PCAN-MiniDisplay

Visualization, Recording, and Playback of CAN Data

User Manual

Document version 2.4.0 (2019-11-19)

Relevant products

Product name	Model	Firmware	Part number
PCAN-MiniDisplay	Display with mounting frame	V1.4.0 or higher	IPEH-002262 (from serial number 150)
PCAN-MiniDisplay Case	Display in casing with push buttons		IPEH-002262-KSM01 (from serial number 12215)

The front page shows the PCAN-MiniDisplay with mounting frame.

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 2.4.0 (2019-11-19)

Contents

1 Introduction

The PCAN-MiniDisplay is used as a human-machine interface for the visualization of CAN data. For the connection to the CAN bus, it has two High-speed CAN connectors. The graphic representation of incoming CAN data is configured before using the unit and then is performed on a TFT display.

The PCAN-MiniDisplay can alternatively be used for data logging. The data traffic is recorded to a memory card and can be replayed or analyzed on a PC later on.

In addition to the built-in PCAN-MiniDisplay there is a version in plastic casing with push buttons available. This can be used to create and evaluate application-specific configurations. Furthermore, the user can directly switch between different configurations during operation.

The specific features of the PCAN-MiniDisplay with casing (IPEH-002262-KSM01) are described in Appendix E on page 82.

Note: This user manual refers to devices that were delivered from July 2017 onwards (see *Relevant products* on page 2). Owners of older PCAN-MiniDisplays please note the information on the firmware update in the section *Internal Statistics* on page 36.

1.1 Properties at a Glance

- CAN connections via 2 High-speed CAN channels (ISO 11898-2)
    - Bit rates from 40 kbit/s up to 1 Mbit/s
    - Complies with CAN specifications 2.0 A/B
    - NXP TJA1041 CAN transceiver with wake-up
- 2 analog inputs
    - Resolution 10 bit
    - Measuring range 0 - 16.5 V
- 1 digital input and 1 digital output
- TFT display with 320 x 240 pixel resolution
- Freely configurable visualization of CAN data via text files
    - Display of texts, CAN data as well as the values of the analog and digital inputs via text labels
    - Formattable output of CAN data as numbers, float numbers, or strings with consideration of enums and multiplexers
    - Display of bitmap images with transparency
    - Signal-dependent display of image lists
    - Presentation of multiple signal curves in plotters
    - Configurable access to key and buzzer functions as well as display brightness via CAN
    - Transmitting CAN frame sequences at the start, operation, and end of a configuration
- Slot for microSD memory card (max. 32 GByte)
- High-speed USB 2.0 port for memory card access using a PC
- Running configurations from the memory card

- Optional auto-start of a single configuration or recording on power-up
- Recording of incoming CAN messages to the internal memory card, optionally including occurring errors
- CAN ID filtering for recordings, separately configurable for each CAN channel
- Conversion of recordings to various output formats using Windows software
- Playback of a recording with optional loop function
- Use of optionally equipped push buttons for menu navigation and to switch between multiple display configurations
- Dimensions: 80 x 55 mm
- Voltage supply from 7 to 30 V
- Operating temperature range from -20 to 70 °C (-4 to 158 °F)

1.2 Scope of Supply

- PCAN-MiniDisplay with mounting frame
- Cables for power supply, CAN, and digital inputs
- 512 MB memory card installed
- Conversion software PEAK-Converter for Windows 10, 8.1, 7 (32/64-bit)
- Configuration examples
- Manual in PDF format

2 Connectors

The connectors on the circuit board are:

- J1 External buttons (2.1 en page 9)
- J2 System (2.2 on page 9)
- J3 and J7 USB (2.3 on page 10)
- J4 JTAG (2.4 on page 12)
- J5 microSD (2.5 on page 12)
- J6 Serial (2.6 on page 13)

For basic use, at least the power supply (J2), a CAN connection (J2), and a microSD memory card (J5) are needed.

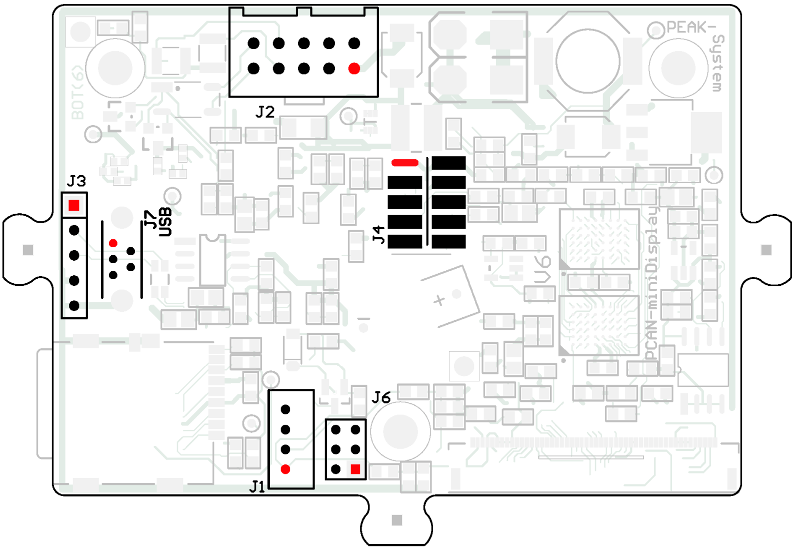


Figure 1: Bottom side of the PCAN-MiniDisplay board;
connectors J1, J2, J3, J4, J6, J7;
pin 1 marked red

2.1 J1 External Buttons

3 push buttons can be connected and used to navigate through the menus of the PCAN-MiniDisplay user interface. This is for maintenance and debugging purposes. For example, a user of the PCAN-MiniDisplay can be enabled to do communication configuration changes or CAN tracing.

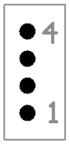


Figure 2: Pin assignment J1 external Buttons

Pin	Function
1	Din1: Down ↓
2	Din2: Up ↑
3	Din3: Enter ↵
4	GND

The 3 push buttons connect each function pin (1 to 3) to GND.

2.2 J2 System

- Power supply
- 2 High-speed CAN channels (CAN1 and CAN2)
- 1 digital output, 1 digital input, 2 analog inputs

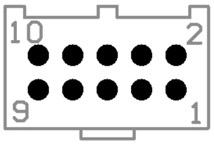


Figure 3: Pin assignment J2 System

Pin	Function
1	Vb (7 - 30 V DC)
2	GND
3	Dout1
4	Din4
5	Ain1
6	Ain2
7	CAN2_L
8	CAN2_H
9	CAN1_L
10	CAN1_H

The digital and analog inputs can be used, for example, for the instrument display. Further technical information for the inputs and the digital output of the J2 System connector can be found in *Technical Specifications* on page 40.

2.3 J3 and J7 USB

The connections J3 and J7 (Mini-USB port) can be used for access to the memory card in the PCAN-MiniDisplay from a PC via a USB connection. The memory card can also be accessed with the PCAN-MiniDisplay switched off. When the device is switched on, the USB connection is interrupted shortly.

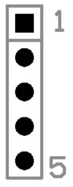


Figure 4: Pin assignment J3 USB

Pin	Function
1	5 V USB
2	D-
3	D+
4	GND
5	GND

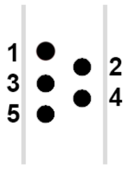


Figure 5: Pin assignment J7 USB (Mini-USB)

Pin	Function
1	5 V USB
2	D-
3	D+
4	not connected
5	GND

2.3.1 Unplugging the USB Connection

Before unplugging the USB cable from the PC or the PCAN-MiniDisplay, the device should be logged out of the operating system. This procedure ensures that the operating system has correctly finished a write process to the memory card of the PCAN-MiniDisplay.

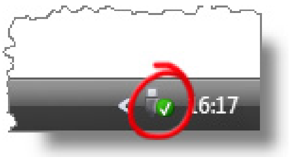


Figure 6: **Safely remove hardware** windows icon in the taskbar notification area

2.3.2 Restriction for PCAN-MiniDisplay Functions

During a USB connection to a PC some functions of the PCAN-MiniDisplay are limited, because the device cannot access the memory card at the same time as the connected PC:

- Scene definitions (instruments) cannot be loaded.
- The recording and playback of CAN traffic (trace) do not work.
- The commands in the `Memory Card` menu do not work.

2.4 J4 JTAG

Access the microcontroller for debugging purposes.

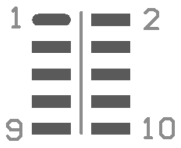


Figure 7: Pin assignment J4 JTAG

Pin	Function
1	GND
2	GND
3	Reset#
4	3.3 V
5	TCK
6	TMS
7	TDO
8	TDI
9	RTCK
10	TRST

2.5 J5 microSD

Slot for a microSD memory card (max. 32 GByte).

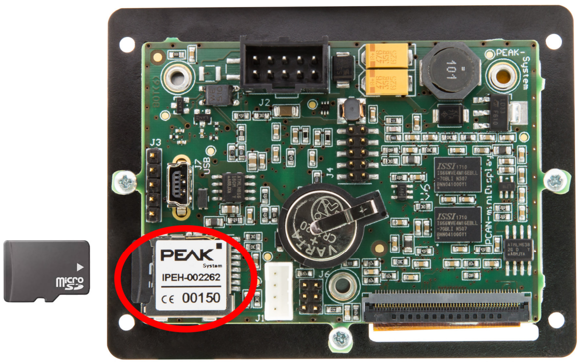


Figure 8: Location of the microSD card slot
on the circuit board of the PCAN-MiniDisplay (bottom view)

2.6 J6 Serial

Provides serial signals for customer-specific extensions.

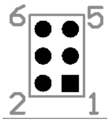


Figure 9: Pin assignment J6 Serial

Pin	Function
1	GND
2	5 V
3	TxD RS-232
4	RxD RS-232
5	DBG
6	Buzzer

Note: The serial pins RxD and TxD are not supported by the standard firmware.

3 Scene Definition

The PCAN-MiniDisplay uses a so-called scene to display CAN data in graphical form.

Note: Running a display configuration as well as recording and playback of CAN messages is not possible at the same time. The PCAN-MiniDisplay can only be used with one of the three functions.

