F0-Special In" and "none" (not used)
 - └ **Output:** "FF-32bit Variable" and "0"
(the Y result is written to the 32-bit variable #0)
 - └ **Parameter:** 7
(number of the already defined characteristic curve)
 - └ **Cycle time:** 50
(the calculation of the raw value takes place every 50 ms)
 - └ **Information:** description of the line
4. Save the configuration as `exercise_1h` on your PC.
 5. Transfer (upload) the configuration file to the MU-TC1 via the CAN bus.

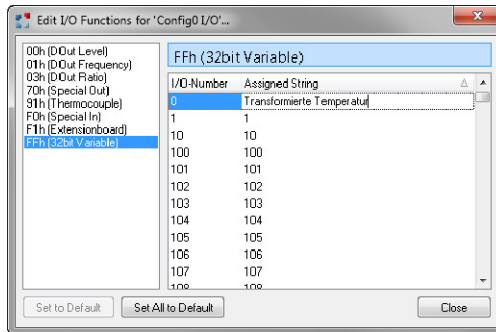
If you put an intact sensor in Port 1A, a temperature is displayed, the 50 °C - corresponds to the measured value. If you heat the sensor with your fingers, the displayed temperature drops accordingly.

In addition, a readable name can be assigned to each 32-bit variable (and other resources).

▶ Do the following to assign a name to each 32-bit variable:

1. Go to the **Function blocks** tab.
2. Select the **Edit > I/O-Functions** menu item.

The following dialog window appears:



3. Rename the 32-bit variable #0 to **Transformierte Temperatur**.

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

4.9 Exercise 2a: Turn on the LED, if Sensor Plugged

If you load the configuration file from exercise 1c again, and the temperature sensor from the Port 1A was removed, a raw value of 0x8000 is transmitted.

Because temperatures below the natural constant $K_0 = -273.15 \text{ }^\circ\text{C}$ can not occur (this would correspond to measured values from 0x8000 to 0xEEEC hex), this range of values is used to represent errors, e.g.:

- └ **0x8000**: no thermal voltage measured (sensor at port not connected or cable break).
- └ **0xE000**: card is not plugged in or respectively not recognized.

The measured values from the sensor can therefore be distinguished by the range of values in valid and invalid.

The mathematical comparison of the measured value with a constant (e.g. 0x8000) requires the use of a function block of the firmware.

▶ Do the following to create the required function block:

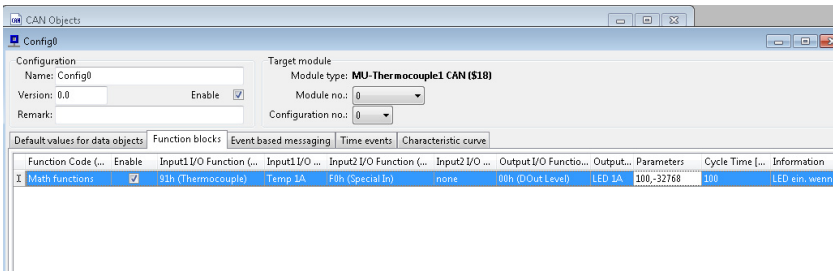
1. Open exercise 1c and save it as `exercise2a`.
2. Double click the Icon **Config0** at the left navigation window.

A new window named **Config0** opens.

3. Select the **Function blocks** tab.
4. Open the context menu with the right mouse button and select **Add Record**.

A table line appears, which represents a function block.

5. Enter the following values for the mathematical comparison described above:



- **Function Code:** MathFunction
- **Enable:** yes, this block should be active.
- **Input1:** "91-Thermocouple" and "Temperature1a"
- **Input2:** "F0-Special In" and "none" (not used)
- **Output:** "00-Dout Level" and "LED 1A" (LED to Port 1A)
- **Parameter:** type of the mathematical function logical comparison with 0x8000

As function is chosen: **In1 > const** and **-32768** is entered as value. The result is a Boolean value (TRUE or FALSE) that controls the LED.

- **Cycle Time:** e.g. 100
(The logical comparison takes place every 100 ms.)
- **Information:** description of the line

6. Save the configuration as `exercise_2a` on your PC.
7. Transfer (upload) the configuration file to the MU-TC1 via the CAN bus.

If you put an intact sensor in Port 1A, the LED lights up. When you remove the sensor, the LED is off.

4.10 Exercise 2b: Slow LED Flashing Below a Threshold

LED 1a should flash when the measured value falls below 30 °C. The LED should be off if the measured value equals 30 °C. In addition, it should shine permanently when no sensor is plugged.

This example demonstrates conditional execution of function blocks. Three different measuring ranges must be checked:

1. If the result is TRUE, the next line is executed to set the LED accordingly.
2. When FALSE, the next line is skipped.

The number of lines to skip is given in the **Parameter** field of the respective examination area (here: 1 each). With this you can create CASE similar structures.

Flashing is implemented via the function **PWM**. A fixed PWM frequency is set by default. The duty cycle decides on the state of the LED 1A: off, flashing, or on.

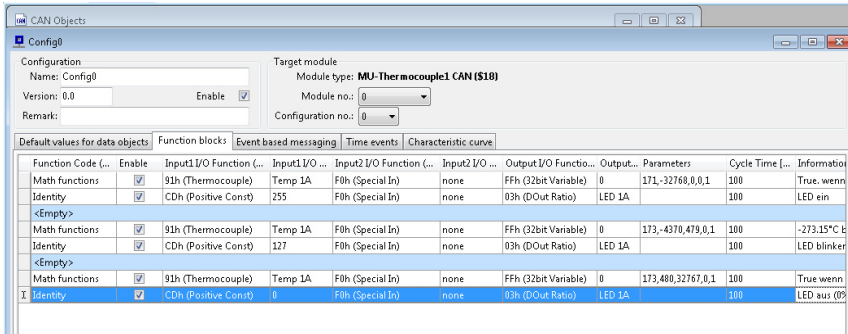
The frequency can be adjusted in 0.1 Hz increments between 0.1 and 10 Hz. The duty cycle (ratio) is set from 0 to 255. The value 127 corresponds to 50 %.