3.1 General Description of an *.ins File

A scene is a collection of instruments, labels, fonts, etc. defined in a text file with the `*.ins` extension. Scenes furthermore include bitmap graphic files (`*.bmp`) and font files (`*.fon`).

The content of a scene definition file is divided in different sections each containing definition entries:

- [global]: general specifications for the scene
- [fontX]: fonts that are used by labels
- [messageX]: messages for the use by variables, multiplexer variables, and for assigning the CAN channel 1 or 2.
- [enumX]: enumeration value lists for a user-defined output of specific values of variables.
- [variableX]: Variables for representing the data of CAN messages or the analog or digital inputs. Output with labels, the plotter, or instruments.
- [txlist_X]: CAN frame sequences that can be transmitted at the start, operation, and end of a configuration

- [labelX]: text outputs of fixed strings or variable
- [instrumentX]: Bitmaps and Bitmap list based instruments for the presentation of variables
- [plotterX]: line writer for the presentation of signal curves

A reference to the keywords to be used in those sections is located in appendix section D.1 on page 47. An example file is shown below.

A definition file can contain **comments** that are led in by double slashes (//). All characters in the line after a double slash are ignored as the `*.ins` file is interpreted by the PCAN-MiniDisplay.

During **load time**, the PCAN-MiniDisplay generates a binary version of a definition (`*.inb`). That one is loaded much faster on future use, because all conversions for panel-internal use are already incorporated. This is the default behavior but can be configured in the Global section.

Altering the definition of a scene can only be done in the text-based file.

Note: If the definition of a scene is altered, the previously generated binary file (`*.inb`) must be manually deleted on the memory card of the PCAN-MiniDisplay, so the loader sees that a new binary must be created from the text file. If you do not delete the binary file, your changes in the definition file will not have any effect.

3.2 Example of an *.ins File

On the memory card or in the download package of the PCAN-MiniDisplay you can find examples of the various configuration options in the Scenes directory.

The following, simple example defines:

- the CAN message "ExampleMsg" with the ID 100h
- the variable "Speed" for the first 8 bits of the transmitted data
- two labels for presenting the variable value
- the display of a background image with an instrument

```
[global]
format=1
compress=1            // 1 => Scene will be compressed after first load
instruments=1         // Number of instruments defined in this scene
messages=1            // Number of CAN messages defined in this scene
variables=1           // Number of variables defined in this scene
labels=2              // Number of labels defined in this scene

[message1]
enable=1              // 1 => Enables message processing of this message
name="ExampleMsg"
canid=0x100           // ID of this CAN message
frametype=0           // 0 => 11 Bit, 1 => 29 Bit
channel=0             // 0 => CAN channel 1, 1 => CAN channel 2
timeout=0             // 1 => Enables timeout function

[variable1]
name="Speed"
messagename="ExampleMsg"
position=0,8          // Bit start and length
byteorder=0           // 0 => Intel, 1 => Motorola
muxtype=0             // 0 => None, 1 => Multiplexer, 2 => Multiplexed
muxval=0              // If muxtype = 2, this variable is updated if its
                         multiplexer (muxtype = 1) has this value
scale=1
offset=0
vartype=0             // Representation of CAN data: 0 => unsigned, 1 =>
                         signed, 2 => float, 4=> string, 5=> enum
datatype=0            // Representation of result: 0 => unsigned, 1 =>
                         signed, 2 => float, 4=> string, 5=> enum
```

```
[label1]
name="label1"          // Name of the label for reference
position=10,20         // X-position,Y-position in pixel; 0,0 is the top
                          left corner
font_idx=0             // Sets the font to be used
length=15              // Number of characters
initval="Current Speed: "
fontcolor=0,175,0      // RGB value
bgcolor=0,0,0          // RGB value black

[label2]
name="label2"          // Name of the label for reference
position=140,20        // X-position,Y-position in pixel; 0,0 is the top
                          left corner
font_idx=0             // Sets the font to be used
fmt_string="%3d km/h"  // Format definition
var_name="Speed"       // Reference to the variable
length=8               // Number of characters
initval="--- km/h"
fontcolor=0,255,0      // RGB value green
bgcolor=0,0,0          // RGB value black
update_rate=0

[instrument1]
pictures=1
name="Background"
no_restore=1
0="Background_320x240.bmp","",0,0,0,0
```

Note: The text file (`*.ins`) must have an empty line at its end.

3.3 Using Several Scenes

It is possible to use more than one scene on the PCAN-MiniDisplay. Scenes can be switched by using the digital inputs (e.g. with push buttons).

To achieve this, a list of scene definition files must be created in a scenes project file (`*.inp`). This list also defines which scene is used on startup.

A reference to the keywords to be used in those sections is located in appendix section D.2 on page 78. An example file is shown below.

```
[general]
version=1

[scenes]
scenes=4
startscene=2
1="speedo0.ins"
2="speedo1.ins"
3="picture1.ins"
4="picture2.ins"
```

Note: The text file (`*.inp`) must have an empty line at its end.

4 Steps to Operation

On power-on the PCAN-MiniDisplay looks for the internally saved startup options. A scene for display or the tracer for CAN recording can be started automatically.

Tip: At delivery, the PCAN-MiniDisplay is configured to start the following scene:
`/MiniDisplay/Scenes/Default/Default.inp`

This chapter describes the steps to put the PCAN-MiniDisplay to operation. Please go through each of the following sections.

4.1 Prerequisites

- microSD memory card, max. 32 GByte
  (512 MByte card is enclosed)
- PC with SD card reader (including adapter for microSD card)
  or USB connection PC to PCAN-MiniDisplay (connected to J3)
- Power supply 7 to 30 V DC (connected to J2)

4.2 Preparing the Memory Card

So that the device starts up properly, the memory card must contain a specific directory structure. This is present on delivery. You can add your own scenes to the directory structure.

- Do the following to prepare the memory card with your scenes:

  1. Establish a USB connection between the PCAN-MiniDisplay and your PC in order to access the microSD memory card.

     As an alternative, remove the card from the PCAN-MiniDisplay and insert it into a card reader attached to your PC.

     The memory card appears as a mass storage device.

  2. Copy one or several scenes that you have created previously (see *3 Scene Definition* on page 13) into the following directory of the memory card: `/MiniDisplay/Scenes/`

     Each scene must have its own subdirectory.

  3. End the USB connection in the PC's operating system and disconnect the USB cable.

     If prepared in a PC, remove the microSD memory card from it and insert it into the slot of the PCAN-MiniDisplay.

- If needed, do the following to restore the directory structure on the card:

  1. Establish a USB connection between the PCAN-MiniDisplay and your PC in order to access the microSD memory card.

     As an alternative, remove the card from the PCAN-MiniDisplay and insert it into a card reader attached to your PC.

     The memory card appears as a mass storage device.

  2. Format the memory card with the FAT32 file system.

  3. Download the PCAN-MiniDisplay download package from our website.

4. Copy the directory `MiniDisplay` including all subdirectories and files to the root directory of the memory card.
   For supplementary information, see Appendix C *Structure on the Memory Card* on page 45.

5. Establish a USB connection between the PCAN-MiniDisplay and your PC in order to access the microSD memory card.

   As an alternative, remove the card from the PCAN-MiniDisplay and insert it into a card reader attached to your PC.

   The memory card appears as a mass storage device.

4.3 Powering On the Device

The PCAN-MiniDisplay is powered on by applying a supply voltage in the range of 7 to 30 V DC to connector J2. The device starts up automatically. If not altered after delivery, the PCAN-MiniDisplay shows the scene with the name `Default.inp`.

If the device is switched off via the **Power off** menu item, it can be switched on again via pin 1 of the J1 connector (see chapter *J1 External Buttons* on page 9). For the housing version, simply press the down arrow key.

4.4 Menus

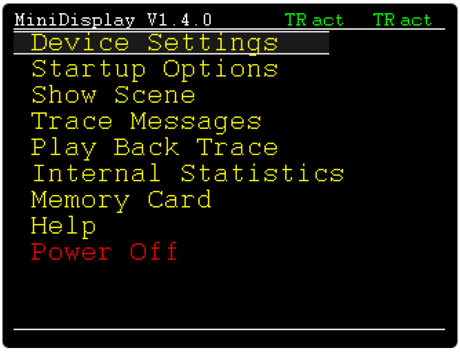


Figure 10: Main menu

The PCAN-MiniDisplay provides a user interface for accessing the individual functions and configuring the settings.

The navigation is done via the digital inputs of connector J1 that can be equipped with push buttons, for example. See section 2.1 on page 9 for details about the connection.

The PCAN-MiniDisplay Case is equipped with the appropriate push buttons. In that case browsing through the menus and functions is done with the buttons Up ↑ and Down ↓. A selected item is activated with the button Enter ↵.

While displaying a scene, pressing the Enter ↵ button takes you back to the main menu.

Tip: The use of the menus is not needed for displaying a scene.

4.4.1 Status Indication

While using the menus, icons on the upper right of the screen indicate the status of the CAN bus communication for the CAN channels 1 and 2 (High-speed CAN).

Icon	Meaning	
`T` `R`	CAN traffic: Blinking: Green: Yellow, red:	T = Transmit, R = Receive Outgoing/incoming CAN messages Regular traffic Faulty traffic

`act`
`pas`
`off`
	Informs about the CAN controller status (**act**ive, **pas**sive, bus-**off**). If the bus-off state was reached due to too many send errors, no more CAN messages are received or transmitted. The PCAN-MiniDisplay automatically resets the CAN controller after a certain time.	

4.5 Setting the Startup Options

Via the main menu entry `Startup Options` you can set the behavior on power-on.

Do the following to display a scene at startup:

Note: At delivery, the PCAN-MiniDisplay is configured to start the scene called `Default.inp`.

1. Set the marker ☒ after `Start scene`.
2. Click on `..` after `Select scene`.
3. Select a scene (`*.ins`) or a project (`*.inp`) to be shown at startup.

   The selected file is displayed beside `Select scene` `..`.
4. Click `OK`.

Do the following to start tracing at startup:

1. To the right of `Start tracer`, select the checkbox and set the marker ☒.
2. At `Tracer timeout`, enter the time span for stopping the tracer if no CAN messages are received. The `Never` entry keeps the tracer running.
3. Click `OK`.