Three functional blocks are necessary for the three range checks and for each an assigned function block to set the PWM ratio (= the corresponding flashing). To structure a configuration separating lines may be added (**Menu Edit > Insert Space Record**).

▶ Do the following that the LED flashes slowly below a threshold value:

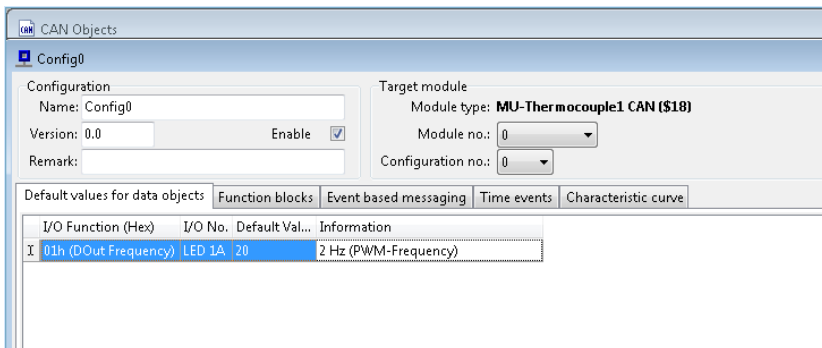
1. Double click the icon **Config0** at the left navigation window.
A new window named **Config0** opens.
2. Select the **Function blocks** tab.

3. Open the context menu with the right mouse button and select **Add Record**.
4. Enter the following parameters:



Finally, you should determine the flashing frequency 2 Hz (default value).

- ▶ Do the following to set the flashing frequency:
 1. Select the **Default values for data objects** tab.
 2. Open the context menu with the right mouse button and select **Add Record**.
 3. Enter the following parameters:



- └ **I/O Function:** 01h (DOut Frequency)
- └ **I/O No.:** LED 1A
- └ **Default Value:** 20
- └ **Information:** description of the line
 4. Save the configuration as `exercise_2b` on your PC.
 5. Transfer (upload) the configuration file to the MU-TC1 via the CAN bus.

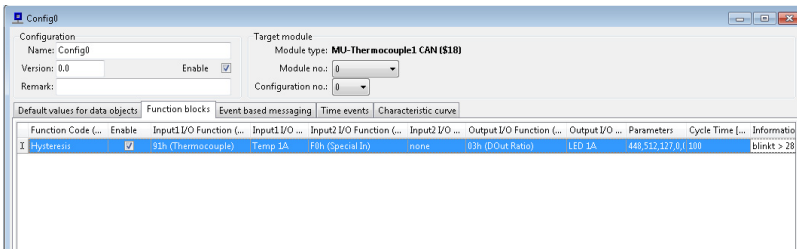
If no sensor plugged into Port 1A, the LED lights up. When a sensor is plugged in and the measured temperature is below 30 °C, the LED flashes. If the temperature rises to at least 30 °C, the LED is off.

4.11 Exercise 2c: Switch on the LED Below a Threshold (with Hysteresis)

The light emitting diode LED 1A should flash until the rising measured value 32°C is reached. It should remain off until the measured value falls below 28°C , and then start flashing.

Flashing is implemented via the function **PWM**. A fixed PWM frequency is set by a default value. The duty cycle decides whether state of the LED 1a is off, flashing, or on.

- ▶ Do the following to switch on the LED below a threshold:
 1. Open exercise 1c and save it as `exercise 2c`.
 2. Double click the icon **Config0** at the left navigation window.
A new window named **Config0** opens.
 3. Select the **Function blocks** tab.
 4. Open the context menu with the right mouse button and select **Add Record**.
 5. Enter the following parameters:

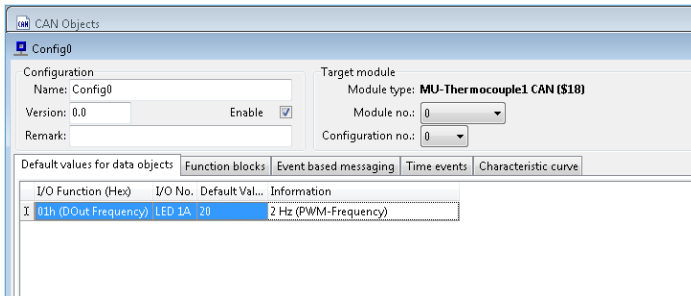


- └ **Function Code:** Hysteresis
- └ **Input I/O-Function:** 90-Thermocouple (Evaluating the measuring temperature)
- └ **Input I/O-Number:** Temp1A
- └ **Output I/O-Function:** PWM-Ratio (Result affects ratio of the LED)

- └ **Output I/O-Number:** LED 1A
- └ **Hysteresis-Parameter:** 448 (= 28 °C), 512 (= 32 °C), 127 (= 50 % Ratio), 0 (= LED off)

Finally, you should determine the flashing frequency 2 Hz (default value).

- ▶ Do the following to set the flashing frequency:
 1. Select the **Default values for data objects** tab.
 2. Open the context menu with the right mouse button and select **Add Record**.
 3. Enter the following parameters:



- └ **I/O Function:** 01h (DOut Frequency)
 - └ **I/O No.:** LED 1A
 - └ **Default Value:** 20
 - └ **Information:** description of the line
4. Save the configuration as `exercise 2c` on your PC.
 5. Transfer (upload) the configuration file to the MU-TC1 via the CAN bus.