5 CAN Tracing

5.1 Recording CAN Traffic

Note: The recording or playback of CAN messages is not possible simultaneously with displaying a scene.

➢ Menu item `Trace Messages`

With this function, the whole incoming CAN traffic including RTR frames and error frames is recorded on the memory card of the PCAN-MiniDisplay. The timing is regarded too.

`Trace CAN 1 / Trace CAN 2`

With these settings, the information to be recorded can be selected. The following options are available:

- `Off` (no recording)
- `Data` (CAN data)
- `Errors` (error frames)
- `Data + Err.` (CAN data and error frames)

`Trace file size`

This option is used to limit the maximum file size of a recording from 10 Mbyte up to 1 GByte.

Trace timeout

If no CAN message is received within this time span, the device switches to sleep mode. To restart the device, a keystroke or a wake-up via the CAN bus is required. The `Never` entry keeps the tracer running.

Note: Sleep mode is not supported for scenes.

Multi file trace

If this option is active, the recording is saved to multiple files. If the maximum size of the trace file `Trace file size` is reached, the recording is continued in a new file. The given `File` name is supplemented by a consecutive number. If the option is inactive, the recording is stopped, when reaching the maximum file size.

Trace type

This option determines the type of recording. The following options are available:

- `linear` The recording is progressive. A two-digit index from 0 to 99 is added to each new trace file.
- `circular` A certain number of trace files is created while recording. If the maximum number of files is reached, the oldest is deleted and the index of all other files is reduced by 1. Finally, a new file is created in which the recording is continued.

File count circular

This value indicates the maximum number of trace files that are created for a recording of the `Trace type` `circular`.

Session type

This setting determines whether the files created for a recording of the `Trace type` `circular` are overwritten or not. The following options are available:

- `Single session` With each recording, the file sequence with the name `trace_xx.btr` is overwritten.
- `Data` A new sequence of files is created with each recording.

Later, a trace file can be used for playback of the recorded CAN messages on the CAN bus (see following section 5.2 on page 29). As an alternative, it is possible to convert the recording on a PC to another format for further use and for evaluation (see section 5.3 on page 30).

Note: A filter may be applied to incoming CAN traffic which allows only CAN messages with specific IDs to pass through. See chapter 5.4 on page 32 for details about filtering.

Do the following to record:

1. Make sure that there is no USB connection between the PCAN-MiniDisplay and your PC.
2. Click on `Start`.

   The recording is saved to the indicated `File`.
3. End the recording with `Stop tracing`.

Indication	Meaning
`File`	Name of the trace file of the current recording. The file name (`trc00000.btr`) is automatically put together with a consecutive number. When recording in multiple files, another two-digit index is added.
`Start time`	Specifies the start time of the recording.
`CAN queue level in %`	Current and maximum fill level of the receive queue (latter in parentheses). If the queue has reached a fill level of 100 percent, most likely some incoming CAN messages were not recorded.
`CAN messages total`	Number of CAN messages that have been recorded.
`File size`	Current size of the trace file in kByte and already used storage space relative to the maximum possible file size. During recording, the trace file grows in 512-byte blocks, each containing 25 CAN messages. Thus, 1 MByte can hold 51200 CAN messages.
`File count`	Number of files recorded. When the recording is done with `Trace type` `circular`, the number is incremented linearly regardless of the current file index.
`Trace type`	Indicates the used trace type.
`Session type`	Indicates the used session type.

5.2 Playing Back Recorded CAN Traffic

➢ Menu item Play Back Trace

The PCAN-MiniDisplay can play back CAN messages from a binary trace file (`*.btr`) onto the connected CAN bus. The timing of the CAN messages, as it occurred originally during recording of the trace file, is preserved.

Do the following to play back a trace file:

1. Make sure that there is no USB connection between the PCAN-MiniDisplay and your PC.
2. In the main menu, select Play Back Trace.
3. Use Select file and select the trace file (`*.btr`) for playback from the list.
4. If you don't want the trace to be played back repeatedly, change the Loop mode from Infinite to the desired number of cycles (1 to 9).
5. If you use a trace file that contains CAN messages from several CAN channels, select the CAN channels Channel 1 / Channel 2 to be used for the playback.
6. Click on Play to start the playback.
7. You can Pause playback. Now you have these options:

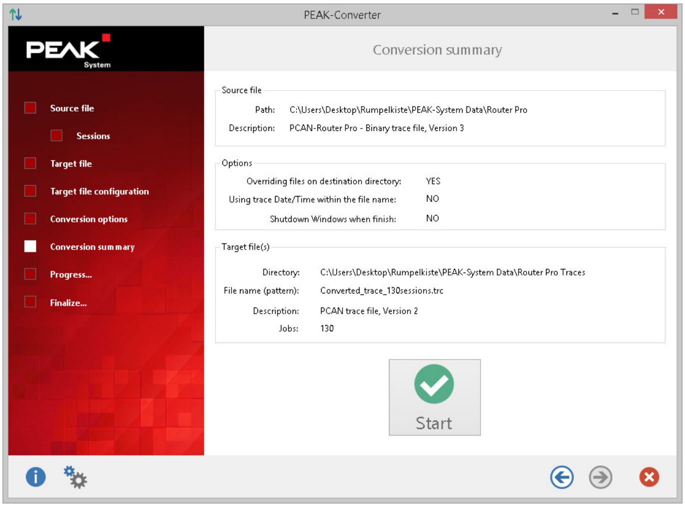
Function	Executed action
Exit	Ends the playback.
Restart	Restarts the playback from the beginning of the trace file.
Reset	Resets the CAN controller. Useful after troubleshooting on the CAN bus.
Continue	Continues the playback from the point where it was interrupted.

5.3 Using the Recorded CAN Traffic on the PC

The recorded CAN traffic can be load with a PC via a USB connection from the memory card of the PCAN-MiniDisplay. It is stored in binary-coded trace files `*.btr` in the following directory:

```
/MiniDisplay/Traces/
```

For further use you must convert the data in an appropriate format. For this, the Windows software PEAK-Converter is supplied on the memory card of PCAN-MiniDisplay.



Conversion summary

Source file

Path:	C:\Users\Desktop\Rumpelkiste\PEAK-System Data\Router Pro
Description:	PCAN-Router Pro - Binary trace file, Version 3

Options

Overriding files on destination directory:	YES
Using trace Date/Time within the file name:	NO
Shutdown Windows when finish:	NO

Target file(s)

Directory:	C:\Users\Desktop\Rumpelkiste\PEAK-System Data\Router Pro Traces
File name (pattern):	Converted_trace_130sessions.trc
Description:	PCAN trace file, Version 2
Jobs:	130

Figure 11: User interface of the PEAK-Converter

Possible conversion targets:

Target format	**File extension**	**Explanation/usage**
PCAN-Trace	`.trc`	Text-based trace format by PEAK-System; Used for viewing and playback of the data in the software PCAN-Explorer or PCAN-Trace.
Vector ASC Trace	`.asc`	Text-based trace format by the Vector company that also can be used by some third-party programs.
Character Separated Values (CSV)	`.csv`	Common, text-based format for import into a spreadsheet (semicolon as separator).

For further use of the trace data proceed as follows:

1. Connect the PCAN-MiniDisplay to the PC with the provided USB cable. The PCAN-MiniDisplay does not need to be powered on.
2. Under Windows, start the `PEAK-Converter.exe` program from the memory card of the PCAN-MiniDisplay which resides in the `/MiniDisplay/Tools` directory.
3. Select a trace file (`*.btr`) as source. You can find the trace files in the following directory: `/MiniDisplay/Traces/`
4. Specify a destination file and select the desired target format.

5.4 Filtering the CAN Traffic at Recording

When recording a trace file, all CAN traffic is processed. However, it is possible to filter the incoming CAN messages.

A text file with a list of the CAN IDs to be filtered is created for this purpose. The listed IDs pass through the filter. All others do not. The filter file is placed on the memory card and loaded via the device settings of the PCAN-MiniDisplay.

Do the following to apply a CAN ID filter:

1. Create a filter file with the extension `*.flt` with a text editor. The format of the file is described in the appendix section D.3 on page 79.
2. List the CAN IDs that can pass the filter.
3. Connect the PCAN-MiniDisplay to the PC with the provided USB cable. The PCAN-MiniDisplay does not need to be powered on.
4. Copy the filter file to the memory card in the `/MiniDisplay/Filters/` directory.
5. On the PCAN-MiniDisplay, navigate to `Device Settings`. Select `Ch.1` `Filter` for CAN channel 1 and `Ch.2` `Filter` for CAN channel 2.
6. Select the previously created filter file (`*.flt`).
7. Use `OK` or `Save&OK` to apply the changed settings.

6 Settings and Maintenance

6.1 Device Settings

Here you specify the settings for the connection to a CAN bus and those for the use of the device.

When you have changed settings, save them permanently with `Save&OK`. If you want to use the changed settings only temporarily (during the current session), choose `OK`. The next time the device is switched on, the previous settings are restored.

6.1.1 Ch.1/2 Detect CAN bitrate

If the bitrate of the connected CAN bus is unknown, the PCAN-MiniDisplay can automatically detect it. This requires data traffic on the CAN bus.

Bitrates from the following series are recognized (kbit/s): 1000; 800; 500; 250; 200; 125; 100; 95.2; 83.3; 50.0; 47.6; 33.3; 20.0; 10.0.

6.1.2 Ch.1/2 CAN bitrate

Manual selection of the bitrate. Besides the fixed bitrate values, the list contains eight user-defined bitrates. They are managed in the `User CAN bitrates` setting.

6.1.3 Ch.1/2 Listen-only mode

If you want the device to not affect the traffic on the CAN bus, i.e. use it as pure monitoring tool, the Listen-only mode must be active (`On`). The device will neither acknowledge nor transmit CAN (error) frames.

6.1.4 Ch.1/2 Filter

CAN ID filtering is used for recording only and can be applied independently to each CAN channel by selecting a filter file (`*.flt`) from the memory card. The selected file is shown beside the menu entry and is applied to the corresponding CAN channel.

To **disable** CAN ID filtering, go on `Exit` in the file selection screen. `None` is then shown beside the menu entry.

For more information about filtering, see chapter 5.4 on page 32.