Heating of the sensor: the flashing is terminated only above 32 °C.

Cooling of the sensor: flashing starts below 28 ° C.

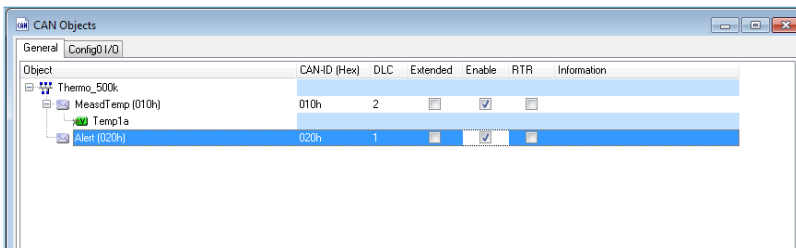
4.12 Exercise 3a: Send Alarm Message Outside a Region

The following example (based on exercise 1c) should transmit a separate CAN message when the measured value is outside a certain range. For this, an additional message needs to be added to the CAN database.

Solution process: Sending the **Alert** CAN message with the ID 0x020 and the length of 1 byte on the **Thermo_500k** bus, but only if the sensor on Port 1A leaves the range from 28 to 33 ° C. This message contains the **ErrFlag** signal.

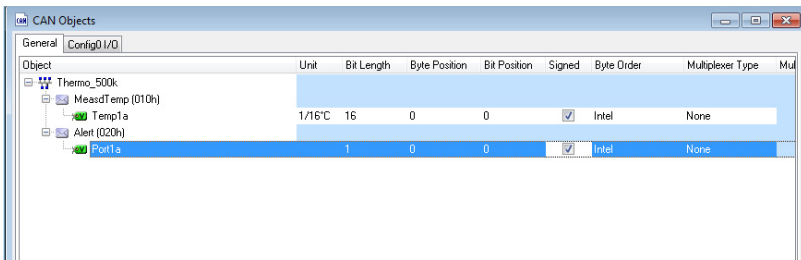
▶ Do the following to set up an alarm message:

1. Add the alarm message in the **General** tab.
2. Open the context menu with the right mouse button and select **Add a new Symbol**.
3. Enter the following parameters:



- └ **Object:** Alert (symbol name)
- └ **ID:** 20h
- └ **DLC:** 1
- └ **Extended:** No, a 11-bit ID is sufficient.
- └ **Enabled:** yes
- └ **RTR:** no, the message is not transmitted on request only.

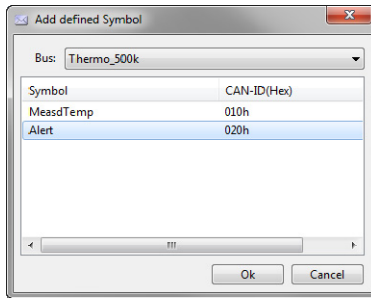
4. Create a new signal within the CAN message.
5. Open the context menu with the right mouse button and select **Add a new Variable**.
6. Enter the following parameters:



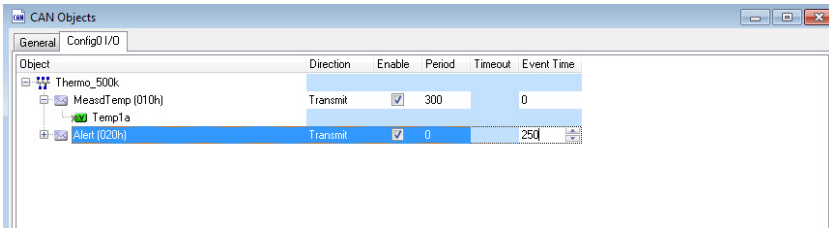
- └ **Object:** Port1a (variable name)
- └ **Unit:** -
- └ **Bit length:** 1
- └ **StartByte:** 0
- └ **StartBit:** 0
- └ **Signed:** yes (irrelevant for 1-bit values)
- └ **Byte Order:** intel format (irrelevant for 1-bit values)

The base of the CAN message is hereby defined. Import them into a configuration **Config0**.

7. Get the **Config0 I/O** tab to the foreground.
8. Open the context menu with the right mouse button and select **Add defined Symbol**.
9. Select the Alert message.

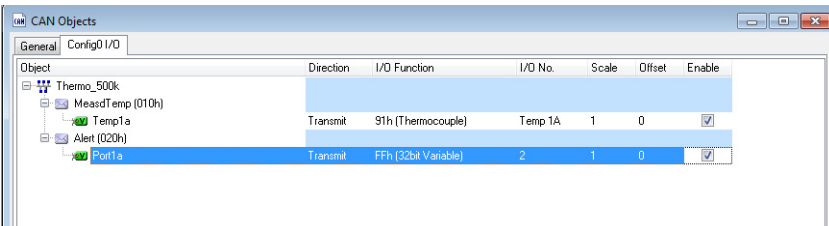


10. Enter the parameters for the message:



- **Direction:** Transmit (the MU-TC1 should act as a transmitter)
- **Enable:** yes, the message should be transmitted.
- **Period:** 0 (no periodically sending, only in case of alarm)
- **Event Time:** 250 ms
(When the alarm is present, the alert message is transmitted periodically.)