6.1.5 User CAN bitrates

To adapt to specific conditions, experts may directly access the bus timing registers (BTR) of the integrated CAN controller. The register settings are analog to those of a SJA1000 CAN controller operating at 16 MHz clock frequency.

Eight user-defined entries can be edited. Each entry contains a 2-byte value (4 hexadecimal digits) for the bus timing registers and an arbitrary name. User-defined bitrates appear later with their name in the `CAN bitrate` list below the fixed bitrate values.

Tip: With the Bit Rate Calculation Tool from PEAK-System you can determine the register values for your bitrate. The software is available free of charge for Android, iOS, and Windows on our website.

For each entry the table for editing shows the parameters that result from the given register values: the bitrate, the sample point (SP), and the synchronization jump width (SJW).

To reset an entry (Name = UserX, Bitrate = undef), set the BTR value to 0000.

6.1.6 Beeper

The PCAN-MiniDisplay can give acoustic feedback to several events. Among other, a change of the CAN bus status is signalized. The `Off` setting disables the acoustic signal function of the device.

6.1.7 Display rotation

When this option is active `On`, the image output is rotated by 180 degrees.

6.1.8 Screensaver timeout

The brightness of the screen is reduced after the selected time span of non-use. The power consumption is therefore reduced.

6.1.9 Show startscreen

After power-on, the PCAN-MiniDisplay shows a bitmap file for a few seconds as start screen. This behavior can be switched on and off here.

The bitmap file can be replaced with your own graphic. For more information, see section 7.1 on page 39.

6.1.10 Date & time

With `Set` the device date and time are adjusted. Date and time are used when saving files to the memory card.

6.1.11 Reset file index

File names of traces to be saved get a number coming from a counter. The current count is indicated in parentheses and can be set to 0 by clicking `Reset`.

6.2 Internal Statistics

The page gives an overview about the device's internals. The specifications are usually used for support purposes. The screen also shows the serial number of the PCAN-MiniDisplay.

Furthermore, hardware functions are available for maintenance of the device. They are described briefly in the following.

Important note: Misapplication of these functions can lead to the unavailability of the device.

Update Firmware

Firmware updates files (`*.bin`) can be placed in the directory `/MiniDisplay/Firmware/` on the memory card. The update process is started by selecting a firmware file.

Important note: Older PCAN MiniDisplays without casing (IPEH-002262, before serial number 150) and casing (IPEH-002262-KSM01, before serial number 12215) must be updated to the firmware 1.1.7 before they can be updated to firmware 1.2.1 or higher.

Factory Defaults

All settings are reset to their default states defined by the current firmware.

Bootloader

Starts the boot loader for a firmware update via CAN.

6.3 Memory Card

Directories and bitmaps on the memory card can be displayed with the PCAN-MiniDisplay.

Note: The PCAN-MiniDisplay cannot access the memory card as long as a USB connection to a PC is established.

Show Directory

Shows the directories on the memory card in order to see which files exist.

View Bitmap

Only bitmap files (`*.bmp`) are shown in the directories (e.g. from instruments).

Click on a bitmap to view it. Click again to leave the bitmap view.

7 Using Own Pictures

The PCAN-MiniDisplay can be used to integrate your own pictures. These can be project-specific images when switching on the display or for individual instrument displays.

The following table provides information about the necessary image properties for using your own images on the display:

Properties of the pictures	
File name	`*.bmp`
Format	Windows bitmap
Resolution	max. 320 x 240 pixels
Color depth	24 bits
Transparency	Without alpha channel, transparency by black-and-white mask (no gradation possible)

The following figure shows how to create your own image with transparency:

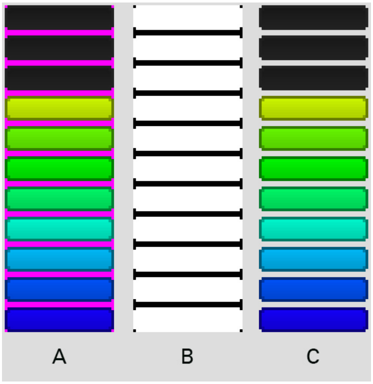


Figure 12: Sample images from the instrument display of the PCAN-MiniDisplay

A: Initial picture without transparency
B: Transparency mask (black and white)
C: Initial picture with transparency

Note: The PCAN-MiniDisplay shows the images with a limited color range.

7.1 Alternative Startscreen

An alternative start screen can be assigned to each PCAN-MiniDisplay. A bitmap file must be located on the internal memory card. By default, the manufacturer start screen (PEAK logo) is displayed.

The following table shows the file name and storage path to be used:

Properties of the start screen	
File name	`Intro.bmp`
Storage path on the memory card	`/MiniDisplay/Intro.bmp`

8 Technical Specifications

Inputs and output	
Digital input (J2, pin 4)	10 kΩ pull-up resistor Switching threshold: 0: <= 1.8 V (switch against ground) 1: >= 6.6 V
Digital output (J2, pin 3)	Low-side switch max. output voltage 32 V max. output current 500 mA
Analog inputs (J2, pin 5 and 6)	Input voltage range from 0 to 16.5 V 10-bit A/D converter: 10 bit Range of values: 0 = 0 V 1023 = 16.5 V

Display	
Type	a-Si TFT active matrix
Resolution	320 x 240 pixels
Active area	57.6 x 43.2 mm (W x L)

Memory card	
Type	microSD
Max. capacity	32 GByte
File system	FAT; FAT32 if capacity > 2 GByte
Name of the USB device	PCAN-MiniDisplay

CAN	
Specification	ISO 11898-2, High-speed CAN 2.0A (Standard format) and 2.0B (Extended format)
Bit rates	40 kbit/s - 1 Mbit/s
Transceiver	NXP TJA1041T, wake-up-capable
Internal termination	none

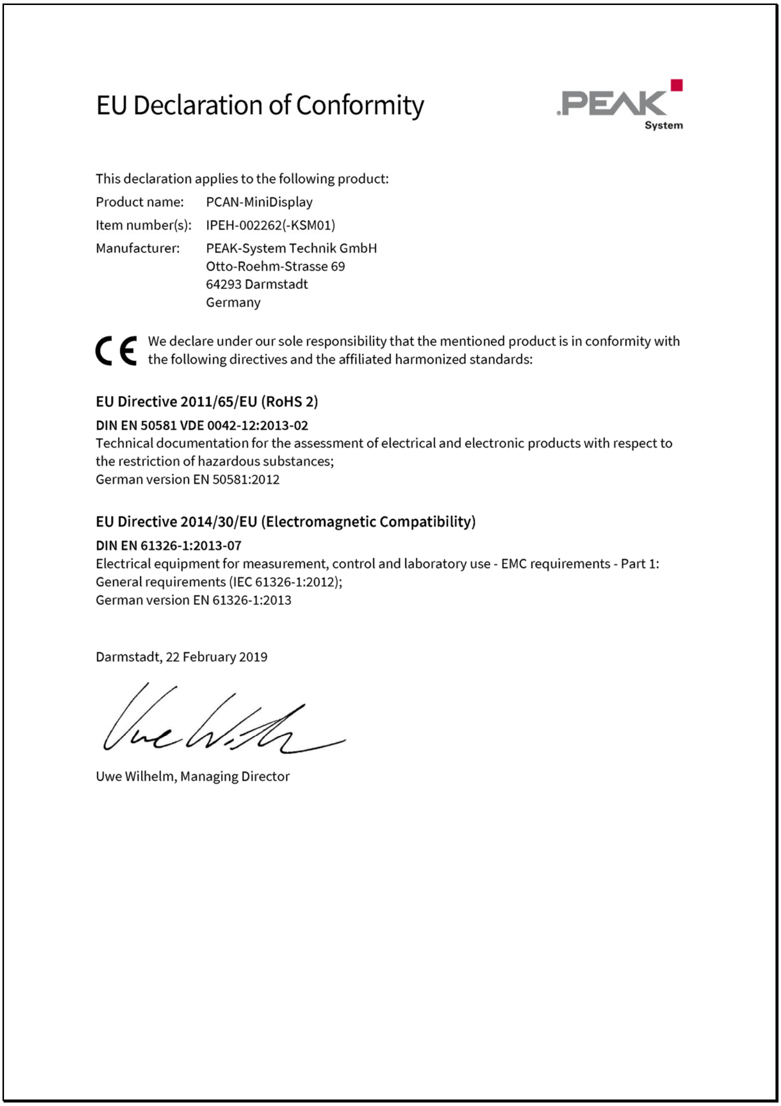
Power supply	
Supply voltage	7 - 30 V DC
Current consumption	135 mA (at 12 V) typical
Current consumption in screensaver mode	70 mA (at 12 V) typical

Environment	
Operating temperature	-20 - +70 °C (-4 - +158 °F)
Temperature for storage and transport	-30 - +80 °C (-22 - +176 °F)
Relative humidity	15 - 90 %, not condensing

Measures	
Size	With frame: 82 x 62 x 18.3 mm (W x L x D) Without frame: 80 x 55 x 18.3 mm (W x L x D) See also dimension drawings Appendix B on page 43
Weight	55 g

Conformity	
EMV	Directive 2014/30/EU DIN EN 61326-1:2013-07
RoHS 2	Directive 2011/65/EU DIN EN 50581 VDE 0042-12:2013-02

Appendix A CE Certificate



EU Declaration of Conformity

This declaration applies to the following product:

Product name:	PCAN-MiniDisplay
Item number(s):	IPEH-002262(-KSM01)
Manufacturer:	PEAK-System Technik GmbH Otto-Roehm-Strasse 69 64293 Darmstadt Germany

CE We declare under our sole responsibility that the mentioned product is in conformity with the following directives and the affiliated harmonized standards:

EU Directive 2011/65/EU (RoHS 2)

DIN EN 50581 VDE 0042-12:2013-02
Technical documentation for the assessment of electrical and electronic products with respect to the restriction of hazardous substances;
German version EN 50581:2012

EU Directive 2014/30/EU (Electromagnetic Compatibility)

DIN EN 61326-1:2013-07
Electrical equipment for measurement, control and laboratory use - EMC requirements - Part 1: General requirements (IEC 61326-1:2012);
German version EN 61326-1:2013

Darmstadt, 22 February 2019

[illegible]

Uwe Wilhelm, Managing Director

Appendix B Dimension Drawing

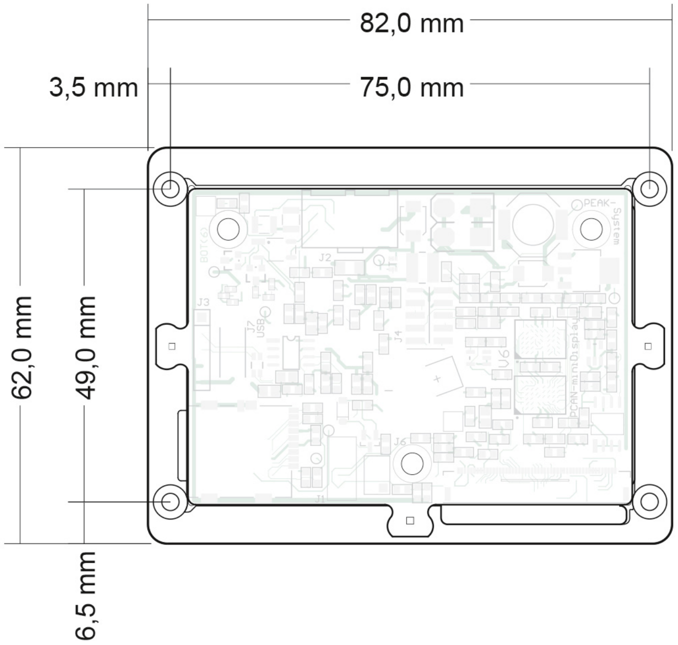


Figure 13: View onto the PCAN-MiniDisplay with mounting frame (IPEH-002262).
The installation depth is 18.3 mm.

The figure does not correspond to the original size.

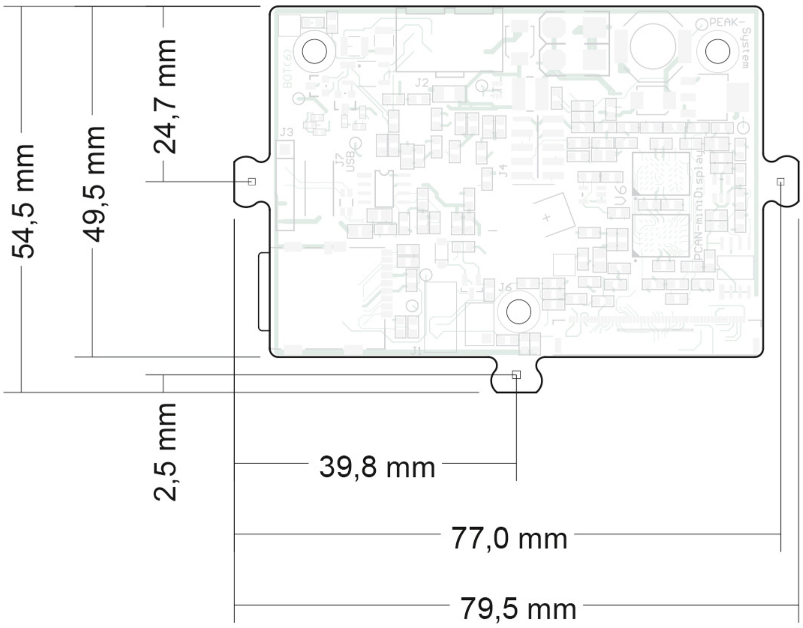


Figure 14: View onto the PCAN-MiniDisplay without mounting frame (IPEH-002262).
The installation depth is 18.3 mm.

The figure does not correspond to the original size.

Appendix C Structure on the Memory Card

The following table shows the structure of the MiniDisplay directory on the memory card and explains the functions.

Directory Structure	Function
/MiniDisplay/	Fixed root directory for files that are accessed by the PCAN-MiniDisplay or that are related to the device.
Intro.bmp	Screen at startup of the device (320 x 240 pixels).
Documentation/	Documentation of the PCAN-MiniDisplay in PDF format.
Filters/	Directory for filter files (*.flt).
Firmware/*.bin	Location of binary files (*.bin) used for updating the PCAN-MiniDisplay firmware. **Please note:** If you use a firmware lower than 1.2.x, you must upgrade to version 1.1.7 before installing a firmware 1.2.x.
Help/*.dhp	Files of the PCAN-MiniDisplay device help (*.dhp). It is recommended to update these files in connection with a firmware update.
HTML/	Data for the presentation of the ReadMe files.
Resources/	This folder contains fonts, bitmaps, and other resources for creating new scenes. PCAN-Explorer projects for testing the PCAN-MiniDisplay example scenes are available as well.
Scenes/	This directory is used for scenes. It contains the default scene and many examples. At delivery of the device, the Default scene is set to automatically be shown at startup.
*.ins	Scene definition file (see chapter 3 page 14).
*.inb	Binary counterpart to a scene (*.ins) for faster loading (see chapter 3 page 14).
*.inp	List of scenes that can be switched with the digital inputs.

Directory Structure	Function
`*.fon`	Bitmap font to be used in a scene.
`*.bmp`	Bitmap to be used in an instrument.
`Tools/`	Software tools to be used with the PCAN-MiniDisplay.
`PEAK-Converter.exe`	Windows program PEAK-Converter for converting a recorded binary trace file (`*.btr`) to another format.
`Traces/`	Fixed subdirectory for traces that are recorded by the PCAN-MiniDisplay.
`trc00000.btr`	Binary-coded trace data from the trace function, usable for playback or otherwise after conversion on the PC; consecutive numbering by the internal counter.

Appendix D Definition Files Reference

The keywords of the following definition files are described in this appendix:

- D.1 *Scene Definition (*.ins)* starting below
- D.2 *Scenes Project File (*.inp)* on page 78
- D.3 *CAN ID Filters (*.flt)* on page 79

D.1 Scene Definition (*.ins)

The following describes all available keywords for a scene definition. They are written into a `*.ins` text file and structured in sections. Section names are indicated by square brackets, e.g. [global]. Entries each are put into a line containing a keyword with an equal character (no spaces!) and the assigned value.

Note: The text file (`*.ins`) must have an empty line at its end.

D.1.1 [global] Section

format

Determines the file format version. Up to now only version 1 is existing and valid.

```
format=1
```

compress

If the compress function is active, a compressed, binary version (*.inb) of the text file (*.ins) is generated, so that the scene can be loaded faster at the next start.

Value	Description
0	No compressed version of the scene is created. The text file (*.ins) is loaded for executing the scene.
1	A compressed version of the scene is created. The binary file (*.inb) is loaded for executing the scene.

Note: If the definition of a scene is altered, the previously generated binary file (*.inb) must be deleted on the memory card of the PCAN-MiniDisplay. Otherwise no new binary file will be created and changes in the text file will be ignored.

protect

The protect function is related to the creation of the binary version of a scene using the compress function. If protect is active, the binary file is encrypted during creation and can only be executed on the device on which it was created.

Value	Description
0	The binary file of the scene will not be encrypted.
1	The binary file of the scene will be encrypted.

Note: If the definition of the scene is confidential, the content of the text file (*.ins) can be deleted after creating the binary file (*.inb). The binary file can be started without any problems via the empty text file, since its contents are ignored as long as a binary file exists.

bitrate0 bitrate1

Determines the bitrate of the High-speed CAN channel 1 (bitrate0) and the High-speed CAN channel 2 (bitrate1). If these keywords are omitted, the bitrates defined in the `Device Settings` menu are valid (`BusOff` at delivery). If a scene with a defined bitrate is closed, that bitrate stays active.

Note: During the device's startup, the bitrate defined in the `Device Settings` menu is active for a short time before the bitrate of the scene is enabled.

Possible bitrates (in kbit/s): 1000, 800, 500, 250, 200, 100, 95, 83, 50, 47, 33, 20, 10

```
bitrate0=250
bitrate1=20
```

instruments

Count of instruments in this scene, range: 1 … 32

```
instruments=4
```

plotter

Count of used plotters, range: 1 … 4

```
plotter=1
```

messages

Count of used CAN messages, range: 1 … 96

```
messages=1
```

enums

Count of used enums, range: 1 … 20

```
enums=2
```

variables

Count of used variables, range: 1 … 96

```
variables=7
```

labels

Count of used labels, range: 1 … 64

```
labels=5
```

fonts

Count of additional fonts that are specified for this project, range: 1 … 27

There are 3 internal fonts available. To use the internal fonts, use font_idx instead of a font name. For more information, see D.1.3 *[fontX] Section* on page 55.

```
fonts=2
```

flashing_time

Flashing frequency for instruments with the “flashing” keyword, range 50 … 2500, default 500. The value is the time in milliseconds for the “on-time” and the “off-time”.

```
flashing_time=250  // 250 ms on, 250 ms off, resulting in 2 Hz
```

D.1.2 [instrumentX] Section

Each instrument is indicated with a [instrumentX] section. X is a consecutive number, starting with 1. The number of instruments is defined with the "instruments" keyword in the [global] section.

```
[instrument1]
```

pictures

Count of pictures per instrument, range: 1 … 99

```
pictures=4
```

name

Name of the instrument (max. 23 characters). Only for documentation. The string must be written with quotation marks.

```
name="turn_left"
```

var_name

Name of the variable that controls the instrument. The string must be written with quotation marks.

```
var_name="speed"
```

initval

Determines the initial value of the instrument until the first real CAN data for this instrument is received.

```
initval=267
```

no_restore

Determines the handling of the background while drawing.

Value	Description
0 (default)	Before the instrument sequence will be drawn, the background is saved. Then the background could be restored before a new instrument sequence is drawn. This can be useful if the sequences have different sizes.
1	The picture sequence will be drawn without saving the background – makes only sense if all pictures of the instrument have the same size and are drawn at the same position. Used for optimization of CPU load.

```
no_restore=1
```

restore

Same as no_restore above, but with inverted logic.

transparency

Defines whether transparency areas must be considered while drawing.

Value	Description
0	A sequence will be drawn not regarding the background. This is used when the background is part of the sequence.