11. Enter the parameters for the signals:



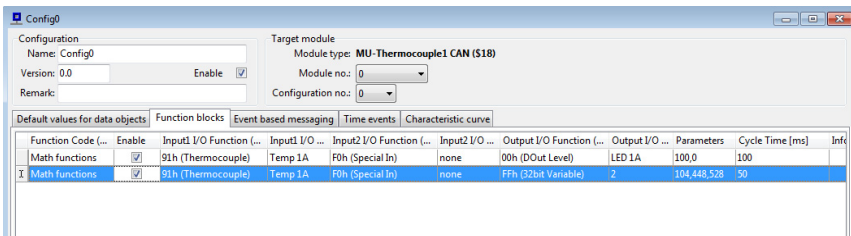
- └ **I/O-Function:** FF 32bit-Variable
(A variable that contains the error flag.)
- └ **I/O-Number:** 2 (the variable number)
- └ **Scale:** 1
- └ **Offset:** 0
- └ **Enable:** yes, this signal (in the message) should be used.

Two further steps are required to solve the task.

- Firstly, a cyclic test is necessary if the temperature measured at Port 1A has left the value range (range check with bool value as a result).
- Secondly, the sending of the message **Alert** has to be triggered.

➡ Do the following to create a cyclic check:

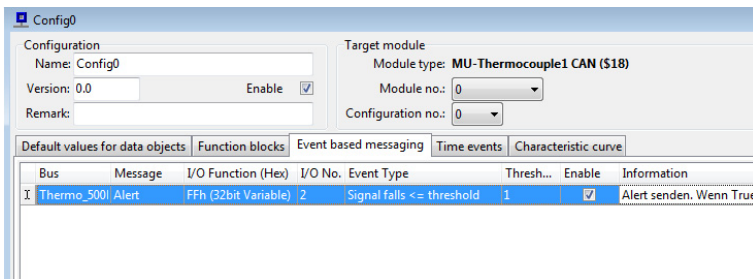
1. Double click the Icon **Config0** at the left navigation window.
This will open a new window titled **Config0**.
2. Select the **Function blocks** tab.
3. Open the context menu with the right mouse button and select **Add Record**.
A new line appears, representing a function block.
4. Enter the following parameters:



- └ **Function Code:** MathFunction
- └ **Enable:** yes, this block should be active.
- └ **Input 1:** "91-Thermocouple" and "Temp1A"
- └ **Input 2:** "F0-Special In" and "none" (not used)
- └ **Output:** "FF 32bit-Variable" and #2 (arbitrarily chosen)
- └ **Parameter:** Type of the mathematical function: TRUE if outside fixed limits: lower limit 448 (= 28 °C), upper limit 528 (= 33 °C)
- └ **Cycle time:** 50 (the logical comparison takes place every 50 ms)
- └ **Information:** description of the line

Finally, based on the Boolean value a CAN message has to be triggered in variable #2.

5. Select the **Event based Messaging** tab.
6. Open the context menu with the right mouse button and select **Add Record**.
A new line appears.
7. Enter the following parameters:



- └ **Bus:** Thermo_500k
- └ **Message:** Alert (this CAN message is to be sent in case of alarm)
- └ **I/O-Function:** the 32-bit variable # 2, which contains the result of the field checks.

- └ **Event Type:** If the result is ≥ 1 (TRUE)
 - └ **Threshold:** the mentioned 1 (TRUE)
 - └ **Enable:** yes, the trigger should be active.
 - └ **Information:** description of the line
8. Save the configuration as `exercise_3a` on your PC.
 9. Transfer (upload) the configuration file to the MU-TC1 via the CAN bus.

For a room temperature below 28 °C at intervals of 250 ms, the message **Alert** (ID = 0x020) is sent. If you heat the sensor with your finger, the transmission stops from 28 °C and until 33 °C are exceeded. Then the message is sent again, which means there is no alarm between 28 and 33 °C.

Depending on the season / temperature other temperature thresholds can be useful. You could also transmit the measured temperature in the alert message, or an above / below flag.

4.13 Exercise 3b: Send Temperature with a Change of at Least 1 °C

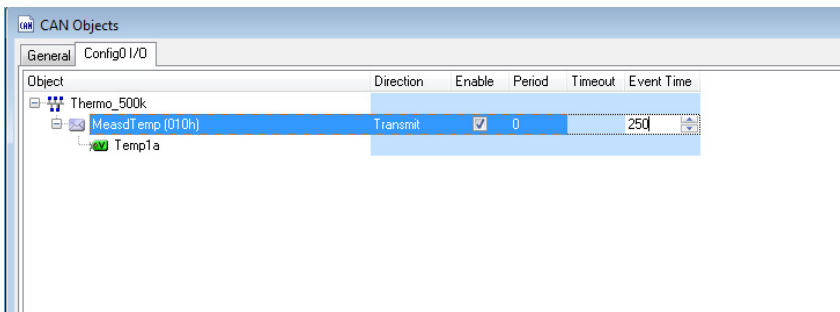
In the following example (based on exercise 1c) the CAN message **MeasdTemp** should only be sent when the measured temperature has changed by at least 1 °C.

For this, the message in the CAN database has to be modified and an appropriate transmission trigger is required.

▶ Do the following to send a message when temperature changes:

1. Change in the tab **Config0 I/O** in the MeasdTemp CAN message the transmission period to 0.
2. Set instead an **Event Time** of **250 ms**.

The message is no longer sent cyclically.

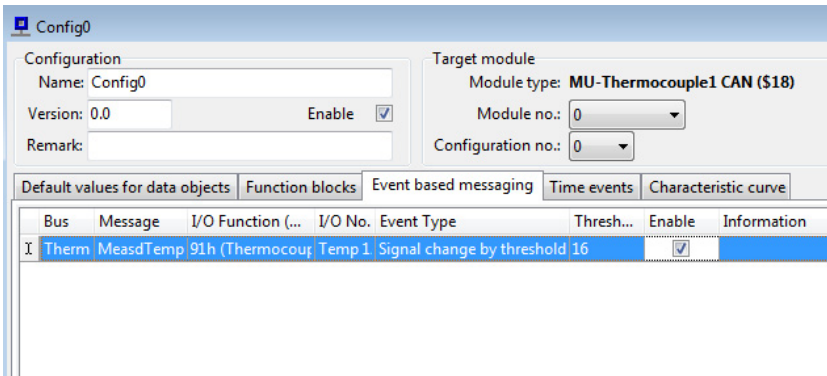


In order to send the message from time to time, the trigger must be defined.

3. Double click the icon **Config0** at the left navigation window. This will open a new window titled "Config0".
4. Select the **Event based Messaging** tab.
5. Open the context menu with the right mouse button and select **Add Record**.

A new line appears.

6. Enter the following parameters:



- **Bus:** Therm_500k
 - **Message:** MeasdTemp
(the temperature message should be sent)
 - **I/O-Function:** 91-Thermocouple
(the temperature measured should be rated)
 - **I/O_Number:** Temp1A
(specifically the temperature measured at Port 1A)
 - **Event Type:** if it changes by a certain amount (Threshold)
 - **Threshold:** 16 (corresponds to 1 °C, because it is always measured in 1/16 degree resolution)
 - **Enable:** yes, the trigger should be active.
 - **Information:** description of the line
7. Save the configuration as `exercise 3b` on your PC.
 8. Transfer (upload) the configuration file to the MU-TC1 via the CAN bus.

The **MeasdTemp** CAN message will now only sent when the sensor heats/cooling or pulled/plugged.

4.14 Exercise 4a: Temperature Output of Internal Compensation Sensors

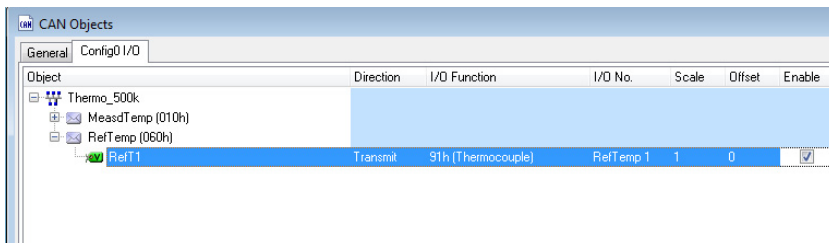
The thermovoltage used for the temperature measurement at the tip of the measuring sensors can be distorted significantly by various influences. A significant source of error are the metallic contacts of the 8 connection ports. This is where thermoelectric voltages emerge, for example by solder joints, etc., whose influence is growing with increasing ambient temperature.

For compensation, the temperature at these contact points is measured and the mathematical equivalent of a counter-voltage is added to the measured value. The temperature at the contact points (so-called reference temperature) can be read out.

Solution process: Sending a **RefTemp** CAN message with the ID 0x060 and the length of 8 bytes on the bus **Thermo_500k**. This message contains the signal **RefT1** that contains the reference temperature of the left card (slot #1).

▶ Do the following to output the temperature of the internal sensors:

1. Open exercise 1c and save it as `exercise 4a`.
2. Create a new message **RefTemp** in the **General** tab.
3. Create a new signal (13-bit) in the **General** tab
4. Import the message including the CAN signal to the configuration and enter the following parameters:



5. Save the configuration as `exercise 4a` on your PC.
6. Transfer (upload) the configuration file to the MU-TC1 via the CAN bus.

With the message **RefTemp** the temperature, which is measured at the sensor ports 1A and 1B is transmitted. It should correspond approximately to the room temperature.

4.15 Exercise 4b: Output of the Module ID

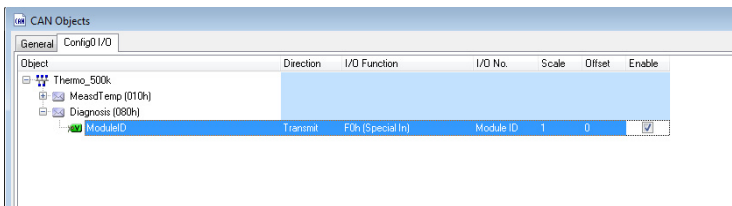
The **Module ID** is a 4-bit value, which is factory-set to 0 and can be modified via the DIP switch inside the MU-TC1. It has several functions, including selecting the appropriate configuration depending on the switch position. For example, you can switch between different configurations that are contained in a PPCAN project file.

Note: If a new configuration is unexplainable, one of the first debugging steps is to determine the Module ID. You may be running a different configuration each time.

Solution process: The **Diagnosis** CAN message is to be sent with the ID 0x080 and the length of 8 bytes on the **Thermo_500k** bus. This message contains the ModuleID signal that indicates the currently set **Module ID**.

Do the following to output the Module ID:

1. Open the configuration from exercise 1c and save it as exercise 4b.
2. Create a new message Diagnosis in the **General** tab.
3. Create a new signal (4-bit, unsigned) in the **General** tab.
4. Import the message including the CAN signal to the configuration and enter the following parameters:



I/O-Function: F0-Special In (one of several status information)

- └ **I/O-Number:** Module ID
The I/O number 16 has already been named.
 - └ **Scale:** 1
 - └ **Offset:** 0
 - └ **Enable:** Yes
5. Save the configuration as `exercise_4b` on your PC.
 6. Transfer (upload) the configuration file to the MU-TC1 via the CAN bus.

The **Diagnosis** message (with the ID 0x080) transmits the Module ID. Modifying the Module ID is not active until the device is reset (e.g. Power Off / On).

4.16 Exercise 4c: Output of the Card Type per Slot

The controller board of the MU-TC-1 can detect 31 different types of measurement cards. The corresponding value from 0 to 31 can be read and displayed. Currently are defined:

0 = no card is plugged (empty slot)

15 = measuring board for 2 thermocouples type K (green) is plugged.