1	While loading the `*.ins` file, every single picture is recalculated with the background. Therefore, instrument0 must be defined with a global background picture as the one and only item with the keyword "no_restore=1". An overlapping of different transparency images is not possible here. There are 2 options available for the recalculation: a) The picture has a transparency color (RGB value 255,0,255). b) The picture has an alpha channel. The alpha channel is defined by a separate grayscale bitmap. The grayscale values define the transparency of the foreground picture. RGB value 255,255,255: foreground picture is available to 100 %, no part of the background can be seen. RGB value 127,127,127: foreground is available to 50 % and 50 % of the background can be seen. RGB value 0,0,0: no foreground picture is available, 100 % of the background picture can be seen.
2	With value 2 is an overlapping of different transparency images possible.

```
transparency=1
```

offset

All pictures from the picture list of this instrument will be moved in x and y direction (offset).

1st parameter: x offset (range: 0 ... 319)
2nd parameter: y offset (range: 0 ... 239)

```
offset=100,50
```

background

Defines an individual background picture for this instrument.

Value	Description
0	No background for this instrument.
1	The instrument has an individual background picture which will be used if transparency=1. While loading the `*.ins` file, the pictures is calculated in the background. By default (background=0) the instrument0 is the global background picture. Nevertheless, for each instrument an individual background picture may be defined. The individual background picture must be the last picture of the picture list. The background picture does not have an alpha channel, so it replaces the global background picture from instrument0. Typically, an individual background pictures has the same size as the single pictures of the instrument.

```
background=1
```

update_rate

Update rate of this instrument in milliseconds. If not defined, the default update rate of 50 ms is used. An instrument may be updated with each new incoming CAN message. However, it could happen that the current value is not read because of a too high update rate. This keyword limits the update rate to ensure that the update of the instrument stays reliable.

```
update_rate=500   // Redraws the instrument
                  // (two times a second at most)
```

Picture List

The picture list of the instrument consists of the number of entries as specified in the keyword "pictures". The value of the variable assigned via "var_name" determines which picture from the list is displayed.

Format of a picture list entry:
`<index>=<pic-name>,<transparency-name>,<xpos>,<ypos>,<low-val>,<high-val>`

Part of picture list entry	Description
<index>	Number of the picture. Must be in the range {0 … pictures - 1}.
<pic-name>	File name of the picture without path. The picture must be stored in the same directory where the `*.ins` file is stored.
<transparency-name>	File name of the grayscale picture (alpha channel). If this picture does not exist, an empty string "" must be provided. File name without path (like <pic-name>).
<xpos>	x position {0 … 319}. Will be added to the parameter x-Offset.
<ypos>	y position {0 … 239}. Will be added to the parameter y-Offset.
<low-val>	The lowest variable value for displaying the picture.
<high-val>	The highest variable value for displaying the picture.

```
0="arrow_left_off.bmp","",20,100,0,3
    // Show picture "arrow_left_off.bmp"
    // at position x=20, y=100
    // if the variable is in the range of 0 to 3
1="arrow_left_on.bmp","",20,100,4,7
    // Show picture "arrow_left_on.bmp"
    // at position x=20, y=100
    // if the variable is in the range of 4 to 7
```

D.1.3 [fontX] Section

Each used font is indicated with a [fontX] section. X is a consecutive number, starting at 1. The number of fonts is defined by the “fonts” keyword in the [global] section.

```
[font1]
```

font

Filename of the bitmap font. The string must written with quotation marks. Extension of a font file name is `*.fon`.

```
font="arial_20.fon"
```

The following fonts are available on the PCAN-MiniDisplay:

- `lucida_13_18.fon`
- `century_13_16.fon`
- `century_16_18_numbers.fon`
  (only digits from 0 to 9)
- `Palatino_Linotype_12_14.fon`
- `Palatino_Linotype_16_20.fon`
- `Palatino_Linotype_24_28.fon`
- `Palatino_Linotype_24_32_numbers.fon`
  (only numbers from 0 to 9)
- `Palatino_Linotype_24_32_numbers_bold.fon`
  (only numbers from 0 to 9)
- `Courier_13_18_bold.fon`
  (bold characters)
- `gear_40_49.fon`
  (numbers from 0 to 7 are displayed as N, 1, 2, 3, 4, 5, 6, R in a size of 40 x 49 pixels)

name

Name of the font, as a reference for an element that uses this font.

```
name="arial_20"
```

type

Type of the font. Currently only monochrome fonts with fixed size are available (type=0).

```
type=0
```

D.1.4 [messageX] Section

The definition of each CAN message is initiated with a [messageX] section. X is a consecutive number, starting with 1. The number of messages is defined with the "messages" keyword in the [global] section.

Variables can refer to the CAN messages defined in this way.

```
[message1]
```

enable

Value	Description
0	Processing of the respective [messageX] is deactivated.
1	Processing of the respective [messageX] is activated.

```
enable=1
```

canid

Specification of the 11-bit or 29-bit CAN ID of a message as an integer value (decimal or hex).

```
canid=256
canid=0x100
```

frametype

Defines the CAN frame type as Standard Frame or Extended Frame.

Value	Description
0	Standard Frame (11-bit CAN ID)
1	Extended Frame (29-bit CAN ID)

```
frametype=0  // Standard Frame with 11-bit ID
```

channel

Defines the CAN channel receiving this message.

Value	Description
0	CAN channel 1
1	CAN channel 2

```
channel=0  // Use first CAN channel
```

timeout

If a [messageX] was not received in the specified time span (up to 2^32 milliseconds possible), all variables of this [messageX] are set to the respective "timeoutval" (see *timeoutval* on page 66).

Value	Description
0	timeout disabled
>0	timeout is enabled. The value specifies the timeout delay in milliseconds.

```
timeout=1000
```

Example for a [messageX]

```
[message1]
enable=1            // Processing of the messages is enabled
name="testmessage1"// Name of the message
canid=0x125
frametype=0         // 0: 11-bit, 1: 29-bit
channel=0
timeout=1000
```

Example for a [variableX] which References to [messageX]

```
[variable1]
name="Var1_mux"
messagename="testmessage1"
timeoutval=133
...
```

D.1.5 [enumX] Section

Each enum is indicated with an [enumX] section. X is a consecutive number, starting with 1. The number of enums is defined with the “enums” keyword in the [global] section.

```
[enum1]
```

name

Name of the enum as a reference to be used with a variable (max. 23 characters).

```
name="enum_test"
```

datatype

Specifies the data type of the variable which uses the enum.

Value	Description
0	unsigned (only positive numbers can be used for the enum)
1	signed (positive and negative numbers can be used for the enum)

```
datatype=0
```

count

Specifies the count of entries in the enum value list. The maximum number of entries in an enum is only limited by the working memory.

```
count=2
```

length

Indicates the length of the longest string that is output by the enum.

```
length=13
```

Enum Value List

The value list of the enum consists of the number of entries as specified in the keyword "count". The value of the variable which uses the enum determines which string from the list is used.

An entry consists of an integer value and the string to be displayed instead of the value. The string should have max. 16 characters length.

```
0="String Value"
```

Example for a [enumX]

```
[enum1]
name="enum_test"
datatype=0          // 0 => unsigned, 1 => signed
count=3             // Number of enum values
length=7            // Number of characters of the longest string
0="Minimum"         // Enum value list
128="Middle"
255="Maximum"
```

D.1.6 [variableX] Section

Each variable is indicated with a [variableX] section. X is a consecutive number, starting with 1. The number of variables is defined with the “variables” keyword in the [global] section.

```
[variable1]
```

name

Name of the variable as a reference to be used with an instrument or a label (max. 23 characters).

```
name="Speed"
```

There are 3 special variable names for addressing device functions or parameters:

```
name="dev_keys"
```

Name your variable `dev_keys` to address the key functions of the PCAN-MiniDisplay. The variable can have the values: 0 for no key pressed, 1 for Enter, 2 for Key up, and 3 for Key down.

```
name="dev_brightness"
```

Name your variable `dev_brightness` to set the display brightness of the PCAN-MiniDisplay. Integer values from 0 to 100 are valid.

```
name=" dev_buzzer"
```

Name your variable `dev_buzzer` to address the buzzer of the PCAN-MiniDisplay. Integer values from 0 to 14 are valid. The buzzer sound is only played once on change of the variable value.

messagename

The name refers to the respective [messageX] (max. 23 characters).

```
messagename=”testmessage1”
```

Note: Since firmware 1.2.x, "messagename" replaces the parameters "frametype" and "canid". If the variable is assigned to a [messageX], "source" is set to "CAN" and the channel is set to the channel value (0 or 1) of the message.

source

Type of source. Available source types: CAN, AIN, and DIN.

```
source=CAN
```

channel

The channel to be used (count starts at 0). The number of supported channels depends on source type.

Source type	Value	Description
CAN	0	CAN channel 1
	1	CAN channel 2
AIN	0	Internal NTC
	1	External input
	2	External input
DIN	4	External input (switch against GND)

```
channel=0
```

position

1st parameter: start bit position of the signal, range: 0 … 63
Note: The bits are counted differently in Intel and Motorola format.
2nd parameter: bit length of the signal, range: 1 … 64

```
position=0,8    // First byte of the CAN data field
position=8,16   // Bytes 2 and 3 of the CAN data field
```

byteorder

Defines the byte order of the CAN data.

Value	Description
0	Intel (“Little Endian”)
1	Motorola (“Big Endian”)

```
byteorder=0  // Intel format
```

muxtype

The data structure of a CAN message is defined by variables. With the help of a multiplexer, different data structures can be defined for the same CAN message. For this purpose, a variable is first created as a multiplexer. Its value specifies which data structure is contained in the CAN message.

"muxtype" specifies whether the variable to be configured is the multiplexer or is contained in the data of the message, depending on a multiplexer.