16 = measuring board for 2 thermocouples type J (black) is plugged.

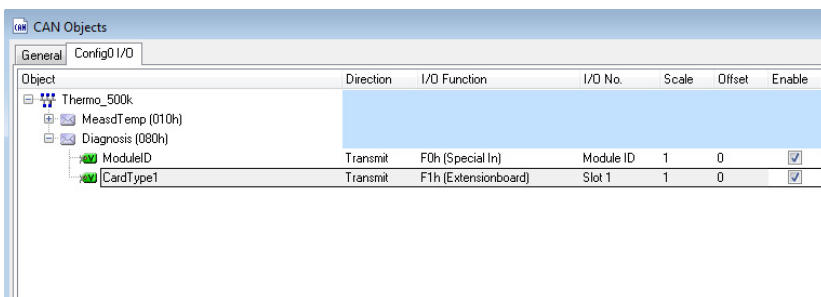
17 = measuring board for 2 thermocouples type T (brown) is plugged.

Solution process: The **Diagnosis** CAN message is to be sent with the ID 0x080 and the length of 8 bytes on the **Thermo_500k** bus.

Diagnosis contains the **CardType1** signal, indicating the type of the detected card in slot 1.

▶ Do the following to output the card type per slot:

1. Open the configuration from exercise 4b and save it as exercise 4c.
2. Create a new Signal (5-bit, unsigned) in the **General** tab.
3. Import the message including the CAN signal to the configuration and enter the parameters:



- └─ **I/O Function:** F1-ExtensionBoard
(one of several status information)
 - └─ **I/O No:** Slot 1
Here, the I/O number 0 has already been named.
 - └─ **Scale:** 1
 - └─ **Offset:** 0
 - └─ **Enable:** yes
4. Save the configuration as `exercise_4c` on your PC.
 5. Transfer (upload) the configuration file to the MU-TC1 via the CAN bus.

With the exception of the Module ID (in byte 0), the **Diagnosis** message (with ID 0x080) also transfers the card type in slot 1 (in byte 4).

In addition, the number sequence can be converted to plain text with the PCAN-Explorer. Based on the exercise 4c the value range from 0 to 31 can be decoded according to card types.

You will find the **ENUMS** section in the Symbol file. Here you can define enumerations:

```
{ENUMS}
enum BoardType(15="K(green)", 16="J(black)", 17="T(brown)",
0="empty")
```

In the message definition, you can use the defined **ENUMS**:

```
{RECEIVE}
[Diagnosis]
ID=80h
Picture=-----aaaa ----- ----bBBBB -----
-----
a=Module ID      unsigned
b=Board-0  unsigned /e:BoardTyp
```

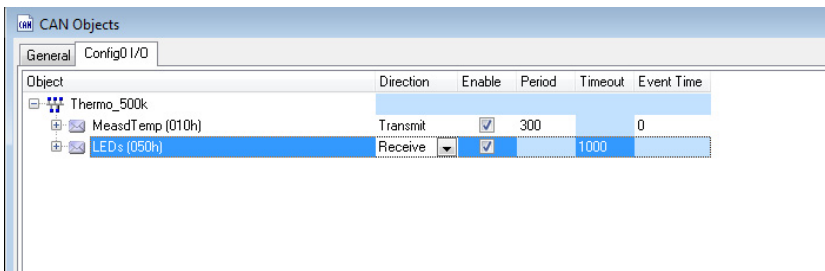
4.17 Exercise 4d: Set the LEDs External

The setting of the LEDs due to device-internal decisions has already been shown in exercise 2. The setting of the LEDs from outside through a CAN message reception is also possible.

Solution process: The **LEDs** CAN message with the ID 0x050 and the length of 4 bytes should be received on the bus **Thermo_500k**. This message contains the signal **LED 1A**, which corresponds with the LED for the sensor Port 1A. If the message to be received is missing, the LED is off after 1 second.

▶ Do the following to set LEDs externally through a CAN message:

1. Open the configuration from exercise 1c and save it as exercise 4d.
2. Create a new message LEDs in the **General** tab.
3. Create a new signal (1-bit) in the **General** tab.
4. Import the message including the CAN signal to the configuration.
5. Enter the following parameters:



- └ **Direction:** Receive
(This message comes from outside and will be evaluated.)
- └ **Enable:** yes, the message should be active.

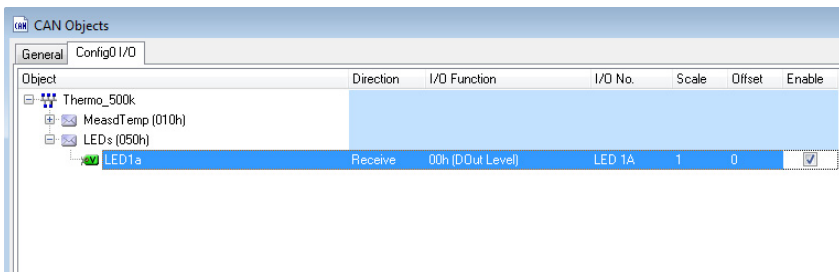
- └ **Period:** Transmission period
(Does not apply to receive messages.)
- └ **Timeout:** 1000 ms
(After this the default value takes effect.)
- └ **EventTime:** Transmission period when an event is present.
(Does not apply to receive messages.)

The message LEDs is defined as a receive message and therefore doesn't have a transmission period.

The **timeout** for this message is 1000 ms.
After that the default value takes effect.

The received signal **LED 1A** acts directly on the corresponding resource.