Value	Description
0	The multiplexer function is not used.
1	The variable is used as a multiplexer. The multiplexer must be defined before all other variables of this CAN message. Only one multiplexer can be created per CAN ID.
2	The variable is contained in the data of the CAN message depending on the value of the multiplexer.

```
muxtype=0
```

muxval

Defines the multiplexer value that can be specified either by decimal (from 0 to 2^{32} -1) or by the hexadecimal value (from 0 to 0xFFFFFFFF). The unsigned value can have a length of 1 to 32 bits.

```
muxval=103
```

scale

Scale for an incoming signal value. The value will be stored as a float (32-bit).

```
scale=2.5
```

offset

Offset for an incoming signal value. The value will be stored as a float (32-bit).

```
offset=5.2
```

maxraw

Maximum raw value (value from CAN message before scale/offset calculation) with internal data type float 32-bit. Any larger value than this is limited to this value. Use either maxraw OR maxval.

maxval

Maximum physical value (value from CAN message after scale/offset calculation) with internal data type float 32-bit. Any larger value is limited to this value. Use either maxval OR maxraw.

minraw

Minimum raw value (value from CAN message before scale/offset calculation) with internal data type float 32-bit. Any smaller value than this is raised to this value. Use either minraw OR minval.

minval

Minimum physical value (value from CAN message after scale/offset calculation) with internal data type float 32-bit. Any smaller value is raised to this value. Use either minval OR minraw.

vartype

Data type of the variable after reading from the CAN message.

Value	Description
0	unsigned, integer
1	signed, integer
2	floating-point number (32 bits)
4	string

```
vartype=1
```

A signed integer variable with an assumed bit length of 8 has a value range from -128 to +127.

datatype

Data type of the variable after reading from the CAN message and further calculation with scale and offset.

Value	Description
0	unsigned, integer
1	signed, integer
2	floating-point number (32 bits)
4	string
5	Enum (for value presentation with an enum)

```
datatype=2
```

The value is internally stored as a floating-point number. This is useful if the value is to be displayed as a fraction number (see labels).

timeoutval

Defines the value of the variable when the "timeout" event occurs (see *timeout* on page 58). This value is calculated with the "scale" and "offset" parameters. If no "timeoutval" is defined, the variable value is set to 0.

```
timeoutval=133
```

enum

The name refers to the respective [enumX] (max. 23 characters).

```
enum=”enum_test”
```

Example of a Variable

```
[variable1]
name="odometer"
messagename="testmessage1" // Name of the message
position=8,16  // Start bit and bit length
byteorder=1    // 0: Intel, 1: Motorola
scale=1.5      // Value=raw * scale + offset
offset=0       // Always float type
vartype=0      // Type of the incoming CAN data:
               // 0: unsigned, 1: signed, 2: float, 4: string
minval=3       // Lower limit is 3
maxval=99999   // Any value > 99999 is set to 99999
datatype=1     // Type of the CAN data after scale & offset:
               // 0: unsigned, 1: signed, 2: float,
               // 4: string, 5: enum
```

Example of a Multiplexer Variable

```
[message1]
enable=1
name="testmessage1"
canid=0x1
frametype=0
channel=0
timeout=0

[variable1]
name="multiplexer"
messagename="testmessage1"
position=0,8
byteorder=1
muxtype=1           // 1=multiplexer
scale=1
offset=0
vartype=0
datatype=0

[variable2]
name="multiplexed1"
messagename="testmessage1"
position=8,16
byteorder=1
muxtype=2           // 2=multiplexed
muxval=103          // This variable is updated if the
scale=1                multiplexer (variable 1) has this value.
offset=0
vartype=0
datatype=0
timeoutval=0

[variable3]
name="multiplexed2"
messagename="testmessage1"
position=8,8
byteorder=1
muxtype=2           // 2=multiplexed
muxval=104          // This variable is updated if the
scale=1                multiplexer (variable 1) has this value.
offset=0
vartype=0
datatype=0
timeoutval=0
```

D.1.7 [labelX] Section

Each text label is indicated with a [labelX] section. X is a consecutive number, starting with 1. The number of labels is defined with the “labels” keyword in the [global] section.

```
[label1]
```

name

Name of the label (max. 15 characters), for documentation.

```
name="odometer"
```

position

X and Y position of label.

1st parameter: X position (horizontal), range: 0 - 31
2nd parameter: Y position (vertical), range: 0 - 239

```
position=10,30
```

font_name

Name of font (max. 63 characters), reference to a font from the section fonts.

```
font_name="Palatino_Linotype_12_14.fon"
```

This selects a font with a width of 12 pixels and height of 14 pixels. See list of available fonts for further font types.

font_idx

Index of an embedded font.

Value	Description
0	“Courier_8_12”, width=8, height=12
1	“Courier_10_14”, width=10, height=14
2	“Courier_12_18”, width=12, height=18

```
font_idx=2
```

length

Number of characters / length of the output string. Maximum is 39 characters.

```
length=5
```

fmt_string

Format specification according to the printf function, known from various programming languages, for output of the variable value.

If no formatting is specified, a standard formatting corresponding to the datatype of the variable is used.

```
fmt_string="%02d km"
```

var_name

Reference to a variable from section variables (max. 23 characters).

```
var_name="speed"
```

initval

Initial value that is output after startup before a relevant variable is received (max. 39 characters).

```
initval="-----"
```

fontcolor

Defines the font color as an 8-bit RGB value. The color components red, green, and blue have a value range from 0 to 255.

```
fontcolor=255,255,255  // white
```

bgcolor

Defines the background color as an 8-bit RGB value. The color components red, green, and blue have a value range from 0 to 255.

```
bgcolor=255,0,0  // red
```

update_rate

Defines the update rate of a label in milliseconds. The label is redrawn with this rate, independent of the reception of a relevant variable.

```
update_rate=100  // Redraw label not faster than every 100 ms
update_rate=0    // Updates label as fast as possible
                 // (each time the variable is received)
```

sector

The value range of a variable can be divided in up to 5 sectors. For each sector a different font color and background color can be defined. Each sector can have the "flashing" keyword set to 1 or 0. A sector can be enabled (1) or disabled (0).

```
sector=1           // sector enabled
range=0,10
fontcolor=255,0,0  // bright red
offcolor=127,0,0   // light red
bgcolor=0,0,0      // black
flashing=1

sector=1           // sector enabled
range=11,244
```

```
   fontcolor=0,255,0   // bright green
   bgcolor=0,0,0       // black
   flashing=0          // no flashing

   sector=1            // sector enabled
   range=245,255
   fontcolor=255,0,0   // bright red
   offcolor=127,0,0    // light red
   bgcolor=0,0,0       // black
   flashing=1
```

Example: An 8-bit unsigned variable has a value range of 0 to 255. When the value is equal 0 to 10, the displayed value is shown in red and flashing. In the range of 11 to 244 is displayed in green. In the range of 245 to 255 it is displayed in red and flashing.

range

A sector is defined for a certain range.

```
   range=20,50
```

This sector is valid if the variable is in the range of 20 to 50 (20 and 50 included).

offcolor

Each sector can have the “flashing” keyword. In case the sector is active and the label is flashing, the color toggles between “fontcolor” and “offcolor”.

```
   offcolor=127,0,0  // middle red
```

Example of a Label

```
[label1]
name="odometer"       // Name of label for reference
position=10,30        // x-pos,y-pos in pixels
                      // 0,0 is top left corner
//font_name=""        // Use either font_name OR font_idx
font_idx=2            // Use integrated font 2
length=8              // Number of characters of the label
fmt_string="%05d km"  // Format string according to printf()
var_name="odomtr"     // Reference to variable
startval="----- km"   // Initial value that will be displayed
fontcolor=0,255,0     // RGB value (bright green)
bgcolor=127,0,0       // RGB value (mid red)

sector=1              // New sector, 1: enabled
range=0,10            // Applicable for values within 0 to 10
fontcolor=0,255,0     // Font color if value in this range
bgcolor=0,0,0         // Background color for this range
flashing=0

sector=1
range=1,20
fontcolor=255,0,0
offcolor=127,0,0
bgcolor=0,0,0
flashing=1
```

Once the keyword “sector” is used, the parameters for “fontcolor” and “bgcolor” are assigned to this sector.

D.1.8 [plotterX] Section

Each plotter is indicated with a [plotterX] section. X is a consecutive number, starting at 1. The number of plotters is defined with the “plotter” keyword in the [global] section.

name

Name of the plotter (max. 15 characters), for reference only.

```
name="Vehicle Speed"
```

position

X and Y position of plotter.

```
position=100,50
```

size

Dimension of the plotter in pixels.

1st parameter: width
2nd parameter: height

```
size=100,50
```

plotcolor

Defines the plot line color as an 8-bit RGB value. The color components red, green, and blue have a value range from 0 to 255.

```
plotcolor=255,0,0      // Color is 100% red
plotcolor=127,127,127  // Color is mid gray
```

bgcolor

Defines the background color as an 8-bit RGB value. The color components red, green, and blue have a value range from 0 to 255.

```
bgcolor=255,255,255  // white background
```

enable

Enables or disables this plotter.

```
enable=1
```

channels

Number of channels for this plotter diagram. Up to 4 channels can be displayed in a plotter. The channel indexes are used to specify name, variable, etc.

```
channels=2
```

timespan

Time in milliseconds that the x-axis should represent.

```
timespan=5000  // 5 s to draw the complete diagram
```

var_nameX

Name of the data source (max. 23 characters). X can be 1, 2, 3 or 4 for up to 4 variables in a plotter diagram. See “channels” keyword.

```
var_name1=speed
var_name2=acceleration
```

yminX

Minimum value of the vertical axis. A value smaller than this is limited to this minimum.

```
ymin1=10
```

ymaxX

Maximum value of the vertical axis. A value greater than this is limited to this maximum.

```
ymax2=250
```

scroll_x

Aktiviert das Scrollen entlang der X-Achse für den Plotter.

```
scroll_x=1
```

D.1.9 [txlist_X] Section

Each transmit list is indicated with a [txlist_X] section. X is one of three transmission types: init, cyclic, or exit. For each transmission type, a maximum of 30 messages can be defined.

```
[txlist_X]
```

txlist_init

The transmit list is executed once when starting a scene.

```
[txlist_init]
```

txlist_cyclic

The transmit list is executed cyclically during the scene.

```
[txlist_cyclic]
```

txlist_exit

The transmission list is executed once when ending a scene.

```
[txlist_exit]
```

After the initiation of the [txlist_X] section, parameters follow whose sequence must be kept:

```
[txlist_X]
count=1
delay,bus,message-ID,frameformat,messagetype,length,data[0],
  data[1],data[2],data[3],data[4],data[5],data[6],data[7]
```

count

Specifies the number of defined CAN messages in the respective transmit list [txlist_X].

```
count=1
```

delay

Sets a delay in milliseconds until the next transmit message is sent.

```
150,bus,message-ID,frameformat,messagetype,length,data[0],…
```

bus

Sets the CAN bus channel for the transmit message.