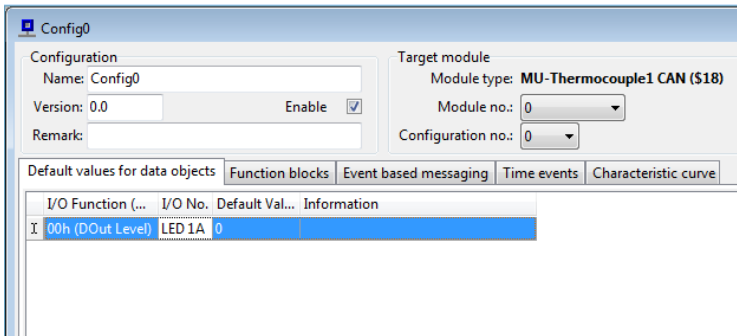
6. Enter the following parameters for **LED 1A** in **Config0**:



- └ **I/O-Function:** 00-Dout Level (one of the LEDs)
- └ **I/O-Number:** LED 1A (the LED for Port 1A)
- └ **Scale:** 1
- └ **Offset:** 0
- └ **Enable:** yes, the signal should be used.

Finally, the default value for the LED must be set. With PowerOn and the timeout case, the LED should be off:
Default = 0.

7. Enter the following parameters:



- └ **I/O-Function:** 00-Dout Level (one of the LEDs)
- └ **I/O-Number:** LED 1A (the LED for Port 1A)
- └ **Default-Value:** 0 (LED should be off)
- └ **Information:** description of the line

8. Save the configuration as `exercise 4d` on your PC.

9. Transfer (upload) the config file to the MU-TC1 via CAN.

The PCAN-Explorer (alternatively PCAN-View) sends a CAN message with ID 0x050, in which byte 0 and bit 0 = 1 are set. The LED is switched on. If another message is sent within one second, the LED remains on.

If the message is not transmitted for more than one second, the default value will take effect and the LED is switched off.

➤ Enter the following code to define a send message in the PCAN-Explorer symbol file:

```
{SEND}
[Leds]
ID=50h
// 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=-----a -----
a=LED 1A bit
```

4.18 Exercise 4e: Control the Blink Function

Blinking of LEDs can also be realized via a PWM function. For this purpose (as expected from PWM), the parameters **Frequency** and **Ratio** are available.

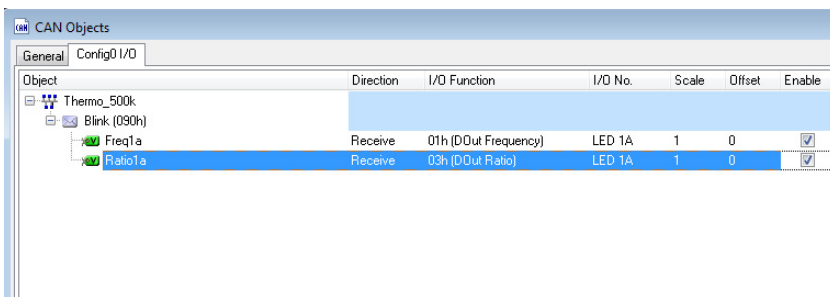
	X	Y
Frequency0	0.1.. 10 Hz	0.1 Hz
Ratio (pulse pause ratio)	0 255 equivalent 0..100%	1

Solution process: The **Blink** CAN message with the ID 0x090 and the length of 2 bytes should be received on the **Thermo_500k** bus. This message contains the two 8-bit signals **Freq1a** and **Ratio1a**, which directly affect the LED 1A.

➡ Do the following to control the blink function:

1. Create a new receive message including the two signals.

These are written directly to the I/O functions **Frequency** and **Ratio** of the LED 1a.



2. Save the configuration as `exercise 4e` on your PC.
3. Transfer (upload) the configuration file to the MU-TC1 via the CAN bus.

The PCAN Explorer (alternatively PCAN-View) sends a 2-byte-long CAN message with ID 0x090, in which byte 0 and byte 1 are assigned with corresponding values for **Frequency** and **Ratio**. The LED flashes according to the set values.

- ▶ Do the following to define the message with the PCAN Explorer in the symbol file from exercise 4e:

```
{SEND
[Blink]
ID=90h
//   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=aaaaaaaa bbbbbbbb
a=Freq1a unsigned
b=Ratio1a unsigned
```

- ▶ Do the following in the PCAN-Explorer:

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


An empty panel with a vertical slider appears.

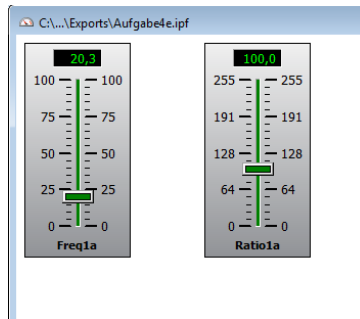
2. Select again the menu item **Tools > Instruments Panel > Create vertical Slider Control**.

Another vertical slider is displayed, which you can place next to the first instrument.

3. Use the left mouse button to drag the **Freq1a** and **Ratio1a** variables one by one from the **Project Browser** to one slider.

Both sliders are now assigned send parameters. This shows the names of the variables in a suitable formatting.

-  **Note:** Possibly the value range of the controller **Freq1a** has to be limited to 100 in the context menu under **Properties**.



4. Save the file as exercise 4e on your PC.

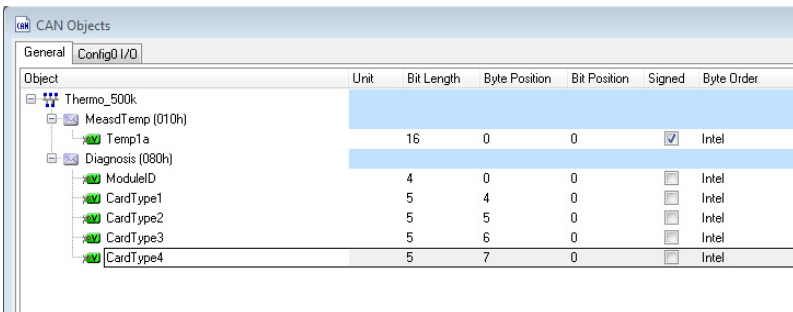
4.19 Exercise 5: CAN Messages on Request

Status information should be output only on request. Based on exercise 4c the types of all the inserted cards, should be output via CAN.

Solution process: The **Diagnosis** CAN message is to be sent with the ID 0x080 and the length 8 bytes on the **Thermo_500k** bus on request (Remote Transmission Request). This message contains the signals **CardType1** to **CardType4**, which indicates the type of cards recognized in the slots.

▶ Do the following to get CAN messages on request:

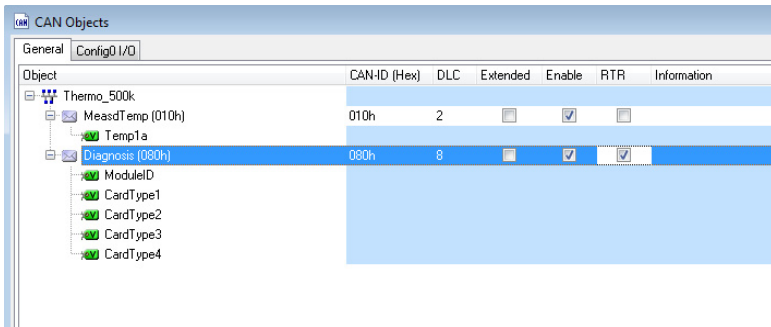
1. Open the configuration from exercise 4c and save it as exercise 5.
2. Add 3 more CAN signals (each 5-bit, unsigned) and set the parameters as follows:



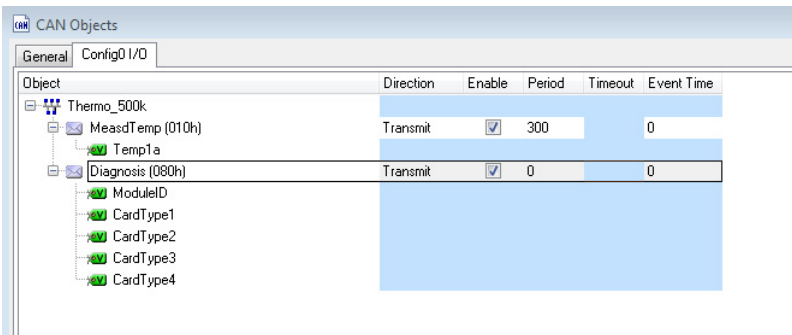
The screenshot shows the 'CAN Objects' configuration window with the 'General' tab selected. The tree view on the left shows the hierarchy: Thermo_500k > MeasTemp (010h) > Temp1a and Diagnosis (080h) > ModuleID, CardType1, CardType2, CardType3, and CardType4. The table on the right lists the parameters for these signals.

Object	Unit	Bit Length	Byte Position	Bit Position	Signed	Byte Order
MeasTemp (010h)						
Temp1a		16	0	0	<input checked="" type="checkbox"/>	Intel
Diagnosis (080h)						
ModuleID		4	0	0	<input type="checkbox"/>	Intel
CardType1		5	4	0	<input type="checkbox"/>	Intel
CardType2		5	5	0	<input type="checkbox"/>	Intel
CardType3		5	6	0	<input type="checkbox"/>	Intel
CardType4		5	7	0	<input type="checkbox"/>	Intel

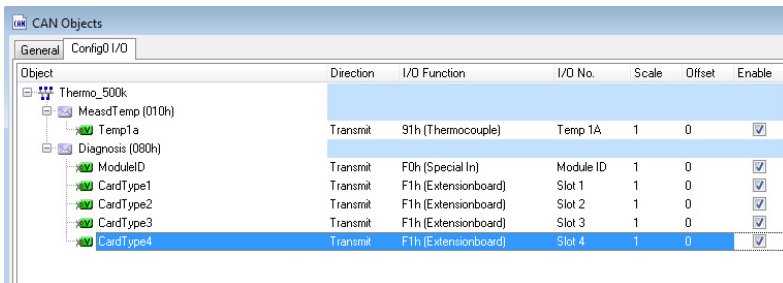
3. Activate the **RTR** check box to change the CAN message to **Transmission Request**.



- Import the three new signals in the configuration: open the context menu with the right mouse button and select **Add defined Variable**.



- Enter these values and assign the virtual module resources:



6. Save the configuration as `exercise 5` on your PC.
7. Transfer (upload) the configuration file to the MU-TC1 via the CAN bus.

The PCAN-Explorer (alternatively PCAN-View) sends a CAN message with the ID 0x080 and the length DLC = 0. The MU-TC1 responds by transmitting the 8 byte long message `Diagnosis` including the 4 card types.

The **Diagnosis** message can be updated accordingly in the PCAN-Explorer symbol file.

▶ Enter the following code to update the message:

```
[Diagnosis]
ID=80h
//   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=----aaaa ----- ----- ----- eeeee ---ffff ---ggggg ---hhhhh
a=Module ID unsigned
e=Board-0 unsigned /e:BoardType
f=Board-1 unsigned /e:BoardType
g=Board-2 unsigned /e:BoardType
h=Board-3 unsigned /e:BoardType
```

Anhang A Literature and References

- └ MU-TC1-CAN hardware manual
- └ PPCAN-Editor 2 integrated software help
- └ Reference of functions blocks
- └ PCAN-View integrated software help
- └ PCAN-Explorer integrated software help
- └ PPCAN protocol reference