Value	Description
0	CAN channel 1
1	CAN channel 2

```
delay,1,message-ID,frameformat,messagetype,length,data[0],…
```

message-ID

Specifies the ID of the CAN message as a 11-bit or 29-bit integer value (decimal or hex).

```
delay,bus,0x180,frameformat,messagetype,length,data[0],…
```

frameformat

Defines the CAN frame type as standard frame or extended frame.

Value	Description
s	standard frame (11-bit CAN-ID)
e	extended frame (29-bit CAN-ID)

```
delay,bus,message-ID,s,messagetype,length,data[0],…
```

messagetype

Here it can be determined whether it is a data frame or a remote frame.

Value	Description
d	data frame
r	remote frame

```
delay,bus,message-ID,frameformat,d,length,data[0],…
```

length

Length of the CAN frame (0 to 8 data bytes)

```
delay,bus,message-ID,frameformat,messagetype,4,data[0],…
```

data[x]

Content of the CAN message in 8-bit integer values (decimal or hex)

```
delay,bus,message-ID,frameformat,messagetype,length,0x11,…
```

Examples of Transmit Lists

```
[txlist_init] //Send once by starting a scene.
count=6
100,0,0x80,s,d,4,0x11,0x22,0x33,0x44
200,0,0x81,s,d,5,0x11,0x22,0x33,0x44,0x55
250,0,0x82,s,d,8,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88
100,1,0x80,s,d,4,0x11,0x22,0x33,0x44
200,1,0x81,s,d,5,0x11,0x22,0x33,0x44,0x55
250,1,0x82,s,d,8,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88

[txlist_exit] //Send once by ending a scene.
count=6
100,0,0x180,s,d,4,0x11,0x22,0x33,0x44
200,0,0x181,s,d,5,0x11,0x22,0x33,0x44,0x55
250,0,0x182,s,d,8,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88
100,1,0x180,s,d,4,0x11,0x22,0x33,0x44
200,1,0x181,s,d,5,0x11,0x22,0x33,0x44,0x55
250,1,0x182,s,d,8,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88
```

```
[txlist_cyclic] //Send cyclically during a scene.
count=4
100,0,0x44,s,d,4,0x01,0x02,0x03,0x04
200,0,0x48,s,d,5,0x21,0x22,0x23,0x24,0x25
100,1,0x44,s,d,4,0x01,0x02,0x03,0x04
200,1,0x48,s,d,5,0x21,0x22,0x23,0x24,0x258
```

D.2 Scenes Project File (*.inp)

Note: The text file (`*.inp`) must have an empty line at its end.

D.2.1 [global] Section

version

Version format. Currently only 1.

```
version=1
```

D.2.2 [scenes] Section

scenes

Count of used scenes.

```
scenes=4
```

startscene

This scene will be displayed after startup, default is 1.

```
startscene=2
```

List of Scenes

Format of a scenes list entry:

```
<index>=<filename>
```

The scenes are numbered consecutively, starting with 1. As parameter, the file name of the scene must be indicated. If a scene is defined with the keyword parameter "compress=1" and was already loaded before, the compressed version will be used. This helps to decrease the loading time.

```
[scenes]
scenes=4
startscene=2
1="speedo0.ins"
2="speedo1.ins"
3="picture1.ins"
4="picture2.ins"
```

D.3 CAN ID Filters (*.flt)

The use of a CAN ID filter (`*.flt` file) is determined in the **Device Settings** menu of the PCAN-MiniDisplay. While a recording CAN IDs and ID ranges that are listed in a filter file can pass the filter. All the others can't.

D.3.1 Format Description Filter File

- A `.flt` text file has several sections that are each introduced by the section name in square brackets.
- Depending on the section, it can contain keywords, CAN IDs, or CAN ID ranges.
- A CAN ID is indicated with a decimal or a hexadecimal value (latter with prefix 0x).
- For performance reasons, it is recommended to list CAN IDs in rising order.
- A comment can be inserted starting with a double slash.

Section/Keyword	Description	Example entries
[global]	This section is obligatory and must contain the following two keywords.	
version	Version format. Currently only 1 is valid.	`version=1`
enable	Enables the filter. Currently only 1 is valid.	`enable=1`

[single_11bit]	Defines 11-bit CAN IDs that can pass the filter.	`0x100`
		`1023`

[range_11bit]	Defines 11-bit CAN ID ranges that can pass the filter.	`4-13`
		`0x200-0x340`
		`// Full 11-bit range:`
		`0x000-0x7ff`

[single_29bit]	Defines 29-bit CAN IDs that can pass the filter.	`0x123`
		`0x11111`
		`125000`

[range_29bit]	Defines 29-bit CAN ID ranges that can pass the filter.	`500-550`
		`0x9000-0x10000`
		`0x1F80000-0x1FA0000`
		`// Full 29-bit range:`
		`0x0000000-0x1FFFFFF`

D.3.2 Example Filter File

```
[global]
version=1
enable=1

// This is a comment
[single_11bit]
0x100
1023  // = 0x3FF - Another comment
0x5AB

[range_11bit]
4-13
0x200-0x340
0x7f0-0x7fe

[single_29bit]
0x123
0x11111
```

```
125000  // = 0x1E848

[range_29bit]
500-550
0x9000-0x10000
0x1F80000-0x1FA0000
```

Tip: On the memory card of the PCAN-MiniDisplay the example `Example.flt` is located in the directory: `/MiniDisplay/Filters`.

Appendix E Model with Casing

This section describes the specific features of the model with casing and push buttons (IPEH-002262-KSM01).

This product model provides an environment to facilitate the development of configurations for the PCAN-MiniDisplay.

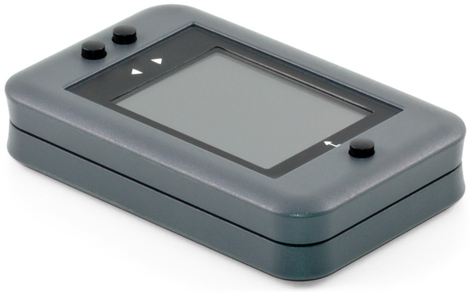


Figure 15: Model with casing and push buttons

E.1 Scope of Supply

- PCAN-MiniDisplay in plastic casing including mating connectors (Phoenix Contact FMC 1,5/10-ST-3,5 1952348 and Phoenix Contact FMC 1,5/4-ST-3,5 1952283)
- 512 MBytes memory card installed
- Conversion software PEAK-Converter for Windows 10, 8.1, 7 (32/64-bit)
- Configuration examples
- Manual in PDF format

E.2 Push Buttons

The three push buttons Up ↑, Down ↓, and Enter ↵ are device-internally connected to J1 of the PCAN-MiniDisplay (see 2.1 on page 9 for details). The use depends on the current mode of the device.

Current Mode	Up ↑	Down ↓	Enter ↵
Scene display	Switch to the next scene if multiple scenes are defined		Leave the scene display and show the main menu
Menus	Skip to previous or next entry of a menu or of a list		Activate the selected entry
Off (supply voltage applied)	No function	Start device	No function

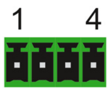
Note: The digital inputs of J1 are also available on the 4-pole outlet.

E.3 Connectors

The model with casing provides three connectors on the back side:

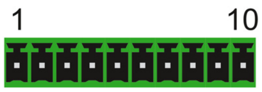
- Mini USB socket
- 4-pole connector for mating spring terminal block (J1 internally)

Pin	Function
1	Din1: Down
2	Din2: Up
3	Din3: Enter
4	GND



- 10-pole connector for mating spring terminal block (J2 internally)

Pin	Function
1	Vb (7 - 30 V DC)
2	GND
3	Dout1
4	Din4
5	Ain1
6	Ain2
7	CAN2_L
8	CAN2_H
9	CAN1_L
10	CAN1_H



E.4 microSD Card Slot

The microSD card slot is accessible by opening the casing.

Do the following to access the microSD card slot:

1. From the bottom of the casing, cautiously remove the four plastic caps.
2. Remove the four screws from the wholes that were covered by the caps.
3. Remove the top of the plastic casing (containing the window).

   The microSD card slot is positioned below the display circuit board, near the single button.

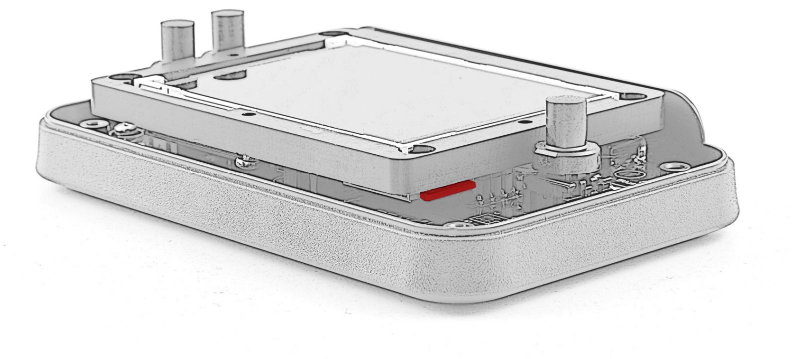


Figure 16: Position of the microSD card slot

4. Slightly insert the card with a suitable tool (e.g. tweezer) until the spring mechanism is released (click sound).
5. Carefully remove the microSD card with a tool.
6. When inserting a microSD card, push it in until the spring mechanism locks (click sound).
7. Follow the first three steps in reversed order to reassemble the casing.

E.5 Technical Specifications Casing Model

The technical data of the PCAN-MiniDisplay Case differ slightly. The differences are given in the following table. Refer to *Technical Specifications* on page 40 for further details.

Technical specifications (only IPEH-002262-KSM01)	
Size	116,5 x 24,5 x 72,5 mm (W x H x D)
Weight	138 g
Ingress protection (IEC 60529)	IP20
Mating connector types	4-pin: Phoenix Contact FMC 1,5/4-ST-3,5 1952283 10-pin: Phoenix Contact FMC 1,5/10-ST-3,5 1952348